LEHRSTUHL FÜR COMPUTERGESTÜTZTE MODELLIERUNG UND SIMULATION
INGENIEURFAKULTÄT FÜR BAU GEO UMWELT
TECHNISCHE UNIVERSITÄT MÜNCHEN

Automatisierte Konformitätsprüfung digitaler Bauwerksmodelle hinsichtlich geltender Normen und Richtlinien mit Hilfe einer visuellen Programmiersprache

CORNELIUS FRANZ DANIEL PREIDEL

Vollständiger Abdruck der von der Ingenieurfakultät Bau Geo Umwelt der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. rer. nat. Thomas H. Kolbe
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. André Borrmann
	2.	Prof. Dr.-Ing. Frank Petzold

Die Dissertation wurde am 23.01.2020 bei der Technische Universität München eingereicht und durch die Ingenieurfakultät Bau Geo Umwelt am 29.04.2020 angenommen.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit einer Automatisierung von Konformitätsprüfungsprozessen im Bauwesen und stellt in diesem Zusammenhang einen neuartigen Ansatz vor, welcher auf der Anwendung einer visuellen Programmiersprache basiert.

Mit Einführung digitaler Methoden, wie insbesondere dem *Building Information Modeling (BIM)*, durchläuft die Baubranche eine Transformation. Die BIM-Methode sieht den konsequenten Einsatz digitaler Bauwerksmodelle über den gesamten Lebenszyklus einer gebauten Anlage hinweg vor. Bei der digitalen Bauplanung stellt neben dem Erzeugungsprozess der Modellinhalte die Qualitäts- und Konformitätsprüfung einen essenziellen Baustein dar. Da sich die einzelnen Inhalte eines Bauwerksmodells auf unterschiedliche qualitative Aspekte beziehen, können bei der Modellprüfung verschiedene Komplexitäts- und Detailstufen unterschieden werden. Die vorliegende Arbeit führt in diesem Zusammenhang ein Ebenenmodell ein, welches die Modellqualität in die Hauptgruppen der datentechnischen, inhaltlichen und gestaltungsplanerischen Qualität untergliedert.

Auf der obersten Ebene dieses Modells steht die Konformität hinsichtlich regulatorischer Anforderungen, wie diese in Normen und Richtlinien des Bauwesens zu finden sind. Aktuell ist der Konformitätsprüfungsprozess mühsam, umständlich und fehleranfällig, da dieser zumeist manuell auf Basis der zweidimensionalen Planung und iterativ bei jeder Planungsänderung durch den zuständigen Planungsingenieur und die Baugenehmigungsbehörde durchgeführt werden muss. Eine Automatisierung dieses Prozesses mithilfe digitaler Methoden ermöglicht es, den Arbeitsaufwand und somit die Kosten zu reduzieren. Gleichzeitig wird die Fehleranfälligkeit minimiert und der der Planungsingenieur von monotoner, iterativer Arbeit befreit, so dass dieser mehr Zeit für zentrale, kreative Arbeiten hat.

Die wesentlichen Herausforderungen bei der technischen Umsetzung eines *Automated Code Compliance Checking (ACCC)* können an der schematischen Grundstruktur des Gesamtprozesses festgemacht werden, welche sich in vier einzelne Prozessschritte gliedert: die Übersetzung des Regelwissens aus den regulatorischen Anforderungen in ein formales und maschinenlesbares Format, die Aufbereitung der Gebäudemodelldaten für die Prüfung, der eigentliche Prüfprozess und schließlich die Aufbereitung der erhaltenen Ergebnisse für die erforderliche Kommunikation mit den Projektpartnern.

Ausgehend von dieser Struktur stellt die vorliegende Arbeit bereits existierende Konzepte für die Sicherung der daten- und informationstechnischen Qualität digitaler Bauwerksmodelle auf Basis formaler Konformitätsüberprüfungen vor und untersucht diese hinsichtlich ihrer jeweiligen Unzulänglichkeiten. Um den bestehenden Herausforderungen zu begegnen und Prozesse zu optimieren, wird die *Visual Code Checking Language (VCCL)* eingeführt. Diese zeichnet sich als visuelle, imperative, strikt-typisierte und objektorientierte Programmiersprache aus und eignet sich explizit für die formale Beschreibung von Konformitätsprüfungsprozessen im Bauwesen. Visuelle Programmiersprachen stellen im Gegensatz zu textuellen Programmiersprachen die beschriebenen Verarbeitungssysteme graphisch und somit besser verständlich für Anwender, die über keine oder aber geringe Programmierkenntnisse verfügen, dar. Der Ansatz wurde prototypisch in einem Demonstrator, mit welchem sich VCCL-Programme für eine praktische Anwendung erstellen lassen, umgesetzt.

Um die VCCL hinsichtlich ihrer Ausdruckskraft und Eignung für die Formalisierung von regulatorischen Anforderungen zu untersuchen, werden in der vorliegenden Arbeit ausgewählter Ausschnitte nationaler Normen mit Hilfe der VCCL kodiert.

Abstract

This thesis deals with the automation of conformity checking processes in civil engineering and presents in this context a new approach based on the use of a visual programming language.

With the introduction of digital methods, as in particular the *Building Information Modeling (BIM)*, the construction industry is undergoing a transformation. The BIM method aims for the consistent use of digital building models over the entire life cycle of a built facility. In digital construction planning, quality and conformity checking is an essential component next to the production process of the model contents themselves. Since the individual contents of a building model refer to different qualitative aspects, different levels of complexity and detail can be distinguished during model checking. In this context, this thesis introduces a level model which subdivides the general model quality into the main groups of data technical, content and design planning quality.

Conformity with regulatory requirements, such as standards and guidelines for the construction industry, can be found at the top-level of this model. Today, the conformity checking process is cumbersome and error-prone, as it is usually carried out manually based on two-dimensional planning and iterative by the responsible planning engineer, as well as the building permitting authority with every planning change.

Automating this process using digital methods has the potential to reduce the amount of work and thus the costs, while freeing the engineer from monotonous, iterative work so that he has more time for central, creative work, and minimizing the risk of errors.

The main challenges in the technical implementation of an *Automated Code Compliance Checking (ACCC)* can be identified by the basic schematic structure of the overall process, which is divided into four individual process steps: the translation of the rule knowledge from the regulatory requirements into a formal and machine-readable format, the preparation of the building model data for the check, the actual checking process and finally the preparation of the results obtained for the necessary communication with the project partners.

In order to meet the existing challenges, the thesis introduces the *Visual Code Checking Language (VCCL)*, a visual, imperative, strictly typed, and object-oriented programming language, which is explicitly suitable for the formal description of conformity checking processes in civil engineering. In contrast to textual programming languages, visual programming languages represent the described processing systems graphically and thus better understandable for users who have little or no programming knowledge. The VCCL was prototypical implemented in a demonstrator with which VCCL programs can be created for a practical application.

To proof the VCCL expressiveness and suitability for the formalization of regulatory requirements, selected sections of national standards are coded with the help of the VCCL in this thesis.

Vorwort

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der Technischen Universität München am Lehrstuhl für Computergestützte Modellierung und Simulation im Zeitraum vom Oktober 2014 bis März 2018 entstanden. Mit der Fertigstellung dieser Arbeit geht mein Promotionsstudium zu Ende und daher möchte ich mich bei allen, die mich während dieser Zeit unterstützt haben, herzlich bedanken.

In erster Linie gilt mein Dank meinem Betreuer, Prof. Dr.-Ing. André Borrmann, für die Anregung zu dieser Arbeit, sein großes Interesse daran und seine stets hervorragende Betreuung.

In gleicher Weise gebührt mein Dank Prof. Georg Nemetschek stellvertretend für die Nemetschek Group, ALLPLAN GmbH und Solibri Inc., die meine Arbeit als Industriepartner mit großem Interesse und Engagement inhaltlich sowie finanziell begleitet haben. Der enge Kontakt mit der Praxis war die Basis für eine Zusammenarbeit mit einem gewinnbringenden Resultat für beide Seiten.

Des Weiteren gilt mein Dank Prof. Dr.-Ing. Frank Petzold, der mir fortwährend als Zweitbetreuer, Mentor und Diskussionspartner zur Seite stand.

Für das Gelingen dieser Arbeit war insbesondere auch mein inspirierendes und diskussionsfreudiges Arbeitsumfeld an der Technischen Universität München von wesentlicher Bedeutung. Daher möchte ich mich bei allen Mitarbeitern des Lehrstuhls Computergestützte Modellierung und Simulation sowie des Lehrstuhls Computation in Engineering für das hervorragende und stets kollegiale Miteinander bedanken.

Abschließend gilt es auch meiner Familie und meinen Freunden Danke zu sagen, die mich immer wieder aufs Neue motiviert haben und mich insbesondere während der zurückliegenden, arbeitsreichen Monate mit Rat und Tat beiseite standen.

Dieser Dank gilt in erster Linie meinen Eltern, die mich auf meinem Weg zur und durch die Promotion stets unterstützt haben. Das besondere Engagement meiner Mutter bei dem Lektorat soll an dieser Stelle nicht unerwähnt bleiben.

Schließlich möchte ich mich ganz besonders bei meiner Frau Janina bedanken, die mir stets den nötigen Rückhalt gegeben und insbesondere in der letzten Phase die nötige Zeit eingeräumt hat, die Dissertation fertig stellen zu können.

Heroldsberg, im Frühjahr 2020

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACCC	Automated Code Compliance Checking
AEC	Architecture, Engineering and Construction
AIA	Auftraggeberinformationsanforderungen
AISC	American Institute of Steel Construction
AG	Auftraggeber
AN	Auftragsnehmer
API	Application Programming Interface
BAP	BIM Abwicklungsplan
BBR	Bundesamt für Bauwesen und Raumordnung
BCA	Building Construction Authority
BCF	BIM Collaboration Format
BERA	Building Environment and Analysis Language
BGB	Bürgerliches Gesetzbuch
BIF	BIM Integration Framework
BIM	Building Information Model
BIM	Building Information Modeling
BIMRL	BIM Rule Language
BLIS	Building Lifecycle Interoperable Software
BMVI	Bundesministerium für Verkehr und digitale Infrastruktur
BOM	BERA Objektmodell
BPMN	Business Process Model and Notation
C3R	Conformance Checking in Construction - Reasoning
CAD	Computer Aided Design
CDE	Common Data Environment
CEN	European Committee for Standardization
CFG	Context Free Grammar
COBie	Construction Operations Building Information Exchange
CORENET	Construction and Real Estate Network
CSTB	Centre Scientifique et Technique du Bâtiment
DBMS	Datenbankmanagementsysteme
DIN	Deutsche Institut für Normung e. V.
EDM	Express Data Manager
FCA	Fire-Code Analyzer
FCM	Fire Compliance Model
FM	Facility Management
FOL	First Order Logic
gbXML	Green Building XML
GSA	US General Services Administration
GUID	Globally Unique Identifier
HOAI	Honorarordnung für Architekten und Ingenieure
HTML	Hypertext Markup Language
IAI	International Alliance for Interoperability
IBC	International Building Code
ICC	International Code Council
ICIS	International Construction Information Society
IDM	Information Delivery Manual
IFC	Industry Foundation Classes
ISO	International Organization for Standardization
LINQ	Language Integrate Query
LOD	Level of Development
LoG	Level of Geometry
LoI	Level of Information
LSC	Life Safety Code
MBO	Musterbauordnung
MVD	Model View Definition
NABau	DIN-Normenausschuss Bau
NIST	US National Institute of Standards and Technology
NLP	Natural Language Processing
OCL	Object Constraints Language
OOM	Objektorientierte Modellierung
OWL	Ontology Web Language
PAS	Publicly Available Specification
PTNB	Plan de Transition Numérique du Bâtiment
RASE	Requirement, Applicability, Selection, and Exception
RDF	Resource Description Framework

RFP	Request for proposal
RSM	Ruleset Manager
SASE	Standards Analysis, Synthesis, and Expression
SBVR	Semantics of Business Vocabulary and Business Rules
SDO	Standards Development Organization
SICAD	Standards Interface for Computer Aided Design
SMC	Solibri Model Checker
SPARQL	SPARQL Protocol And RDF Query Language
SPEX	Standards Processing Expert
SPF	STEP Physical File
SQL	Structured Query Language
STEP	Standard for the Exchange of Product Model Data
SWRL	Semantic Web Rule Language
TGA	Technische Gebäudeausrüstung
UI	User Interface
UML	Unified Modeling Language
VCCL	Visual Code Checking Language
VDC	Virtual Design and Construction
VDI	Verein Deutscher Ingenieure e.V.
VPL	Visual Programming Language
VP	Visual Programming
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition

1 Einführung

1.1 Einleitung

Die vorliegende Arbeit befasst sich mit Methoden sowie Optimierungen zur Qualitätssicherung und -prüfung der Planungsleistungen im Bauwesen mit Hilfe digitaler Werkzeuge. Die Bestimmung und Sicherstellung der Qualität bezieht sich hierbei im Besonderen auf den Daten- und Informationsgehalt digitaler Modelle, welche gemäß der BIM-Methode für die Planungsaufgaben verschiedener Fachdisziplinen zum Einsatz kommen. Die inhaltliche Korrektheit der verwendeten Modelle lässt sich grundlegend in zwei verschiedene Ebenen gliedern, welche in der vorliegenden Arbeit aufbauend aufeinander betrachtet werden:

- Die Korrektheit der im digitalen Bauwerksmodell enthaltenen Informationen hinsichtlich daten- und informationstechnischer Anforderungen, welche sich aus allgemeingültigen sowie projektspezifischen Festlegungen ergeben.
- Die Korrektheit und Konformität der Objektplanung, welche sich aus der Gesamtheit der im digitalen Bauwerksmodell enthaltenen Informationen ergibt, im Blick auf die mannigfaltigen Vorschriften und Richtlinien, welche im Bauwesen in Abhängigkeit des geltenden Rechts anzuwenden sind.

Hierbei stellt insbesondere die Prüfung eines Gebäudemodells im Blick auf die Konformität mit den geltenden Vorschriften im Bauwesen eine komplexe Herausforderung dar. Forschungsansätze der vergangenen Jahrzehnte, welche eine Automatisierung dieser Konformitätsprüfung verfolgen, zeigen bisher nur einen unbefriedigenden Erfolg. Die in den regulatorischen Dokumenten enthaltenen Bauregeln lassen sich von den Endanwendern, welche typischerweise für die Durchführung der Prüfprozesse verantwortlich sind, mit den vorgeschlagenen Methoden nur eingeschränkt bzw. teilweise abbilden.

Die vorliegende Arbeit vertritt die These, dass eine Automatisierung dieser Konformitätsprüfung nur erfolgreich und nachhaltig umgesetzt werden kann, wenn das bereitgestellte Werkzeug für die Beschreibung des Regelwissens für den jeweiligen Benutzer und die übersetzten Inhalte nachvollziehbar bleiben. Gleichzeitig muss gewährleistet sein, dass sowohl alle regulatorischen Inhalte der Bauvorschriften als auch die geometrischen sowie nicht-geometrischen Inhalte der Modelle beschrieben werden können. Ausgehend von dieser Annahme wird eine Methode vorgestellt, welche auf der Anwendung einer visuellen Programmiersprache basiert. Eine visuelle Sprache bietet den wesentlichen Vorteil, dass auch Nicht-Programmieren verhältnismäßig komplexe Anfragen und regulatorische Anforderungen mit Hilfe von Methoden formulieren können, auch wenn diese nicht über tief greifende Programmierkenntnisse verfügen.

Somit erlaubt eine solche Sprache das Bauregelwissen so zu formalisieren, dass der Anwender die entsprechenden Regeln im übersetzten Zustand nachvollziehen und diese bei Bedarf an die eigenen

Bedürfnisse anpassen kann. Auf diese Weise soll die Beschreibung von Regelwissen insbesondere für die typischen Anwender der Konformitätsprüfung, also Architekten und Ingenieure, anwendbar sein.

Mit einer erfolgreichen Automatisierung der Konformitätsprüfung können so die Prüfläufe schneller, effizienter und sicherer durchgeführt werden, was schließlich zu einer Reduktion der Kosten und Risiken, aber auch zu mehr Raum für Innovation und die Entwicklung von Handlungsoptionen führt. Überdies entlastet die Automatisierung den Anwender von einer monotonen Arbeit und führt so zu einer verbesserten Arbeitsqualität.

1.2 Ausgangspunkt

Grundlegender Zweck von Normen, Richtlinien und Technikstandards ist es, Mindestanforderungen festzulegen, die zum Schutz der öffentlichen Gesundheit, Sicherheit und des nachhaltigen Ressourcenschutzes in der bebauten Umwelt erforderlich sind. Diese Festlegungen definieren technische Spezifikationen und vereinheitlichen wesentliche Anforderungen, um beispielsweise die Standsicherheit, Zugänglichkeit, Materialqualität und nicht zuletzt die Sicherheit des Anwenders allgemeingültig gewährleisten zu können. Da sich diese Vorschriften auf unterschiedliche Lebensphasen sowie fachspezifische Leistungsmerkmale eines Bauwerks beziehen, gibt es eine sehr große Anzahl von Regelwerken und Richtlinien. Gemäß dem Geltungsbereich findet die jeweilige internationale, nationale oder aber regionale Gesetzgebung Anwendung. Weiterhin müssen die funktionellen Anforderungen und Ansprüche des Auftraggebers beachtet werden.

Die Zahl von Normen und Vorschriften im Bauwesen ist in den letzten Jahren in erheblichem Umfang gestiegen. In Deutschland sind aktuell etwa 3000 Normen bzw. Vorschriften für das Bauen relevant, was in Deutschland einen Spitzenwert darstellt (Matzig, 2018). Überschneidungen von in Deutschland geltenden regionalen, nationalen und europäischen Normen haben dazu geführt, dass man öffentlich bereits von einem *Vorschriften-Dschungel* spricht (Hahmann, 2015; Matzig, 2018; Passarge, 2010). Diese Vielzahl an Normen führt zu einem signifikanten Mehraufwand und somit zu einer deutlichen Verteuerung der Bau- und somit Wohnkosten. So hat laut Matzig (2018) die Einführung immer neuer Bauvorschriften und Standards in den Jahren 2007 bis 2014 zu einer Verteuerung der Baukosten im Wohnungsbau um 40% geführt. Dabei handelt es sich jedoch nicht um ein explizit deutsches Problem, denn auch in anderen Ländern gibt es eine enorme Anzahl von länderspezifischen nationalen und regionalen Normen (Hahmann, 2015). Trotz der voranschreitenden Vereinheitlichung von Richtlinien im Bauwesen, wie beispielsweise durch die Harmonisierung der geltenden Regelwerke in den Staaten der EU mit der Einführung der Europäischen Normen (EN), ergibt sich eine enorme Anzahl von Vorschriften, welche während der Gestaltungsplanung berücksichtigt werden müssen.

Regelungen und Anforderungen, welche sich aus Rechtsquellen ableiten und somit einen Rechtscharakter erhalten, müssen von den Ingenieuren und Architekten bei der eingereichten Gestaltungsplanung nicht nur berücksichtigt werden, sondern werden auch vonseiten der Baubehörden geprüft, um so schließlich feststellen zu können, ob eine Genehmigung für die Bauplanung erteilt oder aber Nachbesserungen verlangt werden müssen (Hjelseth, 2015b). Folglich handelt es sich bei der Genehmigungsprüfung um einen essenziellen Prozess mit einem erheblichen Einfluss auf den Erfolg des Gesamtprojekts.

Da es bei der herkömmlichen Prüfung um einen weitestgehend manuellen Vorgang handelt, ist der Aufwand, welcher mit der Prüfung der Gestaltungsplanung auf Einhaltung der geltenden Vorschriften

und Normen verbunden ist, besonders hoch. Zudem muss dieser im Falle von notwendigen Anpassungen der Planung von beiden Seiten, Planer und Prüfer, iterativ wiederholt werden. Als Grundlage für den Prozess dienen zumeist herkömmliche Planungsunterlagen, also insbesondere zweidimensionale technische Zeichnungen, welche in ihrer Gesamtheit die Gestaltungsplanung beschreiben.

Aufgabe des zuständigen prüfenden Ingenieurs ist es, die Gestaltungsplanung anhand der zumeist in natürlicher Sprache, Tabellen, Gleichungen und Abbildungen niedergelegten Normen und Regelungen zu untersuchen. Das eigentliche Regelwissen ist also von dem jeweiligen Ingenieur aus den regulatorischen Dokumenten selbst abzuleiten. Es bedarf somit auf dieser Seite eines hohen Maßes an Spezialwissen im jeweiligen Fachbereich und den dazugehörigen Normen sowie einer großen Sorgfalt und Umsicht. Eine weitere Herausforderung besteht darin, dass die Inhalte der zumeist in natürlicher Sprache verfassten Bauvorschriften Ausdrücke mit Mehrdeutigkeit, Unklarheit oder sogar Widersprüche enthalten können (Nawari, 2012a,b).

Gemäß einer Studie von McGrawHill Construction (2007) gibt es im Bauwesen innerhalb der verschiedenen Disziplinen ein hohes Interesse an einer Automatisierung der Prüfprozesse, um eine Reduktion des Aufwands und der Fehleranfälligkeit zu erreichen. Mit der Einführung und Entwicklung digitaler Methoden und einheitlicher Datenstandards für Gebäudemodelle stehen dem Bauwesen Technologien zur Verfügung, die eine sehr geeignete Basis für Optimierungen bilden. An dieser Stelle kann insbesondere ein Einsatz der Methode Building Information Modeling (BIM) hilfreich sein. In BIM-basierten Projekten entsteht im Laufe der Planungsphase ein digitales Abbild der baulichen Anlage, welches die gesamten Informationen für alle Projektbeteiligten und über den gesamten Lebenszyklus des Bauwerks zur Verfügung stellt. Es bietet sich an, diese bereits gebündelten Daten für eine solche Überprüfung eines Modells auf Einhaltung von Normen und Richtlinien zu verwenden und so die Effizienz des Prüfprozesses zu optimieren. Erste Ansätze für eine solche effizientere Gestaltung der Planungsprüfung sowie schließlich auch der Baugenehmigung lassen sich auf internationaler Ebene, aber auch in Deutschland bereits erkennen (Eastman et al., 2009; enbausa.de, 2018; Fiedler, 2015).

Die mit einer automatisierten Planungsprüfung verbundenen Herausforderungen sind jedoch vielfältig. Für eine Automatisierung der Prüfung muss das Regelwissen, welches in den regulatorischen Dokumenten enthalten ist, in ein von Maschinen lesbares Format transformiert werden. Die kognitive und analytische Fähigkeit des menschlichen Gehirns ist nicht vergleichbar zu allem, was bisher in diesem Bereich in Computersystemen implementiert wurde. Daher stellt die Automatisierung dieses Prozesses eine grundlegende Herausforderung für die AEC-Industrie dar (Nawari, 2018; Nawari und Alsaffar, 2015).

Bisherige Forschungsansätze sowie bereits verfügbare Werkzeuge, die sich auf ein solches Automated Code Compliance Checking (ACCC) konzentrieren, weisen bisher erhebliche Defizite auf: (1) Viele der Ansätze arbeiten mit fest implementierten Regeln, sodass der Rechencode und somit der eigentliche Prüfprozess verborgen sind. Als Ergebnis kann ein Prüfingenieur, welcher über keine fundierten Programmierkenntnisse verfügt, das Ergebnis nicht validieren. (2) Darüber hinaus sind viele der Ansätze unflexibel oder aber bieten zu wenig aussagekräftige Methoden in Bezug auf die Anpassung der Verfahren an nationale Anforderungen oder das Hinzufügen neuer Verfahren.

1.3 Motivation

Nicht zuletzt durch die Einführung moderner Softwarelösungen und digitaler Methoden ist im heutigen Bauwesen ein deutlicher Trend hin zu immer komplexeren Bauvorhaben zu beobachten. Diese Entwicklung bringt auch kontinuierlich höhere Anforderungen an die Bau- und Planungsleistungen mit sich (Baumanns et al., 2016). Bauprojekte im Allgemeinen sind von einer hohen Anzahl organisatorischer Abhängigkeiten geprägt, da die am Bauprojekt beteiligten Akteure bei der Planung und Erstellung einer baulichen Anlage über die verschiedenen Phasen in unterschiedlichen Zusammensetzungen und vertraglichen Konstellationen kooperieren. Dadurch ergibt sich eine sehr enge Verzahnung der einzelnen Planungsaufgaben und -leistungen, welche durch eine steigende Komplexität der Bauaufgabe wiederum selbst verstärkt wird. Die Vielzahl von Schnittstellen in Bauvorhaben bringt hohe Transaktionskosten und Fehleranfälligkeit für das Bauprojekt mit sich.

Als wesentlicher Schlüssel für die Optimierung in der Bewältigung dieser Herausforderung gilt die voranschreitende Entwicklung von neuen Technologien, durch die die Baubranche gegenwärtig einen grundlegenden Wandel erlebt. Dieser Trend spiegelt sich auch in den Strukturen und Strategien vieler Unternehmen des Bauwesens wider, welche sich vermehrt für digitale Methoden öffnen und Ressourcen für die Entwicklung in diesem Bereich aufwenden (Baumanns et al., 2016). Nicht zuletzt haben diese Themen durch die in den letzten Jahren entstandenen erheblichen Kostensteigerungen und Verzögerungen bei der Durchführung von deutschen Großprojekten großes politisches und mediales Interesse erhalten und einen Optimierungsbedarf hinsichtlich des Einsatzes digitaler Planungswerkzeuge in komplexen Bauvorhaben aufgedeckt (Kammholz, 2014).

Als Konsequenz wurde auf politischer Seite die *Reformkommission Großprojekte* ins Leben gerufen. Diese hat 2015 in ihrem Endbericht schließlich die Förderung einer stärkeren Nutzung digitaler Methoden durch die Bundesregierung empfohlen (BMVI, 2015a). Für die Umsetzung der Empfehlungen wurde ein konkreter Stufenplan entwickelt, welcher sukzessive die Voraussetzungen dafür schaffen soll, dass digitale Methoden in zunehmendem Umfang bei der Planung und Realisierung von infrastrukturellen Großprojekten angewendet werden können (BMVI, 2015b). Als Grundlage für die Anwendung müssen hierfür insbesondere die digitalen Anforderungen festgelegt, Standards vereinheitlicht und Konzepte zum Planungs- und Bauablauf mit Hilfe dieser Methoden entwickelt werden.

Obwohl sich CAD-Systeme in der Baupraxis weitestgehend etabliert haben, werden viele Prozesse innerhalb des Bauwesens noch nicht stringent im Sinne eines vollständig digitalen Arbeitsablaufs verwendet. Viele Prozesse werden noch immer manuell gesteuert und mittels zweidimensionaler Planung auf dem Papier bearbeitet. Im Blick auf die Nutzung und das Ausschöpfen vorhandener Optimierungspotentiale werden seit einiger Zeit Möglichkeiten und Grenzen neuere Instrumente und Funktionalitäten im Bauwesen ausgelotet (Garrett et al., 2014).

Enorme Optimierungschancen ergeben sich insbesondere im Bereich der Baugenehmigungs- bzw. Baubewilligungsprozesse. Auch diese Prozesse werden heutzutage in vielen Ländern noch weitgehend papiergestützt und manuell durchgeführt. Es handelt sich hierbei um einen essenziellen Vorgang für ein jedes Bauprojekt, da dieses die rechtliche Grundlage und Voraussetzung für die Durchführung eines Projektes darstellt. Diverse Pilotprojekte in Vorreiterländern konnten das Optimierungspotential, welches in einer Digitalisierung und Automatisierung dieser Arbeitsabläufe und Prüfprozesse liegt, zeigen. So konnte beispielsweise in Norwegen von der nationalen Behörde *Statsbygg* durch Formalisierung und

Automatisierung von Prüfprozessen im Bereich der Barrierefreiheit in ersten Tests 60-70% der Fehler reduziert werden (Eastman et al., 2009). In Singapur wiederum konnte durch die Einführung einer nationalen digitalen Plattform für das Baubewilligungsverfahren die Anzahl der benötigten Tage für das Verfahren drastisch reduziert werden (Fiedler, 2015).

Wesentliche Mehrwerte einer (Teil-)Automatisierung der Konformitätsprüfungen ist somit die Qualitätssteigerung des Prozesses an sich, eine wesentliche Beschleunigung insbesondere der iterativen, manuellen Prüfaufgaben und nicht zuletzt eine Vereinheitlichung der Prozesse.

Daher wird in der vorliegenden Arbeit ein neuartiger Ansatz für die formale Beschreibung von Regelwissen aus Bauvorschriften vorgestellt, welcher auf dem Einsatz einer eigens entwickelten visuellen Programmiersprache basiert. Diese Programmiersprache ermöglicht Domäne-Experten, erforderliche Prüfprozesse selbst zu definieren und die Ergebnisse der einzelnen Verarbeitungsschritte interaktiv zu prüfen.

1.4 Aufbau

Die vorliegende Arbeit gliedert sich in zwei Hauptteile.

In **Teil I** werden die wesentlichen Grundlagen für Kooperationsprozesse im digitalen Bauwesen erläutert. Diese dienen als Basis für die Erstellung von qualitativ hochwertigen Gebäudemodellen und sind somit Ausgangspunkt für Prüfungen hinsichtlich qualitativer Kriterien.

Kapitel 2 bietet eine umfassende Einführung in die grundlegenden Konzepte beim Einsatz digitaler Modelle im Bauwesen. Hierbei werden insbesondere kollaborative Prozesse untersucht, welche eine Voraussetzung oder aber einen wesentlichen Einfluss auf Qualitätsprüfung der Modellinhalte haben.

In **Kapitel 3** wird die Qualitätsprüfung selbst hinsichtlich daten- und informationstechnischer Kriterien, welche sich aus grundlegenden aber auch projektspezifischen Anforderungen ergeben, vorgestellt. Hierbei wird insbesondere auf die unterschiedlichen Stufen und Formen der Qualität eines digitalen Bauwerksmodells eingegangen. Hierdurch werden unter anderem auch die Voraussetzungen diskutiert, welche erfüllt sein müssen, um eine Konformitätsprüfung durchführen zu können.

In **Teil II** der Arbeit wird die automatisierte Konformitätsprüfung digitaler Gebäudemodelle hinsichtlich geltender Normen und Richtlinien das Bauwesen behandelt.

Hierfür werden in **Kapitel 4** zunächst die regulatorischen Anforderungen, welche sich aus diversen geltenden Dokumenten und Quellen im Bauwesen ergeben, sowie unterschiedlich Darstellungstypen dieser Anforderungen selbst, vorgestellt.

In **Kapitel 5** wird die allgemeine Struktur eines Automatisierten Konformitätsprüfungsprozesses sowie dessen einzelne Bestandteile jeweils im Detail vorgestellt. Abschließend werden die wesentlichen Herausforderungen diskutiert.

In **Kapitel 6** wird ein umfassender Überblick zu den bisherigen Forschungsansätzen sowie kommerziellen Anwendungen im Bereich des ACCC gegeben.

Kapitel 7 führt schließlich eine neue Methode zu einem ACCC ein, welche die Unzulänglichkeiten bisheriger Ansätze überwinden soll. Bei der Visual Code Checking Language (VCCL) handelt es sich um eine visuelle Programmiersprache, welche sich spezifisch für die Formulierung von Prüfprozessen im Bauwesen auf Basis von digitalen Gebäudemodellen eignet und insbesondere auch von Nicht-Programmierern angewendet werden kann.

In **Kapitel 8** wird die Eignung der zuvor vorgestellten VCCL hinsichtlich ihres Einsatzzwecks untersucht. Hierfür werden diverse Anwendungsfälle für die VCCL exemplarisch dargestellt.

Kapitel 9 fasst die Inhalte und Ergebnisse der vorliegenden Arbeit zusammen und gibt einen Ausblick für weitere Entwicklungen, welche auf dieser Arbeit aufbauen können.

Teil I

Kooperationsprozesse im Digitalen Bauwesen

2 Digitale Methoden im Bauwesen

2.1 Einführung

Zunächst soll eine umfassende Einführung in die grundlegenden Themen und Konzepte zu den digitalen Methoden im Bauwesen gegeben werden. Wesentliche Aspekte sind hierbei die Nutzung und der Austausch der Informationen über die verschiedenen Phasen des Lebenszyklus eines Bauwerks hinweg. Dabei sollen neben den rein technologischen insbesondere auch die methodischen bzw. prozesstechnischen Grundlagen der Zusammenarbeit näher betrachtet werden. Schließlich werden als wesentliche Kriterien für eine erfolgreiche Zusammenarbeit die Modellqualität und die zugehörigen Prüfmethoden, die im digitalen Bauwesen zum Einsatz kommen, im Detail vorgestellt und erörtert.

2.2 Zusammenarbeit im Bauwesen

Die Basis für die Erstellung einer baulichen Anlage ist die gedankliche Entwicklung und Gestaltung des Vorhabens. Die Summe aller Vorgänge, welche diese Entwicklung aber auch die planerische Begleitung während der eigentlichen Erstellung der baulichen Anlage beschreiben, wird als Bauplanung bezeichnet (Nestle und Frey, 2003; Neufert und Kister, 2016). Das Bauwerk selbst, die Planung sowie der Prozess des Errichtens zeichnen sich durch die folgenden Eigenschaften aus:

Einmaligkeit. Wegen der wechselnden Randbedingungen (u. a. Umwelt, Standort) und Anforderungen (u. a. Nutzung, Zweck) zeichnet sich ein Bauprojekt im Wesentlichen durch Einmaligkeit aus. Entsprechend ist jedes Bauwerk als Unikat zu betrachten.

Interdisziplinarität. Die Erstellung und Planung eines Bauwerks bestehen aus unterschiedlichen fachspezifischen Leistungen, welche aus Gründen der Wirtschaftlichkeit von spezialisierten Projektbeteiligten erbracht werden.

Komplexität. Je mehr Beteiligte innerhalb eines Projekts interagieren, desto höher ist die Komplexität dieses Projekts. Aufgrund der beiden vorausgegangenen Eigenschaften haben Bauprojekte in der Regel eine hohe Komplexität im Vergleich zu den Anforderungen anderer Industriebranchen.

Die Planung, Erstellung und Bewirtschaftung eines Bauwerks stellen als jeweils einzelne Lebenszyklusphase eine eigene Herausforderung hinsichtlich der Mitarbeit bzw. Zusammenarbeit zwischen den Beteiligten dar. Um diese Gesamtaufgabe zu bewältigen, arbeiten die beteiligten Ingenieure und Architekten über Zeiträume unterschiedlicher Spanne in verschiedenen Disziplinen und variierenden Formationen zusammen.

Die Planung und Erstellung eines Bauwerks ist durch eine hohe Anzahl stark spezialisierter Beteiligter unterschiedlicher Fachdisziplinen charakterisiert (Luo et al., 2017). Die steigende Komplexität

der Anforderungen für die Bauaufgaben in den letzten Jahren hat zu einer immer tiefer greifenden Spezialisierung der Fachplanung geführt. Gleichzeitig bewirkt eine Verkürzung der Planungszeiträume eine Verdichtung der Schnittstellen zwischen den in einem Projekt involvierten Fachplanern. Um diese Komplexität zu bewältigen ist ein hohes Maß an zweckgerichtetem Zusammenwirken im Sinne einer Kooperation zwischen den einzelnen Planenden notwendig (Borrmann et al., 2015b).

Sowohl in der deutschen als auch der englischen Sprache wird der Begriff Zusammenarbeit sehr häufig mit ähnlichen, aber doch in der Bedeutung unterschiedlichen Ausdrücken belegt. Um im Folgenden die Bauplanung näher betrachten zu können, sollen diese Ausdrücke grundlegend voneinander getrennt und in ihrer Nuancierung näher erläutert werden (Borrmann, 2007; Duden, 2013; Roschelle und Teasley, 1995; Schuman, 2006):

Kooperation beschreibt die parallele Arbeit von einzelnen Teams oder Personen an unterschiedlichen Teilaufgaben auf ein gemeinsames Endergebnis hin. Die einzelnen Beteiligten sind dabei nicht an der Produktion aller Ergebnisse einer einzelnen Teilaufgabe beteiligt, sondern tragen zu dem Gesamtergebnis lediglich bei.

Kollaboration beschreibt, dass einzelne Personen oder Teams parallel an dem Ergebnis einer einzelnen Teilaufgabe arbeiten. Hierbei sind die jeweiligen Beteiligten an der Produktion aller Ergebnisse der Gesamtaufgabe involviert. Die einzelnen Teilschritte und -aufgaben folgen aufeinander und sind somit sequentiell. Hiermit wird also das ineinandergreifende Zusammenarbeiten und somit eine Ausprägung der Kooperation beschrieben.

Koordination beschreibt die Abstimmung von verteilten Elementen. Das Ziel ist neben Informationen auch andere Ressourcen, wie insbesondere gemeinsames Material, bereitzustellen, ohne dass es hierbei zu Konflikten kommt.

Kommunikation beschreibt den Austausch und die Übertragung von Informationen und dient der Verständigung untereinander.

2.3 Digitale Methoden im Bauwesen

Hoher wirtschaftlicher und demografischer Druck sowie die steigende Bedeutung von Umwelt- und Nachhaltigkeitsaspekten haben dazu geführt, dass die Komplexität und Größe von Bauprojekten in den letzten Jahren signifikant gestiegen ist. Gemäß einer Studie von Oxford Economics (2017) werden die globalen Ausgaben im Bereich Infrastruktur bis zum Jahr 2025 voraussichtlich 9 Billionen US-$ pro Jahr übersteigen und somit ist zu erwarten, dass dieser Trend nicht nur anhalten, sondern sich vielmehr verstärken wird.

In der Folge ergibt sich eine zunehmende Verzahnung von immer spezifischeren Planungsleistungen verschiedener Projektbeteiligter, was schließlich dazu führt, dass ein hohes Maß an Kooperation notwendig ist (Borrmann, 2007). Außerdem bringen heutige Bauprojekte Herausforderungen mit sich, welche sich immer schwieriger allein mit konventionellen Methoden bewältigen lassen. In einem Bericht des US National Institute of Standards and Technology (NIST) stellen Gallaher et al. (2004) fest, dass Kosten in Bauprojekten nur dann signifikant gesenkt werden können, wenn (1) die Anzahl der Fehler auf der Baustelle reduziert und (2) die Effizienz der Arbeit in den Büros verbessert wird. Sie führen an,

dass dieses in einem ersten Schritt erreicht werden kann, wenn die Projektbeteiligten Zugriff auf alle benötigten Projektinformationen haben.

Mit der Einführung digitaler Werkzeuge wurde wesentlich dazu beigetragen, die Effizienz der Prozesse im Bauwesen zu steigern. Das Aufkommen von Computern und digitaler Technologien haben zu neuen Formen der Zusammenarbeit zwischen den Akteuren der Baubranche geführt, welche dazu beitragen helfen, die Effizienz zu verbessern (Gonçal, 2017). Die Umstellung von herkömmlichen Methoden auf computergestützte Arbeitsweisen wird gemeinhin als Digitalisierung bezeichnet (Schober et al., 2016).

Der Austausch von Informationen im Bauwesen basiert heutzutage noch immer zu einem großen Teil auf zweidimensionalen technischen Zeichnungen, die sämtliche Bauwerksinformationen enthalten, welche wiederum in den einzelnen Phasen zur Gestaltung, dem Errichten oder aber dem Betrieb benötigt werden (Vismann, 2017). Wie auch andere Industriebranchen befindet sich das Bauwesen allerdings bereits seit einigen Jahren in einem digitalen Wandel. Diese digitale Transformation eröffnet große Chancen für die Anwendung neuer oder aber das Überdenken herkömmlicher Methoden und stellt gleichzeitig neue Herausforderungen, welche bewältigt werden müssen. Bisher verläuft diese Umstellung in der Baubranche nur sehr schleppend und so nutzen gemäß Schober et al. (2016) bisher weniger als 6% der Bauunternehmen digitale Planungsinstrumente vollständig. Zu den Hintergründen einer vergleichsweise langsamen Adaption digitaler Instrumente im Bauwesen zählen die typischen Charakteristika, durch welche sich die Branche auszeichnet: die Abwicklung eines Bauprojekts wird nicht von einem einzelnen Unternehmen durchgeführt, sondern in Zusammenarbeit vieler Akteure. Dabei ist diese Zusammenarbeit auf ein Projekt oder aber auch nur auf eine Bauphase beschränkt, was wiederum zu der bereits genannten Steigerung der Komplexität führt (Borrmann et al., 2015b).

Die voranschreitende Digitalisierung der Bauindustrie ist insbesondere von der Einführung der beiden Begriffe *Virtual Design and Construction (VDC)* und *Building Information Modeling (BIM)* geprägt. Die Bedeutung beider Begriffe ist eng miteinander verwandt.

Kunz und Fischer (2012) bezeichnen mit dem Begriff VDC die Verwendung von multidisziplinären Modellen zur Planung und Durchführung von Bauprojekten und zur Unterstützung der Unternehmensziele z.B. durch bessere Messbarkeit. Mit Hilfe von Modellen werden alle relevanten Sichtweisen auf ein Bauprojekt, also die des Architekten, des Ingenieurs, des Bauunternehmers oder aber des Bauherrn, und somit das gesamte Bauprojekt selbst ganzheitlich computergestützt beschrieben. Dabei stellen die in einem digitalen Bauwerksmodell enthaltenen Informationen alle Aspekte dar, welche entworfen und verwaltet werden können. Weiterhin sind diese Modelle logisch in dem Sinne integriert, dass diese auf gemeinsam genutzte Daten zugreifen können.

Eng verbunden ist dieser Ansatz mit dem Begriff des Building Information Modeling (BIM). Bereits in den 1970er Jahren wurde das Konzept zu dieser Methode erstmals in Forschungsarbeiten von Eastman et al. (1974) über die Erstellung und den Einsatz virtueller Gebäudemodelle veröffentlicht. Eine einzelne einheitliche Definition zu diesem Begriff hat sich seither allerdings nicht etabliert, sondern es existieren vielmehr diverse Deutungen, welche verschiedene Aspekte jeweils unterschiedlich stark hervorheben.

Eastman et al. (2011) bezeichnet BIM als einen Prozess, der die Erzeugung und Verwaltung digitaler Darstellungen physischer und funktionaler Merkmale von baulichen Anlagen beschreibt. Diese Darstellung wird in einem oder in einer Vielzahl von Modellen, den BIMs, verwaltet, welche als gemeinsame Wissensressource für alle Informationen über die bauliche Anlage dient.

Das *National BIM Standard (NBIMS-US)* definiert BIM als (NBIMS, 2017)

> *eine digitale Darstellung der physikalischen und funktionalen Eigenschaften einer Anlage. Ein Gebäudeinformationsmodell ist eine gemeinsame Wissensressource für Informationen über eine Anlage, die eine verlässliche Entscheidungsgrundlage während ihres Lebenszyklus bildet; definiert als vorhanden von der frühesten Konzeption bis zum Abriss.*

BIM umfasst als Konzept also zwei grundlegende Bedeutungen: den Prozess (z.B. das kooperativ Erstellen, die Modellierung) und das Produkt (z.B. das Modell, das digitale Abbild). Über den gesamten Lebenszyklus der baulichen Anlage hinweg, von der frühen Konzeption bis zum Abriss oder Wiederaufbau, dient diese Ressource als verlässliche Grundlage für Entscheidungen. Zentraler Aspekt von BIM ist die Zusammenarbeit der Projektbeteiligten verschiedener Fachdisziplinen, die in dem Projekt und in den verschiedenen Phasen des Lebenszyklus in unterschiedlichen Konstellationen zusammenarbeiten und gemäß ihren Aufgaben Informationen in das jeweilige BIM einfügen, beziehen, aktualisieren oder aber modifizieren (Borrmann et al., 2015b; NIBS, 2012; Succar, 2010). Somit wird das für ein Bauprojekt wesentliche Erfolgskriterium Interdisziplinarität, wie in Abschnitt 2.2 behandelt, adressiert.

Die Bauwerksinformationen werden im Gegensatz zu den herkömmlichen Methoden in der Bauplanung primär nicht in technischen Zeichnungen, sondern in digitalen Modellen vorgehalten, wodurch ein tief greifender Einsatz von Computerunterstützung bei Planung, Bau und Betrieb von Bauwerken möglich wird (Borrmann et al., 2015b). Dieses schließt insbesondere die Koordination der Planung, die Anbindung von Simulationen, die Steuerung des Bauablaufs und die Übergabe von Gebäudeinformationen an den Betreiber ein. Ziel ist es, die digitalen Informationen über die verschiedenen Phasen hinweg konsequent weiterzunutzen, somit Neueingaben und Fehler zu vermeiden und schließlich die Effizienz und Qualität zu steigern. Die verwendeten digitalen Bauwerksmodelle ergeben in ihrer Gesamtheit ein umfassendes digitales Abbild, auch als *digitaler Zwilling* bezeichnet (Pilling, 2016; Weidner, 12.01.2017), eines realen Gebäudes in der jeweiligen Lebenszyklusphase, welches zu dem aktuellen Planungsstand betrachtet wird.

Bei der Einführung digitaler Methoden wie BIM gilt es zu berücksichtigen, dass diese nicht ausschließlich von der Entstehung und Weiterentwicklung technischer Hilfsmittel oder Werkzeuge abhängig ist. Ein Anwender setzt ein technisches Hilfsmittel im Rahmen eines Prozesses ein, um ein Ziel zu erreichen. Neben dem Werkzeug spielen insbesondere die Fähigkeit und der Kenntnisstand des Anwenders selbst sowie die prozesstechnische Methode und Arbeitsweise eine wesentliche Rolle. Dabei ist es sehr hilfreich, wenn entsprechendes Wissen aus vorangegangenen Projekten bereits in Form von Standards oder aber Richtlinien vorliegt. Die Technologie oder Technik in Form von sogenannten BIM-fähigen Softwareprodukten stellt also nur einen Baustein dar und ermöglicht lediglich die Anwendung der Methode. BIM darf also nicht als reine Technologie, sondern muss vielmehr als Methode verstanden werden, die sich mehreren Elementen zusammensetzt, welche einander bedingen (BMWi, 2013; HochTief Vicon GmbH, 2016). Das entsprechende Ineinandergreifen der genannten Komponenten ist in Abbildung 2.1 schematisch dargestellt.

Folgende wesentliche Eigenschaften werden mit dem Building Information Modeling in Verbindung gebracht und gelten als typische Charakteristika:

Geometrie und Semantik. Die Elemente eines Bauwerkmodells sind visualisierbare räumliche Objekte, was bedeutet, dass diese eine geometrische Repräsentation enthalten, durch die das Erscheinungsbild des jeweiligen Objekts beschrieben wird. Darüber hinaus enthält ein Objekt auch semantische

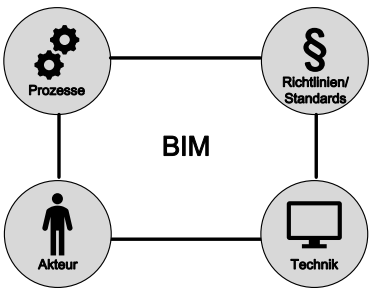


Abbildung 2.1: Wesentliche Aspekte, die bei der Implementierung von BIM berücksichtigt werden sollten

Informationen. In den Computerwissenschaften bezeichnet der Begriff Semantik die Bedeutung von Daten und Informationen (Ernst et al., 2015). Im Bauwesen können einem Bauteil beispielsweise Informationen zu dem Baustoff, Material, zu Herstellungsabläufen und -verfahren sowie zu Nutzungseigenschaften zugewiesen werden. Aus diesem Grund spielt neben der reinen Geometrie von Bauteilen und Räumen auch deren Semantik eine entscheidende Rolle.

Objektorientierung. Die Objektorientierung bezeichnet eine Sichtweise auf ein komplexes System, bei der dieses als Komposition kooperierender Objekte beschrieben wird. Wenn die Modellierung, Verwaltung und Modifikation dem Paradigma der Objektorientierung folgt, wird dieses als Objektorientierte Modellierung (OOM) bezeichnet (Borrmann et al., 2015b). Bei BIM kommt diese Abstraktion zur Anwendung und folglich wird ein Bauwerkmodell als eine Komposition von Komponenten, die wiederum Informationen beinhalten, dargestellt (Eastman et al., 2011). Eine einzelne Komponente ist eine digitale Repräsentation, welche semantische Informationen darüber enthält, was diese selbst darstellt und wie sie sich auf andere Objekte im Modell bezieht. Also können Objekte auch Informationen umfassen, die ihr Verhalten in einem bestimmten Kontext beschreiben. Ein Objekt enthält gekapselte Daten und Prozeduren, die zu einer Entität zusammengefasst sind. Die Struktur und das Verhalten von definierten gleichartigen Objekten kann als Klasse beschrieben werden. Eine solche Klasse stellt somit eine Art Vorlage für die Erzeugung einer Instanz eines Objektes dar. Bei der Instanziierung wird ein Objekt mit einer eindeutigen Identität dargestellt und durch eine eindeutige Identifikation (z. B. *Globally Unique Identifier*) vorgehalten. Wesentliche Voraussetzung dafür ist, dass die in einem BIM-Modell enthaltenen Informationen konsistent und nicht redundant sind. Nur so können Änderungen für die jeweilige Komponente automatisch für alle betroffenen Anwender aktualisiert werden.

Parametrisierung. Als Folge bzw. Ergebnis der Objektorientierung werden beim BIM Parameter verwendet.

Bauteile besitzen beschreibende Attribute, die ein ihrer Semantik entsprechendes Verhalten abbilden. Die Attribute können sich auf alle Informationen beziehen, welche einem Element zugeordnet werden können, wie z.B. Kosten, Materialität oder aber Prozess-bezogene Informationen. In der Praxis werden Attribute bisweilen auch als Parameter bezeichnet, allerdings bezieht sich dieser

Fachbegriff oft auf eine rein geometrische Betrachtung, bei welcher diese Parameter Eigenschaften der geometrischen Repräsentation beschreiben.

Eastman et al. (2011) bezeichnen die Parametrisierung als ein zentrales Konzept von BIM, bei welchem ein parametrisches Objekt als eine Repräsentation beschrieben wird, die aus geometrischen Definitionen und zugehörigen Daten und Regeln besteht. Auf diese Weise ist es möglich, für ein Objekt, welches beispielsweise als eine Wand dargestellt wird, auch das Verhalten zu modellieren, welches eine Wand zeigt, wenn es mit anderen Objekten interagiert (z.B. Verschneidungen mit anderen Bauteilen, Interaktion mit eingebetteten Bauteilen wie Fenster oder Türen).

Eastman et al. (2011) unterscheiden drei verschiedene Stufen der parametrischen Modellierung: In der ersten Stufe werden komplexe Formen oder Baugruppen lediglich durch einen Parameter definiert. Ändert sich dieser Parameter, dann muss das Objekt neu generiert werden, damit die Änderungen wirksam werden. Diese Ebene wird daher auch als parametrische Volumenmodellierung bezeichnet. In der nächsten Stufe wird das Modell automatisch aktualisiert oder neu generiert, wenn die Parameter geändert werden. Das Objekt wird als parametrische Baugruppe bezeichnet. Auf der dritten und letzten Stufe können Parameter eines geometrisch beschriebenen Objekts mit den Parametern eines anderen verknüpft werden, wenn diese beiden Teile miteinander in Beziehung stehen. Beide Teile werden aktualisiert, wenn ein Parameter geändert wird. Das gesamte Modell wird nun als voll-parametrisches (engl. *fully parametric*) Modell bezeichnet, die Methode als parametrische Objektmodellierung.

Konsistenz. Der Begriff Konsistenz beschreibt in diesem Zusammenhang die Widerspruchsfreiheit einer oder mehrerer Datenquellen. Dieses bezieht sich auf ein einzelnes Gebäudemodell an sich, aber auch auf mehrere Modelle eines Projektes untereinander. Um gewährleisten zu können, dass die einzelnen Modelle verschiedener Fachdisziplinen auch in ihrer Gesamtheit gültig sind, muss die Konsistenz regelmäßig überprüft und koordiniert werden. Hierzu wird ein Koordinationsmodell erstellt, welches letztlich durch eine Zusammenführung einzelner bzw. aller relevanten Fachmodelle erstellt wird. Das Koordinationsmodell enthält daher die Summe aller Informationen zu Dimensionen, Positionen und Eigenschaften aller Disziplinen, welche von den Fachmodellen abgebildet werden.

Auch wenn die Planungsleistungen auf Basis von digitalen Bauwerksmodellen durchgeführt werden, verlieren die technischen Zeichnungen nicht zuletzt wegen ihres Einsatzes auf der Baustelle nicht an Relevanz. Allerdings werden technische Zeichnungen, wie etwa Ansichten und Grundrisse, aus den digitalen Gebäudemodellen abgeleitet, damit diese immer die jeweils aktuellsten Informationen aus der zentralen Informationsressource beinhalten. Ansichten und Pläne werden aus den Gebäudemodellen erzeugt und befinden sich dadurch stets in einem konsistenten Zustand zueinander.

Zentrale Datenhaltung. Eine Minimalanforderung für die BIM-basierte Kooperation ist die Erreichbarkeit und Aktualität der Bauwerksinformationen. Um diese gewährleisten zu können, müssen die Informationen zentral vorgehalten werden. Zwar gibt es hierzu diverse technische Lösungen, jedoch bietet sich an dieser Stelle der Einsatz von zentralen Datenplattformen an. In Abschnitt 2.5.6 wird der Einsatz von Datenplattformen im Detail vorgestellt und diskutiert.

Im Folgenden sollen wesentliche ausgewählte Vorteile, welche sich durch den Einsatz der digitalen Methoden im Bauwesen ergeben, näher erläutert werden (Borrmann et al., 2015b; Eastman et al., 2011; Hausknecht und Liebich, 2016; Kulusjärvi, 2012; Mansfeld, 2017; Niedermeier und Bäck, 2015):

Planungsqualität. Ziel der gedanklichen Entwicklung des Bauvorhabens ist es, die gewünschte Leistungsfähigkeit des Produktes zu sichern und den Lösungsweg hin zum Bauwerk möglichst effizient und fehlerfrei zu gestalten. Um dieses zu erreichen, müssen verschiedene Varianten und Lösungen erstellt und gegeneinander abgewogen bzw. bewertet werden. Mit BIM werden die in den Bauwerksmodellen enthaltenen Informationen für die diversen Varianten-Studien und Simulationen einfach und direkt zugänglich gemacht. Im Vergleich zu herkömmlichen CAD-Systemen können durch die Verwendung von parametrischen Bauteilen in BIM-Tools Varianten mit vergleichsweise geringem Arbeitsaufwand erstellt und bewertet werden. Dieses gilt nicht nur für Varianten bzw. Alternativen in der Gestaltungsplanung, sondern auch in den Prozessabläufen, die simuliert werden können. Auf diese Weise können unterschiedliche Methoden auch hinsichtlich der technologischen Abhängigkeiten und benötigten Ressourcen bewertet werden. Dies ermöglicht wiederum für die am Planen, Bauen und Betreiben beteiligten Akteure eine verbesserte Präzisierung der Leistungs- und Terminvorhersage z.B. durch eine Bauablaufsimulation oder aber Aussagen über den erwartenden Energieverbrauch und entsprechende Unterhaltskosten machen. Diese Erkenntnisse und Simulationsergebnisse unterstützen bei einer Lösungs- und frühzeitigen Entscheidungsfindung, wodurch die Zahl der Änderungsvorgänge sinkt und gleichzeitig ein höherer Grad an Vorfertigung möglich wird. Auf diese Weise gibt BIM Planungssicherheit, verbessert die Planungsqualität und senkt die Kosten.

Weiterhin bietet BIM die Möglichkeit, die Planungsqualität selbst, also den planerischen Entwurf, hinsichtlich verschiedener Kriterien zu untersuchen. So können die in den Bauwerksmodellen enthaltenen Informationen dafür genutzt werden, den Planungsstand hinsichtlich spezifischer Kundenanforderungen wie beispielsweise Raumprogramme oder Prozessdistanzwege und allgemeine Anforderungen im Bauwesen wie beispielsweise Normen oder Richtlinien hin zu prüfen. Diese Aspekte werden in Kapitel 3 sowie Teil II der vorliegenden Arbeit im Detail erläutert.

Informationsqualität. Alle Anwendungen auf Basis von BIM basieren auf den im Modell enthaltenen Informationen. Daher kommt der Qualität dieser Informationen und der damit verbundenen Modellprüfung eine wesentliche Rolle zu. Ein BIM-Projekt unterliegt einer mitlaufenden Koordination, was bedeutet, dass die einzelnen Teilmodelle zu gegebenen Zeitpunkten und in wiederkehrenden Intervallen insbesondere hinsichtlich ihrer Konsistenz, aber auch hinsichtlich weiterer Anforderungen, geprüft werden. Durch eine begleitende Qualitätssicherstellung der einzelnen Modelle kann gewährleistet werden, dass alle Modelle eines Projektes und somit die Informationen des Gesamtprojektes möglichst fehlerarm sind (Gijezen und Hartmann, 2010). Hierbei ist wichtig, dass sich alle beteiligten Parteien zu dem Projektziel verpflichten, die Gebäudedatenmodelle durchgängig über das Projekt hinweg anzuwenden.

Die resultierende finale Informationsqualität ist von entscheidender Bedeutung für die Betriebsphase eines Gebäudes, da man hier maßgeblich von der Qualität der Informationen bei dem Übergang in die Betriebsphase abhängig ist. BIM unterstützt bei der Übergabe von Projektdaten in das Datenmanagement für den laufenden Betrieb.

Die Modellqualität wird in der vorliegenden Arbeit in Kapitel 3 im Detail vorgestellt.

Kooperation und Kollaboration. Zentraler Aspekt von BIM ist die Zusammenarbeit der beteiligten Akteure. BIM fördert die Idee eines kollaborativen Entwurfs und einer kollaborativen Planung. Projektbeteiligte können synchron oder aber asynchron an einem oder aber mehreren Modellen arbeiten. Die Kollaboration kann innerhalb eines Gewerks oder gewerkeübergreifend stattfinden. Moderne Technologien für die Zusammenarbeit, wie insbesondere BIM-Server, dienen zur zentralen Verwaltung von Gebäudemodellen und bieten einen erweiterten Funktionsumfang gegenüber Desktop-Systemen.

Kommunikation. Eine reibungslose Kommunikation und ein konsistenter Informations- und Wissenstransfer ist ein wichtiger Teil des Projektmanagements. Dieses ist insbesondere dann gegeben, wenn alle Beteiligten zu jeder Zeit und von jedem Ort aus auf aktuelle Informationen zugreifen können. Entscheidend ist dabei nicht nur die Kommunikation zwischen den Planern, sondern auch mit Bauherren und Entscheidungsträgern. Insbesondere im Planungs- und Bauprozess sind oft schnelle Entscheidungen vonnöten.

Die Anwendung von digitalen Methoden oder Modellen heißt nicht automatisch, dass die Kommunikation zwischen den Beteiligten eines Projektes verbessert wird, allerdings stehen neue Mittel und Methoden für diese Kommunikation zur Verfügung. Insbesondere bei komplexen, dreidimensionalen Konstellationen helfen die Visualisierungen, welche auf Basis von BIM möglich sind, sich einfacher zu verständigen. Durch Referenzierung von Kommunikationsdatenobjekten mit dem Modell, also z.B. den betroffenen Bauteilen, kann die Kommunikation von Änderungen oder identifizierten Fehlern zwischen den Akteuren unterstützt werden. Weiterhin liefert das BIM-Koordinationsmodell alle Informationen, die für Entscheidungen notwendig sind und die dabei helfen, Informationen auch präzise zu beschreiben. Auf Basis des BIM-Koordinationsmodells lassen sich Planungs- und Bauabläufe realitätsnah abbilden. Dadurch hat der Bauherr ein besseres Verständnis für die Entwurfsidee und kann die Auswirkungen von Änderungen leichter erkennen, z. B. wie die Kosten beeinflusst werden können. Transparente Kommunikation ist besonders wichtig bei großen öffentlichen Projekten. An dieser Stelle kann der Bauherr Teilhaber und Interessengruppen (*Stakeholder*) frühzeitig in den Prozess einbeziehen.

Automatisierungspotential. Bei den in den Modellen vorliegenden Informationen handelt es sich um strukturierte Informationen, was bedeutet, dass Ableitungen aus diesen erstellt und formalisiert werden können. Viele bisher manuell durchgeführte Arbeiten, wie die Erstellung von 2D-Plänen oder Mengenermittlungen, lassen sich so einheitlich und automatisiert aus Modellen ableiten und garantieren die permanente Verfügbarkeit eines aktuellen Planungsstands für alle Beteiligten. Nicht zuletzt an dieser Stelle lassen sich erhebliche Effizienzsteigerungen erzielen. Andere Industriezweige, wie beispielsweise die Automobilindustrie, setzen bereits auf eine durchgängige modellgestützte Produktentwicklung und -fertigung und konnten dadurch erhebliche Effizienzgewinne erzielen (Heindorf, 2010).

Zusammenfassend kann festgehalten werden, dass die Einführung von digitalen Methoden ein Schlüsselelement für den Umgang mit den zukünftigen Herausforderungen in der Baubranche sind. Dabei ist der grundlegende Vorteil von BIM, dass die digitale Planung als vorausgehender Schritt die Möglichkeit bietet, Unsicherheiten und Unwägbarkeiten vorab mit Hilfe von Simulationen auszuräumen und so die

Planungsqualität signifikant zu erhöhen. Fehlende Fachkenntnisse im Bereich BIM werden zukünftig zu einem signifikanten Wettbewerbsnachteil für Unternehmen darstellen.

Laut einer Studie des Fraunhofer-Instituts für Arbeitswirtschaft und Organisation nutzen aktuell nur 29% der Akteure der deutschen Baubranche BIM zur Erstellung von Bauteil-orientierten Gebäudemodellen, 10% planen es zumindest für die Zukunft. Der Einsatz von BIM mit Informationen zum Bauablauf als zusätzlicher Planungsdimension wird lediglich von 6% genutzt, bei 7% ist es geplant (Schober et al., 2016). Unternehmen, die bereits die ersten Schritte der BIM-Modellierung praktizieren, stellen hierbei jedoch immer wieder fest, dass sie dadurch die Geschäftsleistung verbessern (Kunz und Fischer, 2012).

2.4 Technologische Grundlagen

Ein wesentliches Fundament für die Anwendung der digitalen Methoden im Bauwesen sind die technologischen Grundlagen, da diese die Basis für die Implementierung der Arbeitsabläufe bilden. In den folgenden Abschnitten werden wesentliche Aspekte technologischer Grundlagen des Building Information Modeling vorgestellt.

2.4.1 Interoperabilität und Informationsaustausch

In der Informationstechnologie wird der Begriff *Interoperabilität* als Fähigkeit, die in einem Informationssystem generierten Informationen für andere Systeme verständlich zu machen, bezeichnet (Schonschek, 2017). Um einen möglichst hohen Grad an Interoperabilität zu erreichen, muss (1) ein verlustfreier Informationsaustausch erreicht und (2) eine korrekte Interpretation der Informationen ermöglicht werden. Beide Kriterien bedingen sich zudem gegenseitig.

Die Gefahr von Informationsverlusten gilt hierbei für Bauprojekte im Allgemeinen, da sich alle Bauvorhaben aufgrund der Struktur und Auftrennung in verschiedene Leistungsphasen durch eine hohe Anzahl an Schnittstellen, über welche die Informationen übergeben werden müssen, auszeichnen. Durch die Kooperation verschiedener Disziplinen an einem Projekt kommt es nicht nur zu unterschiedlichen Sichtweisen auf den Planungsgegenstand Bauwerk, sondern es werden auch unterschiedliche Softwareprodukte und somit unterschiedliche Beschreibungen von Daten eingesetzt. Dadurch ergibt sich schließlich eine informationstechnische Heterogenität. Borrmann et al. (2015b) führen an, dass es zu Brüchen bei dem Informationsaustausch kommt, wenn es aufgrund dieser Heterogenität zu fehleranfälligen Neueingaben kommt. Ziel sollte es sein, diese Neueingaben durch eine konsequente Weiternutzung digitaler Informationen zu vermeiden und so einen Zuwachs an Produktivität und Qualität zu erzielen.

Eine wesentliche Grundlage für den Informationsaustausch ist die Festlegung des Formates, um diese austauschen zu können. Ein Datenformat definiert präzise, wie Daten repräsentiert werden bzw. wie die enthaltenen Informationen zu sind. Entsprechende Formate wurden typischerweise von den Softwareherstellern gemeinsam mit den ersten BIM-fähigen Autorenwerkzeugen auf den Markt gebracht. Ein solches herstellereigenes Format folgt dabei in der Regel den Definitionen und Paradigmen, welche in dem jeweiligen Softwareprodukt herrschen und wird daher auch als proprietär oder nativ bezeichnet (Vries-Baayens, 1991).

Ein wesentliches Problem bei der Verwendung proprietärer Datenformate ist, dass es bei der Übersetzung von Daten von einem Datenformat in ein anderes immer wieder zu Informationsverlusten kommt, da jedes System seine eigene Methode zur mathematischen und strukturellen Beschreibung

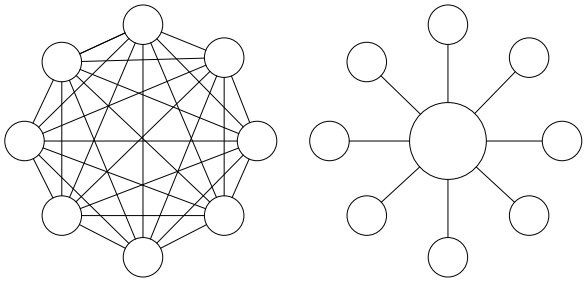


Abbildung 2.2: Schematische Darstellung verschiedener Transaktionssysteme für den Datenaustausch. links: Austausch von Daten mit Hilfe von Direktübersetzern; rechts: Austausch basierend auf einem neutralen Datenformat; angelehnt an Vries-Baayens (1991)

der Geometrie und Semantik definiert. Zudem wird für die Interpretation jedes Datenformats einer Fremdsoftware ein Direktübersetzer benötigt, welcher die Informationen auf das eigene Datenformat überträgt. Um also in alle Richtungen kompatibel zu sein, wird für jede einzelne Schnittstelle ein eigener Direktübersetzer benötigt. Dieses bringt wiederum einen erheblichen Wartungsaufwand mit sich und stellt somit einen erheblichen Nachteil dar (Jubierre, 2017; Vries-Baayens, 1991).

Ein wesentliches Problem bei dem Einsatz proprietärer Datenformate ist, dass mit zunehmender Anzahl von verwendeten Softwaresystemen die Anzahl der Schnittstellen und somit die der benötigten Direktübersetzer exponentiell wächst (siehe Abbildung 2.2). Zudem begibt man sich beim Einsatz von nativen Dateiformaten auch in eine Abhängigkeit eines einzelnen Softwareherstellers, welcher die Wartung und Entwicklung des jeweiligen Formates bestimmt.

Um diese Abhängigkeiten aufzulösen und gleichzeitig die Anzahl der Schnittstellen zu reduzieren, können gemeinsame oder neutrale Datenstandards zum Einsatz kommen. Basis hierfür ist die Definition einer gemeinsamen Beschreibung eines neutralen Datenformats, welches von allen beteiligten Systemen übersetzt und in der Folge genutzt werden kann. Voraussetzung hierfür ist, dass die Systeme eine Schnittstelle zu diesem Datenformat haben. Sollen in der jeweiligen Applikation nicht nur Daten gelesen, sondern auch erzeugt werden können, dann muss diese Schnittstelle schreibend und lesend, also bilateral, sein. Ein wesentlicher Vorteil neben der Reduktion der Schnittstellenanzahl (siehe Abbildung 2.2) ist, dass offene Datenstandards öffentlich dokumentiert und festgehalten werden. Auf diese Weise wird eine Abhängigkeit im Sinne von Fremdentwicklungen vermieden, und die Informationen lassen sich auch zu einem späteren Zeitpunkt wieder aufbereiten (Vries-Baayens, 1991).

Ein wesentlicher Nachteil bei der Verwendung von neutralen Datenstandards ist, dass in Abhängigkeit der Anwendungsfälle nicht alle Informationen abgedeckt werden, welche in einem Projekt ausgetauscht werden sollen. Dieser Faktor ist maßgeblich abhängig von dem Entwicklungsstadium und Reifegrad des verwendeten offenen Standards. Allerdings muss zu Beginn des Projektes sehr genau überlegt und geplant werden, welche Informationen ausgetauscht werden sollen und welches Format in diesem Falle am besten geeignet ist. Wird diese Planung nicht oder nur unzureichend durchgeführt, birgt der Informationsaustausch die Gefahr eines doppelten Informationsaustauschs, nämlich zum Zeitpunkt des Exportieren (*Prä-Prozessor*) und zum Zeitpunkt des Importieren (*Post-Prozessor*) von Informationen (Borrmann et al., 2015b; Jubierre, 2017).

2.4.2 Datenmodellierung im Bauwesen

Einen wesentlichen Anteil an dem Fortschritt der Digitalisierung im Bauwesen haben die Entwicklungen im Bereich der Datenmodellierung der letzten Jahre, in welchen die Methoden zur digitalen Repräsentation von Bauwerksinformationen signifikant vorangetrieben wurden. In den folgenden Abschnitten werden die datentechnischen Grundlagen für die Modellierung von Informationen im Bauwesen vorgestellt.

2.4.2.1 STEP, EXPRESS und EXPRESS-G

Mit Beginn der 1970er Jahre gab es diverse Bestrebungen in Richtung einer branchenübergreifenden, internationalen Vereinheitlichung und Standardisierung für die Repräsentation und Modellierung von Produktdatenmodellen. Unter der Führung des International Organization for Standardization (ISO) wurde an der Entwicklung eines allgemeinen Standards für Produktdatenmodelle gearbeitet und schließlich im Jahre 1984 der *Standard for the Exchange of Product Model Data (STEP)* als *ISO 10303* veröffentlicht (Borrmann et al., 2015a; Laakso und Kiviniemi, 2012).

Der STEP umfasst grundlegende Festlegungen für die Beschreibung von Datenmodellschemata inklusive graphischen Notationen, Speicherformate für die Instanziierung in unterschiedlichen syntaktischen Varianten, vereinheitlichte Definitionen von Programmierschnittstellen sowie diverse semantische Aspekte zu einzelnen Kategorien von Produkten. Mit diesem Ansatz wurde versucht, Informationsmodelle aus unterschiedlichen Disziplinen in einem Datenschema zusammenzufassen. Dieses stellte sich jedoch als problematisch heraus, da die damals existierenden und angewendeten Modelle auf unterschiedlichen Abstraktionsebenen lagen (Laakso und Kiviniemi, 2012).

Daher wurde in einer Weiterentwicklung des Standards eine grundlegende Unterscheidung zwischen Spezifikationen für konkrete (*Application Protocols*) und generische, allgemeingültige Implementierungen (*Integrated Resources*) eingeführt. Ziel der *Application Protocols* ist es, den Informationsbedarf innerhalb einer bestimmten Domäne oder Anwendung explizit zu definieren, indem eindeutig festgelegt wird, welche Informationen ausgetauscht werden müssen. Dieses schafft unter anderem auch eine wesentliche Grundlage für Konformitätsprüfungen, welche auf diesen Definitionen aufbauen. So beziehen sich die *Application Protocols* von STEP besonders auf die Produktmodellierung. Dies wird unter anderem in den Bereichen des Maschinenbaus und der Elektrotechnik genutzt (Laakso und Kiviniemi, 2012). Parallel führten die Entwicklungen im Bereich der Bauwerksinformationen zu dem auf STEP basierenden *Building Construction Core Model (BCCM)* als zentralem Datenmodell für den AEC- und FM-Bereich (Eastman, 1999).

Als die Arbeiten an dem STEP-Standard begannen, lagen die Ziele und Anforderungen, die erfüllt werden sollten, weit über dem, was damalige Technologien zur Datenmodellierung bieten konnten, sodass die Standardisierung lange Zeit im Vorgriff oder parallel zur Implementierung kommerziell verfügbarer Software durchgeführt werden musste. Zu diesem Zeitpunkt war bereits klar, dass eine robuste Datenmodellierung von zentraler Bedeutung für die Handhabung der Komplexität eines einheitlichen Produktmodellstandards ist. Eine Evaluierung damals vorhandener Datenmodellierungssprachen ergab, dass sich diese nicht vollständig für den beabsichtigen Verwendungszweck eigneten. Daher wurde eine eigene Datenmodellierungssprache unter dem Namen *EXPRESS* entwickelt (Kemmerer, 1999). Die

Sprache wurde parallel, aber auch als Basis von STEP entwickelt, um die entsprechenden Datenmodelle, und den Standard selbst definieren zu können.

EXPRESS ist eine deklarative Sprache, mit deren Hilfe objektorientierte Datenmodelle definiert werden können (Schenck und Wilson, 1994). Daher sind Beziehungen, Attribute, Randbedingungen und die Vererbung Kernkonzepte von EXPRESS.

In EXPRESS stellen Entitätentypen das Äquivalent zu einer Klasse, wie diese aus der OOM bekannt sind, dar. Für jeden Entitätentyp können Attribute und Beziehungen zu jeweils anderen Entitätentypen definiert werden. Zudem kommt das objektorientierte Konzept der Vererbung zur Anwendung, wodurch Attribute und Beziehungen an Subtypen weitergegeben werden. Eine Beziehung zwischen zwei Objekten wird dadurch ausgedrückt, dass der Entitätentyp ein Attribut des jeweils anderen Entitätentyps erhält. Eine Besonderheit von *EXPRESS* ist, dass Attribute angelegt werden können, die eine inverse Beziehung explizit deklarieren. Dabei wird keine neue Information modelliert, sondern die Navigation entlang der Beziehung in der entgegengesetzten Richtung ermöglicht. *EXPRESS* bietet darüber hinaus die Möglichkeit, in einem optionalen $WHERE$-Block algorithmische Bedingungen festzulegen, um damit Regeln für die Konsistenz oder Integrität der enthaltenen Daten zu beschreiben. So beinhaltet das $WHERE$-Segment einen oder aber mehrere Booleschen Ausdrücke, die als *Wahr* ausgewertet werden müssen, damit die jeweilige Instanz als gültig eingestuft werden kann (Borrmann et al., 2015a; Kemmerer, 1999; Laakso und Kiviniemi, 2012; Schenck und Wilson, 1994).

Die mit *EXPRESS*-definierten Objekte sind sowohl von Maschinen als auch von Menschen lesbar und können zusätzlich graphisch über den Notation-Standard *EXPRESS-G* oder aber als Instanz über *EXPRESS-I* dargestellt werden (Kemmerer, 1999; Schenck und Wilson, 1994). Zur Veranschaulichung ist in Quellcode 2.1 die EXPRESS-Definition der Entität *IfcProduct*.

```
(*Definition der Entitaet*)
ENTITY IfcProduct
  (*Definition der Subklassen*)
  ABSTRACT SUPERTYPE OF(ONEOF(IfcAnnotation, IfcElement, IfcGrid, [...]))
  (*Definition der Superklasse*)
  SUBTYPE OF IfcObject;
    (*Definition der Attribute*)
    ObjectPlacement : OPTIONAL IfcObjectPlacement;
    Representation  : OPTIONAL IfcProductRepresentation;
    (*Definition der inversen Attribute*)
  INVERSE
  ReferencedBy  : SET OF IfcRelAssignsToProduct FOR RelatingProduct;
  (*Definition der Integritaetsbedingungen*)
  WHERE
    WR1 :   (EXISTS(Representation)
    AND EXISTS(ObjectPlacement))
    OR (EXISTS(Representation)
  [...]
END_ENTITY;
```

Quellcode 2.1: Definition der Entität *IfcProduct* in EXPRESS (BuildingSmart, 2017a)

Der STEP-Standard ist maßgeblich von der Produktmodellierungs- und Fertigungsindustrie beeinflusst, kann aber flächendeckend über mehrere Branchen wie z. B. Automotive, Luft- und Raumfahrt oder Schiffbau eingesetzt werden. Die AEC-Branche ist somit nur eine von mehreren Interessengruppen, die für eine Ausprägung innerhalb von STEP in Frage kommt. Um den individuellen und komplexen

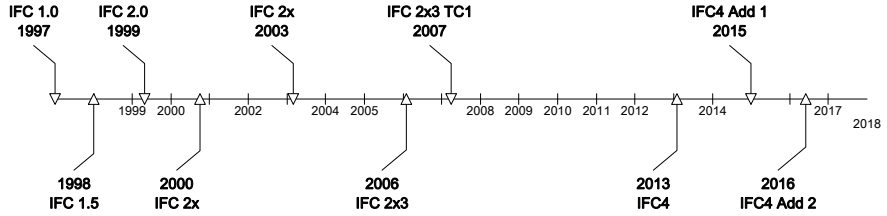


Abbildung 2.3: Chronologie der IFC Versionen (BuildingSmart, 2017a)

Anforderungen bei der Modellierung von Bauwerksinformationen gerecht zu werden, wurde dieses außerhalb der schematischen Definition des STEP-Standard in einem eigenständigen Projekt umgesetzt. Ein erstes Ergebnis dieser Entwicklungen gelang 1996 schließlich mit der Veröffentlichung des allgemeinen Produktmodells für die Bauindustrie, den Industry Foundation Classes (IFC) 1.0. Noch heute gibt es daher große Überschneidungen zwischen STEP und IFC: so bedient sich IFC zur Vermeidung von Redundanzen teilweise der Geometriedefinitionen von STEP (Laakso und Kiviniemi, 2012).

2.4.2.2 Industry Foundation Classes (IFC)

Die Industry Foundation Classes stellen eine umfassende und weitverbreitete Spezifikation für den Informationsaustausch über den gesamten Lebenszyklus eines Bauwerks dar. Dieser Austausch findet zwischen den Projektbeteiligten und über unterschiedliche Softwareanwendungen hinweg während der Gestaltungsplanung, Errichtung, Beschaffung, Wartung und des Betriebs des Bauwerks statt. Die IFC sind ein ISO-Standard (ISO 16739:2013) und wurden im Jahr 1996 als erster Standard für den Austausch von Gebäudeinformationsmodellen veröffentlicht (ISO, 2013-04; Steel et al., 2012).

Der herstellerneutrale Datenstandard wurde in der Version *IFC 1.0* ursprünglich von der Organisation International Alliance for Interoperability (IAI), einem Zusammenschluss aus mehreren namhaften Unternehmen des Bauwesens, herausgegeben (Eastman et al., 2011). Zur besseren Kommunikation der Ziele wurde die Organisation mittlerweile in *buildingSMART* umbenannt und arbeitet seitdem aktiv an der Entwicklung neuer Versionen, um so den Grad der Interoperabilität kontinuierlich zu erhöhen. Der Standard wird bis heute stetig erweitert und optimiert, indem mit jeder neuen Version von Experten und Anwendern diagnostizierte Mängel behoben werden. In Abbildung 2.3 ist eine Übersicht der bisher erschienenen Versionen in einem Zeitstrahl dargestellt.

Die Verwendung offener Standards wie IFC bringt wesentliche Vorteile mit sich, die unter anderem bereits in Abschnitt 2.4.1 näher erläutert wurden. Mittlerweile wird der IFC-Standard in diversen Ländern in der Praxis als wesentlicher Vorteil für Gebäudeeigentümer, öffentliche Verwaltungen und Baubehörden sowie Ministerien offiziell anerkannt und so ist es teilweise bereits verpflichtend, die Modellinhalte in diesem Format zu liefern. Vorreiter bei diesen Entwicklungen sind insbesondere Finnland, Norwegen und Singapur (Fiedler, 2015). Wesentliche Gründe für eine Anwendung von IFC sind (Borrmann et al., 2015b; Eastman et al., 2011; Laakso und Kiviniemi, 2012):

- Herstellerneutralität und Unabhängigkeit bei der Nutzung von Softwareanwendungen
- Nutzung eines einzelnen Datenformates über den gesamten Lebenszyklus eines Gebäudes hinweg

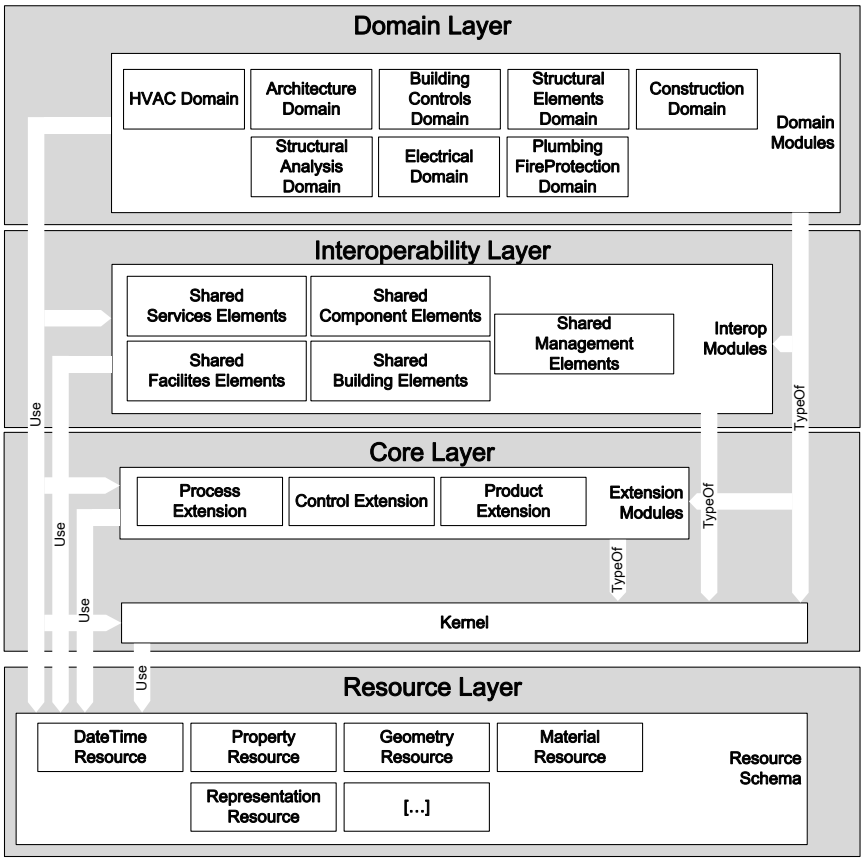


Abbildung 2.4: Schichten-Architektur des IFC-Schema; angelehnt an Borrmann et al. (2015a); BuildingSmart (2017a); Gonçal (2017); Laakso und Kiviniemi (2012)

- Menschenlesbarkeit und Rechtsverbindlichkeit für die Dokumentation und Wiederverwendung der Bauwerksinformationen auch in später Zukunft
- Offene Standards als Grundlage für einen fairen Wettbewerb auf dem Softwaremarkt, welcher langfristig zu besseren Produkten zu vernünftigen Preisen führt (Wettbewerbsgleichheit)

Dabei ist jedoch zu beachten, dass die IFC sich explizit auf den Datenaustausch beziehen und sich nur sehr eingeschränkt für eine etwaige interne Datenhaltung, z. B. in einer Datenplattform, eignet.

Aus technischer Sicht basiert das IFC-Schema auf der Definition von diversen Konzepten, wie Klassen, Attributen, Beziehungen, Eigenschaftssätze und Mengendefinitionen, mit welchen Bauwerksinformationen ganzheitlich beschrieben werden können. Die grundlegende Architektur des Schemas ist in vier Schichten unterteilt (siehe Abbildung 2.4) (BuildingSmart, 2017a):

Ressourcenschicht (*Resource Layer*) enthält die grundlegenden Konzepte, die als Entitäten, wie Geometrie oder Topologie, ausgedrückt werden.

Kernschicht (*Core Layer*) enthält abstrakte Begriffe wie Objekt, Gruppe, Prozess, Eigenschaft-Definition, Beziehungen oder das *Root*-Element. Die in dieser Schicht beschriebenen Konzepte ermöglichen

es, die gemeinsamen Regeln für die Modellierung in IFC zu definieren, die von allen Teilmodellen gemeinsam genutzt werden.

Interoperabilitätsschicht (*Interoperability Layer*) definiert grundlegende Konzepte für die Interoperabilität zwischen den verschiedenen Domänen. In dieser Schicht werden Standardbauteile wie Wand (*IfcWall*), Balken (*IfcBeam*), Decke (*IfcSlab*) usw. definiert.

Domänenschicht (*Domain Layer*) erweitert die Interoperabilitätsschicht um weitere Entitäten und Eigenschaften, um Definitionen für Prozesse in weiteren Bereichen der Baubranche bereitzustellen (Tragwerksplanung, Elektronik, Bauprozessmanagement, usw.).

Mit Hilfe der vier vorgestellten Schichten können Bauwerksinformationen ganzheitlich über den Lebenszyklus der abzubildenden baulichen Anlage hinweg dargestellt werden. Allerdings gibt dieses Schema lediglich einen technischen Rahmen für die Speicherung der Informationen vor. Damit die Informationen schließlich auch einheitlich von einer Softwareanwendung zu anderen gegebenen werden können, müssen die Modelle zwingend korrekt hinsichtlich des vorgegebenen Schemas definiert werden. Ein wesentliches Problem, welches immer wieder im Zusammenhang mit den IFC genannt wird, ist, dass sich von Seiten der Softwarehersteller Interpretationen und Anpassungen finden lassen und darüber hinaus nicht alle Anwender mit den Implikationen des Datenaustauschs über das IFC-Format vollständig vertraut sind. Weiterhin wird kritisiert, dass das Schema zu viele Interpretationsspielräume lässt und daher explizite Vereinbarungen und Absprachen zu der Implementierung unter den Softwarehäusern, sogenannte *Implementers' Agreements*, notwendig sind (Borrmann et al., 2018; Gonçal, 2017).

Die IFC basieren auf dem STEP-Standard, welcher bereits in Abschnitt 2.4.2.1 behandelt wurde. Daher wird das Schema der IFC originär in der Datenmodellierungssprache EXPRESS definiert. Darüber hinaus steht die Schemadefinition von IFC mittlerweile auch als Auszeichnungssprache (XSD) und als ontologische Datenstruktur (OWL) zur Verfügung. Als IFC-Instanz definierte Informationen werden über das STEP Physical File (SPF)-Format oder als XML respektive RDF ausgetauscht. Das neutrale SPF Dateiformat speichert die Informationen über die enthaltenen Objekte in der für Menschen lesbaren ASCII-Struktur.

Die hohe und freie Verfügbarkeit des IFC-Standards hat mittlerweile zu einer weiten Verbreitung geführt, und so bieten diverse Softwarehersteller entsprechende Schnittstellen in ihren Applikationen an. Die Einsatzbereiche von IFC erstrecken sich sowohl über den privaten als auch über den öffentlichen Sektor für Bauvorhaben (Borrmann et al., 2015b; Niedermeier und Bäck, 2015).

In den folgenden Abschnitten wird kurz auf wesentliche Eigenschaften und Besonderheiten der IFC bei der Repräsentation von Informationen eingegangen:

Vererbung bei Entitäten und Objekten. Jede Entität, die in der Kern-, Interoperabilitäts- oder Domänen-Schicht des IFC-Modells definiert ist, erbt über einige Zwischenschritte von der *IfcRoot*-Entität. Dadurch erhält jedes Objekt im IFC-Schema durch Zuweisung eines global eindeutigen Identifikators, die auch als Globally Unique Identifier (GUID) bezeichnet wird, einheitliche Eigentums- und Änderungsinformationen, anhand welcher die Version nachvollzogen werden kann. Im IFC-Modell gibt es drei grundlegende Entitätentypen, die alle jeweils von der Entität *IfcRoot* abgeleitet sind. Diese Entitäten bilden eine wesentliche Basis innerhalb der IFC-Klassenhierarchie (BuildingSmart, 2017a):

- **Objekte** (*IfcObject*): Konzepte für die Beschreibung von behandelten Elementen des IFC-Modells.

Abbildung 2.5: Das Prinzip der objektifizierten Beziehungen illustriert anhand des Beispiels Wand-Öffnung-Fenster; nach Borrmann et al. (2015a)

- **Relationen** (*IfcRelationship*): Konzepte für die Beschreibung von Beziehungen zwischen den Elementen. Diese werden im IFC-Schema als eigene Objekte behandelt. Daher wird in diesem Zusammenhang auch von objektifizierten Relationen gesprochen.
- **Eigenschaften** (*IfcPropertyDefinition*): Konzepte für die Beschreibung von Eigenschaften und Merkmalen, welche den Elementen zugewiesen werden.

Ein Objekt bzw. Element wird durch den abstrakten Supertyp *IfcObject* dargestellt und repräsentiert als Konzept alle physischen Objekte, wie z. B. eine Wand (*IfcWall*), einen Balken (*IfcBeam*) oder ein Fenster (*IfcWindow*), aber auch physisch nicht greifbare Objekte wie Räume (*IfcSpace*) oder räumliche Strukturen (*IfcBuildingStorey*). Darüber hinaus können auch nicht-physische Objekte wie Prozesse (*IfcProcess*), Kosten (*IfcCostValue*), Arbeitsressourcen (*IfcResource*) oder Projektbeteiligte (*IfcPerson*) als Objekte abgebildet werden.

Objektifizierte Relationen. Für die Beschreibung von Beziehungen zwischen Entitäten kommen im IFC Schema sogenannte objektifizierte Relationen zum Einsatz. Hintergrund dieses Konzeptes ist, dass Relationen nicht direkt zwischen Entitäten gebildet werden, sondern dass jeweils ein Relationsobjekt verwendet wird, welches selbst mit beiden in Relation stehenden Objekten in Beziehung tritt. Wesentlicher Vorteil dieses Konzeptes ist, dass es ermöglicht, beziehungsspezifische Eigenschaften direkt am Beziehungsobjekt zu halten und die semantische Verknüpfung von den Objektattributen zu entkoppeln.

Attribute. Die IFC bieten über das *IFC Property Set* (kurz: *Pset*) eine Möglichkeit zur dynamischen Erweiterung für die Zuweisung von Attributen. Anwender können diese Funktion also nutzen, um alphanumerische Informationen zu definieren und diese Entitäten zuzuweisen. Grundsätzlich gliedern sich die *IFC Property Sets* in zwei Gruppen: zum einen die durch buildingSMART standardisierten *Property Sets* (z.B. Pset_WallCommon), zum anderen nicht standardisierte *Property Sets*, die von den Benutzern erstellt und hinzugefügt werden können. Standardisierte Eigenschaftsgruppen können in der Regel an dem Präfix *Pset_* erkannt werden. Für die Definition von Attributen kommt im IFC-Schema das Konzept *IfcPropertyDefinition* zum Einsatz, mit welchem die Eigenschaften von Objekten verallgemeinert werden können. Dieses Objekt kann die spezifischen Informationen über eine Relation mit einem Objekttyp oder einer einzigen Objektinstanz zugeordnet werden. Der Aufbau eines *Property Sets* ist in Abbildung 2.6 dargestellt.

Ein weiteres Konzept, welches in den IFC Anwendung findet, sind inverse Attribute, wie diese bei der Datenmodellierungssprache EXPRESS eingeführt wurde (siehe Abschnitt 2.4.2.1). Durch die Verwendung

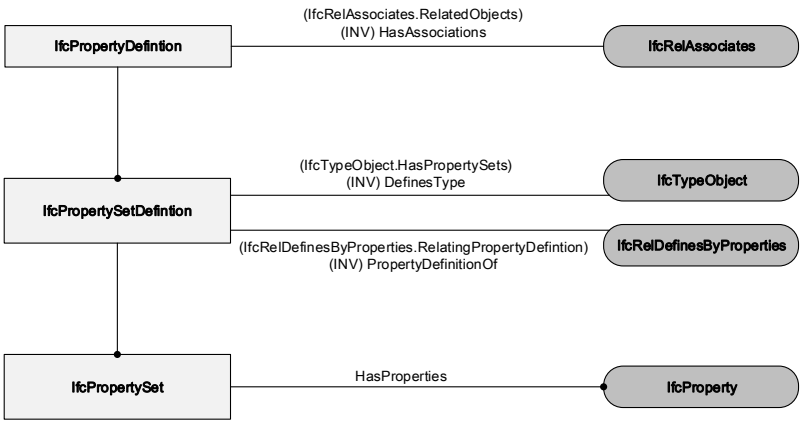


Abbildung 2.6: Schematischer Aufbau eines *IfcPropertySet* (BuildingSmart, 2017a)

von objektifizierten Relationen können Entitäten bilateral miteinander in Beziehung gebracht werden, was bedeutet, dass die Beziehung nicht nur einseitig, sondern von jeweils beiden Seiten definiert wird. Dadurch kann von beiden Seiten aus abgefragt werden, mit welchen Entitäten jeweils eine Beziehung besteht. Eine solche beidseitige Beziehung wird im IFC-Datenmodell über das EXPRESS-basierte Konzept inverser Attribute definiert. Als Ergebnis wird die Navigation durch das Datenmodell deutlich erleichtert. Die Verwendung von inversen Attributen ist bereits bei der Einführung der objektifizierten Relationen in Abbildung 2.5 dargestellt.

Geometrische Repräsentation. Für die Darstellung von geometrischen Informationen dient in dem IFC-Schema das Konzept *IfcShapeRepresentation*, welches eine bestimmte geometrische Repräsentation eines Produktes oder einer Produktkomponente innerhalb eines definierten geometrischen Repräsentationskontextes darstellt. Im IFC-Schema sind die geometrische und semantischen Ebene strikt getrennt. Einem *IfcProduct* werden über eine Relation eine oder aber mehrere *IfcShapeRepresentation* zugewiesen, wodurch die beiden Ebenen voneinander entkoppelt sind. Für das *IfcShapeRepresentation* definiert das Attribut *RepresentationType* die Art der jeweiligen geometrischen Repräsentation, das für die jeweilige

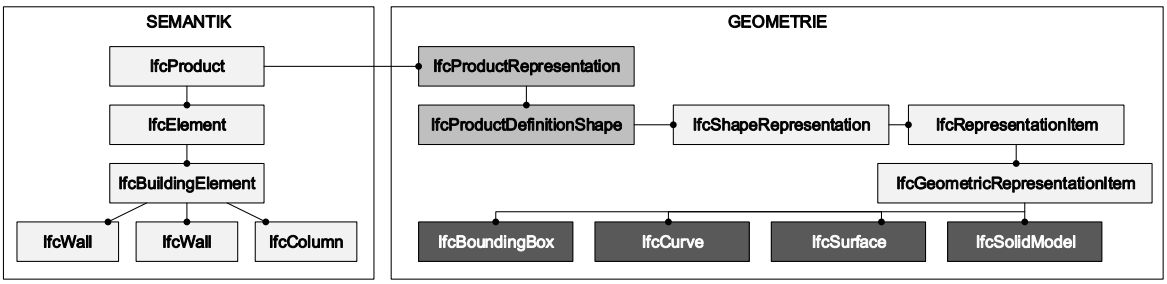


Abbildung 2.7: Verknüpfung der geometrischen Repräsentation mit *IfcProduct*; nach (Borrmann et al., 2015a)

Tabelle 2.1: Geometrische Repräsentationen IFC (BuildingSmart, 2017a)

Gruppe	IfcShapeRepresentation	Beschreibung
Kurven	Curve2D	2-dimensionale Kurven
Geometrische Mengen	GeometricCurveSet	Punkte, Kurven (2- oder 3-dimensional)
	Annotation2D	Punkte, Kurven (2- oder 3-dimensional), Schraffuren und Text (2-dimensional)
	SurfaceModel	flächen- und schalen-basiertes Flächenmodell
Geometrische Körper	SweptSolid	gekehrte Flächenkörper, durch Extrusion und Umdrehung
	BREP	facettierte BREP Darstellungen mit und ohne Hohlräume
	CSG	Boolesche Ergebnisse von Operationen zwischen Volumenmodellen, Halbräumen und booleschen Ergebnissen
	Clipping	Boolesche Differenzen zwischen gekehrten Flächen, Halbräumen und booleschen Ergebnissen
	AdvancedSweptSolid	Flächenkörper, die durch das Kehren eines Profils entlang einer Leitkurve erzeugt werden
Zusätzliche Typen	BoundingBox	vereinfachte 3D-Darstellung durch eine BoundingBox
	SectionedSpine	Querschnitt-basierte Darstellung einer Kurve und ebener Querschnitte. Es kann eine Fläche oder ein Volumen darstellen, die Interpolationen der Querschnitte sind nicht definiert.
	MappedRepresentation	Darstellung auf der Grundlage von gemappten Elementen, die sich auf eine Darstellungslandkarte beziehen. Hinweis: Es kann als eingefügte Blockreferenz angesehen werden. Die Formdarstellung des gemappten Elements hat einen Repräsentationstyp, der den Typ seiner Repräsentationselemente deklariert.

Darstellung verwendet wird. Das Attribut *RepresentationIdentifier* weist die eigentliche geometrische Darstellung zu, welche von der *IfcShapeRepresentation* beschrieben wird (z. B. eine Linie, Fläche, ein Körper, etc.). Die verschiedenen Darstellungsarten der *IfcShapeRepresentation*, welche im IFC-Schema genutzt werden können sind in Tabelle 2.1 dargestellt.

2.4.2.3 ifcOWL - Semantic Web

In den Computerwissenschaften beschreibt der Begriff Ontologie eine Darstellungsweise eines spezifischen Wissensgebiets, einer sogenannten Domäne, welche die Semantik (Bedeutung) des Gebietes erfassen kann. Dieses bedeutet, dass die Ontologie in der Lage ist, zu verstehen, welche Informationen in ihr selbst gespeichert sind und wie diese entsprechend verarbeitet werden müssen. Dieses Selbstverständnis der Ontologie setzt sich aus einer Taxonomie, einer Logik und Relationen, also die Darstellung der internen Zusammenhänge, zusammen (Furrer, 2012).

Gruber (1993) definiert die Ontologie als eine "formale explizite Beschreibung einer gemeinsam verwalteten Konzeption". Eine Ontologie unterliegt einer formellen Struktur, beinhaltet somit eine Logik und ist dadurch für Mensch und Maschine interpretierbar. Gleichzeitig beschreibt sie die Wissensgebiete explizit, d. h. Informationen werden auf Anfrage eindeutig zurückgegeben und es kommt zu keinen Inkonsistenzen im Datenmodell (Genesereth und Nilsson, 2012). Nicht zuletzt soll eine Ontologie die Informationen einer Domäne so beschreiben, dass sie in sämtliche Richtungen austauschbar sind und mit anderen ontologischen Wissensbereichen verknüpft werden können. Dieser Grundgedanke führte in jüngster Zeit zu der Idee des sogenannten *Semantic Web*, einer einzigen vernetzten ontologischen Datenbasis, welche alle Domänen verwaltet und die entsprechenden Daten bei Bedarf abrufen kann. Daher lassen sich viele weitere Definition einer Ontologie finden, die den Fokus vermehrt von der Taxonomie, der Klassenhierarchie, auf die netzwerktechnische Verknüpfung der einzelnen Ontologien untereinander legen, wie beispielsweise bei Guarino und Giaretta (1995).

```
<owl:Class rdf:ID="Region"/>          // Klasse

<owl:Class rdf:ID="Stadt">            // Klasse
<rdfs:subClassOf rdf:resource="#Region"/>
</owl:Class>

<Stadt rdf:ID="Muenchen">           // Instanz
```

Abbildung 2.8: Übersetzung einer Klassenhierarchie in OWL nach (Kost, 2013)

Zur Beschreibung einer Ontologie wurde vom W3C die Auszeichnungssprache Ontology Web Language (OWL) eingeführt (W3C, 2017). Diese Sprache beschreibt den Informationsgehalt einer Ontologie mit Hilfe der drei grundlegenden Sprachebenen *OWL Lite*, *OWL Description Logic (DL)* und *OWL Full* (Lee et al., 2008). Für die Erstellung einer Ontologie muss zunächst eine Basisstruktur für die Klassenhierarchie gebildet werden. Dieses kann mittels der ersten Sprachebene OWL Lite aufgestellt werden, indem die grundlegenden Klassen gemäß der Auszeichnung-Syntax von OWL und mit Hilfe des graphisch-logischen Formulierungssystems Resource Description Framework (RDF) deklariert werden (Pan, 2009; Pauwels et al., 2011). In Abbildung 2.8 ist ein Beispiel für eine Deklaration von Klassen und Instanzen dargestellt.

Auf Basis dieser Klassenhierarchie kann die logische Ebene der Ontologie gebildet werden. Die *OWL DL* beinhaltet eine definierte Menge an logischen Operatoren, die für einzelne, aber auch mehrere Klassen eingesetzt werden können. Eine Auflistung dieser Operatoren ist in Tabelle 2.2 aufgeführt.

Mit Hilfe von relationalen Objekten können die einzelnen ontologischen Klassen zueinander ins Verhältnis gesetzt werden. Ein Beispiel für eine solche Relation und die zugehörige Übersetzung innerhalb von OWL ist in Abbildung 2.9 dargestellt.

Die beiden Sprachebenen setzen gemeinsam ein ontologisches System zusammen und beschreiben dieses umfassend. Ein einfaches Beispiel für ein solches Gesamtsystem ist in Abbildung 2.10 dargestellt.

Beetz et al. (2009) identifizieren wesentliche Interoperabilitätsprobleme bei der existierenden Datenmodellierung, welche sie mit Hilfe des Konzepts der Ontologie überwinden wollen. Die aktuellen Probleme führen sie unter anderem auf die folgenden Ursachen zurück:

Tabelle 2.2: Auflistung logischer Operatoren der OWL Description Logic nach (Lee et al., 2008)

Logische Semantik	Beschreibung	Beispiel
$\neg C$	allgemeine Verneinung (Negation)	kein Bauteil
$\leq R.C, \geq R.C$	qualifizierte Beschränkung (numerisch)	größer für ein bestimmtes Verhältnis einer Klasse
$R =< y, x > \mid < x, y > \in R$	Inverse Funktionen	beinhaltet das Gegenteil von *isInsideOf*
$(< y, x > \in R) \forall (< y, z > \in R) \Rightarrow (< x, z > \in R)$	transitive Funktionen	ein Produkt erbt von Objekt und ein Element erbt von Produkt. Somit erbt das Element vom Objekt.

```
<owl:ObjectProperty rdf:ID="isLargerThan"> // Relation
  <rdfs:type rdf:resource="&Person ; TransitiveProperty"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Room1">
  <isLargerThan ref: resource="#Room2"/>
</owl: Class >

<owl:Class rdf:ID="Room2">
  <isLargerThan ref: resource="#Room3"/>
</owl: Class >
```

Abbildung 2.9: Beispiel für eine logische Relation im ontologischen Modell und die zugehörige OWL-Übersetzung (Lee et al., 2008)

```
<owl:ObjectProperty rdf:ID="wohntIn">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Gebiet"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="bewohntStadt">
  <rdfs:subPropertyOf rdf:resource="#wohntIn"/>
  <rdfs:range rdf:resource="#Stadt"/>
</owl:ObjectProperty>

<owl:DataTypeProperty rdf:ID="Alter">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="&dt;Alter"/>
</owl:DataTypeProperty>

```

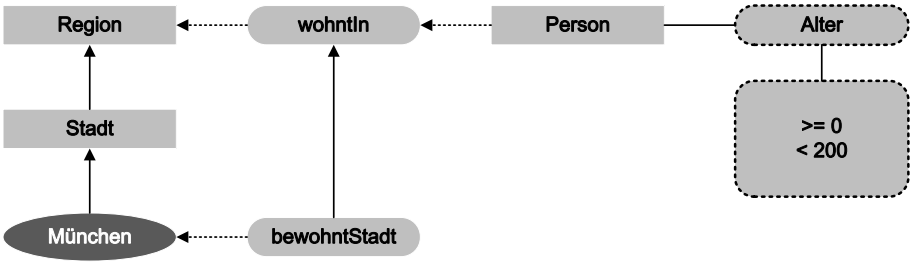


Abbildung 2.10: Beispiel für ein vollständiges ontologisches Modell und die zugehörige OWL-Übersetzung (Kost, 2013)

Mangelnde formale Starre (engl. *lack of formal rigidness*): Die Beschreibung eines Datenmodells mittels IFC basiert nicht auf einer ausreichend strengen bzw. starren mathematischen Grundlage. Mittels Algorithmen, Axiomen und Theoremen könnte ein Modell intelligent gestaltet werden, sodass es auf äußere Veränderungen reagieren kann und dabei die innere Konsistenz bewahrt.

Eingeschränkte Wiederverwendung und Interoperabilität (engl. *limited reuse and interoperability*): Das STEP-Format und die EXPRESS-Modellierungssprache, auf welchem der IFC-Standard beruht,

sind im Bereich der Computerwissenschaften nicht weit verbreitet. So gestaltet sich ein Austausch von Informationen mit anderen Wissensgebieten langfristig als sehr schwierig. Ziel sollte es sein, dass man sich anderen Domänen gegenüber öffnet, um Wissen miteinander zu verknüpfen.

Fehlende integrierte Aufteilung (engl. *lack of built-in distribution*): Nach der Vorstellung einer vollständig vernetzten, digitalen Welt müssen alle Elemente einer Einheit modular austausch- und verteilbar sein. Für IFC sind zwar grundlegende Regeln für den Umgang mit dem Datenmodell definiert, jedoch lassen sich die Datenobjekte der IFC sehr frei nach den jeweils eigenen Bedürfnissen verwenden (beispielsweise das *IfcBuildingElementProxy*). Die fehlende, aber zwingend erforderliche Struktur in der Syntax des IFC-Standards verhindert, dass Daten einheitlich gespeichert werden. Daher bildet dieses Format keine passende Grundlage für einen modularen Austausch von Informationen.

Eine Lösung des Problems sehen Beetz et al. (2009) an dieser Stelle in der Einführung einer ontologischen Struktur der IFC, der ifcOWL. Diese stellt eine Kombination des IFC-Schema und der bereits vorgestellten OWL dar. Dabei handelt es sich nicht um eine Mischform, sondern vielmehr um eine Abbildung des IFC-Formates auf die OWL, um die Modelldaten auf ein ontologisches Niveau zu heben.

Mit Hilfe einer Taxonomie kann der Informationsgehalt eines IFC-Modells über fest implementierte Abbildungsroutinen auf ein ontologisches Schema abgebildet werden. Wie diese einzelnen Klassen ontologisch definiert und voneinander abhängig sind, ist in der sogenannten *Terminology Box* auf Basis von RDF hinterlegt. Eine beispielhafte Übersetzung der Entität *IfcElement* von der EXPRESS-Sprache in ifcOWL ist in Code 2.2 und Code 2.3 dargestellt.

```
1  ENTITY  IfcElement
2  ABSTRACT SUPERTYPE OF (ONE OF(IfcBuildingElement,
3    IfcFurnishingElement, [...]))
4  END_ENTITY;
5
6  ENTITY  IfcBuildingElement
7    ABSTRACT SUPERTYPE OF (ONE OF(IfcDoor, IfcWall, IfcSlab, [...]))
8    SUBTYPE OF (IfcElement)
9  END_ENTITY;
10
11 ENTITY  IfcDoor
12   SUBTYPE OF (IfcBuildingElement)
13 END_ENTITY;
```

Quellcode 2.2: Definition der Entität *IfcElement* in EXPRESS (Beetz et al., 2009)

```
:IfcElement
a owl:Class ;
rdfs:subClassOf owl:Thing

:IfcBuildingElement
a owl:Class ;
rdfs:subClassOf : IfcElement ;
owl:disjointWith  : IfcFurnishingElement

:IfcDoor
a owl:Class ;
rdfs:subClassOf : IfcBuildingElement
owl:disjointWith  : IfcWall, : IfcWindow
```

Quellcode 2.3: Definition der Entität *IfcElement* in IfcOWL (Beetz et al., 2009)

Mit Hilfe der Übersetzung des IFC-Schema von EXPRESS in ifcOWL können nun auch die Instanzen in Form des RDF übertragen werden. Hierfür ist die sogenannte *Assertion Box* zuständig, welche die einzelnen Instanzen des Gebäudemodells den ontologischen Klassen der *Terminology Box* zuordnet. Die Struktur des Abbildungsprozesses ist in Abbildung 2.11 veranschaulicht. Für die Übersetzung eines IFC-Instanzmodells in RDF gibt es einen frei verfügbaren *IFC-to-RDF-Converter* von Pauwels und Oraskari (2016).

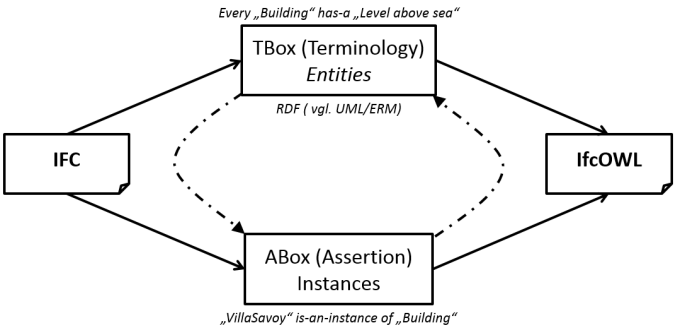


Abbildung 2.11: Abbildung von IFC auf ifcOWL (Beetz et al., 2009)

Neben der Entwicklung von ifcOWL existieren noch einige weitere Ansätze, die auf ähnliche Art und Weise versuchen, das Prinzip der Ontologie auf ein Gebäudemodell und somit auf das Bauwesen zu übertragen; u. a. in (Tan et al., 2010; Zhong et al., 2012). Eine ausführliche Beschreibung und Analyse weiterer OWL-basierter Ansätze ist bei (Pauwels et al., 2011) zu finden.

2.4.2.4 ifcXML

Eine weitere Ableitung aus der IFC-EXPRESS-Definition ist die XML-Schemadefinition ifcXML (BuildingSmart, 2017b). ifcXML nutzt die Konfigurationsmöglichkeiten des Standards ISO 10303-28 für die Übersetzung der EXPRESS-Definition in die ifcXML XML-Schemadefinition XSD. Als Beispiel ist in Quellcode 2.4 die Definition der Entität *IfcProduct* im XSD-Schema dargestellt.

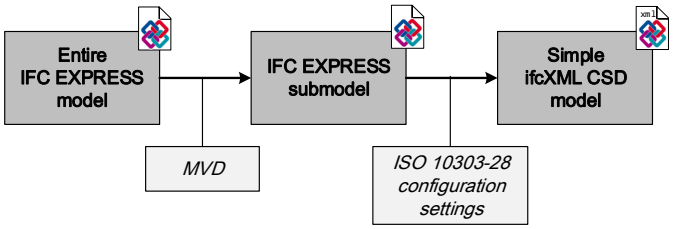


Abbildung 2.12: Erzeugen der ifcXML XSD-Schemadefinition auf Basis der IFC EXPRESS-Definition (BuildingSmart, 2017e)

```
<xs:complexType name="IfcProduct" abstract="true">
  <xs:complexContent>
    <xs:extension base="ifc:IfcObject">
      <xs:sequence>
      <xs:element name="ObjectPlacement" type="ifc:IfcObjectPlacement" [...]/>
      <xs:element name="Representation" type="ifc:IfcProductRepresentation" [...]/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Quellcode 2.4: Definition der Entität *IfcProduct* in ifcXML (BuildingSmart, 2017b)

Mit der Neuentwicklung des IFC4-Standards hat ifcXML eine komplette Neuentwicklung erfahren. So bietet die ifcXML-Repräsentation in der neusten Version die Möglichkeit, mit sämtlichen oder aber mit einer Teilmenge der IFC-Modelldefinitionen zu arbeiten (BuildingSmart, 2017e). Diese Entwicklung wurde unter dem Namen *simple ifcXML* veröffentlicht. Die Idee hinter *simple ifcXML* ist es, anwendungsspezifische Teilmengen des IFC-Datenmodells zu erzeugen, indem zunächst die EXPRESS-Definition mit Hilfe einer MVD-Definition (siehe Abschnitt 2.5.2) eingeschränkt und anschließend in die XSD überführt wird. Dies ermöglicht, die Komplexität zu reduzieren und so die Implementierung deutlich zu vereinfachen. Überdies führt ifcXML zu einem geringeren Ressourcenbedarf, da die Dateigröße in der Regel kleiner ist. Das Prinzip zu *simple ifcXML* ist schematisch in Abbildung 2.12 dargestellt.

2.5 Methodische Grundlagen

Wie bereits in Kapitel 1 erläutert, basiert die BIM-Methodik nicht ausschließlich auf technologischen Grundlagen und Hilfsmitteln, sondern bedarf vielmehr auch methodischer, prozessbasierter Grundlagen, damit die Vorteile von BIM, welche letztlich zu einer Effizienzsteigerung führen sollen, auch zum Tragen kommen. In den folgenden Abschnitten sollen die grundlegenden methodischen Grundlagen des Building Information Modeling, welche zum Teil mit der Einführung weiterer Datenformate zur technischen Unterstützung der Prozesse verknüpft sind, vorgestellt werden.

2.5.1 Modellbasierte Zusammenarbeit

Ein wesentlicher Grundsatz der BIM-Methode ist die modellbasierte Zusammenarbeit. Wie der Name bereits impliziert, ist im Rahmen des Bauprojekts der Ausgangspunkt für die Zusammenarbeit die digitale Abbildung des zu erstellenden Bauwerks sowie des gesamten Bauprozesses selbst. Eine wesentliche Herausforderung hierbei ist, dass es sich bei einem Bauvorhaben um einen komplexen Vorgang handelt, welcher von einer Vielzahl sehr unterschiedlicher Disziplinen sowie den beteiligten Unternehmen abhängig ist. Diese Unternehmen reichen von kleinen und mittleren Unternehmen bis hin zu großen multinationalen Konzernen. Jede dieser Organisationen nimmt für einen unterschiedlichen Zeitraum am Bauprojekt teil und bringt möglicherweise in diesem Zeitraum unterschiedliche Mengen und Arten von Daten in das Projekt ein. Zudem nutzen die Projektbeteiligten unterschiedliche (BIM-basierte oder aber auch nicht-BIM-basierte) Autorenwerkzeuge, um die erforderlichen Informationen zu erzeugen.

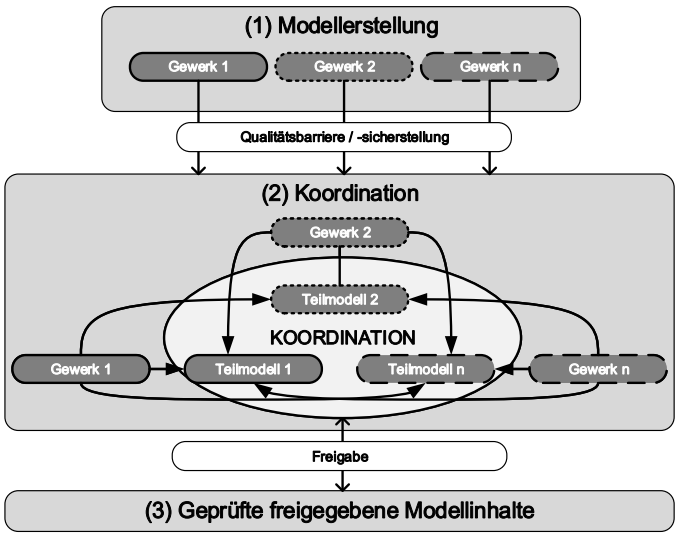


Abbildung 2.13: Grundlegendes Prinzip der modellbasierten Zusammenarbeit; nach (BCA Singapore, 2013)

Um diese Zusammenarbeit dennoch koordinieren zu können, ist es mittlerweile allgemein anerkannt, dass die direkte Verwendung eines einzelnen Modells, welches allen Beteiligten zur Verfügung steht, aus verschiedenen Gründen nicht empfehlenswert ist (Hausknecht und Liebich, 2016). In einem einzelnen Modell können sich Arbeitsbereiche verschiedener Disziplinen überschneiden und die Zuständigkeiten nicht eindeutig zugeordnet werden. Darüber hinaus haben die Projektbeteiligten in der Regel kein Interesse daran, ihre Zwischenstände aufgrund von rechtlichen Rahmenbedingungen und wettbewerblichen Situationen dauerhaft zu teilen. Daher implementieren diverse Richtlinien und Handlungsempfehlungen wie beispielsweise der *Singapore BIM Guide* (BCA Singapore, 2013), die *Publicly Available Specification (PAS) 1192* (British Standards Institution, 2013) oder das Reifegradmodell der britischen *BIM Task Group* (Bew und Richards, 2008) einen domänenspezifischen Ansatz, nach welchem die Modellautoren den vollen Zugriff nur auf ein von ihnen verantwortetes Teilmodell haben. Ein solches Teilmodell beschreibt lediglich einen bestimmten disziplinabhängigen Aspekt des gesamten Bauwerkmodells und wird daher

üblicherweise auch als disziplin- oder domänenspezifisches Teilmodell bezeichnet. Das grundlegende Prinzip dieser Zusammenarbeit ist in Abbildung 2.13 dargestellt.

Da jeder Modellautor für die eigene Teilmenge des Gesamtprojekts verantwortlich ist, wird auch häufig von einem föderalen oder föderativen Ansatz (*Federated Model Approach*) oder dem fach-modellbasierten Arbeiten gesprochen (Porwal und Hewage, 2013; Solihin et al., 2016). Mit diesem Ansatz werden Verantwortlichkeiten und Urheberschaft von Bauelementen sowie Änderungen bei der Ausführung des Bauvorhabens eindeutig geregelt.

In dieser Sichtweise gibt es im eigentlichen Sinne kein physisches Gesamtmodell, welches sämtliche Projektinformationen beinhaltet. Allerdings können die einzelnen Modelle bei Bedarf und für diverse Anwendungsfälle aus den einzelnen Teilmodellen zusammengeführt werden. Dieses ist insbesondere notwendig, wenn diese koordiniert werden müssen. Um die Integrität und Konsistenz des Gesamtmodells zu gewährleisten, müssen die domänenspezifischen Modelle verglichen und zu definierten Zeitpunkten auf Inkonsistenzen oder Kollisionen hin überprüft werden. Zu diesem Zweck bietet der föderale Ansatz das Konzept zu dem zentralen Koordinationsmodell (Solihin et al., 2016; VDI, 2017). Zur Koordination wird dieses Modell aus den betroffenen Teilmodellen zusammengesetzt und dient schließlich als Grundlage für die Koordination und Prüfung.

Für die datentechnische Handhabung der föderierten Modelle bieten sich unterschiedliche Lösungen: Zum einen ein proprietärer Ansatz, welcher die Verwendung mehrerer heterogener Dateiformate, welche in einer verwalteten Datenstruktur abgebildet werden, verfolgt. Die Schwierigkeit liegt bei diesem Ansatz darin, die Informationen aus den einzelnen Modellen in einer gemeinsamen Umgebung zu koordinieren. Für diese Art der Verwendung gibt es unterschiedliche Koordinationswerkzeuge (z.B. *Autodesk Navisworks* oder aber *desiteMD*), welche eine breite Unterstützung verschiedener nativer BIM-Formate bieten.

Ein anderer Ansatz sieht den Einsatz einer gemeinsamen Datenstruktur vor, in welcher die jeweiligen Modellinhalte abgebildet werden. Dieser adressiert maßgeblich die Verwendung von offenen Standards und somit insbesondere die IFC. Auch hier bieten sich diverse Tools an, welche sich für diesen Anwendungsbereich eignen (z. B. der *Solibri Model Checker* oder *simpleBIM*). An dieser Stelle sind auch weitere technische Lösungen wie Datenbankserver, welche häufig auch als Modell- oder BIM-Server bezeichnet werden (z.B. EDM-Modellserver (Jotne EPM Technology, 2004), BIM-Server (Beetz et al., 2010)), denkbar. Die Datenstrukturen dieser Server sind in der Regel eng an dem IFC-Schema orientiert oder bieten einen Direktübersetzer für IFC-Modellinhalte an.

2.5.2 Entwicklung von BIM Daten

Level of Development (LOD). In einem BIM-basierten Bauprojekt entwickeln sich die Modelldaten, welche sich aus der Gesamtheit aller Fachmodelle ergeben, über die Projektdauer stetig hin zu dem finalen Planungsstand, welcher schließlich das zu erstellende Bauwerk als Soll beschreibt. Die einzelnen Entwicklungsstufen, welche ein Bauwerksmodell hierbei durchläuft, werden auch als Reifegrad oder als *LOD* bezeichnet. Dieses Prinzip kann als analog zu einem Durchlaufen der Objektplanung hin zu immer detaillierten Planmaßstäben betrachtet werden.

Gemäß der Definition des NATSPEC (2013) beschreibt der LOD den "minimal erforderlichen dimensionalen, räumlichen, quantitativen, qualitativen und anderen Informationsgehalt, die in einem Modellelement enthalten sein müssen, um die genehmigten Verwendungen im Zusammenhang mit dem jeweiligen LOD zu unterstützen". Dieser Reifegrad muss für jede einzelne Modellübergabe festgelegt

werden und fasst die Anforderungen an die geometrische und semantische Detaillierung eines jeden Bauteils zusammen. Konkret bedeutet dies, dass durch ein LOD Anforderungen für jede projektspezifische Klassifikation und somit für jedes klassifizierte Bauteil festgehalten werden. Da sich der Informationsgehalt eines Objektes aus einer semantischen und geometrischen Ebene zusammensetzt, gibt es jeweils eine geometrische Detaillierung, welche als *Level of Geometry (LoG)*, und eine semantische Detaillierung, welche als *Level of Information (LoI)* bezeichnet wird.

Für die Definition der jeweiligen Reifegrade haben sich mittlerweile nationale Standards entwickelt, welche als Grundlage für die Definition der projektspezifischen Anforderungen herangezogen werden kann (Bopalgni, 2016). Ziel ist es, die Arbeitsweise mit dem LOD aufzuzeigen und dessen Anwendung zu standardisieren, damit das LOD-Konzept besser in kollaborativen BIM-Umgebungen implementiert werden können. Schließlich können solche Spezifikationen auch als standardisiertes Kommunikationswerkzeug eingesetzt werden. Üblicherweise wird in diesen Spezifikationen allerdings nicht strikt vorgegeben, welches LOD zu welchem Zeitpunkt in einem Projekt erreicht werden muss, sondern diese dienen als Grundlage, Referenz und Hilfestellung für projektspezifische Vereinbarungen, welche zusätzlich vertraglich festgelegt werden.

Im Jahr 2008 wurden von der AIA erste LOD-Definitionen als *Building Information Modeling Protocol* veröffentlicht (Bopalgni, 2016). Diese dienten als Grundlage und Vorarbeit für die Einführung der Spezifikationen des BIMForum (2017), welche aktuell zu den meistgenutzten Grundlagendokumenten zählen. Die LOD-Definitionen dienen als Referenz für Anwender in der AEC-Branche und wurden mit dem Ziel entwickelt, den Inhalt von Gebäudedatenmodellen in den verschiedenen Phasen der Gestaltungsplanungs- und Bauphase mit einem hohen Maß an Eindeutigkeit zu beschreiben und so Zuverlässigkeit zu garantieren. In den Dokumenten werden die Eigenschaften von Modellobjekten verschiedener Gebäudesysteme an verschiedenen LODs beschrieben und veranschaulicht. Die klare Definition erlaubt es Modellautoren, zuverlässige Modellinhalte zu generieren, und ermöglicht es den nachgeschalteten Modellnutzern, die Eignung aber auch die Grenzen der erhaltenen Informationen auf einfache Weise zu erkennen.

Für die Definition der Lieferanforderungen des Auftraggebers an den Auftragnehmer haben sich überdies Tools für die Erfassung der Anforderungen an Modellinhalte, sogenannte *Requirement Capturing Tools*, etabliert. Dabei handelt es sich um Softwaretools, mit welchen sehr einfach die Anforderungen an die Modellinhalte, definiert durch den jeweiligen LOD, digital abgebildet und so ausgetauscht, kommuniziert und geprüft werden können. Vertreter dieser Anwendungen sind unter anderem *BIM*Q* der Firma AEC3 GmbH (2017) oder aber LOD Planner (2018).

Information Delivery Manual (IDM). Für eine erfolgreiche Durchführung und ein reibungsloses Aufeinanderfolgen der Planungsprozesse ist es essenziell, dass bereits zu einem frühen Zeitpunkt im Projekt festgelegt wird, welche Informationen einem Projektbeteiligten zu welchem Zeitpunkt zur Verfügung stehen müssen. Nur so kann sichergestellt werden, dass der jeweilige Beteiligte auf die erforderlichen Informationen Zugriff hat und so seiner Aufgabe im Projektprozess nachkommen kann. Als prozesstechnische Grundlage wurde hierfür das IDM von buildingSMART entwickelt (BuildingSmart, 2017c), welches auch als ISO-Standard (ISO 29481-1) spezifiziert wurde (ISO, 2016). Ursprünglich wurde das IDM im Rahmen des BLIS-Projekts im Jahr 2000 eingeführt und später als offizieller Bestandteil der IFC-Standardisierung übernommen (Wix, 2007).

Ziel der Einführung des IDM ist es, die Mitglieder eines Projektteams dabei zu unterstützen, Vereinbarungen zu den geforderten Prozesse und Daten sowie für die die darin enthaltenen Verantwortlichkeiten zu treffen. Es enthält die Informationen, die jedem Projektteilnehmer in jedem Teil der Entwicklung zur Verfügung stellen sollte. Grundsätzlich arbeitet das IDM mit vier Hauptelementen: einer Prozesslandkarten (*Process Map*), den Austauschanforderungen (*Exchange Requirements*), den IFC-basierten Austauschanforderungen sowie einen allgemeingültigen Leitfaden zu den Objekten und Informationen des auszutauschenden BIM-Modells.

Mit Hilfe der graphischen Notation Business Process Model and Notation (BPMN) können die Datenaustauschprozesse konzeptionell in der *Process Map* beschrieben und so einzelne Datenübergabepunkte identifiziert werden. An den definierten Übergabepunkten kommt es jeweils zu einem Austausch von Modellinhalten, welche im Detail definiert werden müssen. Grundlegende Fragestellung bei der Definition dieser *Exchange Requirements* ist, welche Informationen für den anschließenden Prozess zur Verfügung stehen müssen, damit dieser durchgeführt werden kann.

Model View Definition (MVD). Die so identifizierten Austauschanforderungen können schließlich auch technisch in Form einer MVD festgehalten werden (BuildingSmart, 2016b; Hietanen, 2006). Dabei bezieht sich die Spezifikation des Austauschformats zwar auf das offene Datenschema der IFC, allerdings können auch Teilmengen von herstellereigenen Datenschemata adressiert werden. Während der Zweck des IDM darin besteht, Prozesse und Anforderungen an den Datenaustausch zu erfassen, zielt MVD darauf ab, diese Austauschanforderungen anhand des IFC-Schema abzubilden. Zweck von MVD ist es, den Datenaustausch sowohl geometrischer als auch nicht-geometrischer Informationen zu erleichtern, indem eine Sichtweise auf die Daten definiert wird, welche speziell auf die Bedürfnisse für einen bestimmten Verwendungszweck der jeweiligen Informationen zugeschnitten ist. Eine MVD definiert eine Teilmenge des IFC-Schemas, um eine bestimmte Austauschanforderung zu beschreiben, die für einen bestimmten Prozessschritt des Datenaustauschs erfüllt werden muss. Das bedeutet, dass für jeden Datenübergabepunkt eines Bauvorhabens eine solche Anforderung als MVD präzise definiert werden kann, um ein zuverlässiges und nachhaltiges Informationsmanagement innerhalb des Projekts zu gewährleisten.

Für die digitale Repräsentation einer MVD wurde von buildingSMART das auf XML basierende Format *mvdXML* eingeführt (BuildingSmart, 2016b; Hietanen, 2006).

Im Umkehrschluss können die Definitionen als Prüf- und Kontrollmittel eingesetzt werden, da die Qualität und Vollständigkeit eines Modells an der Übergabestelle anhand des MVD-Schemas gemessen werden kann. In der aktuellen Version der IFC4 sind die folgenden MVD als Standard festgelegt:

IFC4 Reference View (RV) wird als Eingabe für Prozesse verwendet, in welchen die Informationen des auszutauschenden Modells nicht bearbeitet werden sollen. Dieses ist beispielsweise der Fall, wenn Modelle koordiniert, also zur Überprüfung hinsichtlich eventuell vorhandener Kollisionen oder Inkonsistenzen kombiniert werden.

IFC4 Design Transfer View (DTV) eignet sich insbesondere, wenn die Informationen des Modells wiederverwendet werden sollen, also beispielsweise eine Weiterentwicklung der Gestaltungsplanung stattfinden soll. Zwar ist die Definition der Teilmenge der IFC in diesem MVD für eine Wiederverwendung optimiert, jedoch bedeutet dies nicht automatisch, dass sich die übermittelten Informationen auch für eine solche Verwendung eignen. BuildingSmart (2016b) weist darauf hin,

dass der MVD-Mechanismus ein "Roundtrip"-Szenario nur bedingt unterstützt. Daher sollten vor einer solchen Verwendung unbedingt entsprechende Testszenarien erfolgen.

Neben den standardisierten MVD-Definitionen wurden auch weitere Spezifikationen entwickelt, um den Datenaustausch für andere Zwecke zu unterstützen. So wurde beispielsweise im Rahmen des BLIS-Projekts im Jahr 2007 eine MVD entwickelt, welche den Austausch von strukturellen Gebäudedatenmodellen oder die Energiebedarfsermittlung unterstützt und bis heute weiterentwickelt (Pinheiro et al., 2018).

Die MVD-Methode stellt ein wesentliches Werkzeug dar, mit dem der Datenaustausch genau definiert und organisiert werden kann. Die Handhabung für Nicht-Programmierer ist allerdings anspruchsvoll. Technisch gesehen handelt es sich bei der MVD um ein XML-Schema, das von Menschen lesbar, aber sehr komplex strukturiert ist, so dass im Prinzip nur Anwender mit Programmierkenntnissen dieses sinnvoll verwenden können. Daher wird MVD in BIM-basierten Projekten noch nicht umfassend genutzt. Die Austauschprogramme werden hauptsächlich von domänenspezifischen Anwendern mit Programmierkenntnissen erstellt. Diese definierten Schemata können dann von den Anwendern als Blackbox verwendet werden. Da eine manuelle Definition dieser Spezifizierung aufgrund der zugrunde liegenden Komplexität der Struktur des Bauwerkmodells sehr umständlich wäre, soll MVD in diesem Punkt technische Hilfestellung leisten.

Modelllieferung. Da sich über die einzelnen Datenübergabepunkte mit den jeweils festgeschrieben LODs neue Projektdaten ansammeln, ergibt sich insgesamt eine Entwicklung der BIM-Daten in Richtung eines virtuellen Abbildes des zu erstellenden Bauwerks. Diese Entwicklung ist schematisch in Abbildung 2.14 dargestellt.

In diesem Beispiel sind zwei beteiligte Disziplinen abgebildet, welche jeweils die Anforderungen aus zuvor festgelegten Anforderungen (siehe *ER*-Strahl) beziehen. Bevor die erstellten Modellinhalte bei der Datenübergabe in die Koordinationsumgebung eingehen können, müssen die Inhalte des einzelnen Modells geprüft und an Hand bereits bestehender Modellinhalte koordiniert werden (siehe Kapitel 3). Dabei wird in der Koordinationsumgebung festgestellt, welcher Teil der gelieferten Inhalte gültig im Sinne von konsistent hinsichtlich der bestehenden Anforderungen ist. Dieser Teil kann als korrekt geliefert angesehen werden und steht somit als zentraler Modellinhalt und potenzielle Referenzgrundlage für weitere Disziplinen zur Verfügung. Inhalte, die nicht als korrekt eingestuft wurden, werden abgelehnt, für die Kommunikation aufbereitet und entsprechend an den Modellautor zur Korrektur zurückgesendet.

2.5.3 Kommunikation

Wie in Kapitel 2.2 beschrieben, spielt bei der modellbasierten Planung und Durchführung von Bauprojekten neben dem Austausch von Modellinhalten die Kommunikation zwischen den Akteuren eine wesentliche Rolle. Hierbei werden projektbezogene Informationen zwischen den Projektbeteiligten ausgetauscht, um unterschiedliche Arbeitsabläufe zu koordinieren, zu dokumentieren und durchzuführen. Zu diesen Prozessen gehören insbesondere die Planung von Aufgaben und Fristen oder das Problem- und Mängelmanagement (Beetz et al., 2015).

In der herkömmlichen Bauplanung gelten die technische Zeichnung und weitere zweidimensionale Dokumente als Referenz und Grundlage für die Kommunikation. Mit dem Einsatz von BIM kann das

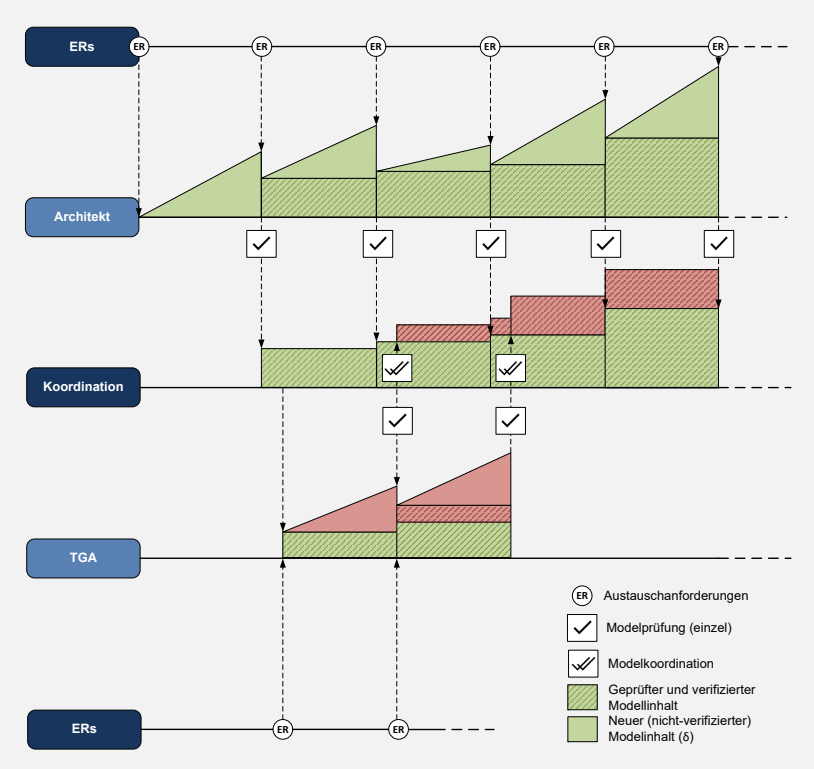


Abbildung 2.14: Entwicklung der Modelldaten in der Koordinationsumgebung durch das Zusammenführen der Teilmodelle

Bauwerksmodell als zentrale Grundlage herangezogen werden. Um die Übertragung und den Austausch der hierzu relevanten Informationen zu harmonisieren, wurde neben dem IFC-Datenformat das sogenannte BIM Collaboration Format (BCF) von (BuildingSmart, 2016a) als Standard eingeführt. In diesem offenen XML-basierten Datenformat ist festgelegt, welche Informationen in den BIM-basierten Prozessen herstellerneutral und allgemeingültig ausgetauscht werden können. Ein wesentliches Merkmal des Formates ist, dass mit diesem keine Bauwerksmodelle, sondern lediglich allgemeine Informationen zu dem vorliegenden Problem selbst, entsprechende Markierungen, räumliche Blickwinkel und Annotationen übertragen werden. Um ein Problem mit Modellinhalten zu verknüpfen, können darüber hinaus Referenzen zu betroffenen Bauteilen gespeichert werden. Auf diese Weise können mit Hilfe jeweils einzelner BCF-Objekte (in BCF als Thema, engl. *topic*, bezeichnet) Mängel bzw. Probleme in einem Bauwerksmodell zunächst markiert sowie dokumentiert und anschließend einer verantwortlichen Person zur Bearbeitung zugewiesen werden. In dieser Weise ersetzt ein einzelnes BCF-Objekt die Revision-Wolke, wie diese in der traditionellen Bearbeitung auf 2D-Plänen verwendet wird.

Während der Planungs- aber auch Ausführungsphase können im Rahmen des Mängelmanagements eines Projektes auftretende bauliche oder andere Mängel mit Hilfe der BCF-Objekte in der Planung und auf der Baustelle systematisch festgestellt, vermerkt und anschließend von den zugewiesenen Beteiligten nach- und ausgebessert werden. Je nach Größe und Detaillierungsgrad des Modells sowie Ausmaß der Planungsaufgabe kann die Zahl der auftretenden Probleme und Mängel leicht in die Hunderte oder

Tausende gehen. Der wesentliche Vorteil bei diesem modell- und prozessorientierten Vorgehen gegenüber der konventionellen Arbeitsweise ist, dass die Informationen, ähnlich einer Heftnotiz, mit einem Modellelement wie beispielsweise einem Raum- oder Bauteil verknüpfen werden können, obwohl diese nicht Teil des Modellinhalts sind. Auf diese Weise können die erstellten Mängel und deren beabsichtige Bedeutung schneller erkannt und besser nachvollzogen werden.

Die entstehenden BCF-Dateien können als Datei zwischen den verschiedenen Projektbeteiligten ausgetauscht werden. Da die Kommunikationsprozesse im Bauwesen allerdings bedingen, dass die Informationen mehrfach zwischen den Akteuren ausgetauscht werden, bietet es sich an, diese zentral und zugreifbar für alle Beteiligten zu speichern. Diese Anwendung bietet sich insbesondere bei der Implementierung eines CDE an (siehe Abschnitt 2.5.6).

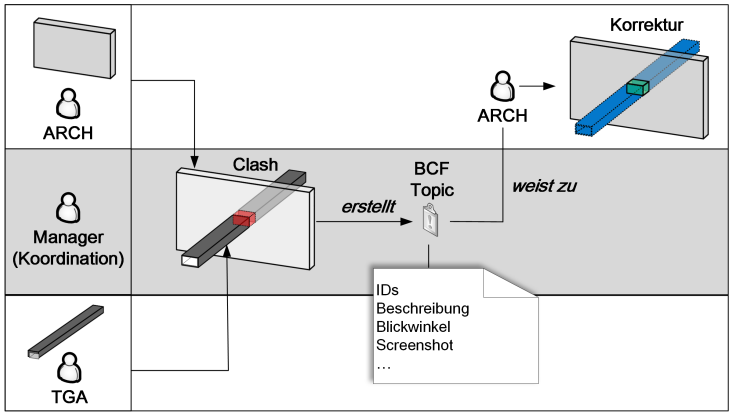


Abbildung 2.15: BCF-basierter Prozess zur Kommunikation von Kollisionskonflikten

In Abbildung 2.15 ist ein beispielhafter Kommunikationsprozess mit Hilfe von BCF dargestellt. In diesem Beispiel identifiziert der für die Koordination zuständige Fachplaner, eine Kollision zwischen zwei Elementen unterschiedlicher Fachdisziplin. Um diesen Fehler zu kommunizieren, erstellt er ein BCF-Objekt, welches die beiden betroffenen Elemente referenziert und die Art des Konflikts umfassend beschreibt, so dass der verantwortliche Fachplaner ausreichende Informationen für die Lösung zur Verfügung hat.

2.5.4 Klassifikationssysteme im Bauwesen

Ein Klassifikationssystem definiert eine organisierte Struktur, an Hand derer Informationen in Kategorien eingeordnet und strukturiert werden können. Im Bereich von BIM können diese verwendet werden, um Bauteile bzw. Objekte eines Modells nach unterschiedlichen Kategorien, z. B. Verwendung oder Funktionstyp, zu klassifizieren. Dabei können einem Objekt mehrere Klassifikationen zugewiesen werden, wodurch mehrere Organisationsebenen eines BIM-Modells entstehen (Gonçal, 2017). Ein Klassifkationssystem hilft bei der Verwaltung, Koordination und Auswertung über verschiedene Disziplinen, da entsprechend klassifizierte Bauteile einfacher gefunden und ausgewählt werden können (Kereshmeh und Eastman, 2016). Darüber hinaus gibt es Forschungsansätze, die eine Optimierung des Datenaustausches durch den Gebrauch individueller Systeme von jedem einzelnen Akteur für den internen Gebrauch oder bestehender Systeme, die als gemeinsame Referenz verwendet werden (Jørgensen, 2011), verfolgen.

Zudem können Klassifizierungssysteme für Anwender auch hilfreich sein, um Objekte eines BIM-Modells mit den Vorschriften, denen sie unterworfen sind, in Beziehung zu setzen. In den letzten Jahren wurden hierzu einige Forschungsarbeiten durchgeführt mit dem Ziel, die Zuordnung zwischen Taxonomien auf der Grundlage von Klassifikationssystemen und Vorschriften zu erleichtern. Cheng et al. (2007) haben einen systematischen Ansatz zur Abbildung von Vorschriften auf branchenspezifische Taxonomien vorgeschlagen, um so die Benutzerfreundlichkeit in ihrer Anwendung zu erhöhen. Durch die Verbindung zwischen Taxonomien, können Anwender strukturiert durch die referenzierten Vorschriften gehen und sehr leicht erkennen, welche Anforderungen für welche Elemente des Baumodells gelten.

Aus technischer Sicht ist in dem IFC-Datenmodell für die Zuweisung einer Klassifikation das Konzept *Classification Association* vorgesehen, mit welchem Verweise auf externe Informationsquellen hinzugefügt werden können. Die Quelle für die Informationen können ein Klassifizierungssystem oder ein *Dictionary*-Service, also z. B. ein Wörterbuchserver, sein. Die Referenz verweist entsprechend auf die externe Quelle oder auf ein Klassifizierungssystem, welches in die Projektdaten eingebettet ist (BuildingSmart, 2017a).

In der Praxis erfolgt die Zuweisung einer Klassifikation allerdings meist über ein Attribut, d.h. dem Objekt wird der entsprechende Klassfikationsschlüssel eines Systems als zusätzliches Attribut übergeben. Eine weitere Möglichkeit ist es, die Klassifikation durch eine entsprechende Benennung der Entitäten (z. B. Systeme, Typen oder Layer) zuzuweisen. Nach dem entsprechenden Attribut oder aber Namen kann anschließend gezielt gesucht und das Modell entsprechend strukturiert werden.

Für die semantische Strukturierung der Informationen des Bauwesens haben sich in den vergangenen Jahren unterschiedliche Klassifikationsstandards entwickelt. Die meisten wurden vor dem Hintergrund eingeführt, gemeinsame Kriterien für die Strukturierung der Informationen in BIM-Modellen, vor allem der enthaltenen Komponenten, nach verschiedenen Kriterien (Funktionen, Aktivitäten, Räume, Systeme, etc.) bereitzustellen.

Bekannte Vertreter solcher Klassifizierungssysteme, die derzeit breite Akzeptanz und Verwendung in der Industrie finden, sind:

Uniclass wurde 1997 vom *Construction Project Information Committee (CPIC)* veröffentlicht mit dem Ziel, Informationen zu organisieren, die allen Beteiligten während des gesamten Lebenszyklus eines Projekts zur Verfügung stehen. Hierzu gehören die Organisation von Bibliotheksmaterialien und die Strukturierung von Produktliteratur (NBS, 2015).

Uniformat kann dazu verwendet werden, Informationen über ein Gebäude in einer Standardordnung auf der Grundlage von Bauelementen (oft als Systeme oder aber Baugruppen bezeichnet) oder Gebäudeteilen zu organisieren, die durch eine jeweilige Funktion charakterisiert sind. Dabei wird keine Rücksicht auf die für die Konstruktion verwendeten Materialien und Methoden genommen (Construction Specifications Institute, 2016b).

Masterformat dient der Organisation von Spezifikationen und die Erstellung von Vertragsdokumenten, welche für Planung und Bau von Gebäuden insbesondere in Ländern wie den USA und Kanada verwendet werden (Construction Specifications Institute, 2016a). Die Spezifikation wurde vom *Construction Specifications Institute* und *Construction Specifications Canada* mit dem Ziel herausgegeben, eine Klassifikation zur Organisation von Informationen zu Bauanforderungen und -aktivitäten bereitzustellen. Der Schwerpunkt bei diesen Informationsanforderungen liegt hierbei allerdings auf Produkten. Wesentliches Ziel des *Masterformat* ist es, die Kommunikation zwischen Planern,

Auftragnehmern und Bauherren zu erleichtern, um den Anforderungen und Kostenvoranschlägen eines Projekts gerecht zu werden.

OmniClass oder aber *OmniClass Construction Classification System* (oder OCCS) (OCCS, 2017) zielt darauf ab, MasterFormat, Uniformat, Uniclass und weitere Klassifizierungssysteme, die sich aus den von der ISO sowie den Unterausschüssen und Arbeitsgruppen der International Construction Information Society (ICIS) entwickelten Standards ableiten, in einem einzigen System auf der Grundlage der ISO 12006-2 zu vereinen. Wegen der umfassenden Beschreibung, gilt *OmniClass* als wichtiges Instrument der Zukunft für die AEC-Branche, welches nicht nur für das Projektmanagement, sondern auch als Klassifizierungssystem für die Suche und den Vergleich von Produkten geeignet ist.

Die einzelnen Klassifizierungssysteme unterscheiden sich stark hinsichtlich der Granularität ihrer Definition und Struktur. Daher muss bei der Verwendung von Klassifikationssystemen berücksichtigt werden, dass sich eine Anpassung an die spezifischen Kriterien und Anforderungen einer individuellen, internen Projektstruktur teilweise schwierig gestalten kann (Kereshmeh und Eastman, 2016; Knopp-Trendafilova, 2010). Teilweise können die beiden unterschiedlichen Strukturen aufgrund der fehlenden Variabilität nicht ohne Weiteres aufeinander abgebildet werden. Beispielsweise muss der interdisziplinäre Charakter des Projekts berücksichtigt werden, sodass die Informationen nach Disziplinen klassifiziert werden können. Allerdings unterscheiden sich die Organisationsstruktur und die Arbeitsweise von Bauherren und Subunternehmern sehr stark voneinander (Knopp-Trendafilova, 2010).

Eine mögliche Lösung für die genannten Probleme ist, dass eine Mehrfachverwendung von unterschiedlichen Klassfikationssystemen bei der föderalen modellbasierten Arbeitsweise zulässig ist, so dass jeder Akteur nach dem jeweiligen Bedarf individuelle, projektspezifische und gleichzeitig auch Standard-Klassifikationssysteme verwendet. Auf diese Weise ist zumindest gesichert, dass das verwendete Klassifizierungssystem, den Bedürfnissen des Anwenders entspricht. Überdies lassen sich die Systeme sinnvoll anwenden, um grundlegende Konzepte, welche auf einer gleichen Bedeutung beruhen, projektübergreifend zu harmonisieren.

2.5.5 Rechtliche Aspekte der Zusammenarbeit

Als rechtliches Grundgerüst und somit als Basis für die Durchführung eines BIM-Projekts dient die Vertragskonstellation zwischen den beteiligten Partnern. In Abbildung 2.16 sind die vorgeschlagene vertragliche Konstellation sowie die zugehörigen typischen Vertragsdokumente dargestellt, welche ein BIM-Projekt neben dem Hauptvertrag auszeichnen (Eschenbruch und Leupertz, 2016).

Als Grundlage für das Projekt legt der Auftraggeber in den sogenannten Auftraggeberinformationsanforderungen (AIA) neben einer allgemeinen Beschreibung des Bauvorhabens fest, welche Ziele er mit der Anwendung von BIM verfolgt und in welchen Fällen digitale Methoden zur Anwendung kommen sollen. Zusätzlich wird in den AIA festgelegt, welche Daten zu welchem Zeitpunkt benötigt werden. Hierzu gehören insbesondere auch die Angaben, in welcher Detailtiefe und in welchem Datenformat die Lieferungen erfolgen sollen (BMVI, 2015b; Eschenbruch und Leupertz, 2016). Zusätzlich kann in diesem Dokument auch festgelegt werden, welche Normen und Richtlinien innerhalb des Projektes im Besonderen zu beachten sind.

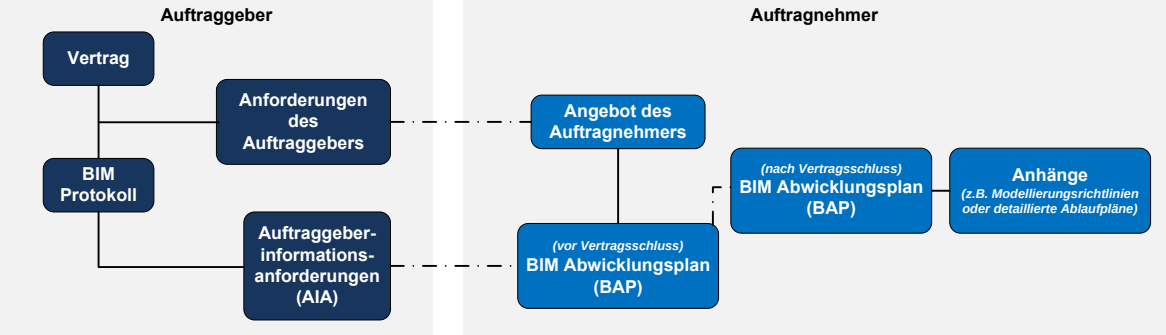


Abbildung 2.16: Vertragliche Konstellation und wesentliche Dokumente eines BIM-basierten Bauprojekts

Typischerweise beauftragt der Auftraggeber den Auftragnehmer, im Gegenzug im BIM Abwicklungsplan (BAP) festzulegen, wie der detaillierte Fahrplan des BIM-Projektes hinsichtlich der Erstellung, Weitergabe und Verwaltung von Daten aussieht. Hierzu zählen essenzielle Fragen, wie insbesondere, wie oft und wann Planungsbesprechungen sowie Zusammenführungen der Fachmodelle mit Kollisionsprüfungen stattfinden. Zudem muss auf die Anwendungsfälle und Ziele, welche in den AIA definiert wurden, konkret Stellung bezogen und Lösungsansätze aufgezeigt werden. So kann beispielsweise detailliert darauf eingegangen werden, welche Teile der Planung bis wann in welcher Detailtiefe geliefert werden, sowie wann und in welchem Umfang Visualisierungen, Mengenermittlungen oder aber Simulationen durchgeführt werden (BMVI, 2015b; Borrmann et al., 2015b). Erfahrungswerte aus bereits absolvierten BIM-basierten Projekten werden typischerweise in Form von Modellierungsrichtlinien oder Handreichungen dokumentiert. Diese können einem BAP zusätzlich als Anhang und einer entsprechenden Referenzierung angehängt werden.

2.5.6 Gemeinsame Datenumgebung

Die Verwaltung der digitalen Informationen und der zugehörigen Prozesse stellt eine der wesentlichen Herausforderungen während der Planung und Durchführung eines BIM-basierten Bauprojekts dar.

Ein einfacher Ansatz, um diesen kollaborativen Ansatz umzusetzen, wäre ein Austausch von lokalen Dateien über die entsprechenden Export- und Import-Schnittstellen. Dieses bringt allerdings in der Regel massive Probleme mit sich, welche unter anderem in der Versionskontrolle verursacht werden, da mehrere Beteiligte an verschiedenen Stellen im Prozess Schlüsseldaten bereitstellen. Bei diesen Ansätzen spricht man daher auch häufig noch von siloartigen Strukturen, bei welchen keine eigentliche Datenintegration herrscht (Dassault Systemes, 2016). Wenn es keine einheitliche und zuverlässige Datenquelle (engl. *single source of truth*) gibt, birgt dieses die Gefahr, dass den Mitwirkenden aussagekräftige kontextbezogene Daten fehlen, die ihnen dabei helfen, bessere Entscheidungen zu treffen.

Daher ist es mittlerweile weithin anerkannt, dass sich für die Speicherung und Verwaltung der unterschiedlichen Modelle und deren Inhalte bei dem fachmodellbasierten Arbeiten insbesondere zentrale digitale Projektplattformen eignen (Young et al., 2009). Grundsätzlich speichern diese Datenplattformen digitale Objekte zentral und können diese für diverse Akteure zugreifbar machen. Darüber hinaus

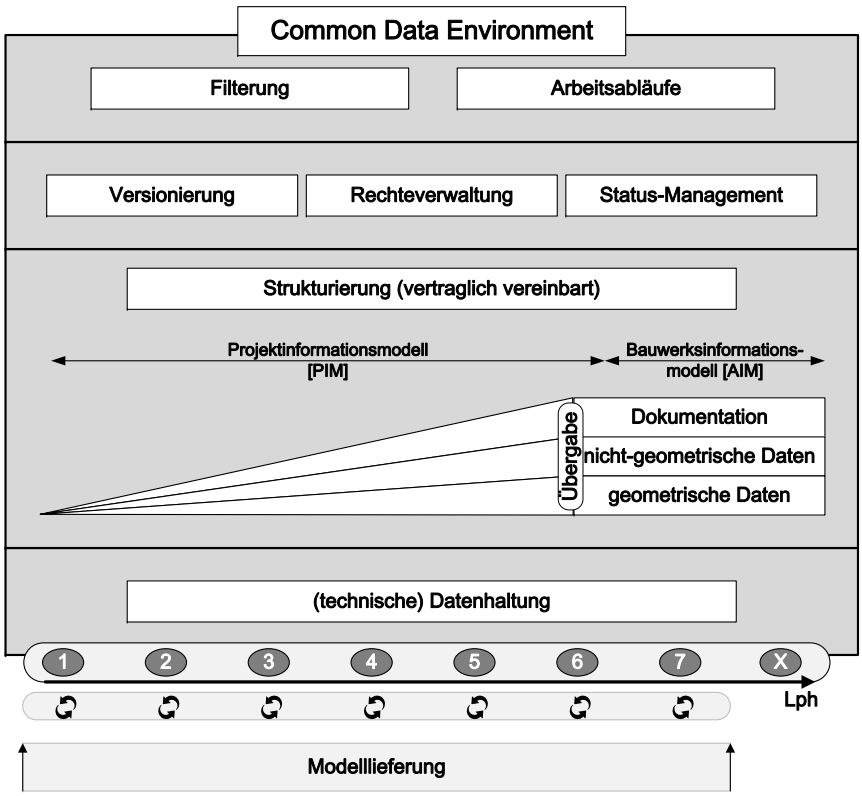


Abbildung 2.17: Technischer Aufbau eines *Gemeinsamen Datenraums* (CDE) als Schichtenmodell; angelehnt an (British Standards Institution, 2013, 2014)

können Objekte in diesen Datenplattformen durch Modifikationen gepflegt und aktualisiert werden, wodurch sich eine Kontinuität ergibt.

Mit der Veröffentlichung der britischen Richtlinie PAS 1192 (British Standards Institution, 2013, 2014) wurde sowohl ein wesentlicher methodischer als auch technologischer Grundstein für diese modellbasierte Zusammenarbeit gelegt. In dem Dokument wurde das Konzept des Common Data Environment (CDE), einem digitalen, gemeinsamen Datenraum für die Projektbeteiligten eines BIM-basierten Bauprojektes, eingeführt.

Ein CDE stellt einen zentralen, digitalen Datenraum dar, welcher sich für die Sammlung, Verwaltung, Auswertung und den Austausch von bauwerksbezogenen Informationen eignet. Alle Projektbeteiligten beziehen die Daten aus diesem Datenraum, und im Gegenzug werden die gültigen erstellten Modellinhalte in dieser Umgebung gespeichert. Das CDE beinhaltet somit sämtliche disziplinspezifische Fachmodelle und digitale Dokumente, welche während Planung und Durchführung des Bauprojektes erstellt und verwendet werden.

Die zentrale Verwaltung der Informationen im CDE bringt den Vorteil mit sich, dass die Gefahr von Redundanzen signifikant reduziert und gleichzeitig die Aktualität und Vollständigkeit der Daten zu jedem Zeitpunkt erhöht werden kann. Weiterhin führt der Einsatz eines CDE zu einer höheren Wiederverwendungsrate von Informationen und vereinfacht die Zusammenführung von Modellinformationen.

Zusätzlich kann das CDE auch für die Archivierung und Dokumentation von Planungsständen eingesetzt werden (VDI, 2017).

Da die Umgebung für alle Projektbeteiligten zugänglich ist, kann sie als Plattform für BIM-basierte kollaborative Prozesse genutzt werden. Zu beachten ist, dass die PAS 1192 (British Standards Institution, 2013, 2014) grundsätzlich keine Empfehlungen für die technische Implementierung der Plattform selbst oder der Prozesse enthält. In diesem Sinne beschreibt der Leitfaden einen breiten Rahmen für die technische Umsetzung eines CDE, stellt aber keine detaillierten Anforderungen, sodass Spielraum für Interpretationen und die technische Umsetzung besteht. Dieses liegt neben der technischen Neutralität der Spezifikation auch daran, dass Aufbau und Funktionen eines CDE in vielerlei Hinsicht von der Anwendung, dem Projektvolumen und der Anzahl der Nutzer abhängig sind.

Dennoch soll an dieser Stelle eine technische Umsetzung eines CDE exemplarisch dargestellt werden. Die Architektur eines CDE kann wie in Abbildung 2.17 als Schichtaufbau beschrieben werden (VDI, 2017). Neben der Datenhaltung ist die Strukturierung der gespeicherten Informationen ein wesentlicher Bestandteil des CDE. Diese Strukturierung muss zu Beginn eines Projektes vereinbart werden und sollte, bei Bedarf vertraglich, laufend aktualisiert werden. Neben der eigentlichen Speicherung der Informationen sollten verschiedene Prozesse und Arbeitsabläufe (engl. *Workflows*), z.B. für den Informationsaustausch, die Modellprüfung, die Versionierung oder die Archivierung, technisch unterstützt werden.

Mittlerweile haben sich bereits diverse Softwarelösungen als Implementierungen eines CDE auf dem Markt etabliert. Ausgewählte Vertreter solcher praxisrelevanter Lösungen werden in (Preidel et al., 2016) vorgestellt.

Relevanz und Bedeutung für die Modellprüfung

Die Einführung eines gemeinsamen Datenraums hat insbesondere für die Modellprüfung und somit auch für die Modellqualität im Allgemeinen eine hohe Relevanz. In der PAS 1192 British Standards Institution (2014) sind verschiedene Status vorgesehen, welche ein Modell bzw. Modellinhalt durchlaufen muss, bevor dieser in das CDE eingeht und für alle Projektbeteiligten als wahrheitsgemäße und verlässliche Quelle zur Verfügung steht. Die jeweiligen Status (geteilt [*shared*], in Bearbeitung [*in progress*], veröffentlicht [*published*], archiviert [*archived*]) sind in Abbildung 2.18 dargestellt.

Eine Überführung von einem Status zum nächsten ist immer dann möglich, wenn entsprechende Prüfungs- und Genehmigungsprozesse durchlaufen wurden. Somit ist also prozesstechnisch sichergestellt, dass eine entsprechende Prüfung zwingend stattfindet. Die Einführung des CDE führt so zwangsläufig zu einem strikten und getakteten Modellprüfungsprozess, bei welchem die einzelnen Modellinhalte jeweils die Qualitätsbarrieren durchlaufen müssen, bevor diese überhaupt als anerkannte Modellinhalte für die jeweils anderen Projektbeteiligten freigegeben werden können. Die Modellprüfung wird so nicht nur zu einem festen Bestandteil des BIM-Prozess in den Endphasen, sondern wird vielmehr zentral in der Modellentwicklung verankert und stellt somit einen Kern dar.

In Anlehnung an die Entwicklung der Bauwerksinformationen durch die einzelnen Modelllieferungen, wie diese in Abschnitt 2.5.2 beschrieben wurden, lässt sich die zentrale Koordinationsumgebung nun also mit einem CDE ersetzen. Die Modellprüfung nimmt hierbei eine Schlüsselposition bei jeder einzelnen Datenübergabe ein und fungiert als zentrale, immer wiederkehrende Leistung. Entsprechend ist die Modelllieferung mit Hilfe eines CDE in Abbildung 2.19 dargestellt.

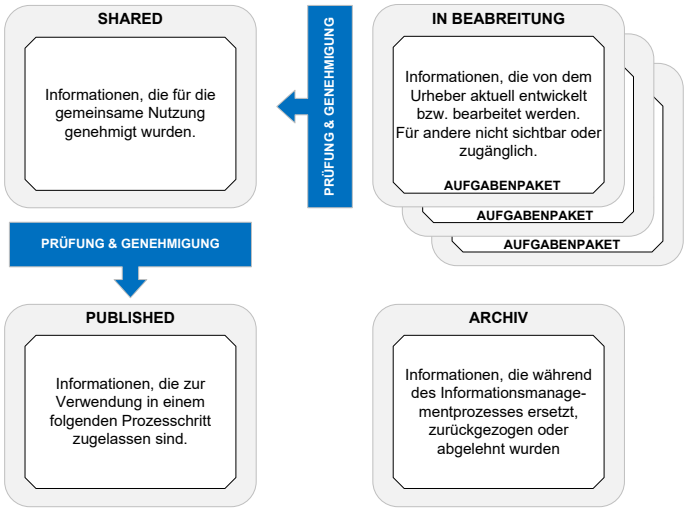


Abbildung 2.18: Status von Modellinhalten, welche gemäß der PAS 1192 durchlaufen werden (British Standards Institution, 2014)

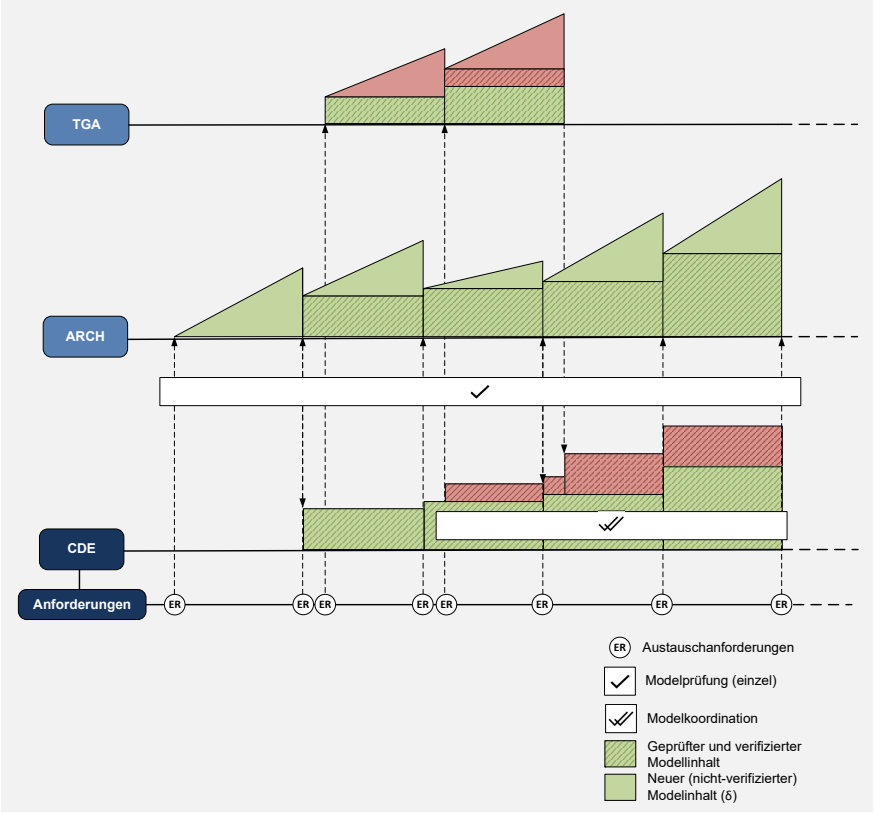


Abbildung 2.19: Entwicklung der Modelldaten in einem gemeinsamen Datenraum durch das Zusammenführen von Teilmodellen

2.6 Zusammenfassung

Das Planen und Errichten eines Bauwerks zeichnet sich aufgrund der Projekt- und Standort-spezifischen Anforderungen durch Einmaligkeit und aufgrund der erforderlichen Zusammenarbeit der verschiedenen Projektbeteiligten durch Interdisziplinarität aus. Darüber hinaus lässt sich eine steigende Komplexität der Bauaufgaben selbst beobachten. Der Einsatz von digitalen Werkzeugen, wie insbesondere die Building Information Model Methode, verspricht an dieser Stelle erhebliche Optimierungspotenziale.

Die BIM Methode sieht den konsequenten Einsatz digitaler Bauwerksmodelle über den gesamten Lebenszyklus einer gebauten Anlage hinweg vor. Die Modelle umfassen neben den Repräsentationen der Bauteile auch alle relevanten semantischen Informationen, einschließlich der jeweiligen Beziehungen zwischen den Objekten selbst. Somit bietet ein Bauwerksmodell als digitale Darstellung des realen Gebäudes eine optimale Grundlage für jegliche Art von rechnerischer Anwendung.

In einem BIM-basierten Projekt entwickeln die Modellautoren ihre jeweiligen disziplinären Teilmodelle. Der Inhalt eines Modells ist maßgeblich von dem beabsichtigten Zweck und der jeweiligen Projektphase abhängig. Jeder beabsichtigte Anwendungsfall eines Modells erfordert einen spezifischen Grad der geometrischen Detaillierung sowie des Informationsgehalts des Modells. Daher muss nicht nur der Austausch der Modelle zwischen den Projektpartnern organisiert, sondern alle enthaltenen Informationen in regelmäßigen Abständen hinsichtlich mannigfaltiger qualitativer Kriterien geprüft werden.

Zu festgelegten Zeitpunkten werden die einzelnen Modelle zu einem Koordinationsmodell zusammengefügt, so dass eine disziplinübergreifende Koordination und Prüfung stattfinden kann. Somit wird sichergestellt, dass die Modellinhalte jeweils für sich aber auch in gemeinschaftlicher Betrachtung fehlerfrei sind.

Der hierfür notwendige Austausch der Informationen zwischen den Beteiligten Akteuren und Systemen ist somit von großer Bedeutung. Hier muss zwischen einem geschlossenen Ansatz, bei dem Produkte nur eines Anbieters oder seiner proprietären Schnittstellen zum Einsatz kommen, und einem offenen Ansatz, der auf Hersteller-neutralen, standardisierten Datenformaten wie den Industry Foundation Classes basiert, unterschieden werden. Um die Zusammenführung der Modelldaten auch technisch zu unterstützen hat sich mittlerweile das Konzept des Common Data Environment, einem gemeinsamen digitalen Datenraum, etabliert.

3 Die Qualität digitaler Bauwerksmodelle

3.1 Einführung

Wie in Kapitel 2 beschrieben, bedeutet die Einführung von BIM für Unternehmen der AEC-Branche eine Transformation der herkömmlichen Kollaborationsprozesse hin zu einem modellbasierten Ansatz. Wie der Name bereits impliziert, sind die verwendeten Modelle und deren Inhalte die wesentliche Grundlage für den Wissensaustausch zwischen den kooperierenden Fachplanern sowie für diverse weitere ineinander verzahnte oder aber aufeinander aufbauende Anwendungsfälle.

Wegen der herausragenden Bedeutung der Modelle spielt die Qualität der Modellinhalte eine tragende Rolle und deren Fehlerfreiheit stellt ein wesentliches Erfolgskriterium dar. Nicht erkannte bzw. behobene Fehler führen unweigerlich zu weiteren Fehlern oder gar zu einer Verkettung derselben. Es ist folglich mittlerweile allgemein anerkannt, dass es sich bei der Prüfung und Qualitätssicherung der Modelle um einen essenziellen Baustein bei der digitalen Bauplanung handelt. Diese spielt insbesondere bei der vertraglich vereinbarten Übergabe der Modelldaten vom AN an den AG, dem sogenannten *data drop* eine zentrale Rolle.

Der Begriff der Qualität kann in Abhängigkeit der Sichtweise unterschiedlich beschrieben werden: in einer objektiven Sichtweise bezieht sich dieser auf die Beschaffenheit eines Objekts, in einer subjektiven hingegen auf die Güte der Gesamtheit der charakteristischen Eigenschaften eines Objektes, Systems oder Prozesses (Duden, 2013). Bei einer subjektiven Betrachtung wird also impliziert, dass für die Feststellung der Qualität eine Prüfung vorausgeht, um die Güte des betrachteten Objekts festzustellen.

Im Kontext von BIM taucht der Begriff Qualität insbesondere im Zusammenhang mit den Begrifflichkeiten Qualitätsprüfung (im engl. meist *quality checking*) oder aber Qualitätssicherstellung (im engl. meist *quality assurance*) (Hjelseth, 2010b, 2015a; Kulusjärvi, 2012; Nisbet et al., 2008) auf. Gemeint ist damit die allgemeine Sicherstellung der Qualität der in digitalen Bauwerksmodellen enthaltenen Informationen. Es ist wichtig festzuhalten, dass es bei dieser Modellprüfung nicht darum geht, die Inhalte eines Modells zu korrigieren bzw. anzupassen. Zentrale Aufgabe ist es, Fehler hinsichtlich maßgebender Qualitätskriterien zu identifizieren, diese inhaltlich z.B. durch Beschreibungen aufzubereiten und entsprechend dem verantwortlichen Projektbeteiligten zu kommunizieren.

In den folgenden Abschnitten soll daher die Identifikation bzw. Festlegung geltender Qualitätskriterien und der eigentliche Prüfprozess voneinander getrennt betrachtet werden.

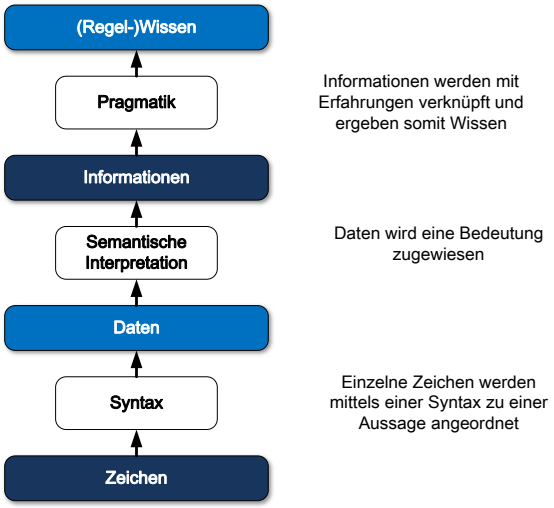


Abbildung 3.1: Entstehung von Wissen aus Daten angelehnt an (Aamodt und Nygård, 1995; Fuchs-Kittowski, 2001)

3.2 Qualitative Kriterien für digitale Bauwerksmodelle

In der Literatur wird der Begriff Modellqualität im Kontext von BIM häufig verwendet, ohne dabei eine einheitliche Definition zu liefern oder sich auf eine solche zu beziehen. So wird beispielsweise in diversen nationalen oder internationalen Richtlinien der Aspekt Modellqualität und -prüfung behandelt (Kulusjärvi, 2012), allerdings fehlt es an einer Konkretisierung der Kriterien, welche geprüft werden müssen, damit ein Modell qualitative Anforderungen erfüllt und damit als qualitativ hochwertig gelten kann. Eine Festlegung von konkreten Qualitätskriterien fällt schwer, weil die Prüfung wie auch die Modellinhalte selbst sehr stark zweckgebunden sind und projektspezifischen Anforderungen unterliegen. Für jeden Prüfvorgang im Rahmen eines Projekts sollte daher definiert und vereinbart sein, welche Kriterien für die jeweiligen Modellinhalte gelten.

Ein klareres Verständnis für die Modellqualität an sich ergibt sich, wenn man die unterschiedlichen Ebenen eines Bauwerksmodell betrachtet, in welche sich dieses gliedert. Wie bereits in Kapitel 2 beschrieben, handelt es sich bei den digitalen Gebäudemodellen um hochstrukturierte Daten, die sich in verschiedene Ebenen gliedern lassen. Da jede einzelne dieser Ebenen der potenzielle Ursprung für Fehler sein kann, müssen diese zu Beginn eindeutig voneinander abgegrenzt werden.

Für eine solche Abgrenzung ist es hilfreich, zunächst die allgemeine Struktur von Datenmodellen, wie diese in der Informationstechnik bzw. dem Wissensmanagement zum Einsatz kommen, zu betrachten. Diese Struktur orientiert sich an den jeweiligen Entwicklungsstadien, welche Daten durchlaufen bis diese schließlich als Wissen bezeichnet werden (Arnold, 2009; Hildebrand et al., 2015). In Abbildung 3.1 ist schematisch die Entstehung von Wissen als Schichtenaufbau, angelehnt an (Aamodt und Nygård, 1995; Fuchs-Kittowski, 2001), dargestellt.

In dem dargestellten Modell wird die Entstehung von Wissen in die vier Schichten *Zeichen*, *Daten*, *Informationen* und *Wissen* unterteilt.

Die Basis stellt eine Menge an nicht weiter interpretierten oder verarbeiteten Zeichen (Rohdaten), das Wissen hingegen die Spitze dar. Der Begriff *Zeichen* beschreibt in diesem Zusammenhang eine Menge von numerischen oder alphanumerischen Zeichen sowie Sonderzeichen. In einer ersten Entwicklungsstufe wird den Zeichen mit Hilfe einer Syntax eine Aussage zugeordnet. Unter dem Begriff *Syntax* versteht man hier ein System von Regeln, nach welchen Zeichen zu einer Aussage angeordnet werden. Die so geordneten Zeichen bilden schließlich Daten, welche sich messen, ordnen und strukturieren lassen (Arnold, 2009).

In einer weiteren Entwicklungsstufe werden die Daten interpretiert und diesen auf diese Weise eine Bedeutung zugewiesen. Daher wird in diesem Kontext auch der Begriff *Semantik* verwendet, welcher die *Bedeutung* oder auch den *Inhalt* eines Objektes bezeichnet. Das Ergebnis dieser semantischen Interpretation bildet die nächste Ebene des Systems, die Informationen. Da Informationen Objekten bereits eine Bedeutung zuweisen, lassen sich mit ihrer Hilfe z.B. Problemzusammenhänge erklären und somit Ziele erreichen.

Das Wissen bildet die oberste Ebene des Systems und entsteht aus der Verknüpfung von Informationen durch Erfahrungen, Wertvorstellungen sowie Fachkenntnisse. Mit Hilfe von Wissen lassen sich zuverlässige Aussagen über eine Sache oder einen Sachverhalt auf Basis von Erfahrungen machen. Entsprechend weist Wissen eine wesentlich höhere Komplexität als die einfache Information auf und ist zugleich eines der höchsten Güter einer Unternehmung, denn auf einer Wissensbasis lassen sich Entscheidungen treffen oder Aktionen einleiten. Das aus Erfahrung stammende Wissen kann schließlich wiederum mit Schlussfolgerungen verknüpft werden, die in Handlungsempfehlungen resultieren. Die Formalisierung solcher Handlungsempfehlungen wird im Kontext der Ingenieurwissenschaften in Form von normativem Wissen bzw. Gesetzen festgelegt, welche wiederum dazu bestimmt sind, das gemeinsame Miteinander zu regeln. So hat beispielsweise die Erfahrung, dass ein bestimmtes Material mit einer gewissen Wahrscheinlichkeit eine ausreichende Festigkeit aufweist, um Stabilität zu garantieren, dazu geführt, dass dieses Material nach den anerkannten Regeln der Technik verwendet werden darf. Weitere Beispiele zu dieser Form des Wissens wird in Kapitel 4 der vorliegenden Arbeit behandelt.

Dieses Modell kann als Grundlage dienen, um Qualitätskriterien auf den verschiedenen Ebenen eines Datenmodells zu definieren und diese anschließend in einem Prüfprozess sicherzustellen. Allerdings müssen an dieser Stelle einige spezifische Anforderungen und Randbedingungen der BIM-basierten Bauplanung berücksichtigt werden.

Wie auch in dem allgemeinen Datenmodell muss bei Gebäudemodellen als Ausgangspunkt zunächst die syntaktische und somit datentechnische Korrektheit betrachtet werden. Je nach gewähltem Datenstandard bzw. -format des Modells, also beispielsweise ein natives Format oder aber IFC, muss gewährleistet sein, dass den Rohdaten entsprechend eine eindeutige Aussage zugeordnet werden kann. Die datentechnische Fehlerfreiheit muss gegeben sein, um anschließend Modellinhalte korrekt lesen zu können. Diese Ebene der Modellqualität ist stark abhängig von den Software-Schnittstellen, mit welchen die jeweiligen Modelle erstellt werden. Daher ist es in erster Linie die Aufgabe der Softwarehersteller, einen reibungslosen Export der Modelle auf höchster datentechnischer Qualität zu gewährleisten. Hilfreich sind hierbei nicht zuletzt auch eine geeignete Dokumention sowie Handlungsempfehlungen, damit es insbesondere im Umgang mit herstellerneutralen Standards wie den IFC zu keinen datentechnischen Fehlern kommt. Durch entsprechende Modifikations- und Einstellungsmöglichkeiten an solchen Schnittstellen kommt allerdings auch auf den Anwender Verantwortung zu, da dieser Einfluss auf den Export nimmt.

Tabelle 3.1: Ebenen der Qualität in einem Gebäudemodell

Kategorie	Verantwortung	Ursprung	Prüfbarkeit
(Gestaltungs-)planerische Qualität	• Modellautor • Gestaltungsplaner (meist identisch mit dem Modellautor) • Prüfer/Prüfingenieur aufseiten der Behörde	• Normen und Richtlinien • kunden- bzw. bauherrenspezifische Anforderungen	• visuell und manuell • objektive Kriterien können z. T. mit Tools (teil-)automatisch geprüft werden • komplexe Prüfungen können teilweise mit Hilfe von Prüftools automatisiert werden
(Modell-)inhaltliche Qualität	• Modellautor • Festgelegter Verantwortlicher, welcher die Modelllieferung schuldet • BIM-Koordinator	• Modellierungsrichtlinien • Vereinbarungen zu den Modelllieferungen (siehe IDM, MVD und LOD in Kapitel 2)	• teilweise automatisiert • manuelle und visuelle Kontrolle • Prüftools
Datentechnische Qualität	• Softwarehersteller • Verantwortlicher für die Datenschnittstellen • Modellautor (Einstellungsmöglichkeiten an den Schnittstellen)	Datenstandards und -spezifikationen	• weitgehend automatisiert • Export- und Importschnittstellen unterliegen einer Zertifizierung und typischerweise sind zusätzliche Qualitätskontrollen in diesen integriert • zusätzliche Prüftools für die Erstellung von Fehlerprotokollen

Liegen die Daten datentechnisch fehlerfrei vor, können die Modellinhalte betrachtet werden. Wie bereits in Abschnitt 2.4.2 ausgeführt, können unter Modellinhalt alle Informationen verstanden werden, welche auf der geometrischen und semantischen Ebene eines Modells enthalten sind. Diese Qualitätsstufe ist maßgeblich abhängig von den Vereinbarungen, welche im Rahmen des Projektes für den Austausch und die Lieferung der Modellinhalte getroffen wurden. Daher ist auf dieser Ebene maßgeblich der Modellautor oder BIM-Koordinator dafür verantwortlich, dass die Modellinhalte gemäß dieser Vereinbarungen korrekt im Modell enthalten sind. Diverse Aspekte dieser Ebene werden nachfolgend im Detail behandelt.

Die Modellinhalte können schließlich dazu verwendet werden, um die planerische Qualität eines Entwurfs bzw. einer Gestaltungsplanung, welche sich aus den gesamten Modellinhalten ergibt, zu bewerten. Dabei werden bei der Bauplanung nicht nur objektive, sondern auch subjektive Kriterien betrachtet. Folgend sind ausgewählte Aspekte, welche üblicherweise bei der Entwurfs- bzw. Gestaltungsplanung betrachtet werden, aufgeführt (Albert und Schneider, 2016; Schneider und Rjasanowa, 2018; Solihin, 2016):

- Einhaltung der Projektvorgaben oder -ziele, z. B. kundenspezifische funktionale Anforderungen, Bauherrenrichtlinien
- Einhaltung der einschlägigen Gesetze, Leitfäden, Normen und Vorschriften (insbesondere Bauvorschriften). Dieses kann die Rücksprache mit den Behörden, wie z. B. der örtlichen Planungsbehörde oder dem Bauaufsichtsamt erfordern, die sich zu grundlegenden Aspekten des Entwurfs äußern.
- Risiken, die mit der Konstruktion verbunden sind, wie z. B. die Verwendung innovativer Komponenten oder nicht standardmäßige Elemente der Konstruktion
- Anfertigung von Notfallplänen, wie z.B. Entfluchtungsplan

- Anforderungen hinsichtlich der Mach- und Baubarkeit, Kosten sowie Programmierung des Entwurfs
- Anforderungen hinsichtlich der Nachhaltigkeit, wie z. B. Standortwahl, Verfügbarkeit von Verkehrsmitteln, Energieverbrauch und Energiequellen, Flexibilität und Langlebigkeit, Materialauswahl, Abbau und Abriss oder Wiederverwendung
- Anforderungen an die ästhetische Qualität

An diesen Kriterien lässt sich erkennen, dass in der BIM-gestützten Bauplanung grundlegend zwischen einer inhaltlichen und planerischen Qualität getrennt werden muss. Ein Modell, welches hinsichtlich der inhaltlichen Qualität Defizite aufweist, kann dennoch hinsichtlich der planerischen Qualität gut sein. Dieses gilt logischerweise auch im umgekehrten Fall. Als Beispiel kann hier die Kollisionserkennung angeführt werden, bei welcher die planerische Qualität von mehreren Teilmodellen untereinander geprüft und koordiniert wird. Auch wenn die inhaltliche Qualität eines Modells gut ist, kann es hinsichtlich der Kollisionen zu wesentlichen planerischen Fehlern kommen. Ähnliche Beispiele lassen sich auch für die Mengenermittlung oder die Validierung des Raumprogramms finden. Auch wenn in einem Modell ein Raumprogramm durch die Vergabe von korrekten Raumnamen und Funktionen dargestellt ist, heißt dieses noch nicht, dass die Anforderungen des Raumprogramms vonseiten des AG, welche zu den planerischen Anforderungen gezählt werden kann, erfüllt wurden. Daraus lässt sich ableiten, dass in einem ersten Schritt die inhaltliche Qualität gegeben sein muss, bevor die planerische Qualität nachhaltig geprüft werden kann, da sich der planerische Inhalt aus dem Modellinhalt zusammensetzt. So wird beispielsweise ein Gebäudemodell, welches keine Raumnamen und -funktionen vorhält, bei einer planerischen Qualitätsprüfung nicht bestehen können, da die Informationsgrundlage für die Prüfung des Raumprogramms fehlt.

Es kann zusammenfassend festgehalten werden, dass es sich bei der Prüfung der inhaltlichen Qualität, um eine objektive Betrachtung der Modellinhalte handelt. Diese Qualität muss grundlegend erfüllt sein, damit die Eingangsinformation für die nachfolgenden Prüfprozesse verwendet fehlerfrei sind.

Bei der Betrachtung der planerischen Qualität kann es qualitative Aspekte geben, welche sich nur schwer oder nicht durch formalisierten Anforderungen bzw. Prüfregeln darstellen lassen. Dabei kann es sich beispielsweise um subjektive Betrachtungen bzw. Prüfungen handeln, bei welcher in verschiedenen Kontexten der gleichen Informationen eine andere Qualität hinsichtlich eines definierten Kriteriums zugewiesen werden kann. Bei einer solchen Prüfung liegt es im Auge des empfangenden Projektbeteiligten (zumeist der Auftraggeber), ob die in den Modellinhalten dargestellte planerische Qualität den eigenen Anforderungen entspricht. In diesen Fällen muss abgewogen werden, welche (Teil-)Kriterien sich formal vereinbaren lassen, so dass man an Hand von Bauwerkinformationen ableiten kann, ob planerische Mindestanforderungen erfüllt sind. Allerdings sind besonders Anforderungen wie die Kriterien zur Messbarkeit des ästhetischen Anspruchs eines architektonischen Entwurfs schwer objektiv prüfbar.

In Tabelle 3.1 sind zusammenfassend die einzelnen Ebenen der Modellqualität zusammengestellt.

Als Ergebnis der vorangegangenen Betrachtung lassen sich grundlegende Kategorien für qualitative Kriterien definieren, deren Aufbau und Struktur in Abbildung 3.2 dargestellt sind. Dabei hat die Auflistung der einzelnen Kriterien keinen Anspruch auf allgemeingültige Vollständigkeit, sondern stellt lediglich eine wesentliche Basis für die Definition der Modellqualität dar. Zur Veranschaulichung werden die Kategorien sowie aufgeführte Kriterien im Folgenden erläutert:

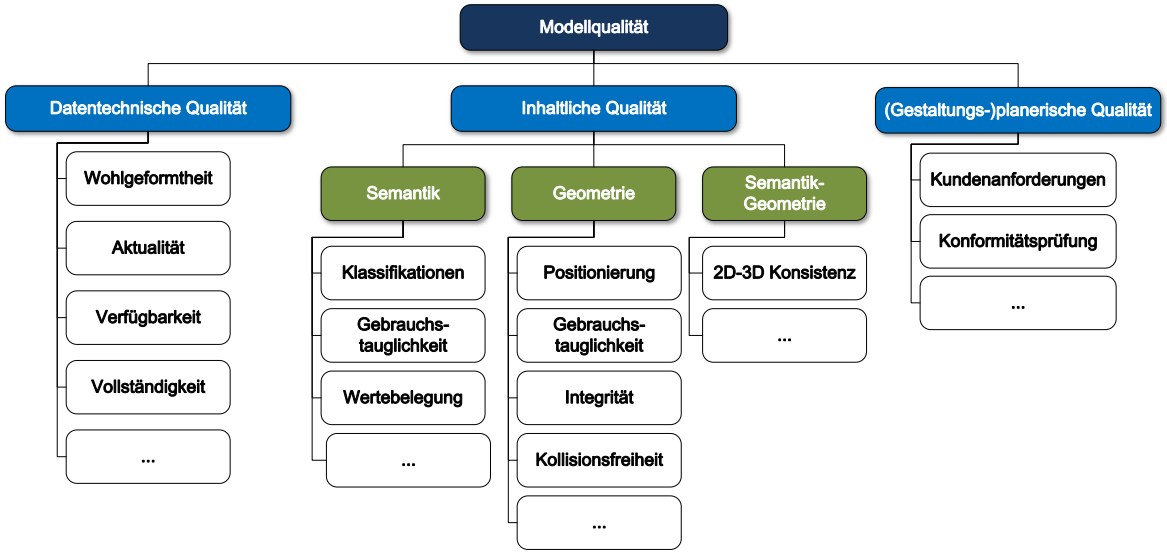


Abbildung 3.2: Kategorisierung der Modellqualitätskriterien

Datentechnische Qualität

Die datentechnische Qualität eines Modells bezieht sich auf die unterste Ebene eines Bauwerksmodells, in welcher die Rohdaten beschrieben sind. Zu dem Zeitpunkt der Prüfung dieser Ebene hat typischerweise noch keine Interpretation der Daten stattgefunden. Treten Fehler auf dieser Ebene des Modells auf, so muss man grundsätzlich davon ausgehen, dass prozess- oder softwaretechnische Probleme vorliegen.

- ***Wohlgeformtheit***

  Wie bereits in Abschnitt 2.4.2 beschrieben, gibt es im Bauwesen verschiedene Möglichkeiten, Daten darzustellen und zu modellieren. Allerdings ist diesen Methoden gemeinsam, dass sie einem festen Schema, einer formalen Syntax, folgen. Ungeachtet der Inhalte müssen Daten gemäß dieses Schemas definiert sein und dieser formalen Syntax folgen, damit diese von einer Instanz (*Parser, Compiler oder Interpreter*) interpretiert werden können und so schließlich zu Informationen, also interpretierten Daten, werden.

  Die syntaktische Ebene eines Modells beschreibt ein Regelsystem, nach welchem die Elemente eines Systems miteinander verknüpft werden und definiert somit die Beziehung dieser Elemente untereinander (Schiffer, 1998). Fehler auf der syntaktischen Ebene stellen im Allgemeinen Verstöße gegen die Satzbauregeln einer festgelegten Sprache dar und führen dazu, dass die interpretierende Instanz den fehlerhaften Teil der Daten zurückweist (Aho et al., 1999). Ein Fehler entsteht auf der syntaktischen Ebene dann, wenn die gespeicherten Informationen der zugrunde liegenden Datenstruktur widersprechen, also nicht mit den formalen Kriterien übereinstimmen. Erfüllen die Daten sämtliche Anforderungen eines Schemas, werden diese als wohlgeformt (engl. *well-formed*) bezeichnet (Ernst et al., 2015).

Diese Gruppe von Regeln betrifft folglich in erster Linie syntaktische Aspekte entsprechend dem Standard, welcher für die Datenmodellierung des jeweiligen Modells definiert wurde. Ein solcher Standard sind beispielsweise die IFC, welche auf unterer Ebene mit dem *STEP Part 21* eine Syntax für Instanz-Dateien festlegen. In einer nachfolgenden Ebene wird durch den IFC selbst darüber hinaus festgelegt, welche enthaltenen Datensätze gültig sind. Da diese Art von Fehlern in der Regel die Interpretation der fehlerhaften Daten verhindert, können diese frühzeitig erkannt werden. Üblicherweise bricht ein lesendes Programm an der syntaktisch/lexikalisch fehlerhaften Stelle ab und es wird eine Fehlermeldung ausgegeben.

Fehlerbeispiele: Falsch geschriebene Signalworte, nicht interpretierbare Zeichenketten, undefinierte Bezeichner (z.B. Variablen, Funktionen, Literale), falsche Verwendung reservierter Symbole (z. B. fehlende Klammern), Typ-Konflikte, falsche Anzahl übergebener Parameter

- ***Vollständigkeit***

  Die Vollständigkeit des Datensatzes ist eine Grundvoraussetzung dafür, dass die Daten von einer interpretierenden Instanz (*Parser, Compiler oder Interpreter*) gelesen werden können. Sind Datensätze nicht vollständig, weil diese beschädigt wurden oder aber der Schreib- oder Transferprozess unterbrochen wurde, können Informationen nicht extrahiert werden. In der Regel führt dieses zu einer Fehlermeldung bei dem Leseprozess.

  Fehlerbeispiele: Beschädigte Dateien, unvollständige Datenübertragung

- ***Aktualität***

  Grundlegend muss die Aktualität der zu prüfenden Daten gewährleistet werden. Eine Prüfung, welche sich nicht auf die aktuellen Datensätze bezieht, ist nicht aussagekräftig und führt zu nicht relevanten Ergebnissen. Wie auch bereits heutzutage in der gängigen Praxis Pläne mit veraltetem Stand abgelehnt werden und lediglich zum Zwecke der Dokumentation archiviert werden, muss dieses ebenso auch mit veralteten Modellständen passieren. Nicht-aktuelle Modellinhalte dürfen daher nicht als Bearbeitungsgrundlage verwendet werden und sollten abgelehnt werden.

  Die Aktualität muss prozesstechnisch (siehe Abschnitt 2.2) gewährleistet und technisch durch eine eindeutige Angabe, welchen Planungsstand das jeweilige Modell beschreibt, identifizierbar sein. Ob ein Modell oder eine Information aktuell ist, kann beispielsweise über Metadaten gekennzeichnet werden. Grundlage für die Einhaltung und Kennzeichnung der Aktualität ist die Verwendung der Status von Modellinhalten wie diese in Abschnitt 2.5.6 vorgestellt wurden.

  Fehlerbeispiele: Veralteter Planungsstand eines Modells

- ***Verfügbarkeit***
  Die Verfügbarkeit beschreibt das Maß, ob ein System oder Daten zu einem bestimmten Zeitpunkt bzw. innerhalb eines vereinbarten Zeitrahmens zur Verfügung stehen. Ähnlich wie bei der Aktualität muss auch die Verfügbarkeit von Modellinhalten prozesstechnisch als auch technisch gewährleistet sein. Dieses ist insbesondere dann relevant, wenn aufgrund der Komplexität und Größe des Bauprojektes eine Vielzahl von Modellen zum Einsatz kommt, welche in einer gemeinsamen Datenumgebung verwaltet werden.

  Fehlerbeispiele: Auf Modellinhalte kann nicht zugegriffen werden, Modellinhalte wurden nicht rechtzeitig für die Prüfung geliefert

Inhaltliche Qualität

Die inhaltliche Qualität bezieht sich auf interpretierte Modellinhalte, welche wiederum wie in Kapitel 2 in unterschiedliche Ebenen aufgegliedert werden. Zum einen beziehen sich die Kriterien auf die semantischen oder aber geometrischen Inhalte des Modells, zum anderen aber auch auf die Wechselwirkungen zwischen diesen beiden Ebenen.

- **Semantik**
  - ***Klassifikationen***
    Wie in Abschnitt 2.5.4 beschrieben, dienen Klassifikationssysteme dazu, Bauwerksmodelle an Hand von festgelegten Kategorien zu strukturieren. Ein Bauteil kann über ein Attribut oder über die Bezeichnung des Modellelements entsprechend klassifiziert werden. Neben allgemeinen standardisierten Klassifikationssystemen kommen in Projekten auch häufig projektspezifische Klassifikationen zum Einsatz, welche in zusätzlichen vertraglichen Dokumenten oder Modellierungsrichtlinien festgehalten werden. Da die Festlegung dieser Klassfikationen die enthaltenen Modellelemente grundlegend definiert, ist die korrekte und vollständige Zuweisung der Klassifikationen essenziell.

    Fehlerbeispiele: Fehlende/falsche Zuweisung einer Klassifikation, Verwendung eines alten oder aber falschen Klassifikationssystems

  - ***Semantische Eignung***
    Eine zentrale Anforderung an die semantische Qualität eines Modells ist, ob die Inhalte den zweckgebundenen Anforderungen entsprechen. Diese Anforderungen ergeben sich aus den projekt(-phasen)-spezifischen Ansprüchen, die als Austauschanforderungen für die Modellliefe-rung formuliert und vertraglich vereinbart wurden. An Hand dieser Anforderungskataloge lässt sich exakt festlegen, welche Leistungen von dem Modellautor in Form der Modellinhalte zu erbringen sind. Eine solche Anforderung kann beispielsweise mit Hilfe des IDM (siehe Abschnitt 2.5.2), als Teil eines BIM-Leitfadens oder als separate Projektlieferung definiert werden. Daher wird in diesem Zusammenhang häufig auch von einer IDM-Prüfung gesprochen (Hjelseth, 2010b; Hjelseth und Nisbet, 2010).

Dabei gilt es zu beachten, dass unzureichende Informationen ebenso als Fehler zu beurteilen sind wie überflüssige Informationen. Die Lieferung von nicht-geforderten Inhalten birgt das Risiko, dass Inhalte nicht konsequent weiter gepflegt werden und es so zu Redundanzen oder Widersprüchlichkeiten kommt.

Beispielanforderungen: Vollständigkeit der Modellinhalte für die Übergabe an andere Disziplinen (bspw. Facility Management (FM) definiert durch COBie), Wand-, Fenster und Dachbauteile müssen einen U-Wert, eine Schalldämmung und eine Brandschutzklasse beinhalten

- ***Zulässige Wertbelegung***
  Für die Belegung der semantischen Informationen lassen sich Bedingungen formulieren, welche an das jeweilige Attribut Anforderungen definieren. In Abhängigkeit des gewählten Datentyps können Gültigkeitsbereiche beispielsweise in Form von sinnvollen Grenzen (Maximal- und Minimalwert) oder aber in Form einer Auflistung gültiger Werte beschrieben werden, welche das Attribut annehmen darf. Mit Hilfe so formulierter Bedingungen lassen sich Fehler verhältnismäßig einfach und schnell identifizieren.
  Wie in Abschnitt 2.4.2 beschrieben sind solche Bedingungen auch bereits fest in den *WHERE*-Statements des *EXPRESS*-Schema und somit auch in dem IFC-Datenstandard niedergelegt. Diese Bedingungen gelten nicht projektspezifisch, sondern domäneübergreifend. Spezifische Anforderungen lassen sich hier beispielsweise mit Hilfe der in Abschnitt 2.5.2 vorgestellten MVD und dem zugehörigen Datenformat *mvdXML* formulieren.

  Beispiele: die maximal zulässige Dicke einer Wand beträgt 2,00 m; die maximal zulässige Höhe einer Wand beträgt 100 m; für Stahlbetonteile muss das Attribut *Material* aus folgender gültiger Liste stammen Stahlbetonstütze *C20/25;C25/30;C30/35*; Alle Stahlbetonteile müssen in ihrem Namen die Zeichenkette *STB* aufweisen

- ***Redundanz- und Widerspruchsfreiheit***
  Die Komplexität eines Bauwerksmodells sowie der häufige Austausch der Informationen birgt die Gefahr, dass es zu redundanten Modellinhalten kommt. Mangelnde Pflege der Modelle und eine unvollständige Prüfung der Einhaltung der Informationsanforderungen können so sehr schnell zu Widersprüchen im Modell führen. Daher muss sichergestellt werden, dass Redundanzen und Widersprüchlichkeiten, sogenannte Inkonsistenzen, vermieden werden. Diese Sicherstellung ist eng verbunden mit der bereits aufgeführten semantischen Eignung.

  Beispiele: Objekte enthalten doppelte Mengenangaben (z. B. *Volumen*), deren Werte sich widersprechen; Klassifikation wird als Objektbezeichner und gleichzeitig als eigenes Attribut angegeben

- **Geometrie**
  - ***Positionierung***
    Bei der geometrischen Beschreibung spielt insbesondere die geodätische Lage eines Bauprojekts eine wesentliche Rolle. Hierbei ist wichtig, dass sich alle raumbezogenen Anwendungen und Dokumentationen einheitlich und ausschließlich an einem festgelegten geodätischen Bezugssystem orientieren. Ein solches System kann beispielsweise mit drei *Gauß-Krüger*-Projektionen definiert werden, um die geometrischen Verzerrungen bei der Abbildung in die Ebene möglichst klein zu halten (Blankenbach, 2015). Bei der Festlegung eines solchen Bezugssystems kommt in dem IFC-Standard das Objekt *IfcProjectedCRS* zum Einsatz, welches ein Koordinatensystem definiert, auf das sich die Übersetzung der lokalen ingenieurtechnischen Koordinatensysteme bezieht. Dabei geben die Attribute *MapProjection* und *MapZone* eindeutig an, welche Projektion auf das zugrundeliegende geographische Koordinatensystem angewandt wird (BuildingSmart, 2017a).
    Für die Zusammenarbeit und ein fehlerfreies Zusammenführen aller Teilmodelle sollte überdies ein einheitlicher Projektnullpunkt definiert werden. Für die Definition eines solchen Punktes sollte eine eindeutige Kennzeichnung definiert werden, sodass insbesondere bei dem Zusammenführen der Modelle Abweichungen schnell erkannt werden können.

    Fehlerbeispiel: Fehlender Projektnullpunkt; Verwendung von globalen Koordinaten (sogenannte *große Koordinaten*)

  - ***Geometrische Integrität***
    Die geometrische Integrität bezieht sich auf eine korrekte inhaltliche Repräsentation von geometrischen Körpern. Wie in Abschnitt 2.4.2 dargestellt, gibt es beispielsweise in den IFC unterschiedliche Arten von geometrischen Repräsentationen, welche für die Beschreibung eines Körpers herangezogen werden können. Für jede dieser Repräsentationen gelten Regeln, welche die Integrität sicherstellen. Für eine integere geometrische Beschreibung ist die Erfüllung dieser Regeln zu gewährleisten.
    So ist die Geschlossenheit der Oberflächen ein allgemeines und wesentliches Kriterium für alle Volumenkörper. Für indirekte geometrische Repräsentationen, beispielsweise beschrieben durch den IFC-Typ *IfcManifoldSolidBrep*, gilt weiterhin die *Euler-Charakteristik* (BuildingSmart, 2017a; Hazewinkel, 2002):

    $$\chi = V - E + 2F - L_{\mathrm{l}} - 2(S - G^{\mathrm{s}}) = 0$$

    mit:
    V, E, F, L_{l}, S: Anzahl der *Knoten*, *Kanten*, *Flächen*, *Loops* und *Hüllen*
    Gs: Anzahl der verschiedenen *Hüllen*-Typen

    Allgemein gilt, dass diese Charakteristik erfüllt sein muss, damit der vorliegende Körper integer ist. Analog lassen sich solche Regeln auch für weitere geometrische Repräsentationen definieren. Neben dieser einen beispielhaft aufgezeigten Regel gibt es weiterhin für indirekte, aber auch direkte Repräsentationstypen eine Vielzahl weiterer Bedingungen, die grundlegend erfüllt

sein müssen. An dieser Stelle sei beispielhaft für die direkten Typen die Regelung genannt, dass sogenannte *Sweep*-Körper sich nicht selbst durchdringen dürfen oder aber keine *CSG* Vereinigungen auf auseinanderliegende Objekte angewendet werden darf.

Fehlerbeispiele: Verletzung der Euler-Charakteristik; *Non-Sense* Objekte; Falsche Definition der Richtung von Flächen (Falsche Richtung der Normalenvektoren); Doppelung oder Übereinanderliegen von Kanten oder Knoten

- ***Geometrische Eignung***
  Wie bereits für die semantischen Informationsgehalte gilt auch für die geometrischen Beschreibungen, dass diese für bestimmte Zwecke ausreichend detailliert und auf die korrekte Art und Weise beschrieben sein müssen. Gemäß den Anforderungen müssen die geometrischen Körper also in einem entsprechenden Detaillierungsgrad beschrieben sein, damit diese für den jeweiligen Zweck genutzt werden können.
  An dieser Stelle können also wesentliche Anforderungen in Bezug auf die geometrische Repräsentation definiert werden. So können beispielsweise eine explizite (Annäherung des Volumenkörpers über angenäherte Außenflächen) oder implizite Repräsentation (direkte Darstellung des Volumenkörpers) in Abhängigkeit des Anwendungsfalls verlangt werden. Für die expliziten Darstellungsformen kann eine maximale Anzahl der verwendeten Flächen für die Annäherung festgelegt werden in Abhängigkeit von dem notwendigen Detaillierungsgrad und der Genauigkeit des jeweiligen Bauteils. Wahlweise kann diese Flächendichte auch im Verhältnis zu dem Volumen betrachtet werden, damit insbesondere unnötig detaillierte Bauteile mit einer vergleichbaren geringen Größe identifiziert werden können. Nicht zuletzt wird auch in den LODs selbst, wie diese in Abschnitt 2.5.2 vorgestellt wurden, festgelegt, welche Bauteile welchen Detaillierungsgrad erfüllen muss, um den Anforderungen zu genügen.
  Unterschiedliche Anwendungsfälle sind unter anderem die Koordination oder die Wieder- bzw. Weiterverwendung von geometrischen Beschreibungen. Für die Koordination von Teilmodellen, z. B. für die Identifikation von Kollisionen, wird typischerweise eine indirekte, für eine Wieder- bzw. Weiterverarbeitung im Rahmen der Gestaltungsplanung hingegen eine direkte geometrische Beschreibung gewählt, da hier die implementierte geometrische Parametrik mit Einschränkungen erhalten bleiben.

  Fehlerbeispiele: Übergabe der indirekten geometrischen Repräsentation für eine Weiterverwendung der Geometrie; die Anforderungen eines LOD an den geometrischen Detaillierungsgrad werden nicht ausreichend erfüllt; indirekte geometrische Repräsentation mit einer unnötig hohen Anzahl an Dreiecksflächen (Typischer Anwendungsfall: abgerundete Geländerstangen) als Indikator auf eine Übererfüllung des geforderten Detaillierungsgrad

- ***Kollisionsfreiheit***
  Ein qualitatives Kriterium, welches bereits weitverbreitete Anwendung findet, ist die Kollisionsfreiheit von Modellen in sich oder zwischen verschiedenen Teilmodellen. Es handelt sich hierbei um eine geometrische Prüfung für die Erkennung von Kollisionen (engl. *clash detection*). Bei

dieser Prüfung werden Komponentenpaare daraufhin untersucht, ob sich deren geometrische Repräsentationen überschneiden. Bei einer einfachen Anwendung wird direkt geprüft, ob sich die Repräsentationen direkt physisch überschneiden. In diesem Kontext wird auch von *harten* Kollisionen gesprochen (engl. *hard clashes*).
Diese Prüfung spielt zwar auch in einzelnen Modellen eine Rolle, allerdings verhindern die meisten BIM-Werkzeuge, dass es zu dieser Art der Überschneidungen kommt, und warnen den Nutzer beispielsweise entsprechend. Bei einer föderativen Arbeitsweise, bei welcher die einzelnen Disziplin-Modelle zu festgelegten Zeitpunkten zusammengeführt werden, um deren Konsistenz zu prüfen, ist die Prüfung hin auf Kollisionen essenziell, da erst ab dem Zeitpunkt der Zusammenführung geprüft werden kann, ob die Planungsstände gemeinsam konsistent sind.

Beispiele: Kollisionsfreiheit des Rohbaumodells; Kollisionsfreiheit in dem Koordinationsmodell, welches aus unterschiedlichen Fachmodellen zusammengeführt wurde, z. B. zwischen Architektur- und TGA-Komponenten

- **Semantisch-Geometrische Ebene**

  Obwohl die beiden Ebenen Semantik und Geometrie in einem Bauwerksmodell getrennt voneinander betrachtet werden können, kommt es zu Wechselwirkungen zwischen den beiden Ebenen. In der Folge kann es auch auf dieser Ebene zu qualitativen Problemen kommen.

  - ***Semantisch-geometrische Konsistenz***

    In diesem Kontext beschreibt der Begriff Konsistenz die Geschlossenheit und somit Widerspruchsfreiheit eines Bauwerksmodells auf semantisch und geometrischer Ebene. Die bauteilorientierte Modellierung hat zur Folge, dass es zwischen den beiden Ebenen Überlappungen gibt, die konsistent gehalten werden müssen. So kann ein Bauteil in einem Gebäude sowohl über die Geometrie als auch über ein Attribut lokalisiert werden, wenn beispielsweise die Lage im Stockwerk über ein Attribut dargestellt wird. Ein weiteres Beispiel für eine solche Überlappung ist die Angabe einer Mengenberechnung. Das Volumen eines Bauteils wird in der Regel als Attribut abgelegt, jedoch wird dieses bereits indirekt über die geometrische Repräsentation dargestellt. Eine solche Redundanz kann prozessbedingt notwendig und sinnvoll sein. Allerdings muss sichergestellt werden, dass diese nicht zu Widersprüchlichkeiten im Modell zwischen den Ebenen führen (Daum und Borrmann, 2013; Stadler und Kolbe, 2007).
    Sollte eine Redundanz beabsichtigt sein, kann es sinnvoll sein, eine zulässige Unschärfe der Information bzw. Toleranz zu definieren. So kann das Attribut *Volumen* zusätzlich als ab- oder aufgerundeter Wert neben der geometrischen Repräsentation angegeben werden.

    Fehlerbeispiel: Ein Bauteil wird laut Attribut dem Obergeschoss zugeordnet, liegt laut geometrischer Darstellung allerdings im Erdgeschoss. Abweichung zwischen dem Wert des Attributs *Volumen* und einer Volumenberechnung aus der geometrischen Darstellung.

  - ***2D-3D-Konsistenz***

    Gemäß der BIM-Methodik werden technische, zweidimensionale Zeichnungen aus einem Bauwerksmodell abgeleitet. Dient das Modell als zentrale Grundlage für die Zeichnungen,

so ist sichergestellt, dass die Zeichnungen keine widersprüchlichen Informationen enthalten. Allerdings ist es schwer zu verhindern, dass Informationen auf Zeichnungen nachträglich manuell hinzugefügt, angepasst oder aber verändert werden. Da die Verbindung zwischen Modell und Zeichnung so unterbrochen werden kann, muss auch hier geprüft werden, ob eine geometrische (z. B. Dimensionen) als auch semantische (z.B. Raumstempel) Konsistenz zwischen Plänen und Modellen gegeben ist (Trzeciak, 2018).

Fehlerbeispiele: Falsche Dimensionen eines Bauteiles im Plan; abweichende Materialauswahl zwischen Plan und Modell; fehlende Bauteile in Plänen; falsche Raumstempel

Planerische Qualität

Sind die definierten Kriterien und Anforderungen der beiden vorangegangenen Ebenen der Qualität erfüllt, so kann schließlich auch die oberste Ebene der Qualität eines Bauwerksmodells betrachtet werden. Auf dieser Ebene werden die Inhalte des Modells hinsichtlich qualitativer Anforderungen geprüft, welche sich aus unterschiedlichen Quellen ergeben. Diese sollen im folgenden kurz vorgestellt werden.

- **Vorschriften, Normen und Richtlinien**

  Zweck dieser Prüfung ist es, festzustellen, ob der Inhalt des Modells mit einer Vorschrift, einem Standard, einer Richtlinie, einer regulatorischen Anforderung oder Ähnliches übereinstimmt. In diesem Zusammenhang wird in der Literatur auch von Konformität der Gestaltungsplanung hinsichtlich der geltenden Regelwerke gesprochen (Hjelseth, 2009a). Der Begriff Konformität bezeichnet allgemein die Übereinstimmung eines Konzepts oder Objekts mit den Normen eines gesellschaftlichen, inhaltlichen oder ethischen Kontextes (Duden, 2013). Im Mittelpunkt steht dabei die Einhaltung klar definierter oder in der Regel vorschriftsmäßiger Bauvorschriften. Die unterschiedlichen Darstellungsweisen sowie die Prüfung dieser Art von Kriterien ist zentraler Aspekt der vorliegenden Arbeit in Teil II. Daher soll an dieser Stelle nicht weiter ins Detail eingegangen werden.

  Beispiele: Regelungen aus den Landesbauordnungen der Bundesländer; internationale, europäische, nationale oder regionale Bauvorschriften zur Barrierefreiheit, Brandschutz oder Materialqualität

- **Spezifische (Bauherren-)Anforderungen**

  Bei der Gestaltungsplanung eines Bauwerksmodells geht es in erster Linie darum, die Anforderung und Funktionen, welche von dem Auftraggeber definiert wurden, zu erfüllen. Der Prozess der Erfüllung ist eine iterative Entwicklung der Planung hin zu einer immer detaillierteren Lösung. Diese Untergliederung in die einzelnen Planungsschritte wird in Deutschland auch von den Leistungsphasen der HOAI (Heinlein et al., 2013) widergespiegelt. Eine wesentliche Voraussetzung für die Einhaltung der gesetzten Ziele ist die regelmäßige Prüfung des aktuellen Planungsstandes, um sicherzustellen, dass die Planung den Anforderungen des Bauherren entspricht und diese auch im finanziellen Rahmen liegt.

Dabei sind die Anforderungen des Auftraggebers üblicherweise weitgehend formal in Form von Spezifikationen beschrieben. Beispiel für solche Bauherrenspezifikationen sind die BIM-Vorgaben der Deutschen Bahn (Deutsche Bahn, 2016) oder die Gestaltungsvorgaben von Krankenhäusern oder Gerichtsgebäuden der GSA (GSA, 2007).

Beispiele: Prüfung auf Einhaltung von Modellierungsrichtlinien; Raumprogramme; gestalterische Vorgaben, wie etwa Anforderungen an Prozessdistanzwege

- **Bürostandards, Best-Practices, Erfahrung- & Fachwissen**

  Neben formalen Anforderungen, die sich aus zentralen oder projektspezifischen regulatorischen Dokumenten oder Anforderungskatalogen ergeben, gibt es fachspezifisches, bürointernes oder aber individuelles Fachwissen, aus welchem sich ebenfalls qualitative Anforderungen an die Modellinhalte ergeben.

  Als Beispiel können an dieser Stelle die erweiterte Kollisionsprüfung und auch die geometrisch-räumlichen Anforderungen zwischen Bauteilen vorgestellt werden. Insbesondere bei großen Projekten bzw. Projektausmaßen und Teilmodellen kann es bei einer Prüfung der harten Kollisionen zahlreichen festgestellten Problemen kommen. Dieses bringt die Herausforderung mit sich, dass die vielen Probleme nicht mehr vernünftig von den jeweiligen verantwortlichen Planungsingenieuren gepflegt werden können.

  Hier gilt, dass die Anzahl der betrachteten Kollisionen bereits erheblich reduziert werden kann, indem lediglich maßgebliche Komponenten betrachtet werden, die zu wirklichen Problemen bei der Planung führen. Die Feststellung einer vorhandenen Kollision sagt per se nichts darüber aus, wie kritisch die vorliegende Überschneidung für ein betroffenes Gewerk letztlich ist. Maßgebliche Probleme sind in diesem Zusammenhang also nur Probleme bzw. Fehler, die auf planerische Fehler zurückzuführen sind und die Mach- und Baubarkeit beeinträchtigen.

  Daher sollte vor der Anwendung einer Kollisionsprüfung beachtet werden, welche Komponenten gegeneinander geprüft und wie viele Probleme aus einer Überschneidung von mehreren Komponenten generiert werden. So können in Abhängigkeit der betroffenen Komponententypen die Überschneidungen beispielsweise entsprechend kategorisiert werden, was wertvolle zusätzliche Informationen für den anschließenden Entscheidungsprozess bringt, beispielsweise wer für die Behebung des Problems verantwortlich ist. Auf der anderen Seite können Überschneidungen zwischen spezifizierten Komponententypen als irrelevant gekennzeichnet werden, sodass diese bei einer Entscheidungsfindung übergangen werden können. Ein Beispiel wäre an dieser Stelle der Trockenbau, welcher in der Regel keine großen Probleme bei Überschneidungen mit sich bringt, da Trockenbauteile keine tragende Funktion besitzen und auf der Baustelle sehr schnell und ohne großen Aufwand angepasst werden können.

  Neben den Überschneidungen muss außerdem geprüft werden, ob Komponenten ein spezifisches räumlich-geometrisches Verhältnis zueinander haben. Ein Beispiel hierfür ist die Prüfung, ob bewegliche Bauteile voll funktionsfähig sind. Wenn der Freiraum vor einem Fenster oder einer Tür nicht ausreichend gegeben ist, dann lässt sich das Fenster gegebenenfalls nicht öffnen und ist somit unbrauchbar. Ähnliche Anforderungen lassen sich für jegliche Komponenten finden, welche einen

operativen Freiraum benötigen. Im Umkehrschluss lassen sich allerdings auch Anforderungen an erforderliche Berührungsflächen, z. B. für Anschlüsse und Mindestauflageflächen, definieren.

Für die Definition dieser unterschiedlichen Anforderungen zwischen jeweiligen Komponenten-Gruppe kann mit Hilfe von geometrisch-topologischen Operatoren mit Parametern, z. B. für Toleranzen, beschrieben werden (Borrmann, 2007; Borrmann und Rank, 2009; Daum und Borrmann, 2014).

Beispiele: Anforderungen an die Baubarkeit, Berücksichtigung von temporären Bauzuständen oder Objekten, wie z.B. Schalung und Traggerüst; Erweiterte Kollisionsprüfung; Überprüfung von Freiräumen, Überprüfung von notwendigen Anschlüssen oder Abständen zwischen Komponenten

Gemäß der eingeführten Kategorien können weitere qualitative Aspekte oder Kriterien eingeführt werden. Die umfassende Modellqualität eines Planungsstandes setzt sich schließlich projektphasen- bzw. projektspezifisch aus einer Auswahl der aufgeführten Qualitätskriterien zusammen. Folglich lassen sich die Kriterien weiterhin in die folgenden Kategorien unterteilen:

Allgemeingültig/Projekt(-phasen)-spezifisch: Ein allgemeingültiges Kriterium muss immer von jedem Modellinhalt erfüllt sein. Ein Projekt(-phasen)-spezifisch Kriterium hingegen gilt nur für ein einzelnes Projekt oder aber eine einzelne Projektphase.

Zeitlich unveränderlich/veränderlich: Ein zeitlich unveränderliches Kriterium muss immer, auch über die einzelnen Entwicklungsstadien eines Modells hinweg, erfüllt sein. Im Gegensatz hierzu gibt es auch Kriterien, welche nur für einen bestimmten Zeitraum oder aber eine einzelne Projektphase gelten.

Objektiv/subjektiv: Objektive Kriterien lassen sich an Hand von formalen Anforderungen beschreiben. Die Betrachtung erfolgt anhand einer neutralen Bewertung. Die Bewertung von subjektiven Kriterien liegt hingegen im Auge des Betrachters und bedarf der Einschätzung eines Menschen. Diese Kriterien lassen sich nicht formal beschreiben.

3.3 Prüfung digitaler Bauwerksmodelle

Bei einer Prüfung des Modells wird die Leistungsfähigkeit mit Hilfe von wohldefinierten Fragestellungen festgestellt. Im Zusammenhang von digitalen Bauwerksmodellen dient eine solche Prüfung also dazu, die Leistungsfähigkeit des Modells und dessen Informationsgehalts zu ermitteln. Bei einer Modellprüfung wird dabei ausschließlich auf Modellinhalte zugegriffen, diese werden aber nicht verändert bzw. manipuliert. Die abgefragten Informationen werden anhand des in einer Prüfumgebung vorliegenden Regelwissens geprüft und schließlich wird ein Bericht über die Einhaltung oder Nicht-Einhaltung der Bestimmungen ausgegeben. Das Regelwissen ergibt sich aus einer Vielzahl von einzelnen Regeln. Unter einer Regel versteht man eine aus bestimmten Gesetzmäßigkeiten abgeleitete, aus Erfahrungen und Erkenntnissen gewonnene, in Übereinkunft festgelegte, für einen jeweiligen Bereich verbindlich geltende Regel (Duden, 2013). Die Ergebnisse einer Modellprüfung werden entsprechend für eine Teilmenge eines Modells (z. B. einzelne Objekte oder aber Objekte einer Klassifikation), einzelne Fachmodelle oder aber das gesamte Modell mit Ergebnisparametern festgehalten:

- ***bestanden*** für bestandene Regeln

- ***nicht bestanden*** für nicht-bestandene Regeln
- ***Warnung*** bei fehlenden Angaben
- ***unbekannt*** für nicht verifizierbare Fehler; die Prüfung konnte kein eindeutiges Ergebnis erzeugen

Der Anwender einer Modellprüfung kann anschließend anhand des Berichts weitere Schritte zur vollständigen Einhaltung der Vorschriften ergreifen und vorhandene Fehler oder Warnungen manuell nachbearbeiten. Hierzu gehört insbesondere eine menschliche Bewertung der Schwere eines Fehlers im Blick darauf, ob ein identifizierter Fehler in der Folge zu einer Nachbesserung führt oder ob dieser akzeptiert werden kann. Handelt es sich bei der Fehlerprüfung um einen automatisierten Prozess, wird auf diese Weise gewährleistet, dass jeder Fehler der Beurteilung eines Menschen unterliegt und so die Verantwortung eindeutig geregelt ist. Als Beispiel für einen nicht-relevanten Fehler sei im Hochbau eine Kollision zwischen einem TGA-Element und einer Trockenbauwand genannt. Auch wenn hier eine (automatisiert identifizierte) Kollision vorliegt, so kann diese in der Praxis sehr schnell von den Bauarbeitern behoben werden, und daher müssen keine Anpassungen im Modell erfolgen. Die Behebung der Kollision wird in diesem Falle direkt auf der Baustelle umgesetzt und ein Mehraufwand durch eine zusätzliche Fehlerverfolgung ist nicht notwendig.

Für die Modellprüfung selbst können je nach Zweck bzw. Verwendung des Modells oder aber des zu prüfenden qualitativen Kriteriums verschiedene Methoden gewählt werden. Kommt ein Bauwerksmodell für unterschiedliche Anwendungsfälle zum Einsatz, so muss dieses auch hinsichtlich der Eignung für diese hin untersucht werden. Die Regelprüfung muss weite Bereiche der AEC-Industrie und ihrer Praktiken adressieren, um so schließlich den gesamten Lebenszyklus einer baulichen Anlage abzudecken. Diese Anforderungen gehen in der Praxis über eine reine Einhaltung von Bauvorschriften hinaus (Solihin et al., 2016) und so sind sehr spezifische Arten der Regelprüfung bspw. spezifischer Kundenanforderungen oder aber Anforderungen für bestimmte Gebäudetypen üblich (Eastman et al., 2009).

Im Kontext der Modellprüfung (engl. *model checking*) tauchen in der Literatur häufig unterschiedliche Begrifflichkeiten auf, die verschiedene Arten und Formen der Modellprüfung beschreiben. So wird unter anderem der Begriff Validierung (engl. *model validation*), Verifikation (engl. *model verification*) verwendet. An dieser Stelle soll die Bedeutung der jeweiligen Begriffe voneinander abgegrenzt werden (Duden, 2013):

Bei einer **Verifizierung** (lat. *veritas: Wahrheit* und *facere: machen*) wird geprüft, ob ein Produkt bei seiner Entwicklung mit den spezifizierten Anforderungen, welche in der Regel in einem Pflichtenheft festgehalten werden, übereinstimmt.

Die **Validierung** (lat. *validus: wirksam*) stellt eine Art Feldexperiment dar, bei welchem kontrolliert wird, ob festgelegte Nutzungsziele erfüllt sind. Es werden also Anforderungen des Kunden, welche die Tauglichkeit beschreiben, geprüft.

In der Literatur werden verschiedene Prüfmethoden voneinander unterschieden (Eastman et al., 2009; Hjelseth, 2010b; Hjelseth und Nisbet, 2010; Solihin, 2016):

Die **Inhaltliche Modellprüfung** ist die einfachste Form der Prüfung und bezieht sich auf die Modellinhalte und deren spezifischen Zweck, für welchen diese erstellt und geliefert wurden. Wie bereits in Abschnitt 3.2 beschrieben, bezieht sich der Begriff Modellinhalt dabei auf die unterschiedlichen

Ebenen des Datenmodells, welche dort behandelt wurden. Die Inhalte werden hinsichtlich Existenz, korrekte Datentypen und Gültigkeit der Inhalte geprüft. Dabei kann die Gültigkeit als ein exakter Wert, ein Wertebereich oder aber eine Enumeration gültiger Werte definiert sein. Das Ergebnis einer solchen Prüfung ist in der Regel eine Auflistung, welche Inhalte im Modell gemäß den Anforderungen fehlen bzw. falsch oder überflüssig definiert sind. Für die technische Umsetzung dieser Prüfung bietet sich insbesondere die Definition von Filterregeln an. Mit einem solchen Filter wird eine Selektion, also die Auswahl von Datenobjekten aus einer Datenmenge, beschrieben. Mit Hilfe dieser Prüfung können insbesondere die Qualitätskriterien der semantischen und der geometrischen Eignung beschrieben werden.

Beispiel: Prüfung der Modellinhalte hinsichtlich der LOD-Anforderungen, Kollisionsfreiheit im Fachmodell

Die **Konformitätsprüfung bzw. Modellvalidierung** ist die häufigste Anwendung der Modellprüfung. Diese Art der Prüfung basiert auf dem Vergleich des Modells mit vordefinierten Kriterien, welche darauf beruhen, dass ein vollständiger und korrekter Modellinhalt (siehe inhaltliche Modellprüfung zuvor) gegeben ist. Der Zweck einer validierenden Modellprüfung ist es festzustellen, ob der Inhalt des Modells mit einer regulatorischen Anforderung übereinstimmt. Die Validierung der Modellprüfung wird anhand definierter Kriterien durchgeführt und führt zu einer *True/False*-Antwort.

Beispiele: Die minimale zulässige Türbreite in einem Bauwerk sollte min. 80 cm betragen; entlang einer Fluchtroute muss ein Freiraum-Korridor von 1,50 m Breite vorhanden sein

Die **Modellkoordination** bezieht sich explizit auf die Prüfung der Konsistenz eines Koordinationsmodells. Das bedeutet, dass bei dieser Prüfung mindestens zwei Fachmodelle betrachtet werden, welche in ein einzelnes Koordinationsmodell überführt werden. Bei der Koordination muss sowohl auf der semantischen, der geometrischen als auch semantisch-geometrischen Ebene die Konsistenz der Informationen geprüft werden. Klassischerweise wird bei der Koordination insbesondere die Kollisionsfreiheit zwischen zwei Modellen unterschiedlicher Disziplinen geprüft.

Beispiel: Kollisionsprüfung zwischen Architektur und TGA-Modell

Das **Entscheidungsunterstützungssystem** ist eine Weiterentwicklung der Konformitätsprüfung, welche als Ergebnis eine Handlungsweisung bzw. -empfehlung ausspricht. Ziel dieser Prüfung ist es, den Fachplaner anzuleiten, eine Lösungsvariante unter bestimmten vorgegebenen Vorschlägen auszuwählen oder verschiedene Möglichkeiten von Lösungsvarianten anzubieten. Diese Form der Prüfung basiert auf zwei Elementen: den Regeln, die eine entsprechende Situationen identifizieren, in welchen Probleme auftreten, sowie der Entwicklung und Darstellung von möglichen Lösungsvarianten. Damit diese Optionen entwickelt werden können, müssen die Sicherheits- und andere Regeln mit möglichen programmierten Korrekturmaßnahmen, also dem

gestaltungsplanerischen Wissen, in der Prüfmethode implementiert sein. Daher kann in diesem Fall auch von einem Entscheidungsunterstützungssystem gesprochen werden. Bisher wird dieses Konzept in der Bauindustrie noch nicht verwendet.

Beispiel: Bei der Prüfung eines Bauwerksmodell hinsichtlich der Barrierefreiheit wird festgestellt, dass die Zugänglichkeit für Rollstuhlfahrer nicht gegeben ist. Daher wird von dem Entscheidungsunterstützungssystem die Planung einer Rampe vorgeschlagen, damit die Zugänglichkeit erfüllt wird.

3.4 Festlegung einer Strategie für die Qualitätssicherstellung

In den vorangegangenen Abschnitten wurden die Grundlagen für die Definition qualitativer Kriterien sowie Konzepte für die Prüfung digitaler Bauwerksmodelle behandelt. Um diese Elemente nun in einem Bauprojekt zusammenzubringen, bietet es sich an, eine Strategie für die Qualitätssicherstellung zu implementieren. Auf diese Weise wird eine möglichst hohe Qualität der Modellinhalte im gesamten Projekt zu einem Kernziel.

Für das Verfolgen einer solchen Qualitätssicherungsstrategie dienen die Vorgaben, welche ein AG in den vertraglichen Dokumenten definiert, als wesentliches Mittel für die Umsetzung. Allgemein empfiehlt es sich, dass ein AG den AN mittels einer formalen Beschreibung zur Durchführung von Maßnahmen für die Qualitätssicherung vertraglich, z.B. als Teil der AIA, verpflichtet. Somit wird die Qualitätssicherung eine zentrale Aufgabe für den AN selbst. Die Ergebnisse einer entsprechenden Prüfung können darüber hinaus in Form von Prüfberichten eingefordert werden und der AN steht somit in einer Bringschuld.

Neben der allgemeinen Verpflichtung muss vom AG ein Anforderungskatalog mit spezifischen qualitativen Kriterien, wie diese in Abschnitt 3.2 vorgestellt wurden, definiert werden. Darüber hinaus sollte bei der Beschreibung auch ein zeitlicher Rahmen bzw. Takt für die Qualitätssicherung vorgegeben werden. Eine Prüfung kann beispielsweise in festen Zeitabständen oder aber an Meilensteine gebunden vorgegebenen werden. Üblicherweise wird die Regelung vereinbart, dass eine Prüfung jeweils bei Modellübergabe stattfindet.

3.5 Zusammenfassung

Die Verwendung von digitalen Bauwerksmodellen kann nur dann einen Mehrwert bieten, wenn sichergestellt ist, dass die enthaltenen Informationen den Ansprüchen an die inhaltliche Qualität gerecht werden. Nicht erkannte Fehler und Unzulänglichkeiten in Modellen führen unweigerlich zu Fehlern, da diese gemäß der BIM-Methode als zentrale Datengrundlage für alle Folgeprozesse gelten. Daher ist die Qualität der verwendeten Modelle von höchster Priorität.

Ein zentrales Projektziel muss daher sein, dass die verwendeten Modelle in Abhängigkeit von dem beabsichtigen Zweck und somit Informationsgehalt und Detaillierungsgrad, verlässliche Informationen enthalten. Um festzustellen, ob die Informationen verlässlich sind, muss in den Projektabläufen eine Qualitätsprüfung und -sicherung verankert werden. Hierzu gehört, dass klar definiert wird, welcher

Projektbeteiligte zu welchem Zeitpunkt welches Modell auf welche Kriterien hin zu prüfen hat und wie die erhaltenen Ergebnisse weiterverarbeitet werden sollen.

Zu einem definierten Zeitpunkt muss ein Teil- oder Koordinationsmodell eine erforderliche Modellqualität definieren. Eine allgemeingültige Definition der Modellqualität fällt schwer, da sich die Qualität auf die verschiedenen Ebenen und Aspekte der enthaltenen Daten und Informationen bezieht, welche in einem Bauwerksmodell beschrieben werden. Je nach beabsichtigten Zweck und Anwendungsfall des jeweiligen Modells, haben die Informationen auf diesen unterschiedlichen Ebenen einen unterschiedlich wichtigen Stellenwert. Grundlegend kann zwischen einer datentechnischen, einer inhaltlichen und einer gestaltungsplanerischen Qualitätsebene unterschieden werden. In jeder dieser Ebenen können eine Vielzahl von qualitativen Aspekten identifiziert werden.

Im Projektverlauf muss bei der Übergabe von Modellinhalten nun eine Auswahl bzw. Komposition der Kriterien definiert werden, welche zu dem gegebenen Zeitpunkt die Anforderungen hinsichtlich der Qualität für den spezifischen Anwendungszweck beschreibt. Die Prüfung selbst kann auf unterschiedliche Art und Weise erfolgen. Neben einer rein inhaltlichen Prüfung ist unter anderem die Koordination von Modellen, eine Prüfung der Konformität hinsichtlich geltender Anforderungen und ein Entscheidungsunterstützungssystem zu unterscheiden.

Um das allgemeine Qualitätsniveau der einzelnen Modelle sowie des übergreifenden Koordinationsmodells kontinuierlich auf einem hohen Level zu halten, bietet sich an in dem Projekt eine Qualitätssicherungsstrategie festzulegen, welche zum Ziel hat, dass sich alle Projektbeteiligten zu einer konsequenten Qualitätssicherung verpflichten.

Teil II

Automatisierte Konformitätsüberprüfung auf Basis von BIM

4 Recht und Normen im Bauwesen

4.1 Einführung

Bauvorschriften gelten als eine der ersten Formen von schriftlich niedergelegtem Regelwissen. Das früheste bekannte schriftliche Baurecht sind die Texte von König Hammurabi von Babylonien, welcher um 2250 v. Chr. lebte. In diesen Texten werden bereits verschiedene Sicherheitsaspekte von Bauwerken beschrieben (Harper und Godbey, 1903). Ähnliche Vorschriften lassen sich auch im Deuteronomium, einem Buch des *Alten Testamentes*, finden. Hier wird unter anderem beschrieben, dass auf dem begehbaren Dach eines Hauses eine Zinne oder Brüstung angebracht werden muss, um zu verhindern, dass Menschen herabstürzen (Deuteronomium 22:8).

Wie bereits zuvor in Abschnitt 3.2 beschrieben, handelt es sich bei Bauvorschriften um Regelwissen, welches aus Erfahrungen, Versuchen, Fehlern oder aber Unfällen resultiert. Unfälle oder aber Katastrophen haben dazu geführt, dass Regeln festgelegt werden, um diese künftig zu vermeiden.

Darüber hinaus gelten Vorschriften als gesellschaftliche Vereinbarung bei der Gestaltung, Errichtung und beim Betrieb von Bauwerken, um eine sichere und verbindliche Ordnung als die Basis für ein gutes Miteinander herzustellen. Mit der Errichtung von Bauwerken wird erheblicher Einfluss auf die Umwelt genommen. Daher wird in diesem Zusammenhang auch von der gebauten Umwelt bzw. der zweiten Umwelt gesprochen (Kromrey, 1981). Durch die Festlegung dieser Eigenschaften und Regeln wird vom Staat die zentrale Aufgabe, die Unversehrtheit der Menschen zu gewährleisten, erfüllt.

Die Vielzahl an Randbedingungen und Unbekannten (vgl. Kapitel 1), welche für ein Bauvorhaben in Abhängigkeit von Typ und Standort gelten, hat zur Folge, dass es auch eine Vielzahl an Vorschriften gibt, deren Inhalte in Teilen sehr komplex sein können (Solihin, 2016). Die Zahl von Normen und Vorschriften für das Bauwesen ist in den letzten Jahren sehr stark gestiegen. In Deutschland sind etwa 3000 Normen bzw. Vorschriften für das Bauen relevant, was einen historischen Spitzenwert darstellt (Matzig, 2018). Dabei handelt es sich jedoch nicht um ein explizit deutsches Problem, denn auch in anderen Ländern gibt es eine enorme Anzahl von länderspezifischen nationalen und regionalen Normen (Hahmann, 2015).

4.2 Quellen für regulatorische Anforderungen im Bauwesen

Folgend sollen die wesentlichen Quellen, aus welchen sich die regulatorische Anforderungen für ein Bauprojekt ableiten, beschrieben werden, um einen Überblick über Art und Inhalt verschiedener Vorschriften im Bauwesen zu geben. Dabei orientieren sich die Ausführungen maßgeblich an dem deutschen Bauwesen und der deutschen Rechtslage. Eine Anwendung auf andere nationale oder gar internationale Rechtslagen ist also nur eingeschränkt möglich.

4.2.1 Deutsches Baurecht

Grundlegende Anforderungen an die gebaute Umwelt lassen sich aus dem geltendem Baurecht ableiten. Der Begriff Baurecht umfasst sämtliche Rechtsnormen und -vorschriften, also gesetzlichen Regelungen, die sich auf das Bauwesen beziehen (Duden, 2013; Vismann, 2017). Ein wesentlicher Unterschied im Vergleich zu sozialen oder moralischen Normen ist, dass Rechtsnormen mit Befehl und Zwang im Zuge der Vollstreckung auch gegen den Willen des Normadressaten durchsetzbar sind. Zudem zählen Rechtsnormen zu dem positiven Recht, was bedeutet, dass diese im Gegensatz zu Naturregeln anthropogen sind, also vom Menschen geschaffen wurden. Rechtsnormen formulieren sogenannte Sollensanordnung, also Ge- bzw. Verbote, nach welchen sich die Normunterworfenen zu richten haben. Diese Anordnungen lassen sich in vier Kategorien unterteilen (Röhl und Röhl, 2008):

Verbot: statuiert eine Unterlassungspflicht

Gebot: statuiert eine Handlungspflicht

Erlaubnis: statuiert ein Handlungsrecht

Freistellung: statuiert ein Unterlassungsrecht

Das Baurecht wird in Deutschland in privates und öffentliches Baurecht unterteilt. Das private Baurecht bezieht sich auf die Rechtsnormen des Zivil-, Grundeigentum und Nachbarrechts und beinhaltet insbesondere die Werkverträge, welche zur Durchführung eines Bauvorhabens geschlossen wurden. Wesentliche Rechtsquelle für die Anwendung des privaten Baurechts ist das Bürgerliches Gesetzbuch (BGB) (Köhler, 1995).

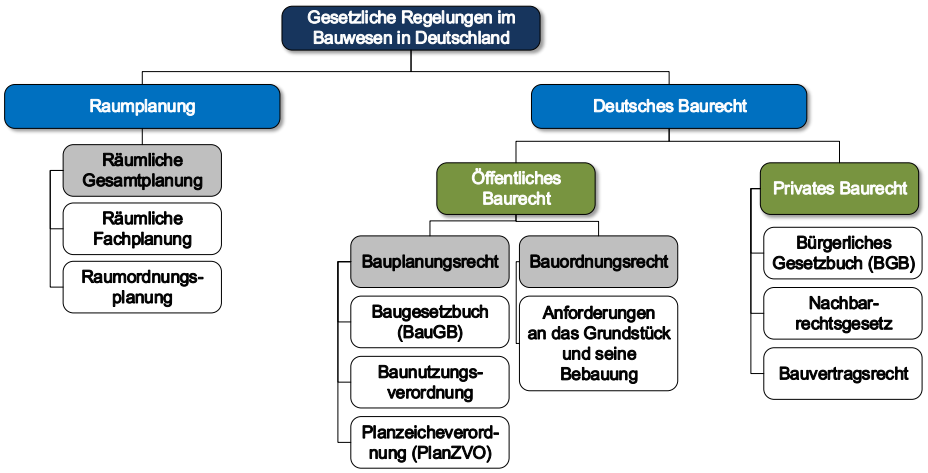


Abbildung 4.1: Überblick zum Deutschen Baurecht (Vitek und Vitek, 2017)

Das öffentliche Baurecht hingegen regelt sämtliche Bereiche, welche die Öffentlichkeit betreffen. Hierzu gehören insbesondere alle Fragestellungen zum Bauplanungsrecht (z. B. Bebaubarkeit von Grundstücken), Bauordnungsrecht (z. B. Sicherheits- und Gestaltungsvorschriften für einzelne Bauvorhaben). Im öffentlichen Baurecht werden unter dem Bauordnungsrecht die Anforderungen an das Grundstück und

seine Bebauung (z. B. Abstandsflächen, verkehrsmäßige Erschließung, Zahl der notwendigen Stellplätze), an einzelne Räume, Wohnungen und besondere Anlagen (zum Beispiel Stellplätze) sowie grundsätzliche Anforderungen an die Ausführung baulicher Anlagen und der wichtigsten Gebäudeteile (zum Beispiel Standsicherheit, Verkehrssicherheit, Brandschutz) zusammengefasst (Albert und Schneider, 2016; Gräber-Seißinger, 2015). Diese Anforderungen sind grundsätzlich in den Landesbauordnungen der Bundesländer, welche auf der Musterbauordnung (MBO) des Bundes basieren, zusammengefasst (Messer und Ammon, 2015). Die MBO fasst als Grundlage für die länderspezifischen Ordnungen diverse Mindeststandards für den Gebäudebau zusammen und gilt somit als ein wesentlicher Bestandteil des öffentlichen Baurechts. Daneben gibt es weitere Einzelregelungen für Fachbereiche wie beispielsweise in Garagen-, Feuerungs-, Versammlungsstätten- und Verfahrensverordnungen.

Baumaßnahmen nach dem öffentlichen Baurecht und der auf Grund der jeweiligen Bauordnung erlassenen Anordnungen werden von Bauaufsichtsbehörden, welche zu den Verwaltungsbehörden zählen, kontrolliert (Messer und Ammon, 2015). Diese Institutionen sind auf diese Weise mit einem Weisungsrecht ausgestattet und können die Einstellung von Bauarbeiten anordnen, insbesondere wenn

- die Ausführung eines genehmigungsbedürftigen Bauvorhabens entgegen den Vorschriften der jeweiligen Landesbauordnung begonnen wurde.
- bei der Ausführung eines Bauvorhabens von den genehmigten Bauvorlagen abgewichen wird.
- gegen baurechtliche Vorschriften verstoßen wird.

Welche Institution diese Position einnimmt und welche Aufgaben dieser zugeordnet sind, wird im Allgemeinen in den Landesbauordnungen geregelt. Üblicherweise ist festgelegt, dass die Landkreise und kreisfreien Städte als untere Bauaufsichtsbehörden, die Regierungspräsidien als obere Bauaufsichtsbehörden und das zuständige Ministerium als oberste Bauaufsichtsbehörde auftreten. Die unteren Bauaufsichtsbehörden sind dafür zuständig, dass die öffentlich-rechtlichen Vorschriften sowie Nutzung und Instandhaltung von Anlagen umgesetzt werden.

Für die Planung und Durchführung von Bauprojekten sind in Deutschland aufseiten des Baurechts insbesondere die Regelungen der MBO, welche in leicht angepasster Form in den Ländern der Bundesrepublik Deutschland, den Landesbauordnungen, Anwendung findet, von besonderer Bedeutung. Dabei ist zu beachten, dass die MBO selbst keinen Rechtscharakter hat. Diesen erhält diese lediglich in Form der jeweiligen Landesbauordnungen (Deutsches Institut für Baunormung, 2017), z.B. in Form der bayerischen Bauordnung (Molodovsky, 2016). Die Anforderungen, welche in der MBO grundlegend enthalten sind, beziehen sich zum einen auf das bebaute Grundstück und zum anderen auf die bauliche Anlage selbst. Zu den Regelungen gehören unter anderem die Erschließung, die Art der baulichen Nutzung, die Abstandsflächen, der Nachbarschutz, das gesunde Wohnen (Belichtung, Raumhöhen, Schall-, Kälte- und Wärmeschutz), die Feuerwiderstandsklassen von Bauteilen, die Eignung von Bauprodukten, die Standsicherheit, die Flucht- und Rettungswege sowie die Sicherheit von Baustelle und Bauwerk (Messer und Ammon, 2015). Als Beispiel für eine solche Anforderung kann an dieser Stelle §6(5) MBO herangezogen werden (Messer und Ammon, 2015):

> *Die Tiefe der Abstandsflächen beträgt 0,4 H, mindestens 3 m. In Gewerbe und Industriegebieten genügt eine Tiefe von 0,2 H, mindestens 3 m. Vor den Außenwänden von Wohngebäuden der Gebäudeklassen 1 und 2 mit nicht mehr als drei oberirdischen Geschossen genügt als Tiefe der Abstandsfläche 3 m.*

Werden von einer städtebaulichen Satzung oder einer Satzung nach § 86 Außenwände zugelassen oder vorgeschrieben, vor denen Abstandsflächen größerer oder geringerer Tiefe als nach den Sätzen 1 bis 3 liegen müssten, finden die Sätze 1 bis 3 keine Anwendung, es sei denn, die Satzung ordnet die Geltung dieser Vorschriften an.

4.2.2 Normen

Der Begriff Norm leitet sich von dem lateinischen Begriff *norma* (dt. *Regel*) ab und beschreibt in der Deutschen Sprache die Festlegung von Regeln für wiederkehrende Wertschöpfung und die damit verbundenen Prozesse (DIN, 2007-03). Durch das Aufstellen von Normen kann sichergestellt werden, dass Produkte oder Prozesse trotz verschiedener Anwendungsorte oder beteiligter Personenkreise bestimmte Festlegungen in definierten Geltungsbereichen einhalten. In (DIN, 2007-03) wird die Normung, der Prozess der Erstellung von Normen, welcher auch als Normungsarbeit bezeichnet wird, als *"Tätigkeit zur Erstellung von Festlegungen für die allgemeine und wiederkehrende Anwendung, die auf aktuelle oder absehbare Probleme Bezug haben und die Erzielung eines optimalen Ordnungsgrades in einem gegebenen Zusammenhang anstreben"* beschrieben. Charakterisiert wird eine Norm durch ihren jeweiligen Normungsgegenstand und das Normungsgebiet. Der Normungsgegenstand beschreibt das Thema, welches genormt werden soll, das Normungsgebiet beschreibt hingegen die Gruppe verwandter Normungsgegenstände. Wesentliche Ziele hinter der Festlegung einer Norm sind (DIN, 2007-03)

Zweckdienlichkeit: Fähigkeit eines Erzeugnisses, eines Verfahrens oder einer Dienstleistung, einen bestimmten Zweck unter festgelegten Bedingungen zu erfüllen

Kompatibilität/Verträglichkeit: Eignung von Produkten, Prozessen oder Dienstleistungen, die gemeinsam unter bestimmten Bedingungen verwendet werden können, um maßgebliche Anforderungen ohne unannehmbare gegenseitige Auswirkungen zu erfüllen

Austauschbarkeit: Fähigkeit eines Produktes, eines Prozesses oder einer Dienstleistung, anstelle eines (einer) anderen verwendet zu werden, um dieselben Anforderungen zu erfüllen

Verminderung der Vielfalt: Auswahl der optimalen Anzahl von Größen oder Arten von Produkten, Prozessen oder Dienstleistungen, um den bestehenden Bedürfnissen zu entsprechen

Sicherheit: Freiheit von unvertretbaren Schadensrisiken

Umweltschutz: Schutz der Umwelt vor unvertretbaren Schädigungen durch Auswirkungen und Betriebsabläufe von Produkten, Prozessen und Dienstleistungen

Schutz des Produktes: Schutz eines Produktes gegen klimatische oder andere schädliche Einflüsse während seiner Benutzung, seines Transportes oder Lagerung

Im Allgemeinen wird durch Normen sichergestellt, dass sich die entstehenden Produkte, Prozesse oder Dienstleistungen besser und sicherer für ihren beabsichtigten Zweck eigenen. Aus wirtschaftlicher Sicht bringen Normen zudem den Vorteil mit sich, dass der Austausch von Waren und Dienstleistungen über Landesgrenzen hinweg durch die erhöhte Kompatibilität erleichtert wird (Hartlieb, 2000).

Nach deutschem Rechtsverständnis ist die Anwendung von Normen grundsätzlich freiwillig. Im Gegensatz zu Gesetzen sind Normen also nicht bindend. Normen erlangen erst dann Rechtsverbindlichkeit, wenn Gesetze auf diese verweisen oder Vertragspartner diese zum Inhalt eines Vertrages machen.

Allerdings werden Normen auch in Fällen, in welchen diese keine Rechtsverbindlichkeit erlangt haben, als Entscheidungshilfe herangezogen. So werden Normen und technische Regeln insbesondere von Gerichten in Verfahren auf dem Gebiet des Mängelgewährleistungsrechts sowie des Delikts- und Produkthaftungsrechts als Referenz verwendet, um zu beurteilen, ob die allgemein anerkannten Regeln der Technik beachtet und somit die verkehrsübliche Sorgfalt eingehalten wurde. Normen können somit als Empfehlungen angesehen werden, deren Einhaltung für Unternehmer im Hinblick auf mögliche Haftungsfälle Rechtssicherheit gibt (DIN, 2016).

Zentraler Kern einer verabschiedeten Norm ist der Konsens, also die allgemeine Zustimmung aller betroffenen Interessengruppen für einen Normgegenstand. Der Konsens ist durch das Fehlen aufrechterhaltenen Widerspruchs gegen wesentliche Inhalte seitens der betroffenen Interessen gekennzeichnet. Daraus leitet sich auch ab, dass der Versuch, die Gesichtspunkte aller betroffenen Parteien zu berücksichtigen und auftretende Gegenargumente auszuräumen, ein wesentlicher Bestandteil der Normierungsarbeit ist. Der resultierende Konsens wird von einer anerkannten Institution in Form eines normativen Dokumentes festgehalten.

Der Begriff *Norm* lässt sich als Überbegriff in die folgenden Kategorien unterteilen (DIN, 2007-03):

Vornorm: Dokument, das von einer normenschaffenden Institution vorläufig angenommen wurde und der Öffentlichkeit zugänglich gemacht wird, damit durch seine Anwendung die notwendige Erfahrung gesammelt wird, die dann die Grundlage einer Norm bildet.

Technische Spezifikation (z. B. DIN SPEC, PAS): Dokument, das technische Anforderungen festlegt, die von einem Erzeugnis, einem Verfahren oder einer Dienstleistung zu erfüllen sind.

Anleitung für die Praxis: Dokument, das Praktiken oder Verfahren für den Entwurf, die Herstellung, die Montage, die Wartung oder Verwendung von Geräten, Konstruktionen oder Produkten empfiehlt.

Vorschrift: Dokument, das von einer Behörde erstellt wird und verbindliche rechtliche Festlegungen trifft.

Technische Vorschrift: Vorschrift, die technische Anforderungen entweder direkt oder durch Bezugnahme auf Normen oder Einverleibung des Inhaltes einer Norm, technischen Beschreibung oder Anleitung für die Praxis festlegt.

In Abhängigkeit des Normgegenstandes und des Geltungsbereichs sind unterschiedliche Institutionen oder Organisationen für das Erstellen und Herausgeben der Normen verantwortlich. In diesem Zusammenhang wird auch von einer normschaffenden Institution gesprochen. Je nach Zuständigkeit für den jeweiligen Normungsgegenstand haben die Dokumente einen Geltungsbereich auf internationaler, nationaler, regionaler oder aber provinzieller Ebene (Zubke-von Thünen, 1999). Auf internationaler Ebene nimmt die Rolle der normschaffenden Institution insbesondere die Internationale Organisation für Normung (engl. *International Organization for Standardization*, ISO) ein. Die ISO hat ihren Sitz in Genf, Schweiz, und hat nach schweizerischem Recht die Rechtsform eines Vereins. In dieser Organisation werden wiederum viele nationale Vertretungen zusammengefasst. Das in Deutschland für die Normung zuständige Deutsche Institut für Normung e. V. (DIN) ist seit 1951 Mitglied der ISO stellvertretend für die Bundesrepublik Deutschland. Auf europäischer Ebene gibt es zudem das Europäische Komitee für Standardisierung (engl. *European Committee for Standardization*, CEN). Im Anhang der vorliegenden Arbeit ist in Tabelle A.1 eine Übersicht der wesentlichen Arten von Normen festgehalten.

4.2.3 Technische und auftraggeberspezifische Richtlinien und Anforderungen

Neben den normschaffenden Instituten gibt es auch nationale Gremien, wie etwa den Verein Deutscher Ingenieure e.V. (VDI), welche ebenfalls fachbereichbezogen Richtlinien entwickeln und diese als anerkannte Regeln der Technik zur Verfügung stellen. So wird in der Satzung des VDI explizit festgehalten, dass das Erstellen von technischen Regelwerken ein wesentlicher Zweck des Vereins ist (VDI, 2017-02). Der Erstellungsprozess in diesen Gremien gleicht den Prozessen in den normschaffenden Instituten sehr stark. Die resultierenden Richtlinien können wie auch die Normen als Referenz herangezogen werden.

Darüber hinaus gibt es weitere fachspezifische Richtlinien, welche für bestimmte Anwendungsbereiche von Auftraggebern, wie insbesondere der öffentlichen Hand, herausgegeben werden. Zu diesen gehören beispielsweise die *Richtlinien für das Aufstellen von Bauwerksentwürfen (RAB ING)*, welche vom Bundesministerium für Verkehr und digitale Infrastruktur (BMVI) für die Beschreibung von Anforderungen an Brückenbauwerke herausgegeben wurden. Ähnliche Anwendung finden sich auch bei Unternehmen wieder, welche ebenfalls eigene Richtlinien herausgeben, um spezifische Anwendungsbereiche einheitlich zu regeln. So gibt beispielsweise die Deutsche Bahn (2017) verschiedene Richtlinien für die Erstellung und den Betrieb der infrastrukturellen Anlagen heraus.

4.3 Beschreibung und Darstellung regulatorischer Anforderungen

Für die Wertschöpfungsprozesse des Bauwesens gelten Vorschriften, Normen und Richtlinien, wie diese in den Abschnitten zuvor beschrieben wurden. Bei dem Gestalten, dem Errichten und dem Bewirtschaften von Bauwerken dienen diese festgelegten Regeln dazu, Technikstandards, wie die Stand- und Betriebssicherheit, Materialqualität und nicht zuletzt die Sicherheit des Nutzers, zu sichern (Neufert und Kister, 2016; Prigge und Neufert, 1999; Schneider und Schlatter, 1996).

Neben den gesetzlichen Vorschriften ist für die Richtlinien und Regelungen im Deutschen Bauwesen satzungsmäßig der DIN-Normenausschuss Bau (NABau) als Organ des DIN zuständig. Der NABau hat die Aufgabe, alle Normungsvorschläge für das Bauwesen zu prüfen und, sofern ein berechtigtes Interesse besteht und die Finanzierung der damit verbundenen Kosten der Geschäftsstelle des NABau sichergestellt ist, zu bearbeiten. Er wirkt über die nationale Normung hinaus bei der europäischen und internationalen Normung seines Bereiches mit. Ferner hat er die Vorbereitung und Anwendung der Normen zu fördern (DIN, 2017).

Da sich die geltenden Regelwerke auf den gesamten Lebenszyklus eines Gebäudes und somit auch die zugehörigen Fachdisziplinen beziehen, ist die Anzahl der zu berücksichtigenden regulatorischen Anforderungen und normativen Dokumente in Abhängigkeit des Standorts groß. Um einen Überblick über die wesentlichen Anwendungsbereiche des Bauwesens zu bekommen, lassen sich die Fachbereiche und somit auch die normativen Dokumente wie folgt gliedern Schneider und Schlatter (1996):

- Architektur, Stadtplanung
- Baustoffe
- Baukonstruktionen
- Bautenschutz, Bauphysik
- Energetisches Bauen
- Konstruktiver Ingenieurbau
- Sanierung, Bauen im Bestand
- Wasser, Abwasser
- Verkehr

Eine Auswahl wesentlicher Normen für das Deutsche Bauwesen ist in Tabelle A.2 im Anhang der vorliegenden Arbeit zu finden.

Präskriptiver vs. leistungsbasierter Ansatz

Ein zentraler Aspekt bei der Beschreibung der Inhalte von Normen ist die Art und Weise, wie die beinhalteten Anforderungen beschrieben werden. Hierbei kann grundlegend zwischen einem präskriptiven bzw. vorschriftsmäßigen (engl. *prescriptive*) und einem leistungsbasierten (engl. *performance-based*) Ansatz zur Beschreibung der regulatorischen Anforderungen unterschieden werden (Becker, 2008; Foliente, 2000; Tavares, 2009).

Bei dem präskriptiven Ansatz wird eine akzeptable Lösung beschrieben, bei dem leistungsbasierten Ansatz hingegen die erforderliche Leistung. Der Unterschied zwischen den beiden Ansätzen wird deutlich, wenn man die unterschiedlichen Herangehensweisen betrachtet, um beispielsweise die Brandschutzsicherheit für ein Bauwerk einzufordern: nach dem präskriptiven Ansatz wird hier festgelegt, aus welchen Materialien das Tragwerk des Bauwerks bestehen darf oder nicht. Mit Herausgabe der Anforderungen wird somit antizipiert, welche Materialien sich hinsichtlich der Brandschutzsicherheit eignen und welche nicht. Bei einer funktionalen Leistungsbeschreibung hingegen wird lediglich die Forderung beschrieben, dass die Gebäudestruktur einem Brand lange genug standhalten soll, damit die Insassen sicher entkommen können. Hierbei wird allerdings nicht im Detail vorgeschrieben, welche Materialien verwendet werden müssen oder dürfen. Es muss hier also bei der Planung schließlich nachgewiesen werden, dass eine Auswahl von Materialien und Dimensionierung von Bauteilen das Ziel der Brandsicherheit erreicht. Bei dem normativen Ansatz hingegen reicht es aus die korrekte Materialauswahl nachzuweisen. Diese Art der Anforderung beschreibt somit die gewünschte Leistungsfähigkeit und das Ziel, welches mit der Formulierung der Anforderung beabsichtigt wurde. Ein konkreter Lösungsweg, wie dieses Ziel erreicht werden soll und muss, wird hier jedoch nicht beschrieben.

Da präskriptive Kriterien einen oder mehrere Lösungswege konkret vorgeben, sind diese Anforderungen für Auftraggeber und Planer einfach zu befolgen, für einen Dritten leicht zu überprüfen und für Bauaufsichtsbehörden einfach durchzusetzen. Auch für die im Rahmen dieser Arbeit verfolgte Überführung von Normen in computerverarbeitbare Algorithmen sind präskriptive Normen ebenfalls leichter zu handhaben. Allerdings gibt es auch einige grundsätzliche Schwierigkeiten im Zusammenhang mit der Anwendung normativer Kriterien, welche eine grundsätzliche Diskussion und ein erhöhtes Interesse an der Entwicklung leistungsbasierter Normen verstärkt haben.

Ein wesentliches Problem liegt darin, dass der präskriptive Ansatz als Innovationsbarriere verstanden werden muss. So wird beispielsweise eine Verwendung von verbesserten oder aber kostengünstigeren Produkten nur aufgrund dessen verhindert, weil ihre Verwendung nicht zulässig ist. Foliente (2000) führt an dieser Stelle als Beispiel auf, dass in den 1960er Jahren die Entwicklung von Schutzsystemen, welche erhebliche lebensbedrohliche Schäden bei Erdbeben hätten vorbeugen können, durch geltende präskriptive Anforderungen maßgeblich behindert und verzögert wurde und somit nicht zum Einsatz kamen.

Da der präskriptive Ansatz kein minimal erforderliches Leistungsniveau beschreibt, müssen Alternativlösungen gesondert aufgezeigt, beantragt, belegt und geprüft werden. Dieses bringt einen erheblichen Mehraufwand mit sich, was wiederum zu Mehrkosten führt. Bauwerke können so nur sehr schwer hinsichtlich der Kosten und Effizienz optimiert geplant und gebaut werden.

Nicht zuletzt ergibt sich ein weiteres Problem im internationalen Handel von Bauprodukten. Eine zentrale Herausforderung bei dem Handel von Produkten oder aber Dienstleistungen ist, dass zwei Länder bzw. Handelsregionen die Gleichwertigkeit der Qualitätskriterien der jeweils anderen Seite feststellen und nachweisen müssen. Erst durch eine solche Feststellung entsprechen die Lösungen gegenseitig dem impliziten Leistungsniveau des anderen Landes, was eine essenzielle Grundlage für einen Austausch ist. In der Praxis stellt sich die gegenseitige Feststellung auf Basis der präskriptiven Anforderungen sehr schwierig dar, da die regulatorischen Anforderungen national sehr stark voneinander abweichen. Nach den Regeln der *Welthandelsorganisation (WTO)* ist das zentrale Kriterium für die Gleichwertigkeit von Produkten oder Dienstleistungen allerdings die gleichwertige Funktion bzw. Leistung(-sfähigkeit) (Meacham, 2010). Auch hier stellen die präskriptiven Anforderungen also ein Hemmnis dar.

Im Gegensatz zu dem präskriptiven gibt der leistungsbasierte Ansatz eine spezifizierte und quantifizierte Beschreibung der erforderlichen Leistung und somit deutlich klarere Orientierungshilfen. Dadurch werden die formulierten Anforderungen auch eher der tatsächlich wachsende Komplexität der Bauaufgaben, beispielsweise bei den architektonischen Entwürfen, gerecht. Gleichzeitig wird der Einsatz neuer und möglicherweise kostengünstigerer Produkte oder Verfahren , die nachweislich das akzeptable Risikoniveau erfüllen oder überschreiten, nicht gehemmt. Tavares (2009) legt dar, dass die präskriptiven brasilianischen Brandschutzrichtlinien bei einer beträchtlichen Anzahl von Brandunfällen in Brasilien den Brandschutz nicht so gewährleistet haben, wie er hätte sein müssen. Hier wäre ein Einsatz von leistungsbasierten Anforderungen sinnvoll gewesen. Tatsächlich haben mittlerweile einige Länder wie Großbritannien, Schweden, Australien, Neuseeland, Kanada und Japan damit begonnen, ihre Brandschutzrichtlinien von einem vorschriftsmäßigen auf einen leistungsorientierten Ansatz umzustellen (Tavares, 2009).

Die genannten Vorteile des leistungsbasierten Ansatzes weisen darauf hin, dass in Zukunft vermehrt auf funktionale Beschreibungen bei der Definition von Anforderungen gesetzt wird, da so das System von den Beschränkungen des derzeitigen präskriptiven Ansatzes befreit wird (Becker, 2008; Foliente, 2000). Allerdings handelt es sich in Deutschland bei den gängigen regulatorischen Anforderungen aktuell noch immer weitgehend um Grund- bzw. Prüfnormen, welche präskriptive Anforderungen enthalten. Dieses bedeutet, dass die Normen deskriptiv einen Sachverhalt darstellen und anschließend die Einhaltung von Randbedingungen einfordern. Der Informationsgehalt einer solchen Vorschrift kann auf unterschiedliche Art und Weise dargestellt werden und reicht von einfach und klar strukturierten Tabellen mit Grenzwerten über graphische Darstellungen oder Gleichungen bis hin zu in Fließtext beschriebenen Sachverhalten. Im Folgenden sollen einige Beispiele für die verschiedenen Darstellungstypen vorgestellt werden:

Fließtexte

Fließtexte sind die übliche Darstellungsform in Vorschriften und Richtlinien. Für die Erstellung von internationalen Normen, technischen Spezifikationen und öffentlich verfügbaren Spezifikationen gelten Grundsätze und Regeln für den Aufbau und die schriftliche Abfassung (DIN, 2012-12). Zu diesen Grundsätzen gehört auch die Verwendung bestimmter modularer Hilfsverben, welche eine Notwendigkeit oder Möglichkeit ausdrücken sollen. In DIN (2012-12) ist eine Aufstellung zu den möglichen Anwendungen enthalten:

- Unbedingte Anforderungen
  - Gebote
  - Verbote
- Bedingte Anforderungen
- Erlaubnisse (Zulässigkeiten)
- Empfehlungen
- Feststellungen

Mit der Festlegung dieser Anwendungen wird beabsichtigt, dass dem Anwender genaue Informationen übermittelt und Unsicherheiten sowie mögliche Interpretationsspielräume gering gehalten werden (DIN, 2012-12; Mattiuzzo und Miesner, 2016). In Tabelle 4.1 ist eine Kategorisierung der jeweiligen modularen Hilfsverben für die Formulierung bestimmter Ausdrücke von Normeninhalten aufgelistet.

Tabelle 4.1: Kategorisierung modularer Hilfsverben für die Formulierung von Normeninhalten (Mattiuzzo und Miesner, 2016)

Festlegung	Bedeutung	Verbform	Formulierungen
Anforderungen	Einhaltung verbindlich, keine Abweichung erlaubt	muss (Gebot)	ist zu | ist erforderlich | müssen |sind zu | haben zu | dürfen nur
		darf nicht (Verbot)	es ist nicht zulässig / erlaubt / gestattet | es ist unzulässig | es ist nicht | es hat nicht | dürfen nicht | sind nicht zulässig
Empfehlungen	Zuraten/Abraten für eine von verschiedenen Möglichkeiten	sollte	es wird empfohlen, dass ... | ist in der Regel ... | sollten
		sollte nicht	es wird nicht empfohlen | sollte vermieden werden
Zulässigkeit	Erlaubnis für eine bestimmte Handlungsweise	darf	ist zugelassen | ist zulässig | dürfen
		braucht nicht	ist nicht erforderlich | müssen nicht | keine ... nötig
Möglichkeit	physische, physikalische oder kausale Möglichkeit/Fähigkeit	kann	vermag | lässt sich ... | können
		kann nicht	vermag nicht | es ist nicht möglich, dass ... | lässt sich nicht

In diesem Zusammenhang beschreiben Anforderungen eine notwendige Beschaffenheit oder Fähigkeit. Das Ziel einer Norm kann ein Objekt oder eine Person sein. Um die Anforderung zu erfüllen, muss das Ziel in seiner Beschaffenheit oder Funktion einem oder mehreren Anforderungskriterien entsprechen, welche in der jeweiligen Norm formuliert werden. Ziel bei der Formulierung der Anforderung ist es, die Einhaltung einer Sache nach Stand von Wissenschaft und Technik zu gewährleisten, um Schäden oder Gefahren zu vermeiden. Daher muss diese konkret und eindeutig formuliert sein und darf nicht stärker detailliert sein, als es dem Zweck angemessen ist. Ausnahmen von Regelungen werden auch als bedingte Anforderungen bezeichnet. Diese sind nur bei entsprechender Nennung genehmigt und müssen ausreichend begründet werden. Von Geboten und Verboten darf als unbedingte Anforderung unter keinen Umständen abgewichen werden. Empfehlungen können ausgesprochen werden, um dem Anwender einen Hinweis für die Verwendung oder Ausführung des normativen Abschnittes zu geben.

Zulässigkeiten oder Erlaubnisse hingegen bestimmen Handlungsrahmen innerhalb bestehender Vorschriften. Eine Über- oder Unterschreitung ist zu vermeiden und darf nur in aufgezeigten Ausnahmen erfolgen (DIN, 2012-12; Mattiuzzo und Miesner, 2016).

Als Beispiel für eine Fließtextdarstellung soll an dieser Stelle eine Anforderung hinsichtlich vorhandener Rettungswege aus der bayerischen Bauordnung (Molodovsky, 2016) dienen:

> ***Art. 31 Erster und zweiter Rettungsweg***
> *(1) Für Nutzungseinheiten mit mindestens einem Aufenthaltsraum wie Wohnungen, Praxen, selbstständige Betriebsstätten müssen in jedem Geschoss mindestens zwei voneinander unabhängige Rettungswege ins Freie vorhanden sein; beide Rettungswege dürfen jedoch innerhalb des Geschosses über denselben notwendigen Flur führen.*
> *(2) Für Nutzungseinheiten nach Abs. 1, die nicht zu ebener Erde liegen, muss der erste Rettungsweg über eine notwendige Treppe führen. Der zweite Rettungsweg kann eine weitere notwendige Treppe oder eine mit Rettungsgeräten der Feuerwehr erreichbare Stelle der Nutzungseinheit sein. Ein zweiter Rettungsweg ist nicht erforderlich, wenn die Rettung über einen sicher erreichbaren Treppenraum möglich ist, in den Feuer und Rauch nicht eindringen können (Sicherheitstreppenraum).*
> *(3) Gebäude, deren zweiter Rettungsweg über Rettungsgeräte der Feuerwehr führt und bei denen die Oberkante der Brüstung von zum Anleitern bestimmten Fenstern oder Stellen mehr als 8 m über der Geländeoberfläche liegt, dürfen nur errichtet werden, wenn die Feuerwehr über die erforderlichen Rettungsgeräte wie Hubrettungsfahrzeuge verfügt. Bei Sonderbauten ist der zweite Rettungsweg über Rettungsgeräte der Feuerwehr nur zulässig, wenn keine Bedenken wegen der Personenrettung bestehen.*

Daher geben Normungsinstitute wie beispielsweise in Norwegen mittlerweile in ihren Handlungsweisungen vor, sich hauptsächlich auf die Formulierung von Fließtext zu beschränken und weitere Darstellungstypen zu Gunsten der Einheitlichkeit zu meiden (Standard Norge, 2017). So werden beispielsweise auch verhältnismäßig komplexe geometrische Gegebenheiten, wie diese insbesondere bei Richtlinien zur Barrierefreiheit vorkommen, mit Hilfe von Fließtexten beschrieben. Ein entsprechendes Beispiel aus der norwegischen Norm NS 11001-1 2009 (NS, 2009):

> *The access route for pedestrians / wheelchair users shall not be steeper than 1:20. For distances of less than 3 metres, it may be steeper, but no more than 1:12. The access route shall have clear width of a minimum of 1,8 m and obstacles shall be placed so that they do not reduce that width.*

Trotz der vorgegebenen Formulierungsweisen und weiterer Rahmenbedingungen bergen Fließtexte die Gefahr, dass es zu Unklarheiten, Mehrdeutigkeiten oder Widersprüchlichkeiten kommt. Dieses liegt insbesondere an dem Interpretationsspielraum des Anwenders bzw. Lesers, welcher die Inhalte der Norm interpretieren muss. Generell gilt, dass die menschliche Sprache per se niemals vollkommen eindeutig ist und daher Spielraum für Interpretationen bietet.

Diagramme und graphische Darstellungen

Insbesondere bei der Beschreibung geometrischer Ausgangssituationen und Gegebenheiten können Vorschriften mit Hilfe von Grafiken für den Anwender verständlicher gestaltet werden. So finden sich in diversen Regelwerken neben den Fließtexten auch graphische Darstellungen bzw. Piktogramme. In Abbildung 4.2 sind beispielhaft verschiedene Abbildungen zur Formulierung von Anforderungen an die Freiräume vor und hinter Türen für eine französische Richtlinie *Annexe 8 : Accessibilité des établissments recevant du public* im Bereich Barrierefreiheit dargestellt (Ministere du logement et de la ville, 2008-05).

Wie im vorigen Abschnitt beschrieben, bergen geometrische Darstellungen die Gefahr, dass es zu großen Interpretationsspielräumen, da Grafiken einen Sachverhalt unter Umständen nicht eindeutig

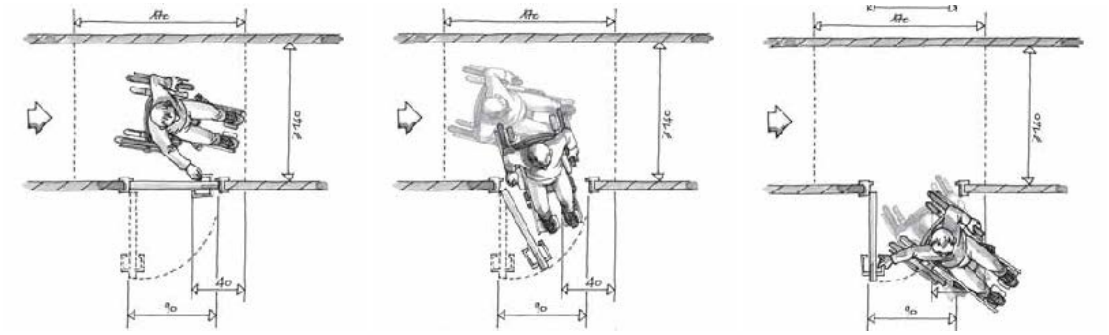
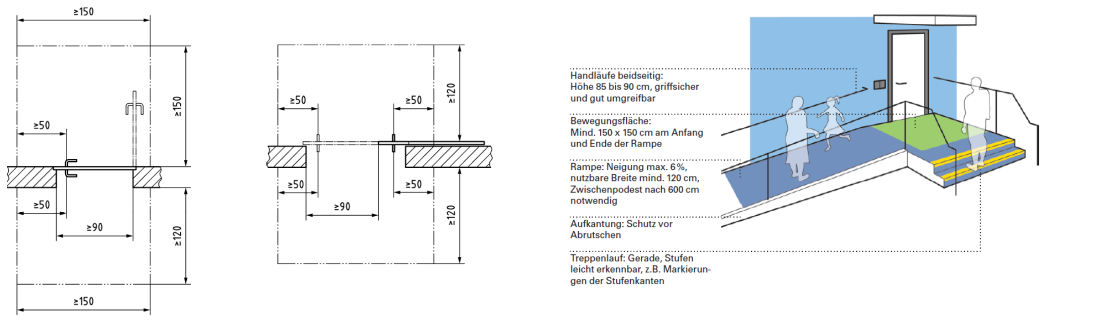


Abbildung 4.2: Graphische Darstellung der Anforderungen an die Freiräume vor und hinter Türen, um die Barrierefreiheit für Rollstuhlfahrer zu erfüllen (Ministere du logement et de la ville, 2008-05).

darstellen. Daher geben einige nationale Normungsinstitute in ihren Handlungsweisungen vor, dass graphische Darstellungen nur als Ergänzung herangezogen werden dürfen, der Sachverhalt selbst jedoch immer im Fließtext dargelegt werden muss, da diese Darstellungsform stringenter ist.

Ein wesentlicher Grund für die Verwendung von Abbildungen ist allerdings, dass ein großer Teil bestehenden Bauvorschriften sich auf Regeln mit räumlichen Sachverhalten bezieht. Solche Sachverhalte stellen beispielsweise topologische oder geometrische Eigenschaften oder Beziehungen zwischen Bauteilen auf. Insbesondere, wenn diese Beziehungen in Kombination mit weiteren geometrischen Randbedingungen, wie bspw. Abstands- oder Zirkulationsflächen, eingeführt werden, ist es bisweilen hilfreich für den Anwender, die Ausgangssituation mit Hilfe von Bildern zu untermauern. Beispiele für die Darstellung von Abstands- und Zirkulationsflächen oder aber geometrischen Beziehungen zwischen Bauteilen sind in Abbildung 4.2 oder aber auch in DIN 18040-1 (DIN, 2010-10) zu finden (siehe Abbildung 4.3).



(a) Anforderungen an die Freiräume vor Türen nach DIN 18040-1 (DIN, 2010-10)

(b) Geometrische Anforderungen an die Außenbauteile vor Türen für die Barrierefreiheit nach (Oberste Baubehörde im Bayerischen Staatsministerium des Innern, 2012)

Abbildung 4.3: Beispiele für die Darstellung von geometrischen Anforderungen in Deutschen Normen und Richtlinien

Ein weiteres Beispiel für eine graphische Darstellung findet sich in Abbildung 4.4. Mit diesem Diagramm wird in der britischen Richtlinie *UK Fire Code Part B4* zum Brandschutz (NBS, 2007) die Position von Öffnungen auf den Außenbauteilen beschrieben. Durch diese Regeln soll verhindert werden, dass Feuer schnell von einem Gebäudeteil bzw. Stockwerk in ein anderes übergreifen.

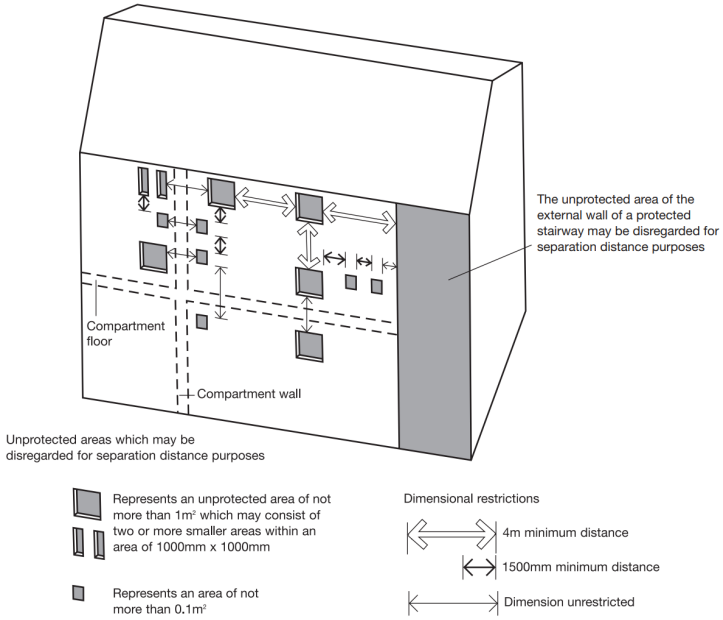


Abbildung 4.4: Graphische Darstellung von Brandschutz-Anforderungen für das Platzieren von Öffnungen in Außenbauteilen von Bauwerken nach UK Fire Code Part B4 (NBS, 2007)

Tabellen

Tabellen finden in Normen bzw. Richtlinien typischerweise dann Anwendung, wenn Parameter als begrenzende Anforderung eingeführt werden. In Datentabellen werden in Abhängigkeit von jeweiligen Eingabedaten spezifische Werte oder aber Wertebereiche eingeführt, welche eingehalten werden müssen. Die Anzahl der Eingabewerte sowie die Komplexität des Tabellenaufbaus können dabei sehr groß werden. Ein entsprechendes Beispiel einer solchen Tabelle aus DIN 18232:2007-11 (DIN, 2007-11) ist auszugsweise in Abbildung 4.5 dargestellt.

Raumhöhe[a]	Höhe der Rauchschicht	Höhe der raucharmen Schicht	Notwendige Rauchabzugsfläche A_w in m²				
			Bemessungsgruppe				
h in m	z in m	d in m	1	2	3	4	5
3,0	0,5	2,5	4,8	6,2	8,2	11,0	15,4
3,5	1,0	2,5	3,4	4,4	5,8	7,8	10,9
	0,5	3,0	6,7	8,7	11,3	15,0	20,4
4,0	1,5	2,5	2,8	3,6	4,7	6,4	8,9
	1,0	3,0	4,8	6,2	8,0	10,6	14,4
4,5	2,0	2,5	2,4	3,1	4,1	5,5	7,7
	1,5	3,0	3,9	5,0	6,5	8,7	11,8
	1,0	3,5	5,9	8,4	10,7	13,9	18,6
	2,5	2,5	2,2	2,8	3,6	4,9	6,9

Abbildung 4.5: Tabelle mit den Angaben zu dem Soll-Wert der Rauchabzugsfläche nach DIN 18232:2007-11 DIN (2007-11)

Der Soll-Wert der Rauchabzugsfläche für einen Raum ergibt sich über die Höhe des Raumes, die Höhe der Rauchschicht und der jeweiligen Brandbemessungsgruppe, d. h. der Stärke des Brandes. Die Höhe des Raumes ergibt sich aus der jeweiligen Gestaltungsplanung, die Höhe der Rauchschicht und die Brandbemessungsgruppe hingegen beziehen sich auf die Brandschutzklasse und die Intensität des Brandes, welche für den Raum überprüft werden soll, und erfordern eine Nutzereingabe.

Der Ist-Wert der Rauchabzugsfläche muss aus der Gestaltungsplanung bezogen werden, denn dabei handelt es sich um einen Wert, der von den vorhandenen Öffnungen des Raumes abhängig ist. Dabei ist zu beachten, dass innerhalb der Norm eine Vielzahl von Korrektur- und Abminderungswerten angegeben wird, welche die Beschaffenheit und Eigenschaften des Raumes einkalkulieren.

Öffnungsart	Öffnungswinkel	Korrekturfaktor c_z
Tür- oder Toröffnungen, Maschengitter		0,7
öffenbare Jalousien	90°	0,65
Dreh- oder Kippflügel	90°	0,65
	≥ 60°	0,5
	≥ 45°	0,4
	≥ 30°	0,3

Abbildung 4.6: Tabelle mit Korrekturfaktoren für den Fenstertyp nach DIN 18232:2007-11 DIN (2007-11)

Zum Beispiel beschreibt die Norm, dass die Öffnungsfläche eines Fensters durch den Fenstertypus und den Öffnungswinkel des Fensters beeinflusst wird. Dieser Umstand wird innerhalb einer Tabelle in Abhängigkeit von Fenstertypus und Öffnungswinkel als Korrekturfaktor c_z angegeben (siehe Abbildung 4.6).

In DIN 18040-1 DIN (2010-10) findet sich zum Thema Barrierefreiheit eine Tabelle, welche für einen jeweiligen Komponenten-Typ entsprechende Maße und Richtwerte einführt (siehe Abbildung 4.7).

	Komponente	Geometrie	Maße cm
	1	2	3
	alle Türen		
1	Durchgang	lichte Breite	≥ 90
2		lichte Höhe über OFF	≥ 205
3	Leibung	Tiefe	≤ 26[a]
4	Drücker, Griff	Abstand zu Bauteilen , Ausrüstungs- und Ausstattungselementen	≥ 50
5	zugeordnete Beschilderung	Höhe über OFF	120 – 140
	manuell bedienbare Türen		
6	Drücker	Höhe Drehachse über OFF (Mitte Drückernuss) Das Achsmaß von Greifhöhen und Bedienhöhen beträgt grundsätzlich 85 cm über OFF. Im begründeten Einzelfall sind andere Maße in einem Bereich von 85 cm bis 105 cm vertretbar.	85
7	Griff waagerecht	Höhe Achse über OFF	85
8	Griff senkrecht	Greifhöhe über OFF	85
	automatische Türsysteme		
9	Taster	Höhe (Tastermitte) über OFF	85
10	Taster Drehflügeltür/Schiebetür bei seitlicher Anfahrt	Abstand zu Hauptschließkanten[b]	≥ 50
11	Taster Drehflügeltür bei frontaler Anfahrt	Abstand Öffnungsrichtung	≥ 250
		Abstand Schließrichtung	≥ 150
12	Taster Schiebetür bei frontaler Anfahrt	Abstand beidseitig	≥ 150

OFF = Oberfläche Fertigfußboden

[a] Rollstuhlbenutzer können Türdrücker nur erreichen, wenn die Greiftiefe nicht zu groß ist. Das ist bei Leibungstiefen von max. 26 cm immer erreicht. Für größere Leibungen muss die Nutzbarkeit auf andere Weise sichergestellt werden.

[b] Die Hauptschließkante ist bei Drehflügeltüren die senkrechte Türkante an der Schlossseite.

Abbildung 4.7: Tabelle mit Maß- und Richtwerten für den jeweiligen Komponenten-Typ nach DIN 18040-1 DIN (2010-10)

Gleichungen

Weiterhin können Anforderungen in Form von Gleichungen dargestellt werden. In diesem Fall wird eine Anforderung als Aussage über die Gleichheit zweier Terme beschrieben. Die Gleichungen können dabei direkt oder aber auch in Form von Fließtext beschrieben werden.

Ein Beispiel für eine direkte Darstellung ist die Berechnung des Temperaturverlaufs durch ein Bauteil gemäß DIN 4108-3 (DIN, 2017-09).

$$R_{\text{min}} = \frac{R_{\text{si}}}{1 - f_{\text{Rsi,min}}} - (R_{\text{si}} + R_{\text{se}})$$

mit:

$$f_{\text{Rsi,min}} = \frac{\theta_{\text{si,min}} - \theta_{\text{e}}}{\theta_{\text{i}} - \theta_{\text{e}}}$$

Ein Beispiel für eine indirekte Darstellung wurde bereits in Abschnitt 4.2.1 angeführt mit der Fließtextdarstellung der Abstandsflächen. Die Anforderung §6(5) MBO (Messer und Ammon, 2015) kann im Grunde allerdings auch direkt als Gleichung dargestellt werden. Hier ein entsprechendes Beispiel:

Die Tiefe der Abstandsflächen beträgt 0,4 H, mindestens 3 m.

$$\bigtriangledown$$

$$T_{\text{Abstandsfläche}} = 0,4 H_{\text{Bauwerk}}$$

Formeln bzw. Gleichungen werden unter anderem auch mit Diagrammen vereint, sodass diese wie Parameter an Piktogrammen dargestellt werden. Beispiele für die Anwendung auf die Abstandsflächen sind unter anderem in der Verwaltungsvorschrift des Sächsischen Staatsministeriums des Innern zur sächsischen Bauordnung (Sächsische Staatskanzlei, 2017) zu finden und in Abbildung 4.8 dargestellt.

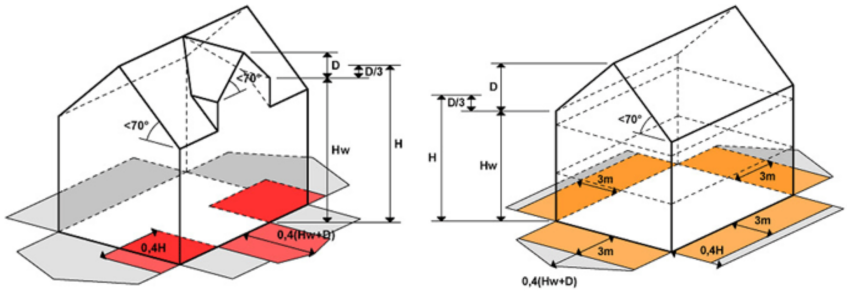


Abbildung 4.8: Darstellung der Anforderungen für Abstandsflächen nach SächsABl als Piktogramme mit Formeln (Sächsische Staatskanzlei, 2017)

4.4 Komplexität regulatorischer Anforderungen

An den diversen Formen und Darstellungstypen, wie diese in den vorherigen Abschnitten vorgestellt wurden, lässt sich erkennen, dass es sich bei vielen der enthaltenen Anforderungen meist um keine simplen Fragestellungen handelt, welche mit einer direkten, einfachen Antwort beantwortet werden können. Je nach Anforderung und Art der Anforderungsbeschreibung ist es notwendig, dass für ein Bestehen genau ein Kriterium, mindestens eines von vielen möglichen, mehrerer Kriterien gleichzeitig oder gar eine komplette Lösungsvariante als gültige Lösung nachgewiesen werden muss. Um die unterschiedlichen Darstellungsformen von regulatorischen Anforderungen zu strukturieren und kategorisieren, ist es hilfreich Komplexitätsstufen einzuführen. Der Begriff Komplexität beschreibt in diesem Zusammenhang, wie viele Prozessschritte notwendig sind, um zunächst die notwendigen Eingangs- bzw. Ausgangsdaten für die Prüfung der jeweiligen Anforderung zu erhalten und diese dann schließlich auch durchzuführen. Die Komplexität hat einen maßgeblichen Einfluss auf den Aufwand, welcher mit einer Formalisierung der enthaltenen Anforderungen verbunden ist.

Solihin und Eastman (2015a, 2016) führen solche Komplexitätsstufen hinsichtlich einer Formalisierung mit Hilfe digitaler Methoden ein und machen so die Herausforderung, welche mit einer solchen Formalisierung verbunden ist, besser greif- und nachvollziehbar. Die Komplexitätsstufen sind eng mit den Prüfmethoden der Modellqualität verwandt, wie diese in Abschnitt 3.3 vorgestellt wurden:

Regeln, die explizite Eingangsdaten erfordern: Viele Anforderungen beziehen sich auf eine einfache Prüfung von Attributen und Relationen, die explizit und direkt in einem Gebäudemodell enthalten sind. Für eine solche Prüfung können die jeweiligen Informationen entweder direkt aus den Entitäten oder aus den zugehörigen Eigenschaften mit anderen Entitäten unter Verwendung der expliziten Beziehungsentitäten bezogen werden. Es müssen keine höherwertigen Informationen aus dem Modell abgeleitet oder aufbereitet werden. Für die Darstellung einer spezifischen Gültigkeit oder Randbedingung können die abgefragten Werte mit Hilfe gültiger Wertebereiche

verglichen werden oder mit Konstanten zueinander ins Verhältnis gesetzt werden (Gleichungen). Das Bauwerksmodell wird nach einer genau definierten Entität (z.B. eine Wand, Tür oder Fensterkomponente) durchsucht und deren Beziehungen abgefragt.

Beispiele:

- Prüfung, ob alle erforderlichen Attribute von Entitäten vorhanden sind. Im IFC-Standard ist genau definiert, welches Bauteil ein Attribut, z. B. *Feuerwiderstandsklasse*, besitzt. Dieser Wert kann direkt über die Entität bezogen (bspw. als *Pset_∗Common.FireRating*) und anschließend hinsichtlich eines Referenzwertes oder eines gültigen Wertebereiches geprüft werden.
- Prüfung von Integritätsbedingungen, wie beispielsweise sinnvolle Grenzen oder gültige Datentypen; z.B. Attribut *Name* mit dem Datentyp *LongName*
- Prüfung der korrekten Zuordnung von Klassifikationen. Auch hier ist eine entsprechende Abbildung von Klassifikationen und der entsprechenden Zuordnung von Bauteilen im IFC-Standard vorgesehen (siehe *IfcClassificationReference*).
- Einfache regulatorische Prüfungen, wie z. B. in IBC 1008.1.2 (International Code Council, 2006) "Fluchttüren müssen schwenkbar oder seitlich schwenkbar sein". Türen können gemäß IFC-Standard über das Attribut *Pset_DoorCommon.FireExit* als Fluchttüren gekennzeichnet und überdies mit dem entsprechenden Öffnungstyp (siehe z.B. *IfcDoorStyle.OperationType*) gekennzeichnet werden. Alle relevanten Informationen für diese Prüfung können somit direkt aus dem Bauwerksmodell übernommen werden.

Regeln, die abgeleitete Informationen bzw. Werte erfordern:

Bei einigen Prüfungen ist es notwendig, einzelne oder mehrere Informationen aus dem Modellinhalt abzuleiten, da diese nicht direkt enthalten sind. Hierbei werden die unbekannten Informationen, z. B. ein Wert, in einem vorhergehenden Schritt der Prüfung in einem definierten Verarbeitungsprozess produziert, jedoch nur temporäre Daten und keine neuen Datenstrukturen erzeugt. Bei diesen Verarbeitungsprozessen handelt es sich in der Regel um keine anspruchsvollen Berechnungen. Daher ist diese Erzeugung der Informationen üblicherweise Teil der Durchführung des Prüfprozesses selbst.

Beispiele:

- Mengenberechnungen, z. B. Volumen, Summe von Stirn- oder Mantelflächen o. Ä. Wenn diese Angaben nicht in dem Autorenwerkzeug an die jeweilige Komponente angeheftet wurden, müssen diese Werte berechnet werden.
- Arithmetische oder trigonometrische Berechnungen, z. B. die Ermittlung des Abstands zwischen zwei Punkten oder die Bestimmung, ob sich ein Punkt innerhalb eines eingeschlossenen Polygons befindet.
- Erzeugen impliziter Beziehungen zwischen Bauteilen, weil diese nicht im Autorenwerkzeug erzeugt oder in das Modell übergeben wurden, z. B. Nachbarschaftsbeziehungen von Bauteilen untereinander oder zu Räumen.

Regeln, welche eine erweiterte Datenstruktur erfordern:

Für die Prüfung dieser Art von Regeln können die erforderlichen Informationen nicht in dem Gebäudemodell enthalten sein, da die entsprechenden Datenstrukturen für die Speicherung dieser Informationen nicht zur Verfügung stehen. Daher müsssen für die Erzeugung eine externe Bibliothek oder Datenstruktur sowie die entsprechenden Berechnungsroutinen herangezogen werden. Typischerweise handelt es sich hierbei um umfangreiche Berechnungen, wie beispielsweise geometrische Algorithmen. Für solch aufwendige Berechnungen wird oft eine zusätzliche Geometriebibliothek oder ein sogenannter Modellierungs- bzw. Geometriekern benötigt, welcher die entsprechenden Berechnungen durchführen kann. Da diese Art der Berechnung von Informationen in der Regel deutlich aufwendiger ist, ist es sinnvoll, die Daten nicht ausschließlich fließend bzw. temporär zu erzeugen, sondern diese auch für eine Weiterverwendung bei weiteren Prüfungen zu sichern.

Beispiele:

- Berechnung eines Graphen, welcher z.B. die Zugänglichkeit zwischen horizontalen bzw. vertikalen Durchgangsbauteilen und Räumen oder eine mögliche Fluchtroute repräsentiert. Sowohl für die Berechnung als auch für die Repräsentation eines Graphen müssen erweiterte Datenstrukturen herangezogen werden, da diese Informationen üblicherweise nicht in Gebäudemodellen dargestellt oder von Autorenwerkzeugen erzeugt werden.
- Berechnung komplexer impliziter Beziehungen zwischen Bauteilen, z.B. geometrisch-topologische Beziehungen zwischen Bauteilen und räumlichen Strukturen.

Regeln, die einen Nachweis der Gültigkeit erfordern:

Hier werden keine präskriptiven Anforderungen beschrieben, sondern es wird lediglich eine leistungsbasierte bzw. funktionale Beschreibung geliefert, welche die Leistungsfähigkeit des gesamten oder einer Teilmenge des Bauwerks beschreiben. Es wird also nicht auf Einhaltung oder Nicht-Einhaltung von definierten Anforderungen geprüft, sondern ein Nachweis (engl. *proof of solution*) geführt, dass die gegebene Lösung gültig ist.

Um eine gültige Lösung zu beschreiben, kann beispielsweise mit einer gültigen Referenzlösung als Vergleichsinstrument gearbeitet werden. Hierbei spricht man dann von einem Referenzverfahren. Eine gültige Lösung kann auch als ein Rahmenwerk aus möglichen gültigen Werten oder Wertebereichen beschrieben werden. Gibt es eine Vielzahl gültiger Lösungen, werden diese in eine Wissensbasis gespeichert, welche kontinuierlich um bewährte Verfahren (engl. *best practices*) erweitert wird.

Referenzverfahren lassen sich nur sehr schwer formal beschreiben, da keine direkten Anforderungen, sondern ein ganzheitliches gültiges Rahmenwerk beschrieben wird.

Beispiel:

- Referenzgebäude bzw. -verfahren nach Energieeinsparverordnung (EnEV)
- Leitfäden oder Richtlinien, welche Konstruktionslösungen bzw. -hilfen vorgeben

4.5 Zusammenfassung

Im Bauwesen gelten Normen, Richtlinien und weitere regulatorische Dokumente, um Anforderungen an die Leistungsfähigkeit eines Bauwerks in Abhängigkeit von dessen Standort und Zweck zu formulieren. Mit der Formulierung dieser Anforderungen soll sichergestellt werden, dass Bauwerke essenzielle Forderungen und Kriterien, wie bspw. Stabilität und Sicherheit der Bewohner, einhalten.

Die einzelnen geltenden regulatorischen Anforderungen ergeben sich in erster Linie aus dem geltenden Recht und der Gesetzgebung, wie z. B. dem Baurecht in Deutschland. Darüber hinaus formulieren Normen und Richtlinien den anerkannten Stand der Technik, welcher zwar nicht als positives Recht gilt, aber zu Beurteilungszwecken von der Justiz im Rechtsfall herangezogen werden kann. Daher machen Unternehmen und Akteure von regulatorischen Anforderungen Gebrauch, um eigene Anforderungen für deren spezifische Zwecke standardisiert und allgemeingültig zu formulieren.

Bei der Beschreibung von regulatorischen Anforderungen kann zwischen einem präskriptiven und einem leistungsbasierten Ansatz unterschieden werden. Der präskriptive Ansatz beschreibt klar definierte Anforderungen und Randbedingungen, die eingehalten werden müssen, auch wenn diese für den spezifischen Einsatzzweck bisweilen nicht das Optimum darstellen. Der leistungsbasierte Ansatz hingegen stellt die gewünschte Leistungsfähigkeit des zu erstellenden Bauwerks in den Vordergrund. Dabei werden keine spezifischen Anforderungen formuliert, sondern es wird lediglich die Beschaffenheit des Ergebnisses beschrieben. Dadurch wird mehr Freiraum für die Entwicklung eines optimalen Lösungsweges eingeräumt.

Gegenwärtig basieren die zentralen, geltenden regulatorischen Dokumente insbesondere auf dem präskriptiven Ansatz und beschreiben Anforderungen in großen Teilen nicht leistungsbasiert. In der vorliegenden Arbeit wird daher auch insbesondere das Augenmerk auf die präskriptiven Anforderungen gelegt.

5 Automatisierung der Konformitätsüberprüfung

5.1 Einführung

Konstruktions- und Planungsfehler im Bauwesen gefährden die Sicherheit und führen unweigerlich zu Ausfällen bei Bau- und Konstruktionsprojekten. Nicht erkannte Fehler oder Versäumnisse können verheerende wirtschaftliche, ökologische und soziale Folgen haben (Love et al., 2013). Daher werden erhebliche Anstrengungen unternommen, die Häufigkeit von Fehlern zu verringern. In Folge von früheren Katastrophen oder Unfällen wurden Bauvorschriften und -standards eingeführt, welche kontinuierlich angepasst werden, damit diese den anerkannten Stand der Technik und angemessenen Grad an Zuverlässigkeit widerspiegeln. Die Prüfung der Entwurfs- und Gestaltungsplanung eines Bauwerks hinsichtlich dieser geltenden Vorschriften und Richtlinien gewährleistet die erforderliche Sicherheit. Daher handelt es sich um einen essenziellen Prozess bei der Durchführung eines Bauprojektes.

Allerdings bleiben Konstruktionsfehler trotz der bisher gesammelten Erfahrungen und des Wissens ein Wesensmerkmal von Bau- und Konstruktionsprojekten (Love et al., 2013). Auch wenn die meisten Fehler während der Gestaltungsplanung erkannt und nachbearbeitet werden, besteht immer die Möglichkeit, dass einige unentdeckt bleiben und so zu Fehlleistungen, Unfällen oder gar zum Verlust von Menschenleben führen. Eine detaillierte Recherche zu der Herkunft und den Auswirkungen von Fehlern in der Bauplanung ist bei Lopez und Love (2012); Love et al. (2013, 2011); Samofalov et al. (2015) zu finden.

Überdies wurde in Studien aufgezeigt, dass die heutigen Prüfmethoden in den Genehmigungsverfahren diverse Herausforderungen und Probleme mit sich bringen (Jeong und Lee, 2009; Shih et al., 2013; Tan et al., 2010). Zu den Faktoren, welche zu einer erhöhten Gefahr führen, dass Fehler in der Planung unentdeckt bleiben werden in der Literatur zumeist die folgenden Aspekte angeführt (Eastman et al., 2009; Nisbet et al., 2008; Solihin, 2016; Uhm et al., 2015):

Manuelle, iterative und monotone Arbeit

Die Prüfung der Gestaltungsplanung wird weitestgehend manuell durchgeführt und ist daher arbeitsintensiv, zeitaufwendig und fehleranfällig. Dieses liegt unter anderem an dem Umstand, dass die Prüfung nicht nur vonseiten der ausführenden Planer, sondern insbesondere auch vonseiten der prüfenden Behörde durchgeführt werden muss. Die Arbeitsintensität nimmt insbesondere dann zu, wenn es zu Iterationszyklen aufgrund von geforderten Nachbesserungen kommt. An dieser Stelle müssen die Prüfprozesse auf gleiche Art und Weise mehrfach durchgeführt werden, da sich die zugrunde liegenden Ausgangsdaten geändert haben.

Daher handelt es sich zu Teilen auch um eine monotone Tätigkeit, bei welcher häufig dieselbe Prozedur mehrfach hintereinander ausgeführt werden muss. So entstehen beispielsweise viele

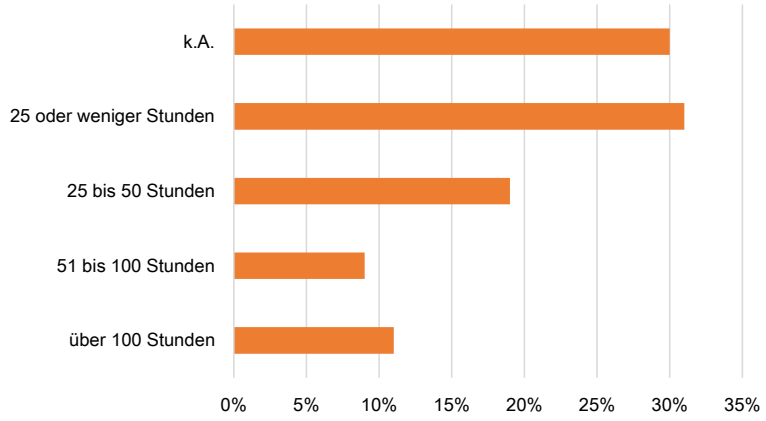


Abbildung 5.1: Ergebnisse einer Umfrage von McGrawHill Construction (2007) zu der Frage, wie viel Arbeitszeit üblicherweise mit der Prüfung der Gestaltungsplanung hinsichtlich Normen und Richtlinien investiert wird

Verbesserungsaufträge aufgrund von Fehlern in der Plandarstellung, da in zweidimensionalen Papierzeichnungen verhältnismäßig häufig Zeichnungsfehler auftreten (Fiedler, 2015).

Laut einer Umfrage von McGrawHill Construction (2007) beschäftigt sich die Hälfte der Architekten und Bauherren bei einem typischen Bauprojekt im Durchschnitt mehr als 26 Stunden mit der Prüfung der Gestaltungsplanung auf Einhaltung von Normen und Richtlinien (siehe Abbildung 5.1).

Komplexität und Anfälligkeit für menschliche Fehler

Da ein Gebäude aus vielen Elementen besteht, die geometrisch komplex und verwoben sind, ist auch die zugehörige Entwurfsprüfung entsprechend komplex und fehleranfällig. Die Ergebnisse werden allerdings manuell produziert und dadurch besteht eine Anfälligkeit für menschliche Fehler oder menschliches Versagen (Acharya et al., 2006; Eastman et al., 2009; McGrawHill Construction, 2007). An dieser Stelle hat insbesondere auch die Subjektivität einen wesentlichen Einfluss, da die Regeln von Seiten des verantwortlichen Prüfers interpretiert werden müssen. Dieses kann zu unzuverlässigen Ergebnissen führen. Das gilt insbesondere, wenn die Formulierung der Anforderungen aufgrund von Mehrdeutigkeiten oder Widersprüchen der natürlichen Sprache Interpretationsspielräume offen lässt. Die regulatorischen Dokumente standardisieren lediglich die Anforderungen und Regeln selbst, nicht aber die Prüfverfahren. Eine Prüfung mittels herkömmlicher Überprüfungsmethoden kann also zu unterschiedlichen Ergebnissen führen und ist somit nicht konsistent (Fiatech, 2012).

Technische Dokumente als Ausgangsdaten

Da noch immer viele der Genehmigungsverfahren nicht auf einer Abgabe digitaler Unterlagen basiert, ist eine vollständige elektronische Verarbeitung der Daten und Informationen nicht möglich. An dieser Stelle kommt es somit zu einem Informations- und Medienbruch im Prüfverfahren. Daher basiert die Prüfung der Gestaltungsplanung gegenwärtig weitestgehend auf zweidimensionalen technischen Dokumenten, aus welchen wiederum die Eingangsinformationen für den Prüfprozess

manuell entnommen werden müssen. Dieses führt dazu, dass Daten mehrfach (manuell) eingegeben werden müssen und die Gefahr von Informationsverlusten steigt. So kommt es etwa zu Fehlern, weil Informationen in den Papierzeichnungen übersehen werden. Zudem ist eine Aufbewahrung der Bestandspläne in Papierform unsicher, denn diese können teilweise oder gänzlich verloren gehen. Hier besteht ein dringender Bedarf einer neuen, effizienten und zuverlässigen Dokumentation und Archivierung (Fiedler, 2015).

Sorgfalt, Erfahrung und Wissen des Prüfers

Die regulatorischen Anforderungen, welche bei der Prüfung beachtet werden müssen, werden vom Prüfer selbst aus den Vorschriften abgeleitet. Für die Auswahl und Prüfung der richtigen Kriterien ist insbesondere der Planer bzw. Prüfer selbst verantwortlich. Dabei spielen insbesondere die Sorgfalt, die Erfahrung und das Wissen des jeweiligen Planers bzw. Prüfers eine wesentliche Rolle. Dies wird durch eine in Südkorea durchgeführte Studie von Uhm et al. (2013, 2015) bestätigt, die zudem feststellt, dass Architekten ihre Entwürfe viel häufiger hinsichtlich der Ausschreibungskriterien als gegen Bauvorschriften oder andere Arten von Entwurfsreferenzen prüfen. Die Ergebnisse dieser Studie sind entsprechend in Abbildung 5.2 dargestellt.

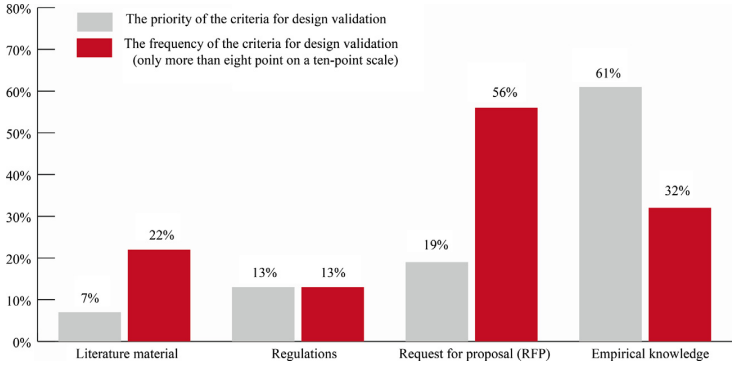


Abbildung 5.2: Priorität der verwendeten Referenzquellen für die Konformitätsprüfung (Uhm et al., 2013, 2015)

Die aufgeführten Probleme, welche die heutige Konformitätsprüfung mit sich bringt, führen zu einem großen Interesse der Beteiligten im Bauwesen an einer Erleichterung dieses Prozesses. In der Studie von (McGrawHill Construction, 2007) bekunden insbesondere Architekten sehr großes Interesse an einer Automatisierung der Prüfprozesse (siehe Abbildung 5.3).

Mit der voranschreitenden Einführung von BIM stehen digitale Werkzeuge und methodische Grundlagen zur Verfügung, um die Effizienz der Prüfung zu verbessern. Dabei verfolgen diverse Forschungsansätze eine Automatisierung der Konformitätsprüfungen, welche in der Literatur häufig als ACCC bezeichnet wird (Dimyadi und Amor, 2013; Eastman et al., 2009).

Automatisierte Systeme sind in der Lage, Aufgaben vorrangig gleichbleibender und standardisierter Art eigenständig zu lösen (Fuchs-Kittowski et al., 1976; Weller, 2008). Diese grenzen sich gegenüber den von Menschen geführten Systemen ab, indem sie die Tätigkeiten vereinheitlicht, ohne Ermüdung und in gleichbleibender Qualität wiederholt durchführen. Mit der Komplexität der Handlungsprozesse steigen auch der Anspruch und die Anforderungen an die Automatisierungssysteme. Es liegt nahe, die

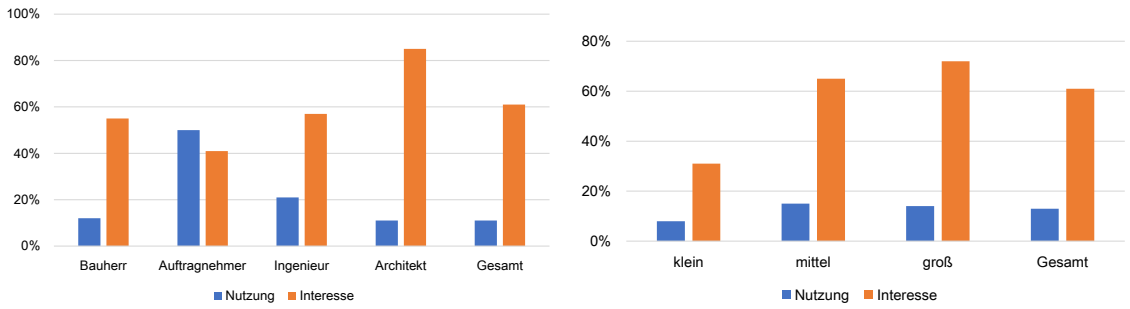


Abbildung 5.3: Ergebnisse einer Umfrage von McGrawHill Construction (2007) zu der Frage, wie viel Arbeitszeit üblicherweise mit der Prüfung der Gestaltungsplanung hinsichtlich Normen und Richtlinien investiert wird

Konformitätsprüfung im Bauwesen, wie diese zuvor mit den diversen zugehörigen Herausforderungen vorgestellt wurden, auf eine Eignung hin zu einer Automatisierung zu untersuchen:

Entlastung bei manueller, monotoner und iterativer Arbeit

Wesentlicher Vorteil einer Automatisierung ist, dass der Mensch bei schwerer körperlicher oder aber monotoner Arbeit und gleichzeitig konstanter Qualität entlastet werden kann. Bei der Prüfung handelt es sich zu weiten Teilen um einen vorrangig gleichbleibenden und standardisierten Prozess, dessen Automatisierung den Menschen unterstützen könnte. Da der Mensch durch die Unterstützung von teil-automatisierten Systemen von monotonen Arbeitstätigkeiten befreit wird, bleibt mehr Zeit für individuelle, kreative Arbeitsprozesse. Dies steigert den Komfort am Arbeitsplatz und mindert gleichzeitig die Fehleranfälligkeit.

Zeitersparnis

Durch eine Automatisierung wird die Durchsatzleistung erhöht, da automatisierte Prozesse schneller durchgeführt werden können. Wird eine Tätigkeit durch ein voll-automatisiertes System unterstützt, hat dies unmittelbare Auswirkungen auf die Arbeitskraft und den Freiraum für Kreativität im Arbeitsprozess.

Konstant hohe Qualität

Für die Automatisierung selbst müssen der Prüfprozess und dessen einzelne Schritte formalisiert werden, was die Möglichkeit, dass Fehler in der Gestaltungsplanung unentdeckt bleiben, auf ein Minimum reduziert. Eine Formalisierung mindert die Komplexität eines (Teil-)Prozesses selbst nicht, aber weniger komplexe (Teil-)Prozesse könne vermeintlich leichter formalisiert und so automatisiert werden, sodass der Mensch mehr Zeit hat, sich auf die kritischen Prüfungen zu fokussieren. Eine Automatisierung muss nicht schlagartig und vollständig eingeführt werden, sondern kann in Zwischenstufen Teilprozesse übernehmen und so den Mensch bei seiner Arbeit schrittweise bei einzelnen Prozessen entlasten. Bei diesen Zwischenstufen handelt es sich also um eine Halbautomatik, bei welcher der Mensch selbst mit seinem Fachwissen weiterhin die Verantwortung trägt und aktiv an der Prüfung teilnimmt.

Tabelle 5.1: Ausgewählte Kennwerte zu den nationalen Genehmigungsverfahren erhoben (Doing Business, 2015) und ergänzt von (Fiedler, 2015)

Land	Rang	Prozessschritte	Dauer [Tage])	Kosten-Index
Hong-Kong, China	1	6	71	15.4
Singapur	3	11	26	15.7
Deutschland	12	9	97	46.7
Norwegen	21	11	110.5	0.6
Schweden	24	7	116	76.3
UK	27	12	88	66
US	34	16	91	16.7
Schweiz	58	13	154	38.1
Frankreich	92	9	184	244.4
Italien	112	11	233	186.4

Digitale Informationen als Grundlage

Zentrale Grundlage und Ausgangspunkt für eine Automatisierung des Prüfprozesses sind die zugrunde liegenden digitalen Informationen. Eine durchgängige Informationsgrundlage ohne Medienbrüche ist somit Voraussetzung für die Automatisierung.

Ein tiefgreifendes Interesse an einer Automatisierung der Konformitätsprüfung spiegelt sich in den diversen nationalen Bestrebungen wider, in welchen neue Wege für eine allgemeine Digitalisierung des Genehmigungsverfahrens sowie eine Standardisierung und (Teil-)Automatisierung der digitalen Konformitätsprüfung entwickelt werden sollen (Fiedler, 2015). Ein gutes Beispiel ist an dieser Stelle Singapur, denn hier hat man bereits in den 1990er Jahren mit der Einführung von CORENET eine solche Entwicklung gestartet. Den positiven Effekt dieser Entwicklung kann man anhand der Kennwerte zu den Genehmigungsverfahren (siehe auch Tabelle 5.1) ablesen: So dauerte ein Prüfverfahren im Jahr 2007 durchschnittlich 103 Tage, im Jahr 2009 38 Tage und im Jahr 2014 lediglich noch 26 Tage (Doing Business, 2015; Fiedler, 2015).

Fiedler (2015) führt aufeinander aufbauende Modernisierungsszenarien durch die Digitalisierung und Automatisierung des Genehmigungsverfahrens in der Bauplanung ein, welche die Effizienzsteigerung durch die Reduktion des Aufwandes verdeutlichen. Diese sind in Abbildung 5.4 dargestellt.

Mittlerweile ist die Automatisierung der Konformitätsprüfung auch im Fokus der internationalen Standardisierungsarbeit, welche von buildingSMART vorangetrieben wird. 2015 wurde ein eigenes Gremium, der *Regulatory Room*, installiert welcher sich mit der Thematik rund um das Thema ACCC beschäftigt (BuildingSmart, 2016c).

Um die Kerninhalte und Herausforderungen einer solchen Automatisierung aufzuzeigen, sollen in den folgenden Abschnitten die wesentlichen Aspekte anhand der allgemeinen Struktur eines ACCC dargestellt werden. In einem abschließenden Abschnitt wird zudem die Automatisierung zusätzlich kritisch diskutiert.

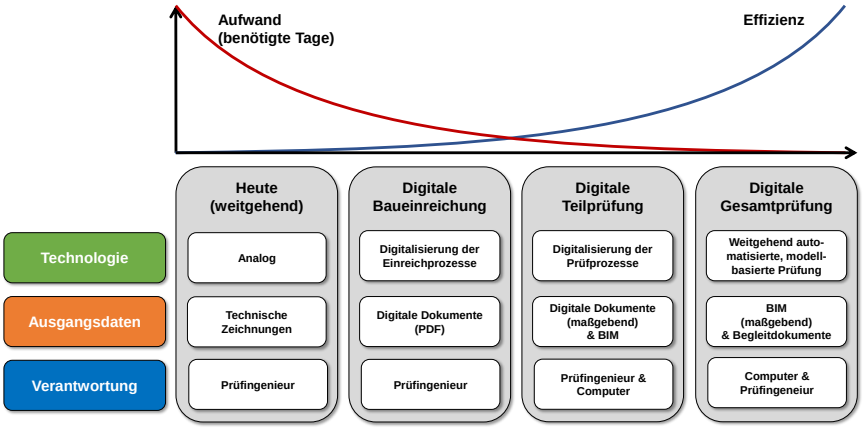


Abbildung 5.4: Entwicklungsstufen der Genehmigungsprüfung in der Bauplanung durch den Einsatz digitaler Bauwerkmodelle und Methoden; angelehnt an die Modernisierungsszenarien bei (Fiedler, 2015)

5.2 Aufbau einer automatisierten Konformitätsprüfung

Um die wesentlichen Herausforderungen einer automatisierten Konformitätsprüfung zu beschreiben, sollen zunächst die allgemeine Struktur sowie die grundlegenden Komponenten eines solchen Prozesses dargestellt werden. Eastman et al. (2009) gliedern den Gesamtprozess in vier Einzelkomponenten: (1) Die Übersetzung der Regeln in eine maschinenlesbare Sprache, (2) die Aufbereitung der Gebäudemodelldaten, (3) die Durchführung des Prüfprozesses und schließlich (4) die Aufbereitung und Darstellung der Prüfergebnisse. Die resultierende Struktur ist in Abbildung 5.5 dargestellt. Anhand dieser Struktur können nun in den nachfolgenden Schritten die Herausforderungen der jeweiligen Einzelkomponenten gezeigt werden, welche sich jeweils stellen.

5.2.1 Übersetzung der regulatorischen Anforderungen

Die Grundlage für eine Automatisierung besteht darin, die regulatorischen Anforderungen, welche in Vorschriften enthalten sind, sowie die damit verbundenen Prüfprozessschritte in eine maschinenlesbare Sprache zu übersetzen. Diese Übersetzung stellt die Kernaufgabe bei der Umsetzung eines ACCC dar.

Die Idee der Digitalisierung von Sprache in mündlicher oder schriftlicher Form existiert seit den Anfängen der Computerwissenschaften und ist auch heute in den verschiedenen Anwendungsbereichen ein relevantes Thema. Es geht darum, den Informationsgehalt von gesprochenem und geschriebenem Wort möglichst präzise in maschinenverarbeitbaren Code zu übersetzen. Das Problem hierbei ist, dass Sprache Informationen in unterschiedliche Formen und Ebenen transportieren kann. Der Sprachwissenschaftler Karl Bühler unterscheidet drei verschiedene Arten von Informationen, welche in Sprache übermittelt werden können (Becker, 2013): Ausdruck (expressiv), Appell (illokutionär, kommunikativ) und Darstellung (deskriptiv, kognitiv). Da sich der Ausdruck der Sprache vor allem auf das Wesen des Senders, also auf ein Individuum, bezieht, kann dieser Aspekt in der vorliegenden Arbeit außer Acht

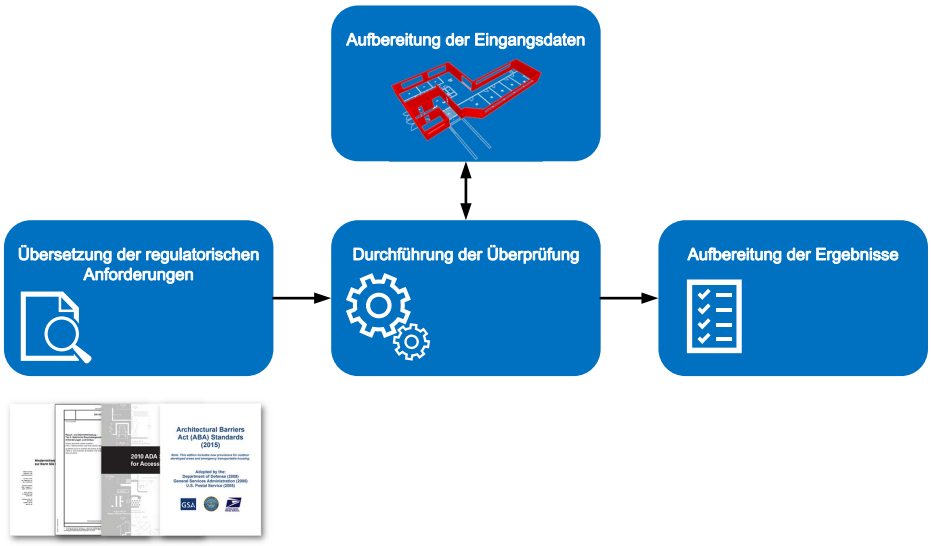


Abbildung 5.5: Allgemeine Struktur des ACCC; angelehnt an (Eastman et al., 2009)

gelassen werden. Die beiden Elemente Appell und Darstellung hingegen beschreiben sehr präzise die Gestalt der Informationen, wie diese auch in Vorschriften enthalten sind. Eine Vorschrift hat die Funktion, dem Bearbeiter einen Sachverhalt darzustellen und anschließend auf dieser Basis die Einhaltung von Regeln und Rahmenbedingungen einzufordern.

Als zentrale Frage stellt sich, wie die enthaltenen Informationen übertragen werden können. Neben einer einfachen manuellen Übertragung gibt es den Lösungsansatz, Sprachinformationen weitgehend automatisiert zu digitalisieren. Für eine solche Übertragung bzw. Übersetzung muss eine Schnittstelle zwischen Mensch und Maschine, welche für die sogenannte Mensch-Maschine-Kommunikation dient (Schenk und Rigoll, 2010), implementiert werden. Ein bekanntes Beispiel für eine solche Schnittstelle ist die Entwicklung von Spracherkennungssoftware, die in den letzten Jahren intensiv vorangetrieben wurde und sich auf eine enge Interaktion von Mensch und Maschine konzentriert. Allerdings müssen bei der Spracherkennung verschiedene Anwendungsformen unterschieden werden, welche sehr grundlegend unterschiedliche Herausforderungen mit sich bringen. So gibt es beispielsweise Diktiersoftware, welche sich auf eine reine Worterkennung fokussiert oder aber Sprachassistenten, welche überdies die Semantik von Spracheingaben, also die eigentliche Intention, erkennen können. Trotz des rasanten technologischen Fortschritts der letzten Jahrzehnte auf diesem Gebiet und den daraus resultierenden Werkzeugen, stellt dieser Bereich immer noch große Herausforderungen an die Entwickler (Becker, 2013).

Bei der technischen Umsetzung lassen sich grundsätzlich zwei Methoden voneinander unterscheiden: Eine vergleichsweise einfache Art der Übersetzung basiert auf der manuellen Übertragung des Prüfprozesses in fest programmierte Programmroutinen oder Methoden. Das bedeutet, dass sich die Digitalisierung der Inhalte eines Codes oder einer Richtlinie auf die Definition von maschinenlesbaren Methoden konzentriert, die für den Anwender in der Regel verborgen bleiben. Daher ist die Lesbarkeit der übersetzten Regeln für den Benutzer eingeschränkt. Die Ausführung des Prüfprozesses ist somit eine

versteckte Prozedur, die nur die ein- und ausgehenden Informationen, nicht aber den Verarbeitungsprozess selbst sichtbar macht und eine Beteiligung eines Benutzers an diesem Übersetzungsprozess nicht berücksichtigt. Diese wird in der allgemeinen Systemtheorie als *Black-Box*-Methode (von Bertalanffy, 1972) bezeichnet und hat den Vorteil der vergleichsweise geringen Fehlerquote des Gesamtprozesses aufgrund der Nähe und dem hohen Standardisierungsgrad der Einzelkomponenten.

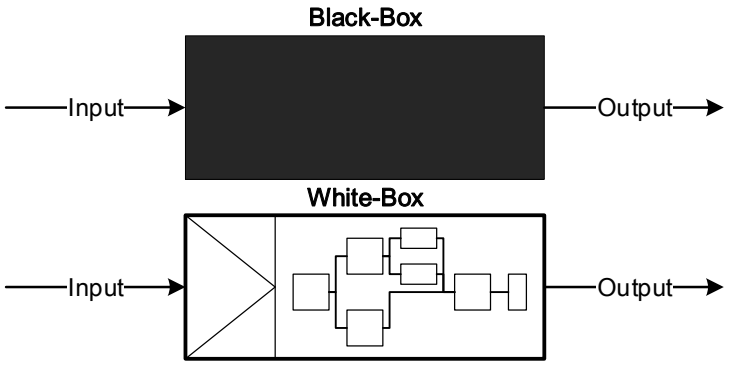


Abbildung 5.6: Schematische Darstellung eines Black-Box und einer White-Box Prozesses, angelehnt an (von Bertalanffy, 1972)

Nach Gross (1996) führen versteckte Verfahren aber auch leicht zu mangelndem Vertrauen in die Ergebnisse. Im Gegensatz hierzu machen *White-Box*-Methoden die internen Verarbeitungsschritte sicht- und damit für den Anwender nachvollziehbar. Um diese Transparenz zu erreichen, muss der Inhalt der übersetzten Richtlinie oder des übersetzten Codes nicht nur für die Maschine, sondern auch für den Anwender lesbar sein. Die einzelnen Vorschriften müssen hierzu mit Hilfe eines Darstellungssystems, welches Zeichen und Regeln definiert, übersetzt werden. Diese Elemente können zur ausreichenden Beschreibung von Objekten, Methoden und Beziehungen verwendet werden. Das Hauptziel besteht nicht nur darin, alle Arten von Informationen, die eine Vorschrift enthalten könnte, abzudecken, sondern auch darin, dem Benutzer die Möglichkeit zu geben, Informationen jederzeit zu verstehen, zurückzuverfolgen und den Fortschritt des Prüfvorgangs nachzuvollziehen. Obwohl die Entwicklung und Implementierung eines solchen Systems im Vergleich zum geschlossenen Prüfverfahren deutlich aufwendiger ist, bieten sie große Vorteile für die Durchführung einer Prüfung. Die Transparenz löst eines der größten Probleme bei der Automatisierung von Prozessen: Trotz des zunehmenden Digitalisierung- und Automatisierungsgrades in der Bauindustrie sind die Planer selbst für die Ergebnisse jedes Prozessschrittes verantwortlich. Diese Verantwortung kann aus rechtlichen Gründen nicht auf eine Maschine oder Anwendung übertragen werden. Die Ergebnisse eines automatisierten Prozesses müssen mit Vorsicht behandelt und im Zweifelsfall manuell überprüft werden. So ist es in der AEC-Branche weitgehend üblich, die Ergebnisse stichprobenartig hinsichtlich ihrer Plausibilität zu prüfen, z. B. durch eine manuelle Berechnung oder den Vergleich mit Faustregeln. Solche Plausibilitätsprüfungen müssen ebenfalls in einem automatisierten Rahmen für die Konformitätsprüfung möglich sein, erfordern aber transparente und einsehbare Prozessschritte.

5.2.2 Aufbereitung der Eingangsdaten

Die wesentliche Grundlage für die Durchführung des eigentlichen Prüfprozesses sind die Bauwerksinformationen, welche in ihrer Gesamtheit die zu prüfende Gestaltungsplanung darstellen. Wie bereits in Abschnitt 4.4 gezeigt wurde, beziehen sich die anzuwendenden Regeln auf jeweils unterschiedliche Eingangsdaten, welche entweder direkt, indirekt oder über die Verwendung einer erweiterten Datenstruktur in den Bauwerksinformationen enthalten sind.

Bevor eine Prüfung durchgeführt werden kann, müssen für eine Regel die Eingangsdaten spezifisch aus den Bauwerksinformationen abgeleitet werden. In diversen Publikationen, wie insbesondere Eastman et al. (2009); Solihin et al. (2017), wird darauf hingewiesen, dass es sinnvoll ist, nur die relevanten Informationen aus den weitverbreiteten Datenmodellen wie insbesondere IFC zu extrahieren und nur diese für die Prüfung heranzureifen.

Zum einen wird darauf verwiesen, dass bei einer Prüfung eine Vielzahl von Regeln und somit Berechnungen an den Informationen durchgeführt werden muss. Die Komplexität und Struktur von IFC führen zu erheblichen Leistungsproblemen und hemmen so die Verarbeitung. Einige Arbeiten, wie beispielsweise bei Pauwels und Roxin (2016), schlagen daher eine allgemeine Vereinfachung der IFC-Datenstruktur vor. Darüber hinaus liegen manche Informationen, welche für die Prüfung notwendig sind, nicht direkt in dem Bauwerksmodell vor bzw. die Datenmodelle lassen auch eine Abbildung der notwendigen Informationen nicht zu. Ein konkretes Beispiel für die Aufbereitung von Bauwerksinformationen für die Konformitätsprüfung liefern Dimyadi et al. (2016a, 2014) mit der Einführung des Fire Compliance Model (FCM). Das FCM wurde für eine Fallstudie entwickelt und beschreibt im neutralen XML-Format alle Mindestangaben, welche für eine auf die Evakuierung bezogene Brandschutzplanung erforderlich sind. Hierzu gehören im Allgemeinen die Geometrie der Räume, insbesondere der Fluchtwege, die Lage der Fluchttüren und deren Abmessungen sowie die Art der Tätigkeiten in den einzelnen Räumen. Typischerweise wird erwartet, dass räumliche Geometrieinformationen in einem Bauwerksmodell zur Verfügung stehen, da es sich um eine wesentliche architektonische Designkomponente handelt. Allerdings fehlt im Modell eine Angabe, welche Wege als Fluchtwege genutzt werden können. Aus diesem Grund müssen spätestens bei der Berechnung der Distanz von Fluchtwegen Methoden oder Operatoren eingesetzt werden, welche diese Informationen aus den Bauwerksinformationen ableiten. In Abbildung 5.7 sind unterschiedliche Teilmengen eines Modells dargestellt, welche sich für unterschiedliche Prüfungen nutzen lassen.

Bei einer Methode zur Automatisierung muss also sehr genau bedacht werden, wie die Informationen aus Bauwerken extrahiert und für die Prüfung aufbereitet werden können.

Ferner müssen vor einer etwaigen Konformitätsprüfung oder einer Ableitung relevanter Informationen die Korrektheit, der erforderliche Detaillierungsgrad (siehe auch LOD in Abschnitt 2.5.2) und die Konsistenz des Gebäudemodells als wesentliche Grundvoraussetzung für den Prüfprozess gegeben sein, damit belastbare Ergebnisse erzeugt werden können (Kulusjärvi, 2012). Entsprechende qualitative Kriterien sowie Prüfprozesse sind in Kapitel 3 ausführlich beschrieben.

5.2.3 Durchführung der Überprüfung

Die Durchführung der Überprüfung bringt die aufbereiteten Bauwerksinformationen und die maschinenlesbaren Prüfroutinen zusammen, um diese schließlich zu einem Ergebnis zu führen. Mögliche

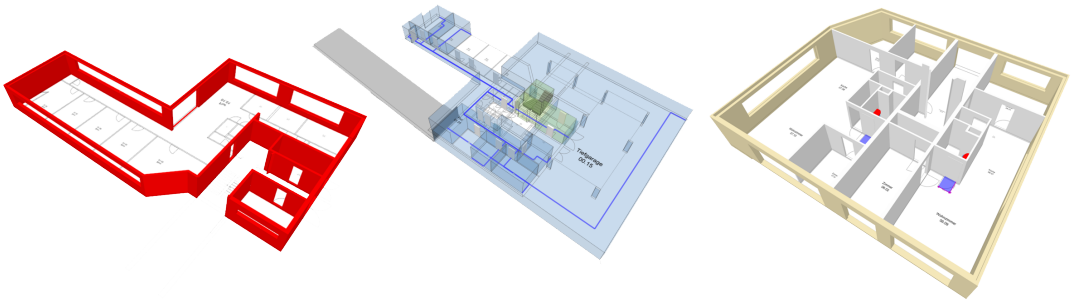


Abbildung 5.7: Darstellung unterschiedlicher Teilmengen eines Modells, welche für unterschiedliche Prüfungen zum Einsatz kommen. Links: Brandschutz; Mitte: Fluchtwegeanalyse; Rechts: Barrierefreiheit

Ergebnisse oder Verfahrensweisen zu einer Prüfung wurden in Abschnitt 3.3 vorgestellt. Generell gilt für diesen Prozess, dass die Fehlerfreiheit der Ergebnisse einer Konformitätsüberprüfung in dem Verantwortungsbereich des Anwenders liegt. Daher kommen bei der zurzeit üblichen, manuellen Überprüfung von Regelwerk und Gestaltungsplanung insbesondere die Erfahrung, die Umsicht und die Sorgfalt des Sachbearbeiters zum Tragen. Es ist empfehlenswert und auch weitgehend Praxis, dass jeder einzelne Schritt einer Überprüfung an Hand von Erfahrungswerten oder überschlägigen Rechnungen hinsichtlich seiner Plausibilität kontrolliert wird. Durch das schrittweise Vorgehen bei der manuellen Bearbeitung lässt sich diese Überprüfung sehr gut in den Arbeitsprozess integrieren. Eine Automatisierung hingegen bringt die Gefahr mit sich, dass diese Kontrolle an Bedeutung verliert und die Verantwortung allein der Maschine übertragen wird, was bereits aus rein rechtlichen Aspekten kritisch gesehen werden muss (Kamaruddin et al., 2016; Mahbub, 2009).

Seit dem Einzug der Computerwissenschaften in das Bauwesen besteht dieses Problem in vielen Bereichen, wie z.B. der Berechnung der Statik, bei dem der Bearbeiter nicht nur auf die erhaltenen Zahlen eines maschinellen Berechnungsprozesses vertrauen kann, sondern in der Pflicht steht, diese Ergebnisse gleichzeitig oder nachlaufend ständig zu überprüfen.

Mangelnde Transparenz und Übersichtlichkeit einer Methode erlauben keine Plausibilitätsüberprüfung während des Prozesses und stellen somit die Ergebnisse grundsätzlich infrage. Dieses Problem kann lediglich gelöst werden, indem der Überprüfungsprozess so transparent und offen wie möglich gestaltet wird.

Die Durchführung der Prüfung ist unter diesem Aspekt maßgeblich von der Maschinen-verarbeitbaren Darstellung der regulatorischen Anforderungen abhängig und daher sei an dieser Stelle auf den vorherigen Abschnitt 5.2.2 verwiesen. Zudem ergibt sich hierdurch eine Diskussion über die Automatisierung sowie einen zugehörigen Grad dieser. In der vorliegenden Arbeit ist diese entsprechend in Abschnitt 5.3 zu finden.

5.2.4 Aufbereitung der Ergebnisse

Ein letzter wesentlicher Aspekt besteht in der Visualisierung, Kommunikation und Weitergabe der erhaltenen Ergebnisse eines Prüfprozesses. Neben der Beschreibung der Probleme für den Anwender

selbst, geht es bei der Prüfung nicht nur darum, Probleme und Fehler in der Planung festzuhalten, sondern diese entsprechend zu kommunizieren und dem verantwortlichen Projektbeteiligten in entsprechender Weise mitzuteilen, sodass dieser Korrekturen vornehmen kann.

Stichprobenartige Überprüfungen, Visualisierungen und einfache Plausibilitätsprüfungen können dem Anwender dabei helfen, die erhaltenen Ergebnisse hinsichtlich ihrer Fehlerfreiheit zu kontrollieren.

Dabei sollte auch beachtet werden, dass es auch bei einer Automatisierung zu mehreren Prüfdurchläufen kommen kann, sodass eine Versionierung und Historie der Kommunikation sinnvoll ist. In Kapitel 2 wurde der Aspekt Kommunikation bereits im Detail vorgestellt. An dieser Stelle kann daher festgehalten werden, dass die Entwicklung in diesem Bereich bereits sehr weit vorangeschritten ist und sich die Ergebnisse auf vielfältige Art und Weise aussagekräftig darstellen lassen. Im Sinne der Interaktion sollte insbesondere die graphische Aufbereitung der Ergebnisse dynamisch möglich sein, d. h. dass zu jedem einzelnen Schritt einer Überprüfung die aktuellen Elemente und Ergebnisse dargestellt werden können.

5.3 Diskussion der Automatisierung

Die steigende Rechenleistung von Computersystemen hat in den diversen Sektoren von Wirtschaft und Industrie dazu geführt, dass immer mehr Prozesse vereinfacht und automatisiert werden, um bei einem gleich bleibenden Zeitkontingent mehr zu erwirtschaften und die Fehlerquote der Prozesse zu optimieren (Spath, 2013). In den letzten Jahren ist in der Baubranche das Thema Automatisierung vermehrt in den Vordergrund gerückt. Unter diesem Begriff ist im Allgemeinen die Übertragung von Funktionen des Produktionsprozesses, insbesondere Prozesssteuerungs- und -Regelungsaufgaben vom Menschen auf künstliche System zu verstehen (Brich, 2014).

Die diversen Vorteile einer Automatisierung wurden bereits zu Beginn dieses Kapitels intensiv behandelt. Im Zusammenhang mit der Prüfung von Normen und Richtlinien gilt es allerdings auch, kritisch zu betrachten, inwiefern eine Automatisierung im Sinne einer Übertragung von Prüffunktionen an künstliche Systeme für eine Anwendung in der Praxis belastbar ist. Diese kritische Betrachtung der Automatisierung wird sehr oft auch anhand der *Ironien der Automation* dargestellt, welche von Bainbridge (1983) eingeführt wurden. Folgende ironische Problemlagen entstehen bei der Automatisierung:

Menschen werden von Entwicklern als wesentliche Fehlerquelle betrachtet und deshalb durch Automatisierung ersetzt. Allerdings sind auch diese Entwickler Menschen und damit anfällig für Fehler. Dies führt dazu, dass eine Reihe von Fehlern auf Entwicklungsfehler zurückgeführt werden kann.

Aufgaben, die sich nicht automatisieren lassen, weil sie möglicherweise zu komplex sind und sich nicht vollständig spezifizieren lassen, werden auf den Menschen als schwächstem Glied in der Prozesskette übertragen.

Menschen werden durch Automatisierung ersetzt, weil die Systeme die Aufgaben besser durchführen können. Menschen überwachen und prüfen allerdings weiterhin die Systeme auf die Korrektheit der Arbeitsprozesse. In Störfällen muss der Mensch dann eingreifen und gegebenenfalls manuell übernehmen.

Die zuverlässigsten Automatisierungssysteme erfordern den höchsten Aufwand an Trainingsmaßnahmen, weil sich im täglichen Betrieb keine Gelegenheit für aktive Kontrolle und Auseinandersetzung

mit dem System bietet. Unzuverlässige Systeme hingegen erfordern regelmäßiges aktives Eingreifen und Hineindenken in die funktionalen Zusammenhänge und erhalten damit die manuellen Kontrollfähigkeiten aufrecht, was den Trainingsaufwand reduziert.

Anhand dieser Ironien kann man erkennen, dass dem Menschen trotz der Automatisierung noch immer eine wesentliche Rolle zukommt, da dieser nicht nur als Prüf- bzw. Kontrollinstanz fungiert, sondern im Notfall auch in der Lage sein muss, die Kontrolle wieder zu übernehmen. Ist ein Prozess allerdings bereits automatisiert und eingespielt, so werden dem Menschen in seiner Rolle oft nicht die geeigneten Mittel, z. B. in Form von Wissen, zur Verfügung gestellt, um diese Funktion zu erfüllen (Wiener und Curry, 1980). Häufig ist in diesem Zusammenhang von dem sogenannten *Automatisierungsdilemma* die Rede.

Dieser Aspekt ist somit hochrelevant für die Gestaltung einer Mensch–Maschine Interaktion, um ein adäquates Situationsbewusstsein zu unterstützen. Bei der Kommunikation zwischen Mensch und Maschine muss sichergestellt werden, dass der menschliche Anwender zu jedem Zeitpunkt die notwendige Information über den Zustand des Prozesses und die Möglichkeiten für Kontrolleingriffe der Automatisierungssysteme in intuitiver Weise erhält. So kann die Verantwortung trotz der Automatisierung und eines gewissen Grades an Selbstständigkeit der Maschine bei dem Anwender verbleiben. Allerdings geht der allgemeine Trend eher dahin, dass der Mensch mehr und mehr aus der Kontrollschleife herausgenommen und eine zunehmende Distanz zum tatsächlichen Kontrollprozess geschaffen wird. Ein Effekt ist, dass praktische Kenntnisse darüber verloren gehen, wie ein Prozess durchzuführen oder aber zu steuern (Wiener und Curry, 1980).

Gleichzeitig ist sicherlich auch unbestritten, dass die Einführung von Automatisierungssystemen zum einen zu einer Produktionssteigerung und zum anderen zu einer Erhöhung der Sicherheit beigetragen hat. In diesem Kontext wird auch von einer weiteren *Industriellen Revolution* gesprochen, welche durch Elektronik und IT sowie eine passive Automatisierung von Produktionsprozessen gekennzeichnet ist (BMWi, 2013).

Zusammenfassend kann festgehalten werden, dass die Automatisierung von standardisierten Prozessen ein enormes Potential durch eine Qualitätssteigerung, Beschleunigung und Vereinheitlichung bei einer gleichzeitigen Verbesserung der Arbeitsverhältnisse für die Anwender birgt. Es muss jedoch darauf geachtet werden, dass die Verantwortung und alle zugehörigen Kontrollprozesse beim Anwender verbleiben. In diesem Zusammenhang ist also eine Vollautomatisierung im Sinne einer vollständigen Übertragung der Verantwortung auf ein künstliches System daher kritisch zu prüfen.

5.4 Zusammenfassung und zentrale Herausforderungen des ACCC

Die Konformitätsprüfung der Gestaltungsplanung hinsichtlich geltender Normen und Richtlinien spielt im Bauwesen eine essenzielle Rolle. Die Qualität der Planungsgrundlagen muss kontinuierlich auf ihre Richtigkeit und Einhaltung der geltenden Anforderungen überprüft werden. Gegenwärtig ist dieser Kontrollprozess mühsam, umständlich und fehleranfällig, da dieser zumeist manuell auf Basis der zweidimensionalen Planung und iterativ bei jeder Planungsänderung durch den zuständigen Planungsingenieur durchgeführt werden muss.

Eine Automatisierung dieses hochrelevanten Prozesses mit Hilfe von digitalen Methoden, wie insbesondere dem Building Information Modeling, birgt das Potenzial, den Arbeitsaufwand und somit

die Kosten zu reduzieren und gleichzeitig den Planungsingenieur von monotoner, iterativer Arbeit zu befreien, damit dieser mehr Zeit für zentrale, kreative Arbeiten hat.

Die zentralen Herausforderungen bei der technischen Umsetzung einer automatisierten Konformitätsprüfung kann an der schematischen Grundstruktur des Gesamtprozesses festgemacht werden, welche sich in vier einzelne Prozessschritte aufteilt: die Übersetzung des Regelwissens aus den regulatorischen Anforderungen in ein formales und maschinenlesbares Format, die Aufbereitung der Gebäudemodelldaten für die Prüfung, der eigentliche Prüfprozess und schließlich die Aufbereitung der erhaltene Ergebnisse für die erforderliche Kommunikation mit den Projektpartnern. Aus den Ausführungen der vorigen Abschnitte lassen sich als Schlussfolgerung die folgenden zentralen Herausforderungen als Fragestellungen formulieren (Eastman et al., 2009; Nisbet et al., 2008; Solihin, 2016):

- Welcher Ansatz ist geeignet, um die Inhalte von Regelwerken, Richtlinien sowie benutzer-/projektspezifischen Anforderungen mit einer enormen Variationsbreite und Komplexität zu formalisieren?
- Wie kann die komplexe Struktur von Bauvorschriften erfasst werden und gleichzeitig für Experten in der Domäne zugänglich und einfach zu bedienen sein, um Regeln selbst zu formulieren und die erhaltenen Ergebnisse auf deren Korrektheit hin zu überprüfen?
- Können Sprachverarbeitungstechniken eingesetzt werden, um die vorhandenen Codes in natürlicher Sprache in eine maschinenlesbare Form umzuwandeln?
- Mit welchen Methoden können Domäne-Experten bei der Definition von Regeln unterstützt werden, insbesondere bei der räumlichen Semantik? Wie können Entitäten und Operatoren mit räumlicher Semantik als Bestandteile einer solchen Darstellung bereitgestellt werden, um ihre direkte Verwendung für Bauvorschriften zu erleichtern?
- Eignet sich eine domänenspezifische, standardisierte Sprache und wie müsste diese hinsichtlich Struktur und Grammatik aufgebaut sein? Was sind die wesentlichen Merkmale und Fähigkeiten einer solchen Sprache, um einer komplexen und großen Variation der Regeln gerecht zu werden, ohne dass die Grammatik auch bei zukünftigen Erweiterungen geändert werden muss? Gibt es weitere Anforderungen an eine solche Sprache, damit diese schließlich auch in der Praxis sinnvoll Anwendung findet?
- Gibt es ein geeignetes Format, um die formalisierten Anforderungen zu erfassen, sodass das enthaltene Wissen verlustfrei ausgetauscht werden kann?
- Wie muss ein Gebäudeinformationsmodell aufbereitet werden, um eine geeignete Grundlage für die automatisierte Prüfung zu schaffen?
- Welche Informationen müssen von dem digitalen Gebäudemodell für den Überprüfungsprozess zur Verfügung gestellt werden und über welche Methoden werden diese Informationen gewonnen, falls sie nicht im Modell enthalten sind? Sind Teilmengen eines Modells, sogenannte Modellsichten, erforderlich?
- Wie können die Informationen, welche in Gebäudedaten vorliegen, so transformiert werden, dass das formalisierte Regelwissen auf diese angewendet werden kann?
- Wie sollte ein automatisierter Überprüfungsprozess strukturiert sein, um die Gestaltungsphase eines Bauwerks zu optimieren?

- Wie können die nach dem Prozess erhaltenen Ergebnisse veranschaulicht und hinsichtlich ihrer Plausibilität überprüft werden?
- Welche Methoden der Mensch-Maschine-Interaktion können zur Unterstützung der Experten herangezogen werden? Wie lassen sich die Ursachen von Fehlern auf intuitive Art und Weise darstellen?
- Können auch Prozeduren zur automatischen Korrektur von Fehlern erfasst werden?

6 Stand der Technik

6.1 Überblick

In den vergangenen Jahrzehnten wurde eine Vielzahl von Ansätzen mit dem Ziel entwickelt, eine automatisierte Regel- oder aber Konformitätsprüfung für das Bauwesen zu ermöglichen. Die in Abbildung 6.1 dargestellte zeitliche Abfolge und Anzahl dieser Entwicklungen zeigt auf, welche anhaltende und mittlerweile stark zunehmende Bedeutung die Automatisierung der Prüfprozesse für das Bauwesen hat.

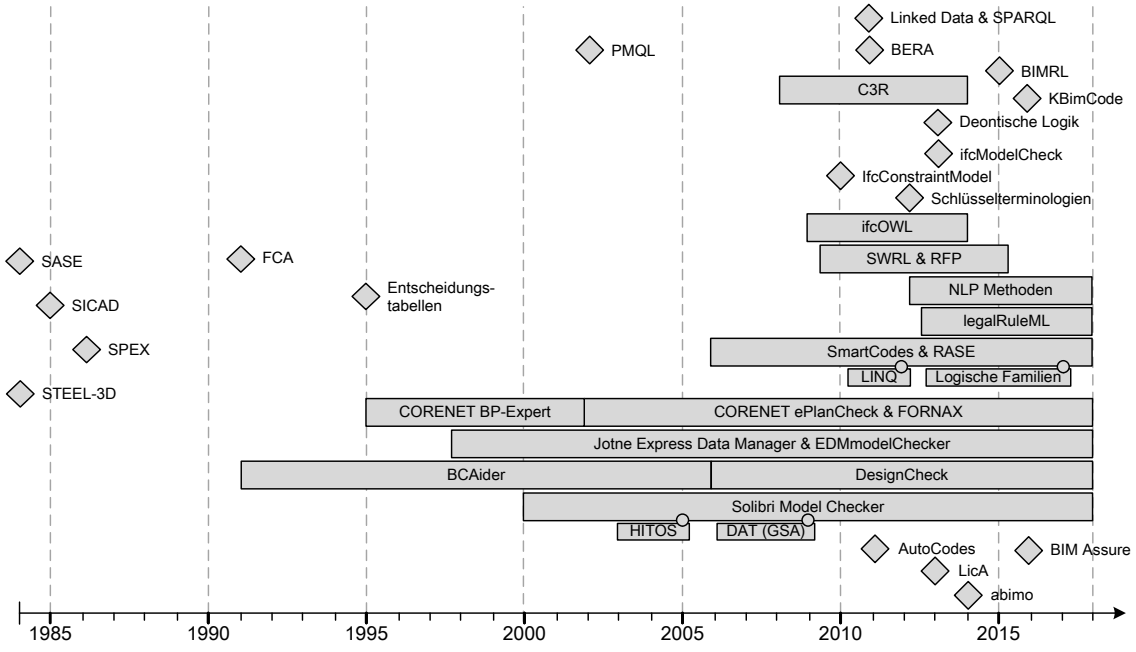


Abbildung 6.1: Chronologische Entwicklung des Automated Code Compliance Checking (ACCC); angelehnt an (Dimyadi und Amor, 2013)

Einen Überblick zu zurückliegenden sowie aktuellen Entwicklungen im Bereich der Automatisierten Konformitätsprüfung bieten Eastman et al. (2009), Dimyadi und Amor (2013) sowie Nawari (2012b).

In den folgenden Abschnitten werden ausgewählten Ansätze, wenn möglich thematisch zusammengefasst vorgestellt, um so den aktuellen Stand der Technik aufzuzeigen.

6.2 Ansätze auf Basis logischer Familien

Einer Reihe von Ansätzen zur Regelprüfung von digitalen Gebäudemodellen ist gemein, dass sie sich maßgeblich darauf konzentrieren, Regelspezifikationen formal zu strukturieren und so in eine maschinenlesbare Form zu bringen. Diese Ansätze führen hierbei einen Formalismus ein, welcher sich dazu eignet, Inhalte von Regelspezifikationen als berechenbare Objekte und so gegebenes Regelwissen digital darzustellen.

Wendt (1998) beschreibt einen Formalismus als “Menge von Vorschriften zur Transformation von Formen“. Aus den eingeführten Formen können nach vorgegebenen Regeln wiederum neue Formen erzeugt werden. Formen und Transformationsregeln können dabei so ausgewählt werden, dass sich diese für den beabsichtigen Zweck, in dem vorliegenden Falle die Darstellung von Regelwissen, eignen.

Ein wesentliches Merkmal dieser Ansätze ist, dass sie zumeist auf der Annahme basieren, dass die verwendeten BIM-Modelle sämtliche Informationen beinhalten, welche für die Prüfung der Regeln benötigt werden. Der Aspekt einer Aufbereitung der Modelldaten, wie dieser in Abschnitt 5 beschrieben wurde, wird hierbei also weitgehend außer Acht gelassen. Das bedeutet, dass die Formalisierung des Regelwissens im Vordergrund steht, nicht aber die Frage, wie die eingehenden Daten, die für die Prüfung der Regel erforderlich sind, erzeugt werden. Vielmehr stehen bei den Bemühungen die Regeln selbst im Vordergrund, welche in einer lesbaren und somit anwenderfreundlichen Form in natürlicher Sprache geschrieben sind, wie dieses bei Bauvorschriften häufig der Fall ist. Ziel ist es, Strukturen zu finden, mit denen Normen und Richtlinien, die in natürlicher Sprache verfasst sind, in computerverarbeitbarer Form abzubilden.

Ein Beispiel für die Formalisierung ist die Anwendung von logischen Familien. Ein zentraler Vertreter dieser Familien ist die Aussagenlogik, welche die Verknüpfungen zwischen atomaren, also unteilbaren, Elementaraussagen beschreibt. Sie gilt als die einfachste Form der Logik (Schenke, 2013) und geht auf die Überlegung des englischen Mathematikers George Boole (1815–1864) zurück, dass jedes System auf diese Verknüpfungen zwischen atomaren Elementen zurückzuführen ist. Ergebnis dieses Ansatzes sind die sogenannten Booleschen Ausdrücke, mit deren Hilfe Systeme über verschiedene Domänen hinweg in ihre Einzelteile zerlegt werden können.

Die Aussagenlogik kann somit auch auf das im Bauwesen angewandte Regelwissen angewendet werden und dieses als solch verknüpfte atomare Aussagen darstellen. Bauvorschriften lassen sich so abstrahieren und mit mathematischer Präzision formalisieren. Mit Hilfe von logischen Operatoren können Aussagen sowohl zur Mengenlehre als auch zur logischen Falsifizierbarkeit getroffen werden. Hierfür stehen neben den Operatoren die beiden Konstanten *True* und *False* zur Verfügung, die jeweils identifizieren, ob eine Aussage vollständig zutrifft oder nicht. Diese drei Elemente der Syntax - Mengenlehre, logische Operatoren und Falsifizierbarkeit – bilden eine Grundlage, um die Inhalte einer Vorschrift zu formalisieren. Die formalen Symbole der Aussagenlogik sind in Abbildung 6.1 dargestellt.

Tabelle 6.1: Formale Symbole der Aussagenlogik (Schenke, 2013)

$\wedge$	Konjunktion	für “und"
$\vee$	Disjunktion	für “oder"
$\neg$	Negation	für “nicht"
$\rightarrow$	Implikation	für “wenn ... dann"
$\leftrightarrow$	Äquivalenz	für “genau dann ... wenn"

Eine Ausprägung der Aussagenlogik, die Prädikatenlogik, auch First Order Logic (FOL) genannt, findet weit verbreitete Anwendung. Diese logische Familie baut auf den drei grundlegenden Komponenten der Aussagenlogik auf und erweitert diese um quantitative Aussagen/Operatoren. Die sogenannten Quantoren, definiert als logische Symbole $\forall$ Allquantor und $\exists$ Existenzquantor, sind vergleichbar mit den Sprachfragmenten "für alle" oder "es gibt" und helfen bei der Formulierung, ob eine Aussage auf eine Gesamt- oder Teilmenge zutrifft. Dadurch lässt sich die logische Falsifizierung wesentlich individueller gestalten, da diese nicht nur Aussagen über *wahr* ($True$) oder *falsch* ($False$) trifft, sondern diese gleichzeitig auf eine definierte Menge begrenzt. Ein Beispiel für eine in der Prädikatenlogik formulierte mathematische Aussage ist (Schenke, 2013):

$$\forall X \forall Y (X < Y) \rightarrow (X + 1 < Y + 1)$$

"Für alle X und für alle Y gilt X < Y. Daraus folgt, dass X+1 < Y+1."

Darüber hinaus können mit Hilfe der FOL über sogenannte nichtlogische Symbole auch außermathematische Aussagen formuliert werden. Hierzu gibt Schenke (2013) folgendes Beispiel:

$$gibt(haensel, gretel, ein\_brot)$$

„Hänsel gibt Gretel ein Brot."

Die Objekte $haensel$, $gretel$, ein_brot und $gibt()$ werden als nichtlogische Bestandteile bezeichnet, da sie sich außerhalb einer mathematischen Formulierung bewegen. Das Gebilde $gibt()$ wird als Prädikatssymbol P bezeichnet und fungiert als Funktion, die für eine bestimmte Anzahl von Objekten eine Aussage formuliert. Das Beispiel stellt also eine Instanziierung der allgemeinen Formel $gibt(X, Y, Z)$ mit den Objekten $haensel$, $gretel$ und ein_brot , welche als Funktionssymbole p (oder q) bezeichnet werden, dar. Eine Aussage mit Prädikats- und Funktionssymbol gemäß der Prädikatenlogik kann also allgemein als

$$P(p, q, [\dots])$$

definiert werden, wobei $p, q, [\dots]$ eine beliebige Menge an Funktionssymbolen darstellt. Mit dieser Definition können sowohl neue Aussagen formuliert, als auch bestehende in ihre Bestandteile zerlegt werden (Schenke, 2013). Die Aussagen- und Prädikatenlogik stellen die beiden wichtigsten logischen Formalismen der Informatik und weiterer Wissenschaften dar, mit welcher Aussagen sowohl für Mensch als auch Maschine les- und interpretierbar formuliert werden können. Ein Beispiel für eine etablierte Implementierung der Prädikatenlogik in den Computerwissenschaften ist die Programmiersprache $PROLOG$ (Bramer, 2013; Eastman et al., 2009; Schenke, 2013). Neben der FOL gibt es noch weitere logische Familien, die auf der Aussagenlogik aufbauen und weitere logische Elemente in die Syntax einführen, um Aussagen auf andere Art und Weise präzise zu formulieren.

Einer der ersten Ansätze zur Darstellung von Regelwissen im Bauwesen in einer systemunabhängigen, maschinenlesbaren Form mit Hilfe der FOL und formalen Formulierungen wurde von Kerrigan und Law (2003) vorgestellt. In dem Ansatz wird jedoch kein digitales Gebäudemodell als Eingabedatum für die Konformitätsprüfung verwendet, sondern alle erforderlichen Fakten mittels eines interaktiven Systems, bei dem der Nutzer eine Vielzahl von Fragen beantworten muss, gesammelt. Die Fakten werden anschließend von einem Programm genutzt, um die Übereinstimmung des Projekts mit den US-Vorschriften zum Schutz der Umwelt formal zu überprüfen. Regeln mit geometrischer oder räumlicher Semantik waren nicht Gegenstand des Forschungsprojekts.

Conditions					Actions		
Location	Type	Wall position with respect to ground	Primary heating source	Thermal resistance value: R-value (RSI)	Compliance checking result	Comments	Reference index
Toronto (Region A, ON)	House	Above	Electricity	R-value > =4.4	Pass	Good	MNECH 3.3.1.1 (Toronto) Web Link 1
			Propane, oil, heat pump	R-value > =3.0	Pass	Good	
			Natural gas	R-value > =2.9	Pass	Good	
			Electricity	R-value < 4.4	Fail	Increase insulation layers so that R > =4.4 RSI	
			Propane, oil, heat pump	R-value < 3.0	Fail	Increase insulation layers so that R > =3.0 RSI	
			Natural gas	R-value < 2.9	Fail	Increase insulation layers so that R > =2.9 RSI	
		Below	Electricity	R-value > =3.1	Pass	Good	MNECH 3.3.2.1 (Toronto) Web link 1
			Propane, oil, heat pump	R-value > =3.1	Pass	Good	
			Natural gas	R-value > =2.1	Pass	Good	
			Electricity	R-value < 3.1	Fail	Increase insulation layers so that R > =3.1 RSI	
			Propane	R-value < 3.1	Fail	Increase insulation layers so that R > =3.1 RSI	
			Natural gas	R-value < 2.1	Fail	Increase insulation layers so that R > =2.1 RSI	
	Building	Any	Any	Any	Exceptional	N.A.	MNECB Web link 2

Note: N.A.=not applicable.

Abbildung 6.2: Entscheidungstabelle zur Überprüfung des Wärmedurchgangswiderstandes einer Außenwand in Toronto (Tan et al., 2010)

6.2.1 Entscheidungstabellen

Bereits in den 1960er Jahren führte Fenves (1966) eine Methode zur automatisierten Konformitätsüberprüfung im Bereich des Stahlbaus ein, welche auf der Anwendung von Entscheidungstabellen (engl. *decisions tables*) basiert. Mit Hilfe dieser Tabellen sollten die Regularien des American Institute of Steel Construction (AISC) (American Institute of Steel Construction, 2015) in eine von Computern interpretierbare Sprache übersetzt werden (Fenves et al., 1995). In einer Entscheidungstabelle werden einzelne Regeln der Gestaltungsplanung als logische Entscheidungsfälle zeilenweise abgebildet. Im Prinzip stellt also jede Zeile eine einzelne formale logische Aussage dar.

Die auf diese Weise aufeinander aufbauenden Regeln führen zu einer zwingend beinhalteten logischen Falsifizierbarkeit, sodass bei den abgebildeten Entscheidungsfällen eine eindeutige Zuordnung, ob die jeweilige Aussage wahr oder falsch ist,vorgenommen werden kann. So ist nicht nur die Lesbarkeit für Mensch und Maschine gegeben, sondern die Tabelle gibt auch für komplexe Entscheidungsfälle durch das Zusammenwirken vieler einzelner Regeln innerhalb einer Entscheidungstabelle einen eindeutigen Wert zurück. Das System bildet damit den Inhalt eines Regelwerks präzise ab (Nawari, 2012b). Ein Beispiel für eine solche Entscheidungstabelle ist in Abbildung 6.2 dargestellt.

Im Jahr 1984 wurde vom US National Bureau of Standards (NIST) als Grundlage für eine umfassende Sammlung von Regelwissen in Form von Entscheidungstabellen das sogenannte SASE-Datenmodell eingeführt (Fenves et al., 1987). Mit Hilfe des Standards Analysis, Synthesis, and Expression (SASE) können Entscheidungstabellen aus den verschiedenen Fachdisziplinen des Bauwesens gesammelt, gespeichert und verwaltet werden. Dieses Datenmodell stellt den ersten Schritt in Richtung einer umfassenden automatisierten Konformitätsüberprüfung dar. Auf Grundlage der Entwicklung der Entscheidungstabellen und des SASE wurden diverse Applikationen entwickelt, welche das Prinzip der Entscheidungstabellen in die Praxis überführen. Als ein Ergebnis wurde im Jahr 1984 von der *Carnegie Mellon University* die Applikation *STEEL-3D* vorgestellt, welche die Regelwerke des AISC im Bereich der Gestaltungsplanung

von Stahlrahmen mit Hilfe der Entscheidungstabellen in ein CAD-System integrierte und auf diese Weise eine erste Konformitätsüberprüfung direkt während der Gestaltungsphase des Bauwerks ermöglichte.

Das SASE-Datenmodell wurde schließlich von Lopez und Elam (1985) aufgegriffen und in eine CAD-Umgebung eingebettet. Die so entstandene Standards Interface for Computer Aided Design (SICAD) unterstützt den Planer nicht nur durch die Bereitstellung der Regelwerke während der Gestaltungsplanung, sondern fördert auch die Interaktion zwischen Nutzer und System, indem es bei fehlenden oder widersprüchlichen Informationen Nutzereingaben einfordert. Eine weitere Implementierung des SASE-Datenmodells wurde im Jahre 1986 als Applikation Standards Processing Expert (SPEX) eingeführt, welche sich insbesondere auf Konformitätsüberprüfungen bei der Auswahl von Material und geometrischen Formen konzentriert Dimyadi und Amor (2013).

Delis und Delis (1995) verfolgen mit ihrem Prüfwerkzeug Fire-Code Analyzer (FCA) einen Ansatz, welcher der Idee zu den Entscheidungstabellen nahe kommt. Mit dem FCA können einzelne konditionale Entscheidungsfälle mit Hilfe einer Implikation, also in der "Wenn-Dann"-Form, abgebildet und durch eine eigene Syntax individuell und somit detailliert ausgestaltet werden. Durch die vorgegebene Syntax und Vorgehensweise ist die formulierte Vorschrift logisch falsifizierbar und kann auf ihre Gültigkeit hin überprüft werden. Das resultierende FCA-System besteht aus einer Wissensdatenbank und einer interpretierenden Instanz (eng. *Interpreter*). Bei der Implementierung orientierte man sich hauptsächlich an den Anforderungen und Paragraphen des Life Safety Code (LSC) (National Fire Protection Association, 2017), einem weltweit anerkannten Sicherheitsstandard im Bereich des Brandschutz.

Im FCA setzt sich eine formalisierte Regel aus einem Bedingungs- und Konsequenz-Objekt, welche zueinander in Abhängigkeit stehen, zusammen. Nur wenn die voranstehende Bedingung, welche auch aus mehreren Einzelbedingungen bestehen kann, erfüllt ist, kann auch die Konsequenz erfüllt sein. Im Grunde handelt es sich also auch hier um die Definition einer Implikation. Das Konsequenz-Objekt bildet in der Regel ein Ergebnis oder eine Aktion, welche zu dem Resultat der Überprüfung führt (Delis und Delis, 1995). Als Beispiel für die Übersetzung eines Regelwerks mit Hilfe des FCA soll an dieser Stelle der Paragraph 12.3.1 der *National Fire Protection* (National Fire Protection Association, 2017) dienen:

> *The fire resistance rating of enclosures in health care occupancies protected throughout by an approved automatic sprinkler system may be reduced to 1 hour in buildings up to an including, three stories in height.*

Dieser Absatz kann nun wie folgt mit der Programmiersprache LISP übersetzt werden:

```
1 (IF   (AND  (THE SPRINKLER-PRESENCE OF A BUILDING IS YES)
2     (?SPACE IS IN CLASS VERTICAL-OPENINGS)
3     (THE SUPERSPACE OF ?SPACE IS ?ZONE)
4     (?ZONE IS IN CLASS FIRE-ZONES)
5     (LISP << (THE STORIES-ABOVE-GRADE OF BUILDING) 4)))
6 (THEN   (A REQUIRED-ENCLOSURE-RATING OF ?SPACE IS 1)))
```

Quellcode 6.1: Übersetzung der NFP 12.3.1 mit Hilfe von LSIP im FCA nach (Delis und Delis, 1995)

In Zeile 1 bis 5 sind durch das Signalwort IF alle Bedingungen definiert, welche erfüllt sein müssen, damit die Konsequenz – markiert durch das Signalwort $THEN$ - in Zeile 6 zum Tragen kommt. Dieses bedeutet, dass der Regel-Interpreter zunächst alle Anforderungen der Bedingungen überprüft und

Frame : DOOR1			
Slot	*ValueClass*	*Source*	*Comment*
CONNECTED_SPACE	NIL	CALC	ROOMS and CORRIDORS connected by this DOOR
NET_BREADTH	NUMBER	INP	Feet
RATING	NUMBER	INP	Hours
SWINGS_INTO	NIL	INP	ROOM or CORRIDOR into which this DOOR swings
TRAVEL_DIS_TO_EXIT	NUMBER	CALC	Travel Distance in feet
BREADTH	NUMBER	CALC	Gross Breadth in feet
FCA_PROBLEM	STRING	CALC	Description of Problem
COORDS	NIL	INP	(X,Y) or (X,Y,Z)
INROOM_TRAVEL_DIS	NUMBER	CALC	Travel Distance in feet
AREA	NUMBER	CALC	Square feet
WALL	STRING	CALC	Name of WALL in which opening occurs
HEIGHT	NIL	INP	Measured in feet
NUM_LEAVES	NUMBER	INP	Number of leaves
DOOR_TYPE	NIL	INP	Special features of door (e.g., smoke barrier)
EXIT_PATH	NIL	CALC	List of Nodes
VISION_PANEL	NIL	CALC	Name of DOOR's vision panel, if any

Abbildung 6.3: Datenobjekt Tür mit den zugehörigen Parametern (Source = „INP“) und geometrischen Algorithmen (Source = „CALC“) im FCA (Delis und Delis, 1995)

damit beginnt, die Anweisung aus Zeile 1, also die Existenz einer Sprinkleranlage, zu prüfen. Sind alle Bedingungen erfüllt, kann die Konsequenz in Zeile 6 schließlich in Kraft treten und der Feuerwiderstand um eine Stunde gemindert werden.

Um die in der Regel enthaltenen Informationsabfragen durchführen zu können, muss auch der entsprechende Informationsgehalt von einem Datenmodell zur Verfügung gestellt werden. Hierfür ist in dem FCA das eigens eingeführte *frame system* verantwortlich, welches die Informationen eines Gebäudemodells als einzelne Objekte hierarchisch strukturiert und mit zugehörigen Parametern zur Verfügung stellt. Da die im FCA behandelten Regelungen stark geometrisch orientiert sind, gibt es eine Sammlung geometrischer Algorithmen, mit deren Hilfe zusätzliche Informationen aus den Objekten des *frame system* erzeugt werden können. Jedes einzelne Objekt des Datenmodells besitzt neben seinen festgelegten Parametern auch eine Liste von anwendbaren geometrischen Operationen, welche die zusätzlichen Informationen über den festgelegten Algorithmus aufbereiten (Delis und Delis, 1995). Eine Liste ist beispielhaft in Abbildung 6.3 für das Datenobjekt Tür dargestellt.

6.2.2 Deontische Logik

Eine weitere logische Familie, welche für die Formalisierung von Regelwissen verwendet werden kann, ist die Deontische Logik. Die Deontologie, auch Deontische Ethik oder Pflichtethik genannt, umfasst sämtliche Theorien, die sich mit Pflichten und Rechten von Handlungen beschäftigen (Schenke, 2013). Aufbauend auf der Aussagenlogik wird in der Deontischen Logik die Syntax um weitere logische Symbole erweitert, welche für die Formulierung von Aussagen verwendet werden können. In der deontischen Logik werden die Symbole O (”es ist obligatorisch, dass”), F (”es ist verboten, dass”) und P (”es ist erlaubt, dass”) zu der Syntax hinzugefügt (Salama und El-Gohary, 2013; Schenke, 2013). Schenke (2013) führt ein Beispiel für die Anwendung der deontischen Logik an, bei dem G ein Element der klassischen Aussagenlogik darstellt:

$$OG \leftrightarrow \neg P \neg G$$

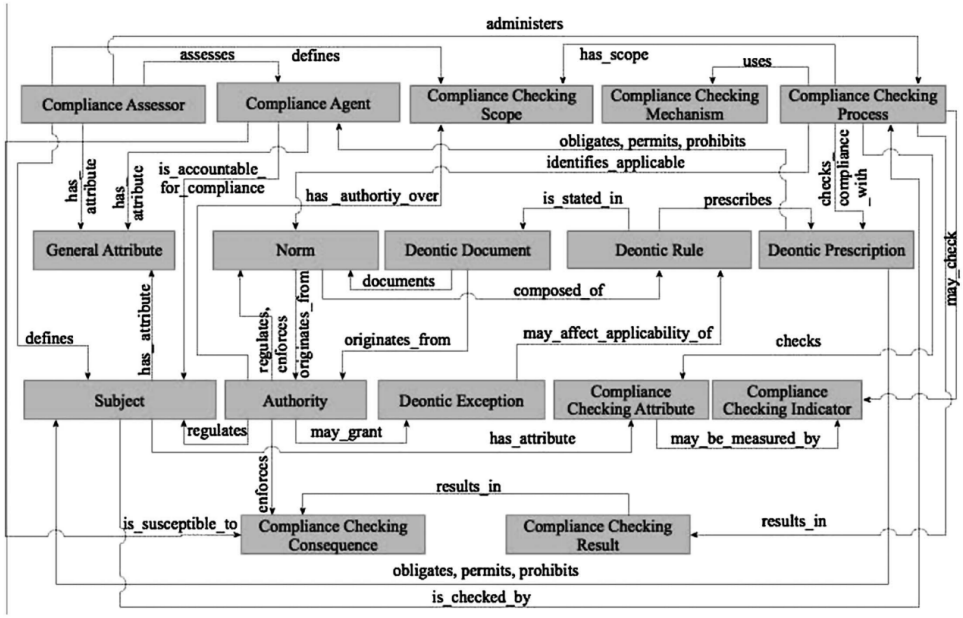


Abbildung 6.4: Darstellung der Konformitätsprüfung mit Hilfe des Deontischen Modells (Salama und El-Gohary, 2013)

Wenn Aussage G obligatorisch ist, folgt daraus, dass es nicht erlaubt ist, dass G nicht zutrifft.

$$FG \leftrightarrow \neg PG$$

Wenn Aussage G verboten ist, folgt daraus, dass es nicht erlaubt ist, dass G zutrifft.

Eine Vielzahl von Normen und Vorschriften im Bauwesen stellt zunächst einen Sachverhalt dar und fordert anschließend die Einhaltung von Randbedingungen und Grenzwerten ein. Daher bietet es sich an, die Operatoren der Deontischen Logik zu verwenden und so die Aussagen möglichst direkt und präzise zu formalisieren.

Salama und El-Gohary (2013) führen ein deontisches Modell ein, mit dessen Hilfe Bauvorschriften auf Basis der deontischen Logik formalisiert und so digital abgebildet werden können. Das Modell verwendet dabei drei Basiselemente: Konzepte, Relationen und deontische Axiome. Die Konzepte repräsentieren übergeordnete Entitäten des normativen Denkens oder der Domäne Bauwesen (z. B. ein Bauteil, eine Norm, ein Ergebnis einer Prüfung, eine Handlungsweisung). Relationen verbinden die Konzepte miteinander und stellen so hierarchische oder interkonzeptionelle Systeme her. Die deontischen Axiome spezifizieren die Definitionen der Begriffe und Beziehungen in der Deontologie. Auf diese Weise werden die Einschränkungen abgebildet. Das resultierende deontische Modell ist in Abbildung 6.4 dargestellt.

Das Modell von Salama und El-Gohary (2013) stellt einen ersten Schritt in Richtung der Verwirklichung eines ACCC-Systems für das Bauwesen dar, welches mit der Deontischen Logik arbeitet. In einer ersten Evaluierung des Modells wurde gezeigt, dass es für die Beschreibung formaler Anfragen, automatisierte Prüfung von Redundanzen, Expertenbewertung und anwendungsorientierte Bewertung verwendet

werden kann. Allerdings sind diese Entwicklungen nicht weiter verfolgt worden, und es gibt keine weiteren praxisrelevanten Studien zu der Tragfähigkeit dieses Ansatzes.

6.3 Ansätze auf Basis von Auszeichnungssprachen

Bereits seit einigen Jahrzehnten dienen Auszeichnungssprachen (engl. *markup languages*) in den Computerwissenschaften dazu, Texte und andere Daten zu gliedern und zu formatieren, um diese lesbar für Maschinen zu machen. Gegenwärtig findet diese Methode eine weit verbreitete Anwendung, um Informationen zwischen verschiedenen Computersystemen auszutauschen. Die Idee einer Auszeichnungssprache ist es, den Inhalt eines Dokuments gemäß einer definierten Syntax so zu kennzeichnen und zu strukturieren, dass dieser für Maschinen les- und interpretierbar wird. Im Gegensatz zu Programmiersprachen stehen bei den Auszeichnungssprachen keine Befehle oder Deklarationen im Vordergrund, sondern die Übermittlung von Informationen (Liu und Özsu, 2009). Einer der bekannteste Vertreter der Auszeichnungssprachen neben HTML ist die Extensible Markup Language (XML), welche unabhängig von der Plattform oder Implementierung zwischen Computersystemen eingesetzt werden kann und als offener Standard im Internet weit verbreitet ist (Liu und Özsu, 2009). Zwar kann XML im Prinzip vollkommen offen verwendet werden, jedoch gibt es Spezifikationen, wie insbesondere die des World Wide Web Consortium (W3C), die grundlegende Regeln im Umgang mit dieser Sprache vorgeben.

6.3.1 SMARTcodes

Ein wesentlicher Ansatz im Bereich des ACCC, der auf der Anwendung der Auszeichnungssprachen basiert, sind die Bestrebungen des International Code Council (ICC) zu den *SMARTcodes*. Die US Organisation ICC entwickelt Richtlinien und Vorschriften im Bereich der Gebäudesicherheit und des Brandschutzes für den Bau von Wohn- und Geschäftsgebäuden in den USA. Die Bauvorschriften der meisten US-Städte, -Counties und -Staaten stützen sich auf diese ICC-Codes. Im Jahr 2006 wurde von der ICC das *SMARTcodes*-Projekt mit dem Ziel gestartet, eine objektorientierte Technologie zur digitalen Darstellung der ICC-Codes einzuführen (AEC3 UK, 2017).

Ein wesentlicher Unterschied zu bisherigen Forschungsansätzen ist, dass dieser Ansatz direkt mit dem Text von Bauvorschriften selbst arbeitet. In ersten Testreihen konnte gezeigt werden, dass Anwender diese Vorgehensweise verhältnismäßig schnell verstehen und sich die Anzahl der Fehler, die bei der Erstellung gemacht werden, in kurzer Zeit reduziert. Da man sich dadurch erhebliche Verbesserungen bei der Zeit- und Kosteneffizienz erhoffte, wurde letztlich entschieden, diesen Ansatz weiterzuverfolgen (AEC3 UK, 2017).

Die *SMARTcodes* stellen im Grunde ein Protokoll dar, mit welchem wiederkehrende Elemente bzw. Fragmente einer Richtlinie standardisiert und vereinheitlicht werden können. Identifizierte und definierte Elemente werden in einer Bibliothek gespeichert und können bei weiteren Übersetzungen wieder verwendet werden. Das bedeutet, dass bei dieser Methode der Text des regulatorischen Dokuments lediglich vorverarbeitet und nicht in Maschinen-verarbeitbaren Code übersetzt wird.

Für die Formalisierung der Inhalte verwenden die *SMARTcodes* die RASE-Syntax (Hjelseth, 2010a; Hjelseth und Nisbet, 2011; See, 2008). Diese Syntax enthält die vier verschiedenen Klassen $Requirement$, $Applicability$, $Select$ und $Exception$, welchen die identifizierten Elemente bzw. Fragmente eines Textes zugeordnet werden. Die Kategorien haben die folgenden Bedeutungen:

Tabelle 6.2: Zusammenhang zwischen den Kategorien der RASE-Syntax und den Booleschen Operatoren (Hjelseth, 2010a)

	-	**NICHT**
UND	R	A
ODER	E	S

- *Requirement*: eine Anforderung, welche erfüllt werden muss.
- *Applicability*: ein einzelnes Objekt oder eine spezielle Form dieses Objekts mit den zugehörigen Eigenschaften. Damit das Objekt überhaupt relevant ist, muss es in Zusammenhang mit einer Anforderung stehen.
- *Select*: Spezifizierung eines Objekts, welches mit einer Anforderung in Zusammenhang steht.
- *Exception*: Ausnahmereglung inklusive der zugehörigen Parameter, wann diese eintritt.

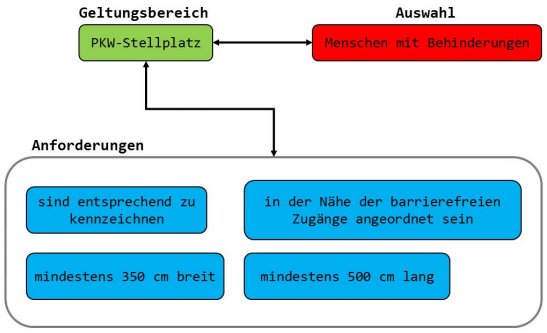
Die Kategorien der RASE-Syntax lassen sich auch mit Hilfe der Booleschen Operatoren darstellen. Dieser Zusammenhang stellt sicher, dass eine logische Falsifizierbarkeit einer RASE-Aussage gegeben ist. Die Abbildung ist in Tabelle 6.2 dargestellt.

Die Inhalte einer Vorschrift werden dabei also nicht abstrahiert oder abgeleitet, sondern lediglich kategorisiert und somit formalisiert. Durch die stark limitierte Anzahl an Kategorien soll die Anwendung der Syntax insbesondere auch für Ingenieure, welche keine Kenntnisse über Programmiersprachen besitzen, möglich sein. Weiterhin sollen auch komplexe Inhalte ohne erheblichen Interpretationsaufwand formalisiert werden können, indem diese in die jeweiligen Basiskomponenten zerlegt werden. Das Ergebnis einer RASE-Kategorisierung kann direkt im Fließtext der Richtlinien dargestellt und markiert werden. Das Resultat ist in dieser Form sowohl für Maschinen als auch für Menschen lesbar (Hjelseth und Nisbet, 2011) und ermöglicht außerdem auch Textsuchen direkt in dem transformierten Dokument (Solihin, 2016). Für eine erste praktische Anwendung der RASE-Methode wurden Regeln des *International Energy Conservation Code* implementiert (Eastman et al., 2009; International Code Council, 2017b).

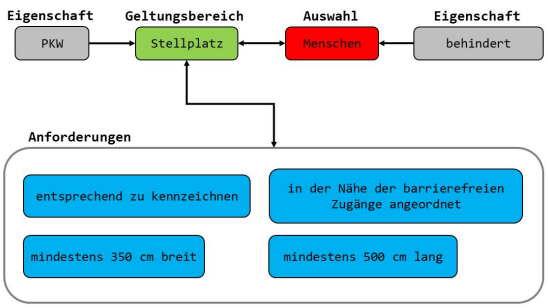
Um die Anwendung der RASE-Syntax besser nachvollziehen zu können, soll diese anhand eines Ausschnittes einer Deutschen Vorschrift im Bereich der Barrierefreiheit aus der DIN (2010-10) gezeigt werden:

> *PKW-Stellplätze, die für Menschen mit Behinderungen ausgewiesen werden, sind entsprechend zu kennzeichnen und sollten in der Nähe der barrierefreien Zugänge angeordnet sein. Sie müssen mindestens 350 cm breit und mindestens 500 cm lang sein*

Diese Anforderung kann nun entsprechend kategorisiert und mit den entsprechenden *Tags* markiert werden. Das Ergebnis ist in Quellcode 6.2 dargestellt. Bei der Übersetzung sind nur noch die relevanten Teile des Fließtextes zu erkennen. Elemente, welche nicht kategorisiert wurden, weil diese nicht von Relevanz sind, entfallen.



(a) Allgemeine Darstellung der formalisierten Anforderung



(b) Instanzierte Darstellung der formalisierten Anforderung mit den zugehörigen Zwangsbedingungen/Eigenschaften

Abbildung 6.5: Graphische Darstellung der formalisierten Anforderung nach (Hudeczek, 2017)

```
<R>
  <a>PKW-Stellplaetze</a>
  <s>Menschen mit Behinderungen</s>
  <r>sind entsprechend zu kennzeichnen</r>
  <r>in der Naehe der barrierefreien Zugaenge angeordnet sein</r>
  <r>mindestens 350 cm breit</r>
  <r>mindestens 500 cm lang</r>
</R>
```

Quellcode 6.2: RASE-Operatoren der Anforderung für PKW-Stellplätze nach (Hudeczek, 2017)

Die vorgestellte Regel besteht aus zwei Sätzen, die gemeinsam eine zusammenhängende Anforderung beschreiben. Die Anforderung beinhaltet, dass PKW-Stellplätze für Menschen mit Behinderungen entsprechend gekennzeichnet, angeordnet und dimensioniert werden müssen. Die Regel wird daher zunächst als übergeordnetes Element, welches sich in Unterelemente gliedert, mit R-Tags gekennzeichnet. An dieser Stelle wird ein Großbuchstabe verwendet, um zu markieren, dass es sich hierbei um ein übergeordnetes Element handelt, welches weitere Elemente hierarchisch unter sich angeordnet hat. Als Objekt, auf welche die Anforderung angewendet werden soll, werden in dem Text die *PKW-Stellplätze* eingeführt und entsprechend mit a-Tags gekennzeichnet. Da es sich hierbei spezifisch um die Stellplätze für *Menschen mit Behinderungen* handelt, wird diese Spezifizierung mit s-Tags gekennzeichnet. Schließlich folgen vier Einzelanforderungen, welche sich jeweils auf die zuvor spezifizierten Stellplätze beziehen. Diese werden jeweils mit r-Tags gekennzeichnet. Das resultierende Ergebnis lässt sich zur Veranschaulichung nun auch wie in Abbildung6.5 graphisch darstellen.

Die Formalisierung mit Hilfe der RASE-Syntax bezieht sich auf rein textliche Darstellung und so ist eine Transformation von Vorschriften, welche weitere Darstellungsarten, wie insbesondere Piktogramme oder aber Tabellen, enthalten, nicht ohne Informationsverlust möglich (Hjelseth und Nisbet, 2011). Zudem erfolgt die Übersetzung nach heutigem Stand ausschließlich manuell. Der Übersetzungsprozess muss daher mit äußerster Sorgfalt erfolgen, da es ansonsten schnell zu Fehlern kommen kann. Je nach Umfang und Komplexität der Anforderungen eines Regelwerks kann der Aufwand stark variieren. In vielen Fällen ist eine intensive Nachbearbeitung/-Kontrolle des Textes unumgänglich. Eine weitere Einschränkung dieses Ansatzes zeigt sich in den begrenzten Übersetzungsmöglichkeiten mit den limitierten Kategorien,

welche bei der Übersetzung verwendet werden dürfen. Trotz einer sorgfältigen Ausarbeitung kann es teilweise zu Unstimmigkeiten in Formulierung und Ausdrucksweise kommen, da Informationen nicht erfasst werden können. Dieses kann insbesondere vorkommen, wenn eindeutige Übersetzungen nicht möglich sind.

Die RASE-Syntax stellt den Grundstein für das *SMARTcode*-System dar, welches schematisch in Abbildung 6.6 abgebildet ist. Dieses System beinhaltet ein Wörterbuch, welches sämtliche Definitionen von Begriffen für Objekte und Eigenschaften speichert, die für die Modellprüfung relevant sind. Dazu gehören insbesondere auch alle Eigenschaften der Objekte, Datentypen und Einheiten. In einer Softwareumgebung unterstützt dieses Wörterbuch den Anwender bei der Identifikation von Schlüsselwörtern und deren logischer Rolle im Kontext. Dadurch sollen Fehler bei der Interpretation der Bauvorschriften reduziert werden. Darüber hinaus wird in diesem Wörterbuch auch die Information hinterlegt, wie die identifizierten Objekte mit dem Datenschema der IFC zusammenpassen. Auf diese Weise können aus dem Wörterbuch auch MVD abgeleitet werden, welche wiederum für die Regelprüfung verwendet werden (Eastman et al., 2009; See, 2008).

Für die Ausführung der eigentlichen Regelprüfungen müssen die übersetzten Inhalte noch in ein IFC-Constraint-Modell überführt werden, welche dann wiederum direkt an einem IFC-Modell geprüft werden kann (AEC3 UK, 2017; Hjelseth und Nisbet, 2011). Dabei handelt es sich um eine standardisierte Darstellung der Regeln, deren Funktionsweise in Abschnitt 6.5 im Detail erläutert wird. Ziel des *SMARTcode* Systems ist es somit, ein Anforderungsmodell für das IFC-Format zu erstellen, welche als Musterlösung betrachtet werden kann. Gegen dieses Anforderungsmodell kann schließlich jedes beliebige IFC-Instanzmodell geprüft werden, um so Konflikte oder Bereiche zu identifizieren, in denen das Gebäudemodell nicht die Informationen enthält, die zur Beurteilung der Konformität erforderlich sind.

Da die vorverarbeiteten RASE-Dokumente nur über diesen Weg in maschinenlesbare Anforderungen übertragen werden können, ergeben sich wesentliche Einschränkungen des Ansatzes. So können etwa notwendige Verarbeitungsschritte (z. B. das Ableiten von Werten oder aber geometrischen Berechnungen), wie diese sehr oft in regulatorischen Dokumenten notwendig sind, nicht ohne Weiteres umgesetzt werden. Es können lediglich die Anforderungen abgebildet werden, welche über das *IFC-Constraint*-Modell beschrieben werden können.

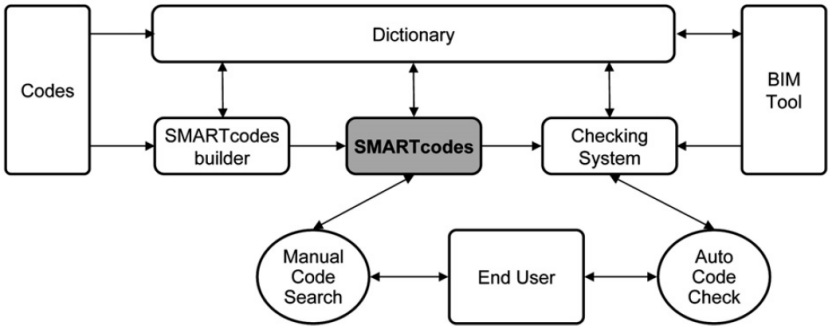


Abbildung 6.6: SMARTcode System; angelehnt an (Eastman et al., 2009)

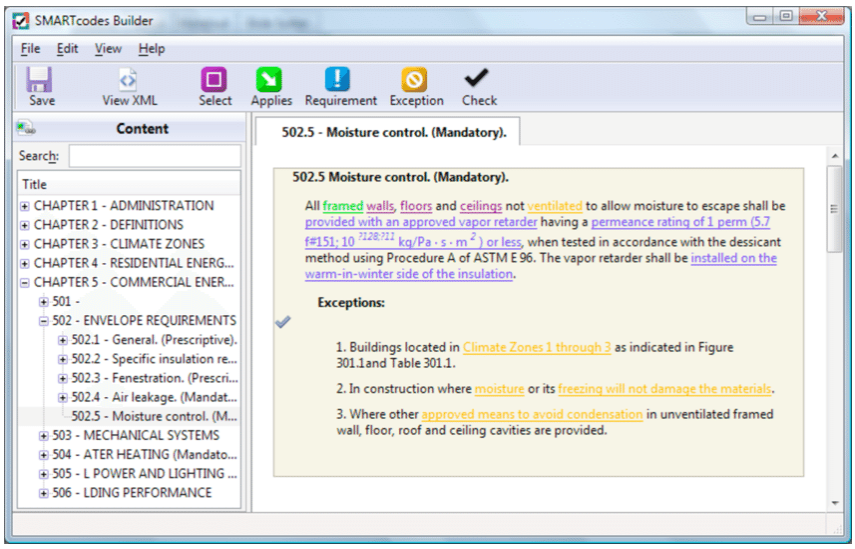


Abbildung 6.7: Oberfläche des SMARTcode Builder (Macit İlal und Günaydın, 2017)

In einer Testphase konnten Endanwendern aufseiten der ICC das Modellprüfsystem in einer Web-Umgebung testen (Eastman et al., 2009). Parallel wurde ein Windows-basierter Prototyp entwickelt, welcher ebenfalls von den ICC-Mitarbeitern und Fachplanern verwendet werden konnte (siehe Abbildung 6.7). In beiden Umgebungen konnten die ICC-Codes übersetzt und bei Bedarf Änderungen oder Ergänzungen eingepflegt werden. Grundlegende Funktion des Windows-Prototyps ist ein XML-Editor mit einem elektronischen Äquivalent eines Markers, welcher mit verschiedenen Farben jedes identifizierte Konzept mit einem Markup markieren kann. Die Softwareanwendung wurde schließlich als *SMARTcode Builder* veröffentlicht (Macit İlal und Günaydın, 2017).

2010 wurden die Entwicklungen des *SMARTcodes* Projekts vom ICC aufgrund einer auslaufenden Finanzierung gestoppt. Seitdem werden diese von den Firmen *AEC3 UK* und *DigitalAlchemy* weitergeführt (Dimyadi und Amor, 2013).

Wie bereits in dem vorigen Abschnitt beschrieben wurde, beschränkt sich die RASE-Syntax mit den eingeführten Kategorien nur auf bestimmte Elemente eines in natürlicher Sprache verfassten normativen Dokumentes. Daher kommt es bei dieser Vorverarbeitung immer wieder dazu, dass Elemente nicht erfasst werden können, obwohl diese relevante bzw. nützliche Informationen enthalten. Um diese Unzulänglichkeiten hinsichtlich einer Aussagekraft zu überwinden, führt Hudeczek (2017) eine Erweiterung der RASE ein, indem er (1) weitere Kategorien und (2) Attribute für die markierten Elemente einführt. Zu den eingeführten Kategorien gehören insbesondere Verweise zu Quellen, aus welchen weitere Informationen gewonnen werden können wie insbesondere Grafiken, Bilder, Tabellen oder andere normative Dokumente, sowie Datenwerte, welche in dem Dokument beschrieben werden. In ersten Anwendungsfällen für Deutsche Normen im Bereich der Barrierefreiheit konnte bereits nachgewiesen werden, dass der Einsatz dieser Kategorien aufgrund der Relevanz und Gebrauchshäufigkeit gerechtfertigt ist (Hudeczek, 2017).

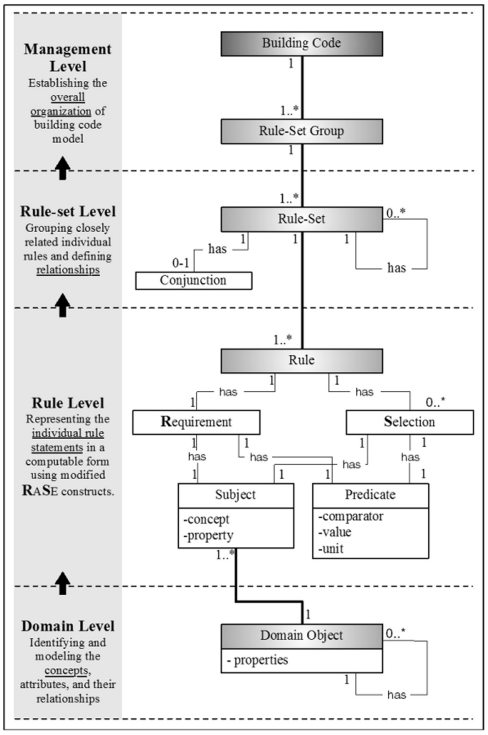


Abbildung 6.8: Hybrides vierstufiges Modell für die Darstellung von Bauvorschriften (Macit İlal und Günaydın, 2017)

6.3.2 Macit İlal und Günaydın (2017): SMARTcodes und logische Familien

Macit İlal und Günaydın (2017) kombinieren den semantischen Modellierungsansatz aus dem SMARTcodes-Projekt mit den theoretischen Grundlagen von (Fenves, 1966). Ziel ist es, die Beziehungsdarstellung von Regeln untereinander in einer gemeinsamen Prüfumgebung zusammenzuführen, wie dieses bei dem allgemeinen *SMARTcodes*-Ansatz nicht gegeben ist. Macit İlal und Günaydın (2017) argumentieren, dass so eine Sicherstellung der Korrektheit und Konsistenz der gesamten Regel-Darstellung ermöglicht werden soll.

Um dieses Problem zu lösen, führen sie ein hybrides vierstufiges Modell ein, welches eine getrennte Darstellung von Domäne-Konzepten, individuellen Regelaussagen, Beziehungen zwischen Regeln und der Gesamtorganisation des Baurechts ermöglicht. Daher ergeben sich schließlich die vier aufeinander aufbauenden Stufen Domänen-, Regel-, Regelsatz- und Managementebene. Das resultierende Modell ist in Abbildung 6.8 dargestellt.

Auf der dritten Ebene des Modells kann eine logische Hierarchiebeziehung zwischen Regeln aufgebaut werden. Der SMARTcode-Ansatz wurde zwar grundlegend übernommen, aber in eine vierstufige Darstellung überführt, die ihn durch die Beseitigung von Redundanzen und das Hinzufügen logischer Beziehungen verbessert. Konzepte, einzelne Regeln und Beziehungen zwischen den Regeln bzw. der Gesamtstruktur einer Bauvorschrift können gesondert dargestellt werden, sodass auch kleinere Änderungen des Modells möglich sind, um spezifische lokale Randbedingungen zu berücksichtigen. Bisher

wurde der Ansatz anhand der Regeln des *İzmir Municipality Housing and Zoning Code (IMHZCode)* getestet (Macit İlal und Günaydın, 2017). Weitere Tests stehen an dieser Stelle noch aus.

6.3.3 Nawari (2012c): SMARTcodes und LINQ

Nawari (2012c) verknüpft in seinem Forschungsansatz den SMARTcodes-Ansatz mit der XML-Repräsentation ifcXML sowie mit der Anfragesprache Language Integrate Query (LINQ), welche sich insbesondere für einen Umgang mit auf Auszeichnungssprachen basierten Daten eignet und Teil des *Microsoft .NET Framework* ist. LINQ bringt in dem Forschungsansatz die XML-basierten Informationen eines Gebäudemodells mit den formalisierten Inhalten von Richtlinien, welche wiederum als RASE-Dokument niedergelegt sind, zusammen.

In der Anfragesprache LINQ sind eine Reihe von Technologien zusammengefasst, welche letztlich auf der Integration von verschiedenen Abfragefunktionen direkt in herkömmliche Programmiersprachen des *.NET Framework* wie insbesondere *C#* und *VB.Net* basieren. LINQ erlaubt es, Abfragen direkt als Teil des herkömmlichen Quellcodes zu deklarieren. Diese Technologie ermöglicht es, Abfrageausdrücke zu schreiben, welche die umfangreichen Metadaten-, Kompilierzeit- und Syntax-Prüfungsfunktionen der *Microsoft*-Entwicklungswerkzeuge nutzen.

Nawari (2012c) erweitert in seinem Forschungsansatz die Standard-Abfrageoperatoren von LINQ um zusätzliche domänenspezifische Operatoren. Dabei werden LINQ-Standardabfrageoperatoren durch die neuen Operatoren ersetzt, welche dann zusätzliche Funktionalitäten wie Abfrageoptimierung oder aber Datenübersetzung durchführen können (Nawari, 2018). Zusätzlich implementiert Nawari (2012a) eine auf LINQ-basierte, speicherinterne XML-Programmierschnittstelle, welche die Kommunikation mit *ifcXML* und RASE erleichtert. Als Ergebnis lassen sich in dem Framework LINQ-Abfragen für die *ifcXML*- und gleichzeitig RASE-Daten definieren und ausführen. Als Beispiel führt Nawari (2012c) die Abfrage *"Finde alle Wandbauteile, deren Höhe größer als 12 ft ist."*, welche in Quellcode 6.3 dargestellt ist, an.

```
1 IEnumerable<XElement> WallHeight =
2 from
3 item in ifcWalls.Descendants("Item")
4 where
5 (int) item.Element("Walls") * (decimal) item.Element("Height") > 12
```

Quellcode 6.3: LINQ-Abfrage für eine sortierte Liste aller Wände mit einer Höhe größer als 12 ft (Nawari, 2018)

Auf ähnliche Weise werden die Informationen einer SmartCode-XML-Datei überführt. So können schließlich LINQ-Abfragen für beide Datenquellen, Gebäudemodell und Regelwissen, miteinander kombiniert werden. Um die Tragfähigkeit des Ansatzes zu zeigen, übersett Nawari (2012a) für einen Teil des *National Green Building Code Standard* (ICC 700-2008) die enthaltenen Regeln mit Hilfe des vorgestellten Ansatzes.

6.3.4 Ebertshäuser und von Both (2013): ifcModelCheck

Auch Ebertshäuser und von Both (2013) verwenden Auszeichnungssprachen in ihrem Tool *ifcModelCheck*. Im Auftrag des Bundesamt für Bauwesen und Raumordnung (BBR) entwickelten sie ein Inspektion-Tool für IFC, um die Planungen aller Projektbeteiligten eines Pilotprojektes hinsichtlich der Dokumentationsrichtlinien des BBR zur Gebäude- und Immobiliendokumentation oder aber allgemeine BIM-Qualitätskriterien automatisiert zu überprüfen. Grundidee dieses Ansatzes ist es, die Elemente *SELECT TYPE*, *PROPERTY*, *CRITERION* und *TARGET VALUE* in den Regeln der Vorschriften zu identifizieren, um so den Informationsgehalt zu formalisieren.

```
// Example - "Does [every Window] have a [U-value] [less than] [1.2] ?"
// Identified Elements
SELECT TYPE:  [every Window]
PROPERTY:     [U-value]
CRITERION:    [less than]
VALUE:      [1.2]
// Resulting method
<code shell>
Main Rule Run Method {
from ITEM in [SELECT TYPE]
where
[PROPERTY] [CRITERION] [VALUE]
select [ITEM]
}

```

Quellcode 6.4: Übersetzung einer Anforderung im ifcModelCheck (Ebertshäuser und von Both, 2013)

Hierbei spezifiziert die *SELECT TYPE*-Komponente die gewünschte Menge von Instanzen, welche geprüft werden sollen. Mit dem *PROPERTY*-Element wird als weitere Spezifikation die Eigenschaft zu der zuvor gewählten Menge definiert (z. B. der UV-Wert im *Property Set* des Fenstertyps).

Für die Überprüfung des Inhaltes dieses Attributs wird in der sogenannten *TARGET VALUE*-Komponente eine entsprechende Soll-Ausgabe des Wertes zugeordnet. Durch die Definition der geeigneten Bedingung mit der *CRITERION*-Komponente wird für das Verhältnis zwischen Ist- und Sollwert in den sogenannten Prüfkriterien der letzte Teil der Beschreibung der Regelklausel festgelegt.

Ein entsprechendes Beispiel für die Anwendung der Elemente ist in Quellcode 6.4 dargestellt.

Um das Formulieren der Vorschriften zu erleichtern, stehen dem Anwender in der Umgebung die Objekt- und Datentypen des IFC-Datenmodell sowie die Elemente der Object Constraints Language (OCL), einer Sprache zur Beschreibung von Randbedingungen bei der Modellierung von Informationssystemen, zur Verfügung. Die formulierten Vorschriften werden anschließend in das XML-Format übertragen und gespeichert. *ifcModelCheck* dient nicht nur als Prüf-, sondern insbesondere auch als Analysewerkzeug zur Bewertung von unterschiedlichen Simulationsergebnissen von Varianten.

Wie allerdings in den Ausführungen zu erkennen ist, sind die Möglichkeiten zur Prüfung stark auf die Attribut-Ebene des Modells beschränkt. Eine Weiterverarbeitung von Informationen mit höherwertigen Methoden (z. B. geometrische Verarbeitungen) ist nicht möglich.

6.3.5 Weitere domänenspezifische Auszeichnungssprachen

Die leichte Anwendung und Lesbarkeit von Auszeichnungssprachen hat dazu geführt, dass diese nicht nur weitverbreitet sind, sondern auch diverse Abwandlungen und Spezifikationen eingeführt wurden, welche sich für die Formalisierung von spezifischem Domänenwissen eignen.

Für die Formulierung von Regelungen und Vorschriften wurde der *RuleML*-Standard eingeführt (Olken et al., 2011; RuleML Inc., 2017). Da *RuleML* insbesondere in *Business-Rule-Management*-Systemen zum Einsatz kommt, wurden 2012 die beiden offenen Standards *LegalDocML* und *LegalRuleML* eingeführt (OASIS, 2017). Mit Hilfe diese Spezifikationen ist es möglich, Rechtstexte, z. B. Gesetze, Verordnungen, Verträge und Rechtsprechung, als Quellen von Normen, Richtlinien und Regeln zu formalisieren. Ziel hinter der Einführung dieser Datenformate ist es, aktuelle Rechtsauffassungen und deren Veränderungen im Laufe der Zeit zu erfassen.

Ein wesentlicher Aspekt bei der Einführung war, dass der wörtliche Inhalt, die Strukturen sowie Darstellungsaspekte eines Dokumentes für den menschlichen Leser aufgrund der Vertrautheit mit dem Dokument essenziell sind und daher auch nach einer Anwendung der Auszeichnungssprachen aufrechterhalten bleiben sollen. Aus diesem Grund bildet *LegalDocML* den wörtlichen Inhalt und die Struktur für den gesamten Lebenszyklus eines Dokumentes ab. Die Präsentation wird dabei mit Hilfe eines *Stylesheets* in HTML oder eines ähnlichen Formates gespeichert. So wird sichergestellt, dass die Struktur und der wörtliche Inhalt erhalten bleiben. Diese Präsentation ist mit einer *LegalRuleML*-Definiton verknüpft, welche den logischen Inhalt und Semantik des Dokumentes abbildet. Für die Beschreibung der Inhalte in dem *LegalRuleML*-Dokument wird *RuleML* als verwendet.

Bei der Verwendung von *LegalDocML* und *LegalRuleML* werden vier Aspekte eines juristischen oder aber normativen Dokuments betrachtet (BuildingSmart, 2017d; OASIS, 2017):

- der Inhalt, welcher eine Reihe von Wörtern und Satzzeichen, welche wiederum die Sätze des Textes bilden.
- die Präsentation, welche beschreibt, wie die Informationen äußerlich aussehen, z.B. die Farbe des im Dokument verwendeten Textes, die Schriftart in den Überschriften und andere solche Formatierungsfragen.
- die Struktur, welche beschreibt, wie die Informationen organisiert sind, z.B., die Identifizierung der Textteile mit Überschriften, Klauseln oder aber Verweisen.
- die Semantik, die beschreibt, was die Information darstellt oder bedeutet.

Mit den formalisierten Inhalten können Informationen zwischen den Parteien ausgetauscht, strukturierte Inhalte aus den Texten gesucht, extrahiert oder automatisch weiterverarbeitet werden. Bisher können Gesetzgeber, Rechtspraktiker oder Betriebswirte Inhalt der Rechtstexte nur sehr beschränkt vergleichen, kontrastieren, integrieren oder anderweitig verwenden, da dieses einen erhöhten manuellen Aufwand mit sich bringt.

Durch eine Erweiterung bzw. Abwandlung der Syntax dieser Sprachen können diese nicht nur auf Rechtsdokumente unterschiedlicher Art sondern auch auf nicht-juristische Dokumente angewendet werden. Eine mögliche Verwendung zur Erfassung von Bauvorschriften stellen Dimyadi und Amor (2013) und Dimyadi et al. (2017) vor. Um die Anwendung auf Bauvorschriften nachzuweisen, übersetzen sie unter anderem Paragraphen der Dimyadi et al. (2017); NZBC (2017), eine Sammlung von Bauvorschriften

in Neuseeland, welche diverse Leistungskriterien für bauliche Anlagen spezifiziert. In Quellcode 6.5 ist die Übersetzung eines solchen Paragraphen mit *LegalDocML* und in Quellcode 6.6 die zugehörige Übersetzung mit *LegalRuleML* dargestellt. Da sich der Ansatz noch in einem frühen Stadium befindet, bleibt abzuwarten, inwieweit sich dieser für eine Automatisierung der Konformitätsprüfung anwenden lässt.

```
<paragraph eId="NZBC_C4.3">
  <heading eId="NZBC_C4.3_heading">
    Movement to Place of Safety
  </heading>
  <content eId="NZBC_C4.3_content">
    <p>The
      <i>evacuation time</i>
      must allow occupants of a building ...
    </p>
  </content>
</paragraph >
```

Quellcode 6.5: Abbildung des Pragraph C4.3 der NZBC in LegalDocML (Dimyadi et al., 2017)

```
nzbcC43DefaultVisibility: => MinSmokeVisibility($room)=10m nzbcC43VisibilityException:
Size($room)<100m2
MinSmokeVisibility($room)=5m nzbcC43MinEvacuationTime: FireEvent,
$MinTimeSmokeVisibility = min{ TimeTakenFor(SmokeObscuration($room),MinSmokeVisibility($room)): for $room
    in $Rooms},
$MaxTime = min{ MinTimeSmokeVisibility, TimeTakenFor($FractionalEffectiveDoseCarbonMonoxide, 0.3),
    TimeTakenFor($FractionalEffectiveDoseThermalEffect, 0.3)},
$MoveSafetyTime = max{ distance($PlaceOfSafety,$room)/$EvacuationSpeed: for $room in $Rooms}
=>[Obligation] $MoveSafteyTime < $EvacuationTime < $MaxTime
```

Quellcode 6.6: Abbildung des Pragraph C4.3 der NZBC in LegalRuleML (Dimyadi et al., 2017)

6.4 Ansätze auf Basis des Semantic Web

Wie bereits in Abschnitt 2.4.2.3 beschrieben, erlauben die Methoden rund um das *Semantic Web*, wie insbesondere RDF und OWL, die explizite Darstellung von Regelwissen.

Eine Reihe von Autoren (Beetz et al., 2007, 2009; Gonçal, 2017; Patil et al., 2005; Pauwels et al., 2016, 2011; Pauwels und Zhang, 2015a,b) haben in den letzten Jahren unterschiedliche Ideen und Ansätze zur Verbesserung der Interoperabilität in den BIM-Prozessen und insbesondere bei der Prüfung dieser Modelle vorgeschlagen, die auf Technologien des *Semantic Web* basieren.

Ein wesentlicher Vorteil dieser Technologie ist, dass diese es erlauben, Verbindungen zwischen verschiedenen Datenquellen herzustellen, ein Konzept, das auch als *Linked Data* bezeichnet wird (Bizer et al., 2009). Bei der Definition von Regeln ist die resultierende Automatisierung und Deduktion, also die Ableitung neuer Erkenntnisse aus einem Datenmodell, ein wesentlicher Vorteil (Gonçal, 2017). Um diese Verknüpfungen herzustellen, müssen Informationen in der Repräsentationssprache formalisiert werden, die innerhalb der semantischen Domäne verwendet wird - in der Regel dem RDF (W3C, 2004a), einem erweiterbaren Datenmodell, das auf gerichteten, beschrifteten Graphen basiert. Wenn die Informationen

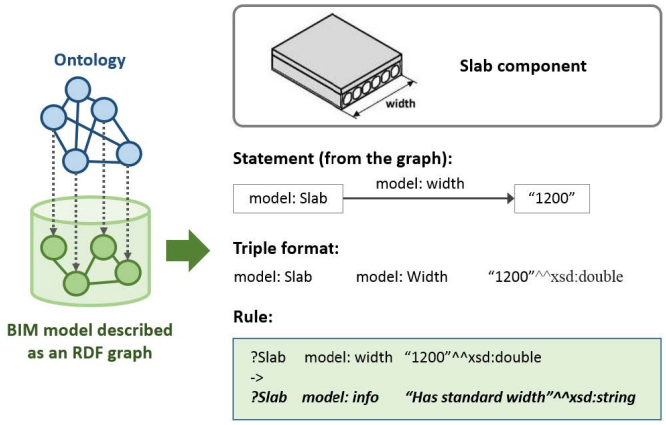


Abbildung 6.9: Beispiel einer einfachen Inferenzregel, die für die Vorverarbeitung von Informationen eines Modells verwendet wird (Gonçal, 2017)

einer Domäne dargestellt und mit Informationen aus anderen Domänen verknüpft werden, die ebenfalls diese Sprache verwenden, können sie von Programmen und Diensten, die in der Lage sind, die Sprache zu lesen, automatisch gelesen und verarbeitet werden.

Darüber hinaus erlaubt das RDF-Datenmodell die formale Darstellung von Inferenzregeln (IF-$THEN$-Anweisungen) in Verbindung mit den durch RDF dargestellten Daten (Gonçal, 2017). Wenn eine Regel z. B. in Form von IF-$THEN$-Anweisungen mit Hilfe von RDF definiert wird, kann diese gemeinsam mit allen bestehenden Regeln auf ein Datenmodell erneut angewendet werden. Dabei werden neue Erkenntnisse hergeleitet und weitere wertvolle Informationen zu dem Datenmodell gewonnen. In diesem Zusammenhang wird auch von sogenannten Schluss- oder Inferenzregeln gesprochen, da es diese erlauben, von bestehenden zu neuen Aussagen überzugehen. Vereinfacht ausgedrückt wird also eine Schlussfolgerung gezogen.

Im RDF-Format dargestelltes Wissen kann somit um neue Erkenntnisse (beschrieben im IF-Teil der Regel) erweitert werden, welche wiederum logisch hergeleitet wurden (beschrieben im $THEN$-Teil der Regel). Darüber hinaus können auch Bedingungen oder Einschränkungen mit Hilfe von logischen Operatoren abgebildet werden. Im Kontext der Modellprüfung stellen Bloch und Sacks (2018) eine detaillierte Analyse von Inferenzregeln insbesondere im Vergleich zu Methoden des maschinellen Lernens vor.

Nach (Eastman et al., 2009) können solche Regeln schließlich Ergebnisse wie *pass*, *fail*, *warning* oder *unknown* liefern, je nach Art der Prüfung, die für jede mögliche Situation erforderlich ist. Diese Eigenschaften von semantischen Web-Technologien können dazu genutzt werden, verschiedene Situationen rund um das Gebäudeinformationsmanagement zu adressieren, insbesondere wenn die Kombination von Informationen aus verschiedenen Domänen erforderlich ist. In Abbildung 6.9 ist ein Beispiel für die Anwendung von semantischen Inferenzregeln schematisch dargestellt (Gonçal, 2017).

Costa und Pauwels (2015) führen aus, dass es zwei grundlegende Bedingungen für die Anwendung von semantischen Inferenzregeln gibt: Damit der IF-Teil einer Regel erstellt werden kann, muss die Datenstruktur des RDF-Diagramms bekannt sein, in welchem die Informationen eines BIM-Modells dargestellt werden. Dabei können die RDF-Informationen aus einer Ontologie, die zur Beschreibung des Modells verwendet wurde, einfach extrahiert werden. So können z. B. die im IFC-Standard beschriebenen

Subclass	Constraints of a building
General	* Total_m²≤900(m²) * Total_breadth≤35m * Total_width≤50m
Rooms	* building⊑∃object.space & floor & story & wall * 3= number_of_Building.babyroom 50(=7m×7m) minimum_size_of_each_babyroom * 2= number_of_Building.babyclass 30(=6m×5m) minimum_size_of_each_babyclass * 1= number_of_Building.displayroom 140≤size_of_each_displayroom≤160 * 1= number_of_Building.utility 100≤size_of_each_utility≤110 * 1= number_of_Building.teacher's room * 2= number_of_Building.stairwell * 4= number_of_Building.toilet
Toilet	* Three toilet ⊑∃located next to babyroom 30≤size_of_each_toilet≤35 * one toilet ⊑∃located next to displayroom 40≤minimum_size_of_toilet
Stairway	* Two stairway⊑∃located on the each corner * stairway⊑∃connects_story

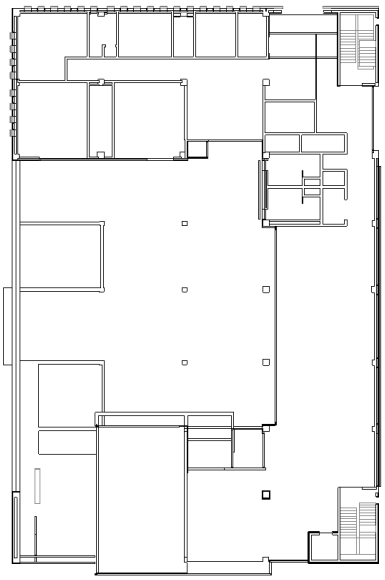


Abbildung 6.10: Randbedingungen für die Gestaltungsplanung eines Grundrisses (Lee et al., 2008)

BIM-Modelle nach der *ifcOWL*-Ontologie in RDF-Daten transformiert werden. Weiterhin muss das Modell die für seine Prüfung notwendigen Informationen ebenfalls enthalten, da die Regel nicht angewendet werden kann, wenn die Daten unvollständig sind.

In ihrem Forschungsansatz übertragen Lee et al. (2008) das Prinzip der Ontologie mit Hilfe von OWL und dem IFC-Datenmodell an dem Pilotprojekt *Gunpo Worker's Welfare Centre* auf das Bauwesen. In ihrem Forschungsbeitrag zeigen sie auf, dass sich auf Grundlage einer Klassenhierarchie des Gebäudemodells Relationen und logische Operatoren Randbedingungen definieren lassen. Mit der Definition der relationalen Objekte innerhalb der Klassenhierarchie konnten eine Reihe von Randbedingungen für die Gestaltung eines Grundrisses (siehe Abbildung 6.10) implementiert werden.

In Quellcode 6.7 ist eine solche Randbedingung beispielhaft als OWL-Übersetzung dargestellt. Hierbei handelt es sich um eine starre, feste (eng. *hard-coded*) Implementierung für das Beispielprojekt.

```
<owl:ObjectProperty rdf:ID="isLargerThan">
<rdfs:type rdf:resource="&Person ; TransitiveProperty"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Room1">
<isLargerThan ref: resource="#Room2"/>
</owl: Class >
<owl:Class rdf:ID="Room2">
<isLargerThan ref: resource="#Room3"/>
</owl: Class >
```

Quellcode 6.7: Beispiel für eine logische Relation im ontologischen Modell und die zugehörige OWL-Übersetzung (Lee et al., 2008)

Pauwels et al. (2011) stellen ein semantisches, regelbasiertes System zur Überprüfung der Gebäudeperformance vor. Dieses System wurde in einem Testfall eingesetzt, um die Einhaltung der akustischen Vorschriften in BIM-Modellen zu überprüfen. Zu diesem Zweck wurden semantische Inferenzregeln mit der Sprache *N3Logic* implementiert, um Instanzen von Gebäudemodellen zu überprüfen, die in einer domänenspezifische Ontologie beschrieben wurden. Bei der Prüfung wurden zwei Normen betrachtet, welche diverse akustische Vorschriften beinhalten und sowohl auf nationaler als auch europäischer Ebene Anwendung finden.

Ein weiteres Beispiel für die Anwendung semantischer Regelsprachen findet sich bei de Farias et al. (2015). In diesem Forschungsansatz wird eine Ontologie als Teilmenge des IFC-Standards eingeführt. Als Teil dieser Ontologie werden unter anderem SWRL-basierte Regeln definiert, welche spezifische Anforderungen eines Auftraggebers beschreiben. Durch die Kombination von Ontologie und Regeln wird die Bedeutung der Konzepte eines BIM-Modells von ihrer Semantik, welche in den IFC-Dateien definiert ist, gelöst, da auf diese Weise die Aussagekraft der Modelle erhöht werden kann, ohne die Interoperabilität auf Basis der IFC zu beeinträchtigen. Als Ergebnis des Forschungsprojektes wurde ein *Parser* entwickelt, welcher aus einer in EXPRESS dargestellten IFC eine IFC-Ontologie generiert. Eine aktuelle Version des IFC zu RDF Konverter wurde von Pauwels und Oraskari (2016) veröffentlicht. Die in SWRL formulierten Regeln stehen hierbei an der Spitze der Ontologie und können mit Hilfe eines Ontologie-Editors erstellt werden. In dem Forschungsprojekt wurden die Ontologie sowie die Regeln in die RDF-Datenbank *Stardog* geladen, eine Umgebung, die auf die Anwendung von SWRL-Regeln auf die Daten-Triple spezialisiert ist.

6.4.1 Semantic Web Rule Language

Die Semantic Web Rule Language (SWRL) ist ein vom W3C eingeführter Standard für die formale Abbildung von Regelwissen (W3C, 2004b). Der Standard findet seit seiner Einführung im Jahr 2004 breite Akzeptanz und Anwendung. Mit SWRL können Regeln mit hochwertiger und abstrakter Syntax definiert auf der Basis einer ontologischen Datenstruktur entwickelt und angewendet werden. Die Idee zu Ontologien in der Informationstechnik und deren Einsatz im Bauwesen wurde in Abschnitt 2.4.2.3 bereits behandelt.

SWRL basiert auf einer Kombination der *OWL DL* und *OWL Lite*. SWRL beschreibt die Form einer Implikation, also zwei Hauptelemente: einen Vorläufer (Körper) und ein Folgeelement (Kopf). Die beabsichtigte Bedeutung kann wie folgt gelesen werden: Wenn die im vorangehenden Abschnitt angegebenen Bedingungen eingehalten werden, dann müssen auch die im nachfolgenden Abschnitt genannten Bedingungen eingehalten werden (W3C, 2004b).

Eine Verwendung der SWRL für das Bauwesen wird unter anderem von Zhong et al. (2012), Beach et al. (2015) und Uhm et al. (2015) vorgeschlagen. Eine beispielhafte direkte Anwendung der SWRL für Bauvorschriften ist in Quellcode 6.8 dargestellt, in welchem die Übersetzung des Korean Building Act Clause 34-1 zu finden ist. Der Korean Building Act 34-1 lautet (Korea Legislation Research Institute, 2008):

> *On each floor of a building, direct stairs leading to the shelter floor or the ground other than the shelter floor shall be installed in the way that the walker distance from each part of the living room to the stairs is not more than 30 meters: Provided, that in the cases of a building of which the main structural part is made of a fireproof structure or non-combustible materials, the walking distance of not more*

than 50 meters may be established, and in cases of a factory prescribed by Ordinance of the Ministry of Land, Infrastructure and Transport, which is equipped with automatic fire extinguishers, such as sprinklers, in an automated production facility, the walking distance of not more than 75 meters may be established.

```
leadingTo(Stairs,Ground) ^ walkingDistance(Stairs, 30) -> Pass(34-11)
leadingTo(Stairs,ShelterFloor) ^ mainStructuralPartMadeOf(Building, FireProofStructure) ^
    walkingDistance(Stairs, 50) -> Pass(34-12)
leadingTo(Stairs,ShelterFloor) ^ Usage(Bulding,Factory) ^ prescribedUse(Building,
    OrdinanceOfMinistryOfLand) ^ hasSystem(Building,Sprinklers) -> Pass(34-13)
Pass(34-11) ^ Pass(34-12) ^ Pass(34-13) -> Pass(34)
```

Quellcode 6.8: SWRL-Übersetzung des Korean Building Act Clause 34 (BuildingSmart, 2017d)

Regelwissen kann mit Hilfe von SWRL formalisiert werden. Für die Anwendung und Prüfung dieser Regeln wird allerdings eine interpretierende Instanz, auch *Reasoning Engine* bzw. *Reasoner* genannt, benötigt. Für die Erstellung und Definition dieser Regeln steht eine Vielzahl von Editoren und Softwareumgebungen zur Verfügung, wie insbesondere die Umgebung Protégé (Stanford University, 2016) der Stanford University.

Eine wesentliche Einschränkung von SWRL ist, dass Regeln nicht konditional miteinander verbunden werden können. Damit können Ergebnisse von Regeln nicht an nachfolgende weitergegeben werden. Weiterhin können in der Sprache keine verarbeitenden Methoden bzw. Operatoren definiert werden, d. h. es ist nicht möglich, Prozeduren auszudrücken, die aus Eingabedaten zusätzliche Informationen erzeugen. Einfache Funktionen, wie beispielsweise die mathematische Berechnung von Flächen, sind mit Hilfe der SWRL somit nicht möglich (BuildingSmart, 2017d).

6.4.2 Uhm et al. (2015): Kombination von Context Free Grammar und SWRL

Uhm et al. (2015) konzentrieren sich in ihrer Studie zur automatisierten Konformitätsprüfung eines BIM-basierten Gebäudeentwurfs auf die in Ausschreibungen enthaltenen Anforderungen an Gebäude in Südkorea. Diese Anforderungen sind untergliedert in sogenannte *Request for proposal (RFP)*, welche wiederum diverse Anforderungen zu der Hauptfunktion, Form und Benutzerfreundlichkeit an ein Gebäude spezifizieren. Bei Bauvorhaben in Südkorea sind diese Anforderungen das Hauptaugenmerk für die Planer bei der Gestaltungsplanung.

Als Ausgangspunkt für ihren Ansatz, analysieren Uhm et al. (2015) 27 dieser RFP für Projekte in den Bereichen Wohnen, Büro, Ausstellung, Krankenhaus, Sportzentrum und Gerichtsgebäude. Diese regulatorischen Dokumente enthalten insgesamt über 1800 Sätze. Für die Analyse verwenden Uhm et al. (2015) das Konzept der Context Free Grammar (CFG), mit welcher sich natürliche Sprache in vier Kategorien untergliedern lässt: Objekt (Substantiv), Methode (Verb), Strenge (Modal) und Andere.

Das Ergebnis dieser Analyse sind einzelnen Elemente, sogenannte Morpheme, welche sich in Unterkategorien gliedern und visualisieren lassen. In Abbildung 6.11 ist beispielhaft ein resultierender CFG-Baums dargestellt. In diesem Beispiel wird der Satz

"Jeder Ein-Patienten-Raum soll eine Mindestfläche von 120 m^2 haben."

in die jeweiligen Morpheme untergliedert (Uhm et al., 2015).

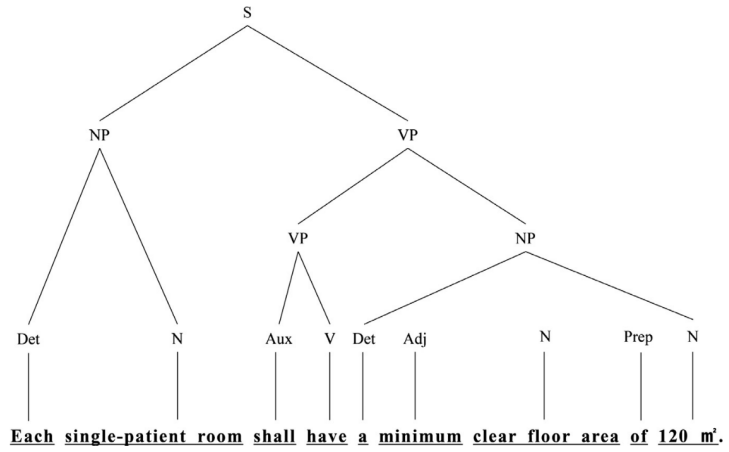


Abbildung 6.11: Übersetzung eines RFP-Beispielsatz in Form eines CFG-Baums (Uhm et al., 2015)

Mit Hilfe dieser Methode konnten Uhm et al. (2015) drei Objekttypen, 29 Methodentypen und fünf modale Ebenen in den ausgewählten regulatorischen Dokumenten identifizieren. Die identifizierten Methodentypen sind in Abbildung 6.12 dargestellt.

Die regulatorischen Aussagen aus den Dokumenten werden mit Hilfe der identifizierten Objekte und der SWRL übersetzt. Um Redundanzen bei der Übersetzung weiterer RFP zu vermeiden, wurde eine Datenbank (*Thesaurus*) entwickelt, in der die identifizierten Typen gespeichert werden können. Dieses Wörterbuch dient als Hilfsmittel für die Übersetzung weiterer regulatorischer Aussagen. Um die formalisierten Regeln auch ausführen zu können, haben Uhm et al. (2015) einen *SWRL-to-Python*-Übersetzer entwickelt, welcher die Regeln in Python-Skripte übersetzt, welche in der Softwareanwendung *abimo* ausgeführt werden können. Darüber hinaus wurde in diesem Prototyp auch das Thesaurus implementiert.

Ein Beispiel für eine SWRL-Regel und die entsprechende Übersetzung ist in Quellcode 6.9 dargestellt. In diesem Beispiel prüft die Methode *isAdjacent*, ob zwei Räume sich eine Wand oder eine Decke teilen. Die Methode selbst steht hierbei als geometrischer Algorithmus in der *Checker Engine* zur Verfügung und kann als Python-Methode aufgerufen werden.

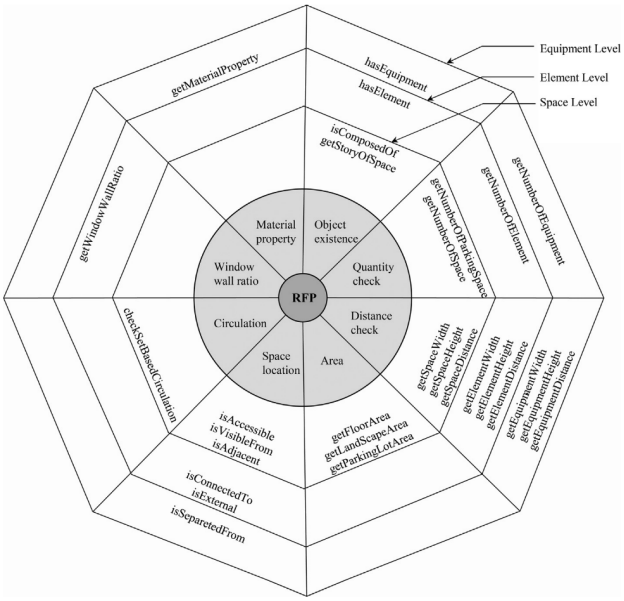


Abbildung 6.12: Identifizierte Methodentypen mit Hilfe von CFG (Uhm et al., 2015)

```
<SWRL>
  Food depots (?x) ^ Kitchen (?y) ^ isAdjacent(?x, ?y) ! AcceptEntity(?x)
<Python>
  Import kbCheckModule;
  Checker = kbCheckModule.kbChecker()
  SpaceListA = Checker.FindSpace('food depot')
  SpaceListB = Checker.FindSpace('kitchen')
  for spaceA in SpaceListA:
  for spaceB in SpaceListB:
  IF Checker. IsAdjacent(spaceA,spaceB) == 1:
    Checker.SetErrorResult(1)
  Else: Checker.SetErrorResult(0)

```

Quellcode 6.9: SWRL-Darstellung der identifizierten Methode *isAdjacent* und die entsprechende Übersetzung in Python (Uhm et al., 2015)

In Abbildung 6.13 ist ein Ergebnis der Prüfung dieser Regel in der Softwareumgebung *abimo* dargestellt.

6.4.3 Ansatz von Bouzidi et al. (2012)

Die Digitalisierung und Formalisierung von Bauvorschriften gilt als wesentlicher Bestandteil des Plan de Transition Numérique du Bâtiment (PTNB), ein Strategieplan des französischen Ministeriums für Wohnungswesen, territoriale Gleichstellung und ländliche Angelegenheiten für die Vorbereitung des digitalen Einsatzes im gesamten Bauwesen und insbesondere in kleinen Strukturen (Bouzidi et al., 2012; Mission Numérique Bâtiment, 2015). Zu diesem im Jahr 2015 gestarteten Stufenplan gehört unter anderem ein Forschungsprogramm für die Entwicklung innovativer Techniken. Teil dieses Programms ist ein Forschungsprojekt, mit welchem das Centre Scientifique et Technique du Bâtiment (CSTB) beauftragt

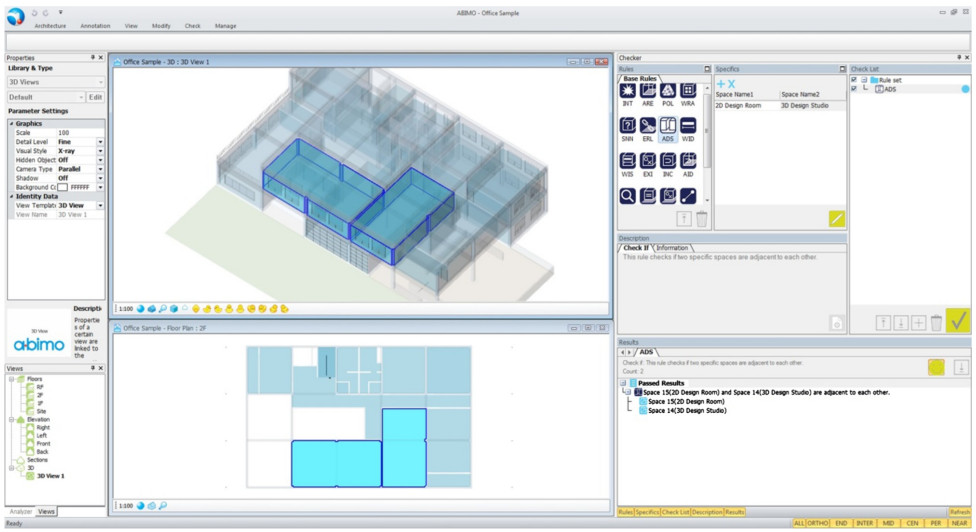


Abbildung 6.13: Visualisierung des Ergebnisses der Prüfung *isAdjacent* (Uhm et al., 2015)

wurde. Sein Ziel ist, die generelle Durchführbarkeit der digitalen Regelprüfung nachzuweisen und eine Methodik zu entwickeln, die unter praxisnahen Bedingungen anwendbar ist. Die Methodik soll es erlauben, das Wissen der Domäne über die Interpretation der Vorschriften zu erfassen und in eine von einem Computer verarbeitbare Darstellung zu überführen.

Im Rahmen einer Machbarkeitsstudie zur Digitalisierung/Formalisierung von Bauvorschriften wurde der Prozess in zwei wesentliche Prozessschritte zerlegt: In einem ersten Schritt werden semi-formale Aussagen (*semi-formal assertions*) identifiziert, welche aus *Wenn-Dann*-Klauseln bestehen. Hierbei umschreiben Fachexperten mit Unterstützung einer Software-Toolbox die in natürlicher Sprache ausgedrückten Bauvorschriften in diese semi-formale Darstellung. Die verwendete Toolbox besteht hauptsächlich aus einem Texteditor mit einer dedizierten Syntax-Hervorhebung und Autokomplettierungsfunktion. Diese Hervorhebung basiert auf einer Ontologie, die sich zum einen aus einer Sammlung bereits identifzierter, konstruktionsorientierten Vokabeln und zum anderen aus der IFC4-Ontologie (definiert mit Hilfe von ifcOWL) zusammensetzt. Basierend auf der Definition dieses Vokabulars wurde eine einfache Grammatik definiert, um die Umschreibung der in der natürlichen Sprache ausgedrückten Bauvorschriften in eine semi-formale Darstellung zu übersetzen. Dieser Ansatz basiert im Wesentlichen auf den Forschungsarbeiten von Hjelseth und Nisbet (2011) und ist somit softwareunabhängig.

In dem zweiten Schritt werden die semi-formalen Regeln in formale Regeln mit Hilfe von *mvdXML* transformiert (siehe auch Abschnitt 2.5.2). Wesentlicher Vorteil dieses Ansatzes ist, dass MVD eine standardisierte und anerkannte Methode ist und sich sehr gut auf IFC-Modelle anwenden lässt.

Allerdings führen Fahad et al. (2016) in diesem Zusammenhang auch einige Grenzen dieser Methode an: Zwar lassen sich einfache Zusammenhänge, Kardinalitäten oder die Existenz von Objekten mit Hilfe von *mvdXML* sehr leicht prüfen, Informationen für weiterführende Analysen lassen sich allerdings nicht ohne Weiteres aus dem Datenmodell extrahieren. Weiterhin ist *mvdXML* zu weiten Teilen auf die Ausdruckskraft des IFC-Schema beschränkt, da es auf diesem basiert. Begriffe oder Konzepte, die nicht in der IFC definiert sind, können auch mit Hilfe von MVD nicht abgebildet werden. In der französischen

Brandschutzverordnung findet sich beispielsweise der Begriff *Protected Area*, welcher mehrere Bedeutungen und spezifische Eigenschaften (Mischzonen, Räume, Öffnungen, Wand- und Deckenverkleidungen, Brandschutzwerte, etc.) zusammenfasst. Dieser Begriff hat allerdings kein direktes Äquivalent in der IFC. Um diese Beschränkung zu überwinden, wurden im Rahmen des Forschungsprojekts zusätzlich *Semantic Web*-Technologien eingesetzt (Bouzidi et al., 2012).

Hierzu wurde zunächst eine Domäne-Ontologie definiert, welche die wesentlichen Konzepte und die Beziehungen zwischen den Vokabeln auf der Basis von OWL definiert.

Um die regulatorischen Anforderungen in eine formale Sprache zu übersetzen, kommt zunächst die Semantics of Business Vocabulary and Business Rules, mit welcher auch Endanwender arbeiten können, zum Tragen. Mit Hilfe von SBVR können in natürlicher Sprache verfasste Beschreibung formalisiert werden. Typischerweise wird SBVR zur Beschreibung von komplexen *Compliance*-Regeln, wie z. B. operative Regeln für ein Unternehmen, Sicherheitsrichtlinien oder Standardkonformität eingesetzt. Erst nach dieser Übersetzung werden die formulierten Anforderungen mit Hilfe von SPARQL in Anfragen übersetzt. Dieser Übersetzungsschritt muss allerdings größtenteils manuell erfolgen. Jeder elementare Prozessschritt wird als SPARQL-Abfrage definiert und kann auf Abruf durchgeführt werden. Die Beziehung zwischen den SBVR-Regeln und SPARQL-Abfragen kann mit Hilfe von RDF dargestellt werden (Bouzidi et al., 2012). Diese Methode kann somit auch dazu verwendet werden, Informationen aus dem Datenmodell zu extrahieren und Informationen außerhalb des IFC-Schemas darzustellen.

6.4.4 Yurchyshyna et al. (2009): Conformance Checking in Construction — Reasoning (C3R)

Yurchyshyna und Zarli (2009) konzentrieren sich in ihrer Forschungsarbeit auf den Aufbau einer ontologischen Wissensbasis, mit Hilfe derer schließlich Konformitätsüberprüfungen durchgeführt werden können. Das Konzept hinter diesem Ansatz ist es, Konformitätsanforderungen in der Abfragesprache SPARQL Protocol And RDF Query Language (SPARQL) (siehe Abbildung 6.14) zu formulieren. Ein wesentlicher Vorteil hierbei ist, dass mit Hilfe von SPARQL nicht nur die Inhalte der Regeln selbst, sondern auch Informationen aus einem gegebenen Gebäudemodell, welches selbst in der ontologischen Datenbasis dargestellt ist, abgeleitet und verwendet werden können. In dem vorgestellten Forschungsansatz handelt es sich hierbei allerdings nur um Ableitungen sehr einfacher Natur. So wird beispielsweise die Beschreibung von räumlichen geometrisch-topologischen Zusammenhängen und Relationen weitgehend vernachlässigt.

Ein Beispiel für die Übersetzung einer einfachen Vorschrift ist in Quellcode 6.10 zu finden. In diesem Beispiel wird die Aussage *"Jede Tür sollte mindestens 90 cm breit sein"* in SPARQL dargestellt.

```
1 SELECT ?door display xml
2 WHERE
3 { ?door rdf:type ifc:IfcDoor
4 OPTIONAL { ?door ifc:overallWidth ?width
5 FILTER ( xsd:integer(?width) >= 90)}
6 FILTER (! bound( ?width) )}
7
```

Quellcode 6.10: Beispielanfrage mittels SPARQL (Yurchyshyna und Zarli, 2009)

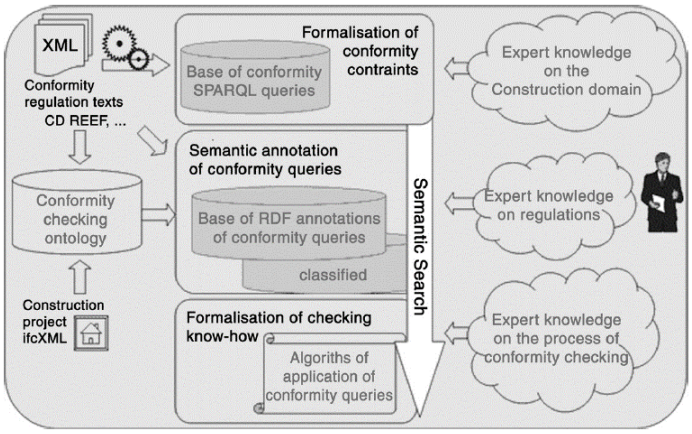


Abbildung 6.14: Aufbau des Softwaretools C3R (Yurchyshyna et al., 2009)

In einem zweiten Schritt des Forschungsansatzes kommt das Potenzial der ontologischen Datenstruktur zum Tragen. Aus den in SPARQL verfassten Vorschriften wird mit Hilfe der OWL (siehe Abschnitt 2.4.2.3) ein ontologisches Schema formuliert. Parallel hierzu werden alle relevanten Daten eines IFC-Gebäudemodells in dem *ifcXML*-Format aufbereitet und auf das ontologische Schema abgelegt. Bisher laufen die beschriebenen Prozessschritte nicht selbstständig, sondern müssen von einem Experten durchgeführt oder aber begleitet werden, der sein Fachwissen über diverse Hilfsmittel, insbesondere mit graphbasierten Schemata in RDF, in den Überprüfungsprozess einbringt. Daher kann an dieser Stelle lediglich von einem halb-automatisierten Prozess gesprochen werden (Yurchyshyna et al., 2009).

Das mit den Gebäudedaten gefüllte ontologische Schema bildet schließlich die Wissensbasis, auf welcher die einzelnen oder kombinierten SPARQL-Anfragen durchgeführt werden können. Technisch gesehen basiert die Durchführung einer solchen Suchanfrage auf einer Graph-Projektion, einem sogenannten Homomorphismus von zwei Graphen, welche die Validierung einer Wissensbasis erlaubt. Hierzu wird die Aussage der Suchanfrage, die prinzipiell auch den Inhalt einer Vorschrift enthalten kann, so in einen RDF-Graphen übersetzt, dass dieser das Gegenteil der Anfrage aussagt. Anschließend kann der Graph hinsichtlich seines Wahrheitsgehalts mit der ontologischen Wissensbasis überprüft werden. Ist das Ergebnis dieser Überprüfung eine Schnittmenge von Anfrage und Ontologie, so entspricht mindestens ein Element des Gebäudemodells der gegenteiligen Vorschrift, und die Konformität von Regelwerk und Modell ist nicht gegeben (Yurchyshyna et al., 2009). Der schematische Aufbau des Gesamtsystems von C3R ist in Abbildung 6.14 dargestellt.

6.5 IfcConstraint-Modell

Mit Veröffentlichung der Version 2x2 wurde in das IFC-Datenschema die Entität *IfcConstraint* eingeführt, mit welcher eine Einschränkung, ein Grenzwert oder eine Randbedingung dargestellt werden kann. Diese definierten Randbedingungen können auf ein Objekt oder den Wert einer Eigenschaft angewendet werden. Die resultierenden Randbedingungen lassen sich mit sogenannten mathematischen Allaussagen vergleichen, welche Aussagen über alle Elemente einer Domäne treffen. So kann eine solche Bedingung

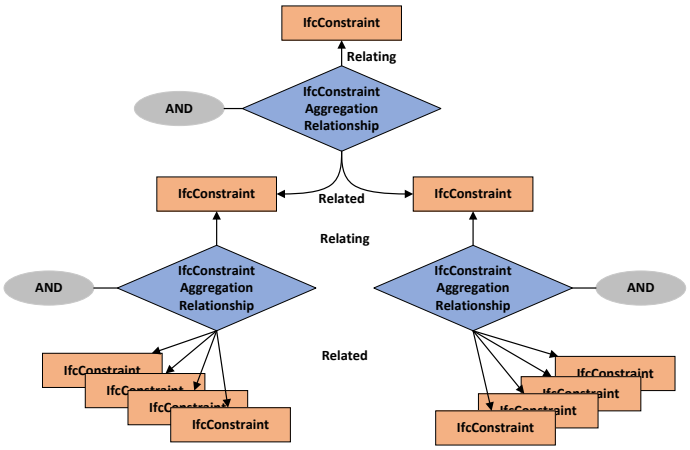


Abbildung 6.15: Kopplung von IfcConstraints im IFC-Datenschema (BuildingSmart, 2007)

beispielsweise ausdrücken, dass die Regel X für alle Elemente, welche sich durch die Eigenschaft Y auszeichnen, gilt. Diese Aussage kann mathematisch dargestellt werden als (Niemeijer et al., 2009)

$$\forall x \in \mathbb{N}(x \geq 0)$$

Technisch hat das *IfcConstraint* seinen Ursprung im *IfcConstraintResource*-Schema und kann auf jedes Objekt angewendet werden, das ein Subtyp von *IfcObjectDefinition* oder *IfcPropertyDefinition* ist (über die Relation *IfcRelAssociatesConstraint*). Überdies können Einschränkungen auf bestimmte Ressourcenobjekte, wie z.B. eine *IfcProperty*, angewendet werden (über die Relation *IfcResourceConstraintRelationship*) (Hjelseth, 2010a). Zudem kann definiert werden, ob es sich bei der Bedingung um ein harte ("muss erfüllt sein"), eine weiche ("sollte erfüllt sein") oder lediglich um einen Hinweis handelt. Somit kann eine Beschränkung qualitativer (objektiver Zwang) oder aber auch quantitativer (gemessener Zwang oder Metrik) Natur sein.

Über das Kriterium kann also eine objektive Einschränkung und somit der Zweck der Prüfung dargestellt werden. Im Zusammenhang mit Bauvorschriften können beispielsweise Zwangswerte, gültige Wertebereiche oder Aufzählungen definiert werden (z.B. muss der Wert von X größer als A, aber kleiner als B sein) und so kann der gültige Lösungsbereich eingegrenzt werden.

$IfcConstraints$ können somit auch als Hilfsmittel herangezogen werden, um Regelwissen und insbesondere Integritätsbedingungen, die in Vorschriften enthalten sind, abzubilden (Hjelseth, 2010a).

Wesentlicher Vorteil dieser Methode ist neben der einfachen Erstellung von allgemeingültigen Randbedingungen die neutrale Darstellung direkt im Datenmodell mit Hilfe des IFC-Schemas sowie die Möglichkeiten zur Parametrisierung der definierten Randbedingungen. Zudem führen Niemeijer et al. (2009) den Vorteil an, dass man mit Hilfe der *IfcConstraints* und der Verankerung im Schema keine Bedingung übersehen kann und diese so auch nicht vergessen werden.

Hjelseth (2009b) verwenden das *IfcConstraint*-Modell für den bereits vorgestellten RASE-Ansatz. Die mit Hilfe der RASE-Syntax übersetzten Regeln können als solche *IfcConstraint* dargestellt werden und

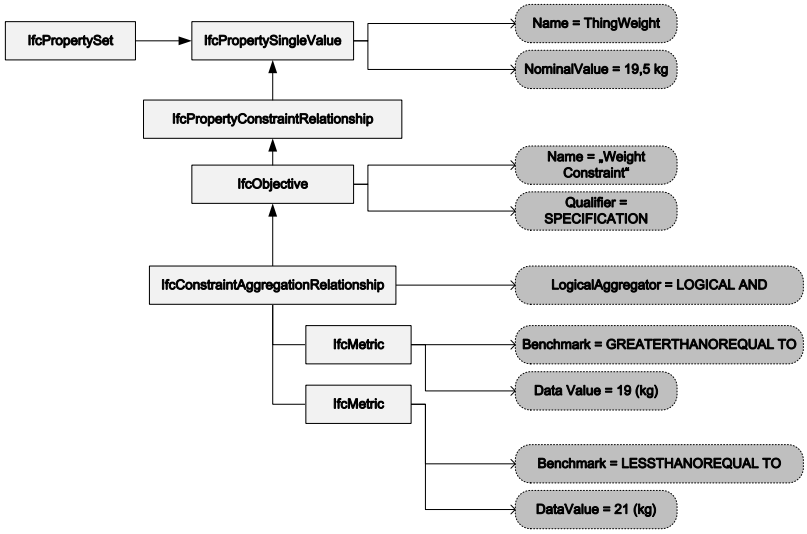


Abbildung 6.16: IfcConstraints IFC-Datenschema siehe BuildingSMART; angelehnt an (BuildingSmart, 2007)

sind somit anwendbar. Aufgrund der Abhängigkeit von den explizit verfügbaren Daten im *Constraint*-Modell, welches heutzutage nur von wenigen Anwendungen unterstützt wird, hat dieser Ansatz jedoch keine weiteren wesentlichen Fortschritte gemacht.

Niemeijer et al. (2009) untersuchen die Machbarkeit der Überprüfung eines IFC-Modells mit Hilfe von *IfcConstraints* in der prototypischen Umsetzung *Check-mate*. Ziel dieser Untersuchungen ist es nachzuweisen, dass man mit Hilfe der *Constraints* an Gebäudemodellen computergestützt überprüfen kann, ob ein Gebäudeentwurf alle Bauvorschriften, Anforderungen usw. erfüllt. Als Beispiel wird hierzu die Massenanpassung im Wohnungsbau, bei welcher Kunden die Anforderungen an die Gestaltungsplanung ohne die Hilfe des Architekten verändern, herangezogen.

Niemeijer et al. (2014) stellen in ihrer Arbeit eine Methode vor, welche mit Hilfe von NLP automatisiert aus natürlicher Sprache interpretierte Designbeschränkungen der niederländischen Bauvorschriften für Dachgaubenerweiterungen ableitet und diese dann in *IfcConstraints* transformiert. Diese *Constraints* können anschließend auf Gebäudeinformationsmodelle angewendet werden. Niemeijer et al. (2014) halten fest, dass mit ihrer Methode in einer prototypischen Umsetzung 44 % der Testdaten ohne menschliche Eingriffe interpretiert werden konnten.

Ein einfaches Beispiel für die Definition einer solchen Zwangsbedingung ist in Quellcode 6.11 dargestellt. Hier wird für den Typ *IfcFontStyle* der Datentyp $STRING$ festgelegt und anschließend mit dem Signalwort *WHERE* definiert, dass der Wert aus einer festgelegten Enumeration stammen muss.

```
TYPE IfcFontStyle = STRING;
  WHERE
    WR1 : SELF IN ['normal','italic','oblique'];
END_TYPE;
```

Quellcode 6.11: Definition einer Zwangsbedingung für den Typ IfcFontStyle gemäß des EXPRESS-Schemas von IFC (ISO 16739)

Ähnliche Integritätsbedingungen lassen sich so beispielsweise auch für Zahlenwerte definieren. In Quellcode 6.12 beschränkt der *WHERE*-Block die Länge *IfcPositiveLengthMeasure* auf Werte größer null. Die Definition dieser Randbedingung ist allgemeingültig und wird auf alle Elemente angewandt, welche ein Attribut vom Typ *IfcPositiveLengthMeasure* tragen. Für Überprüfungen müssen allerdings oft nur einzelne Elemente eines bestimmten Typs untersucht werden (Niemeijer et al., 2009).

```
TYPE IfcPositiveLengthMeasure = IfcLengthMeasure;
  WHERE
    WR1 : SELF > 0.;
END_TYPE;
```

Quellcode 6.12: Definition einer Zwangsbedingung für den Typ IfcPositiveLengthMeasure gemäß des EXPRESS-Schema von IFC (ISO 16739)

6.6 Ansätze auf Basis von Natural Language Processing (NLP)

Eine weitere Gattung von Forschungsansätzen beschäftigt sich mit der automatisierten Extraktion von Informationen aus regulatorischen Dokumenten in reiner Textform. Eine weit verbreitete Methode für diesen Zweck ist die Anwendung von NLP-Techniken wie *Entity Recognition*, *Satzaufteilung*, *Tokenisierung*, *Tagging* oder *syntaktisches Parsing*. Im Allgemeinen handelt es sich bei dem NLP um eine Sammlung von Methoden und Algorithmen, die zu der Gruppe der künstlichen Intelligenz gezählt werden und sich auf die Bestimmung und Extraktion der Bedeutung von Texten in natürlicher Sprache beziehen. Bereits in den 1960er Jahren begannen die Entwicklungen in diesem Bereich als Teil- und Mischgebiet aus künstlicher Intelligenz und Linguistik.

Dabei konzentrieren sich NLP-Algorithmen im Wesentlichen auf einen teil- bzw. vollautomatisierten Wissens- und Informationsaustausch zwischen Mensch und Maschine. Die zentrale Herausforderung bei der Anwendung von NLP-Methoden ist das Verstehen von natürlicher Sprache, was auch die Extraktion von Regeln aus Bauvorschriften beinhaltet. Die entwickelten Methoden sollen dem Computer die Möglichkeit geben, aus natürlichsprachlichen Text, wie z. B. Bauvorschriften, Bedeutungen abzuleiten und logische Regeln für die Weiterverarbeitung zu generieren. Dabei stützen sich diese Methoden unter anderem auch auf die vorhergesagte Wahrscheinlichkeitsverteilung der zu extrahierenden Sprachausdrücke, um die Treffsicherheit der Übersetzung zu erhöhen.

Anwendungen dieser Methoden für Bauvorschriften finden sich in einigen neueren Forschungsarbeiten, wie insbesondere bei Nawari und Alsaffar (2015); Zhang und El-Gohary (2012, 2015, 2016a,b,c); Zhang et al. (2017).

Zhang und El-Gohary (2012, 2015, 2016a,b,c); Zhang et al. (2017) stellen in ihren Publikationen unterschiedliche Ansätze vor, NLP-Methoden zunächst auf regulatorische Anforderungen aus Bauvorschriften automatisch anzuwenden und anschließend die erhaltenen Informationen in logische Klauseln zu transformieren und so zu formalisieren.

Der Prozess besteht hierbei in der Regel aus fünf wesentlichen Schritten: Textklassifikation, Informationsextraktion, Anwendung von Transformationsregeln, Implementierung und schließlich die Evaluierung.

In der Kernverarbeitungsphase werden relevante Texte innerhalb eines Textkorpus mit Hilfe von Textklassifikation identifiziert. Anschließend werden Zielinformationen in diesen relevanten Sätzen extrahiert und transformiert. Dabei wird eine Verarbeitung von Teilsätzen durchgeführt, sodass zunächst nur bestimmte Begriffe extrahiert und verarbeitet werden. Hierbei helfen domänenspezifische semantische Methoden bei der Analyse von zusammengesetzten Satzstrukturen, da eine vollständig automatisierte Extraktion-Technik zu komplex wäre und nur vage Ergebnisse liefern würde. Zum Nachweis der Tragfähigkeit wurden die vorgestellten Methoden auf den *International Building Code (IBC)* in der Version aus dem Jahr 2009 (International Code Council, 2006) angewendet, um hier die wesentlichen Informationen einzelner quantitativer Anforderungen automatisiert in logische Klauseln zu transformieren.

Bei dem IBC handelt es sich um von der ICC entwickelte Modellrichtlinien, welche sich sowohl auf Gesundheits- als auch mit Sicherheitsaspekten für Bauwerke konzentrieren und vorschriftsmäßige und leistungsbezogene Anforderungen formulieren. Wesentliches Ziel des IBC ist es, die öffentliche Gesundheit und Sicherheit zu schützen und gleichzeitig unnötige Kosten und die Bevorzugung bestimmter Materialien oder Bauweisen zu vermeiden (International Code Council, 2006).

In einer der letzten Publikation geben Zhang und El-Gohary (2016c) eine Trefferquote in der Genauigkeit der Erkennung von Verstößen von 100 % bei perfekten und 87,6 % Präzision zu 98,7 % Trefferquote mit unvollständigen Modellinformationen an.

Zhang et al. (2017) halten allerdings fest, dass es eine Reihe von Faktoren gibt, welche es schwierig machen, Regeln aus Verordnungen zu extrahieren: Oft seien die Regeln unvollständig, widersprechen sich oder seien gar unvereinbar mit der Realität der AEC-Industrie, für welche diese entwickelt werden. Darüber hinaus konzentrieren sich die bisherigen Entwicklungen ausschließlich auf quantitative Aussagen, welche nicht zu den komplexen Inhalten hinsichtlich des Regelgehaltes gezählt werden können. Weitergehende Studien, welche sich auf eben diese Herausforderung fokussieren, stehen aus, und es bleibt abzuwarten, inwiefern hier NLP-Methoden verlässliche Ergebnisse liefern können.

Weiterhin ist an dieser Stelle anzumerken, dass sich die zitierten Forschungen ausschließlich auf die quantitative Anforderungen beziehen, welche nicht den wesentlichen Anteil der in regulatorischen Dokumenten beschriebenen Regeln darstellt. Die automatisierte Erkennung, Übersetzung und Prüfung dieser Angaben ist vergleichsweise simpel.

6.7 Lee (2011): Building Environment and Analysis Language (BERA)

Eine weitere Gattung von Ansätzen zum ACCC ist die Einführung bzw. Verwendung domänenspezifischer Programmier- bzw. Abfragesprachen. Viele der zuvor vorgestellten Ansätze haben den wesentlichen

Nachteil, dass lediglich die Inhalte von Regelwerken selbst, nicht aber die Aufbereitung von Informationen, die nicht in der Richtlinie oder in dem zu Grunde liegenden Datenmodell enthalten sind, fokussiert werden. Die formale Strukturierung der Inhalte von Normen garantiert eine eindeutige Falsifizierbarkeit für einzelne Entscheidungsfälle, jedoch können die Formulierungen mit zunehmender Komplexität der abgebildeten Vorschrift deutlich an Übersichtlichkeit verlieren, so dass die Lesbarkeit durch einen menschlichen Nutzer nur noch schwer möglich ist und gleichzeitig der Aufwand für eine Übersetzung stark zunimmt.

An dieser Stelle versprechen Ansätze, die auf domänespezfischen Sprachen beruhen, eine Lösung dieses Problems, da diese einen größeren Freiraum für die Formulierungen von Aussagen einräumen. Zu den grundlegenden Einsatzgebieten dieser Sprachen gehört die Erstellung von Teilmengen, die Validierung von Informationen oder die Ausführung von Ad-hoc-Anfragen. Dabei basiert eine Anfragesprache auf der festen Definition von Operatoren, deren Ausführung eindeutige Ergebnisse liefern (Daum, 2018).

Die Verwendung einer solchen Sprache ermöglicht die Abbildung deutlich komplexerer Sachverhalte auf einfache Art und Weise, jedoch nimmt die Gefahr von Inkonsistenzen, wie z. B. Fehlinterpretationen, durch den weiter gesteckten Rahmen der Syntax deutlich zu. Damit die Konsistenz dennoch gewahrt bleibt, muss der Syntax der Sprache ein fester Rahmen gegeben werden, an welchen sich der Anwender bei seinen Übersetzungen hält. Dieser Rahmen ist maßgeblich von einer (vgl. *domänenspezifisch*) oder mehreren (vgl. *domänenübergreifend*) Domänen abhängig, in welcher sich die Sprache bewegt. Innerhalb der Arbeit beziehen sich die Ansätze insbesondere auf den Bereich des Bauwesens oder aber dessen Teilbereiche. Daher ist es sinnvoll, der Syntax ausschließlich für diesen Bereich feste Definitionen vorzugeben und so den Handlungsrahmen des Anwenders zu begrenzen (Eastman et al., 2009).

Für die Übersetzung von komplexem Regelwissen aus Bauvorschriften führt Lee (2011) die hochgradig domänenspezifische Sprache BERA ein. Ziel der Einführung von BERA ist es, eine einfach zu bedienende, regelkonforme Spracharchitektur mit hoher Wiedergabetreue, Erweiterbarkeit und Kompaktheit anzubieten (Lee et al., 2015). Im Gegensatz zu deklarativen Ansätzen wie SPARQL oder SQL handelt es sich um eine imperative Sprache, welche einige Ähnlichkeiten zu der Programmiersprache Java aufweist. Allerdings richtet sich BERA nicht explizit an Softwareentwickler, sondern an Domäne-Experten wie Architekten, Ingenieure, Planer, Gutachter und Eigentümer. Lee (2011) formuliert die folgenden Ziele für die Entwicklungen rund um BERA:

Einfache Anwendung (Ease of use): Die Zielkunden von BERA sind insbesondere Anwender ohne fundierte Programmierkenntnisse. Die Sprache ist daher darauf ausgelegt, zum einen auf die Informationen eines Gebäudemodells so effizient und einfach wie möglich zuzugreifen und andererseits mittels einer leicht verständlichen Syntax auch die Verarbeitung dieser Daten zu erleichtern.

Portabilität (Portability): Um die Komplexität der Sprache auf einem niedrigen Niveau zu halten, baut BERA auf dem BERA Objektmodell (BOM), einer vereinfachten, abstrahierten Version der komplexen Bauwerksstruktur von IFC auf (Solihin, 2016). Dieses garantiert, dass jede Anfrage von BERA generisch gestaltet werden kann und für jedes Modell in gleicher Art und Weise durchgeführt wird. Auf diese Weise kann die Sprache auch als Standard für viele unterschiedliche Gebäudemodelle oder aber BIM-Datenplattformen dienen. Darüber hinaus wird damit beabsichtigt, dass dem Nutzer ein intuitiver Zugriff auf den Informationsgehalt eines Datenmodells ermöglicht wird. In seinem Forschungsansatz konzentriert sich Lee (2011) zunächst ausschließlich auf räumliche und

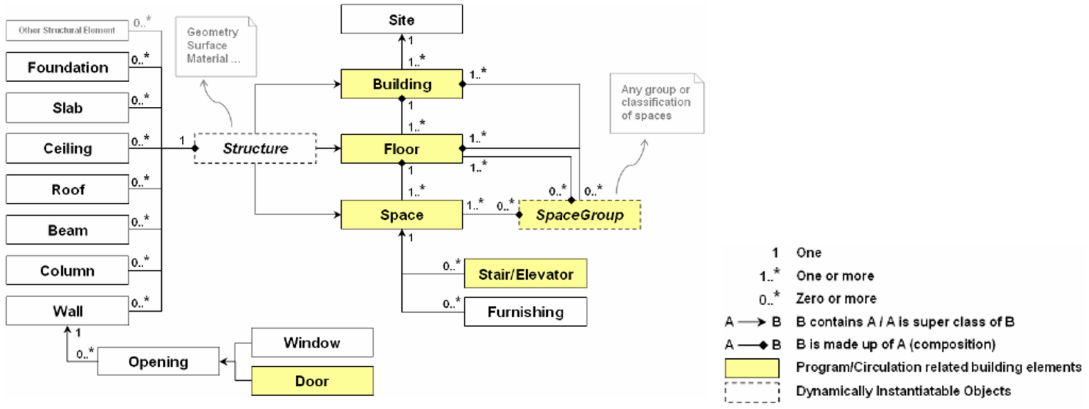


Abbildung 6.17: Abbildung räumlicher Informationen im BOM (Lee, 2011); das BERA Objektmodell ist eine starke Vereinfachung des IFC-Datenmodells

geometrische Informationen bzw. Anfragen. Dieses spiegelt sich auch in dem BOM wider, welches vereinfacht in Abbildung 6.17 dargestellt ist.

Erweiterbarkeit(Exensibility): Als ein solcher Standard kann die Syntax von BERA ähnlich wie bei anderen Programmiersprachen sukzessive um neue Elemente, Objekte und Funktionen erweitert und ausgebaut werden. Dabei bleibt es dem Anwender überlassen, welche Elemente dieser in BERA integrieren möchte. Diese Eigenschaft und Flexibilität für den Anwender grenzt BERA deutlich von anderen existierenden Ansätzen ab.

Welchen Vorteil diese Abstraktion des Modells mit sich bringt, zeigt sich in dem Sprachdesign von BERA, was am Beispiel in Quellcode 6.13 verdeutlicht wird.

```
1 // Wanted: name of a space object called "department"
2 // IFC-centered statement
3 space.ifcRelAssociatesClassification.ifcClassificationReference.ifcLabel.getString();
4 // BOM-centered statement
5 space.department;
```

Quellcode 6.13: Vergleich eines IFC- und BOM-orientierten Befehls in BERA (Lee, 2011)

Das Sprachdesign von BERA baut auf vier zentralen Bestandteilen auf:

Reference Directives: Wie in bekannten Programmiersprachen lassen sich auch in der BERA-Umgebung externe Bibliotheken oder aber Datensätze integrieren. Dieser Teil der Syntax ist vergleichbar mit den Direktiven *import* in *Java*, *using* in *C#*, oder *include* in *C/C++*.

Object Model Definition and Declaration: Durch die Vereinheitlichung der Struktur des Gebäudemodells in dem BOM lassen sich Objekte und Parameter für das Sprachdesign präzise definieren. (Lee, 2011) implementiert die Deklarationen von Objektklassen mit den zugehörigen Parametern in der sogenannten *Object Model definition and declaration* und macht diese so in BERA für den Anwender als Klassen zugänglich. Auf die Informationen eines solchen Objektes kann in der

BERA-Syntax mittels der sogenannten Punkt-Operation (engl. *dot notation*) intuitiv zugegriffen werden. So repräsentiert beispielsweise der Befehl

Space.Floor.name = "Level 1"

den Zugriff auf den Parameternamen der Klasse *Floor*, welche wiederum eine Subklasse des Objektes *Space* ist, und belegt diesen über die Operation = mit der Zeichenkette *Level 1*. Innerhalb dieses Befehls ist also nicht nur eine Zugriffs-, sondern auch eine logische Operation enthalten. Die verfügbaren logischen Operatoren für unterschiedliche Datentypen in BERA sind identisch mit denen von weitverbreiteten Programmiersprachen. Darüber hinaus bietet BERA Elemente wie Rekursion, Negation, Vererbung, Polymorphismus, algebraische Operationen und so weiter, die bereits aus einschlägigen Programmiersprachen bekannt sind (Nawari, 2018).

Rule Definition: Für die Definition von Vorschriften gibt es in der Syntax die sogenannte *rule definition*, welche eine objektorientierte Formulierung von Regelwissen erlaubt. Ähnlich wie bereits bei den Klassen des BOM werden auch die Regelungen als Objekt deklariert, wobei diesen zusätzlich zur Initialisierung weitere Objekte übergeben werden können. Wird ein Regel-Objekt in einem BERA-Befehl mit dem entsprechenden Parameter initialisiert, wird der Inhalt dieser Vorschrift direkt ausgeführt. In dem Beispiel in Quellcode 6.14 wird in Zeile 1 die Regel $myrule$ definiert, welche bei der Initialisierung ein Objekt der Klasse *Space* übergeben bekommt. Bei Aufruf dieses Objektes in einem BERA-Befehl wird der Inhalt der Regel, also die Anforderungen in Zeile 2 bis 4 direkt an diesem Objekt überprüft.

```
1 Rule myrule(Space space) {
2   space2.area > 1000;
3   space2.Floor = "Level_1";
4   space2.security = "public";}
```

Quellcode 6.14: Definition einer Regel in BERA (Lee, 2011)

Execution Statement: Der letzte Bestandteil von BERA beschreibt die Deklaration von sogenannten *execution statements*, welche als freie, objektorientierte Routinen in einem BERA-Befehl aufgerufen werden können. Ein Beispiel für eine solche Aussage ist die Anweisung *get(object)*, welches in Abhängigkeit des übergebenen *object* den Zugriff auf dieses Objekt beschreibt.

Mit der resultierenden Sprache lassen sich schließlich sowohl Informationszugriffe, als auch Vorschriften für das Gebäudemodell formulieren. Im Rahmen eines Forschungsprojekts und für den Nachweis der Tragfähigkeit wurde für BERA ein Editor und Compiler in die Umgebung des Solibri Model Checker (siehe Abschnitt 6.12) implementiert. (Lee, 2011) zeigt mit diesem *BERA Language Tool* und einigen Anwendungsbeispielen die verschiedenen Funktionen von BERA auf.

```
1 Space midOffice {
2   Space.area > 600;
3   Space.area < 900;
4   Space.height > 9;
5   Space.name = "office";
6   Space.name != "shared";}
7 get(midOffice);
```

Quellcode 6.15: Abfrage einer definierten Raumklasse eines Gebäudemodells mit BERA (Lee, 2011)

In Quellcode 6.15 ist die Abfrage einer definierten Raumklasse aufgeführt und mit Hilfe von BERA dargestellt. In Zeile 1 wird zunächst das Objekt *midOffice* vom Typus *Space* (Raummodell) deklariert und in Zeile 2 bis 6 mit Werten belegt. Über den Befehl *get(midOffice)* in Zeile 8 werden nun in dem korrespondierenden BOM alle Objekte gesucht, die den definierten Randbedingungen des Objektes *midOffice* entsprechen. Das Ergebnis dieser Abfrage kann abschließend graphisch im SMC dargestellt werden.

```
1 Space start = getSpace("laboratory") + getSpace("lobby");
2 Space end {
3     Space.area > 600;
4     Space.Floor.number > 0;
5     Space.Floor.height > 10;
6     Space.name = "office";}
7 Path myPath = getPath(start, end);
8 get(myPath);
```

Quellcode 6.16: Abfrage von Laufwegen innerhalb eines Gebäudemodells mit BERA (Lee, 2011)

In Quellcode 6.16 ist ein Beispiel für eine Laufweganalyse mit Hilfe von BERA dargestellt. Ähnlich wie bereits bei der Raumanalyse zuvor werden in diesem Fall die zwei Objekte *start* und *end* der Klasse *Space* deklariert. Die Klasse *Path* ist ein Beispiel für ein Objekt von BERA, welches nicht direkt zu dem BOM gehört, aber zu der Weiterverbreitung von räumlichen Informationen eingeführt wurde. Innerhalb dieser Klasse sind Algorithmen zur Analyse von Raummodellen als Funktionen hinterlegt, welche mittels der Daten des BOM ausgeführt werden können. Das Ergebnis der Laufweganalyse kann abschließend wiederum im SMC graphisch dargestellt werden.

BERA wurde bereits praktisch eingesetzt für die Prüfung der amerikanischen Richtlinien und hier insbesondere den Richtlinien des *Hospital Design Guide* oder des *U.S. Courts Design Guide* (Lee, 2011). In diesem Fall sind die hauptsächlichen Anwendungsgebiete von BERA die Validierung von Evakuierung-Strategien und Raumprogrammen (Nawari, 2018).

Zusammenfassend kann festgehalten werden, dass sich die Entwicklung von BERA sehr stark auf die Formulierung von räumlich bzw. geometrisch orientierte Richtlinien fokussieren. Allerdings werden innerhalb des vorgestellten Ansatzes viele Instrumente präsentiert, die den Grundstein für eine Ausweitung dieser Sprache legen. Auf diese Weise zeigt BERA ein Potenzial für den Einsatz im Bereich des ACCC auf. Allerdings fehlt der imperativen Sprache eine allgemeingültige logische Basis, welche notwendig ist, um eine höhere Vielseitigkeit zu erreichen, insbesondere um komplexere Strukturen

zu definieren. Einen generellen Ansatz zur Analyse von Bauwerksmodellen liefert BERA nicht, zumal allgemeine Operationen wie beispielsweise die Selektion nach Typzugehörigkeit fehlen.

6.8 Solihin (2016): BIM Rule Language (BIMRL) und Concept Graphs

Ein weiterer Ansatz, welcher auf der Einführung einer Programmier- bzw. Anfragesprache basiert, ist die BIMRL (Dimyadi et al., 2016b; Solihin, 2016). Solihin (2016) versucht mit der Einführung der imperativen Programmiersprache BIMRL einerseits, effiziente Abfragen von Informationen aus dem Bauwerksmodell inklusive der enthaltenen Geometrie auch mittels räumlicher Operatoren zu ermöglichen. Gleichzeitig soll BIMRL als standardisierte Regeldefinitionssprache fungieren können und so eine geeignete formale Form für die Speicherung und Austausch von Regeln darstellen.

BIMRL arbeitet auf Grundlage eines relationalen Datenbank-Schemas, welches die IFC-Struktur zu einem sternförmigen Schema vereinfacht. Ähnliche Schemata werden beispielsweise für zentrale Datenbanken verwendet, die sich aus mehreren heterogenen Quellen zusammensetzen und für Analysezwecke optimiert werden sollen. Mit der Einführung dieses vereinfachten Schemas sollen Informationen aus Gebäudemodellen mit Hilfe von Suchanfragen flexibel und schnell abgefragt werden können. Um die Datenbank mit den entsprechenden Informationen zu befüllen, werden IFC-Modell mit Hilfe eines *ETL*-Moduls (*Extract, Transform, Load*) in das vereinfachte Schema geladen.

Die BIMRL-Regelsprache besitzt eine SQL-ähnliche Syntax, die jeweils in drei Elemente untergliedert ist:

CHECK: definiert SQL-ähnliche Filterbedingungen, um betroffene Elemente/Objekte aus dem Gebäudemodell auszuwählen. In diesem Abschnitt können auch mehrere, unabhängige Abfragen gleichzeitig definiert werden, die nacheinander durchgeführt werden und somit verknüpft sind.

EVALUATE: definiert die Auswertungsfunktion, welche letztlich das Regelwissen darstellt. Die definierte Prozedur wird dabei auf die im Abschnitt CHECK erhaltenen Elemente angewendet. In diesem Abschnitt können auch mehrere Auswertungen miteinander verkettet werden, was bedeutet, dass das Ergebnis einer Auswertung an die nachfolgende weitergegeben werden kann. Überdies stehen in diesem Bereich eine Reihe von Operatoren zur Verfügung, mit welchen Informationen aus dem Gebäudemodell verarbeitet werden können, wie insbesondere geometrische Operatoren.

ACTION: definiert die Prozedur, welche angestoßen wird, wenn die Ergebnisse für die Auswertung vorliegen. Mit dieser Funktion kann beispielsweise eine Handlungsempfehlung, eine Manipulation oder ein Kommunikationsprozess angestoßen werden.

Für die Definition und Erweiterung der *CHECK*-, *EVALUATE*-, oder *ACTION*-Elemente gibt es eine *Plug-In*-basierte Programmierschnittstelle. Die Flexibilität und Erweiterbarkeit der Sprache erlaubt es, extrem weitreichende Regeln ohne großen Programmieraufwand zu schreiben, mit Ausnahme der Erweiterungen der Operatoren im *EVALUATE*-Abschnitt, welche fest vorgegeben sind. Überdies können mit der BIMRL auch Variablen definiert werden, wodurch die Regeln auch parametrisiert werden können.

```
CHECK
IfcSpace S, IfcSpace E
WHERE upper(S.LongName) like 'VESTIBULE%' and
upper(E.LongName) like '%PATIENT ROOM%'
COLLECT S.ElementID StartID, S.Name StartName,
Starting PointSpace to AvoidNearest ElevatorNearest Staircase
S.LongName StartLName, E.ElementID DestID,
E.Name DestName, E.LongName DestLName;
EVALUATE
ComputePath(StartID, DestID, "AVOID StartElemName='12003'
or EndElemName='12003'") Output ?PathFound;
ACTION
When ?PathFound = 1
{ PRINT RESULT SAVE INTO TABLE CIRCPATH DRAW COLOR RED
With BACKGROUND (IfcSpace) HighLight (startID, DestID)
COLOR CYAN TRANSPARENCY 0.75
Save Into X3D 'c:\temp\pathFound.x3d'};
When ?PathFound = 0
{ PRINT RESULT SAVE INTO TABLE CIRCPATH APPEND DRAW
with BACKGROUND (IfcSpace) HighLight (startID, DestID)
COLOR MAGENTA
Save Into X3D 'c:\temp\pathNotFound.X3D'};

```

Quellcode 6.17: Beispielanfrage mittels BIMRL (Solihin et al., 2016)

In Quellcode 6.17 ist ein Beispiel für den Einsatz der BIMRL im Bereich der Barrierefreiheit dargestellt. Die dargestellte Regel fragt zunächst alle vorhandenen Paare von Vorräumen und Patientenzimmern in dem *CHECK*-Statement ab. Im *EVALUATE*-Teil der Regel berechnet und bewertet so die Funktion *ComputePath()* den kürzesten Pfades für jedes Paar. Im *ACTION*-Statement werden die jeweiligen Pfade entsprechend der Bewertung visualisiert und als Ergebnis ausgegeben.

Überdies führen Solihin und Eastman (2015a) ein Dokumentationswerkzeug ein, um die Anforderungen präziser und detaillierter zu erfassen. Solihin und Eastman (2015a) verwenden hierzu Konzeptgraphen (*engl. concept graphs*), um das semantische Wissen von Regeln darzustellen. Bei der Wahl der Methode wurde insbesondere darauf geachtet, dass die Darstellung des Regelwissens dazu verwendet werden kann, Softwareentwicklern bei der Implementierung der Regeln zu helfen. Daher dient dieser Ansatz auch in erster Linie dazu, eine einheitliche Auslegung und Darstellung der Vorschriften zu erreichen, sodass diese anschließend von Software-Experten umgesetzt werden können. Die Konzeptgraphen eigenen sich aufgrund ihrer Aussagekraft und Konformität mit der First Order Logic.

Ein *Concept Graph* definiert mit Hilfe von Konzept-Knoten (Rechteck), konzeptuellen Beziehungs-Knoten (Ellipsen) und Funktions-Knoten (Diamantform) ein semantisches Netz. Ein Konzept-Knoten repräsentiert dabei typischerweise ein Objekt, kann aber auch erweitert werden und so eine ganze atomare Regel darstellen. Solihin und Eastman (2016) verwenden diese Methode, um eine Regel als Reihe von miteinander verbundenen Graphen, wie in Abbildung 6.18 schematisch gezeigt, abzubilden. Die Knoten werden mit gerichteten Kanten miteinander verbunden und ergeben so einen Graphen.

In Abbildung 6.20 ist eine Regel als *Concept Graph* dargestellt, welche die Existenz der Eigenschaften *Name* und *LongName* für alle IFC-Entität *IfcSpace* prüft. Weiterhin wird geprüft, ob der Entität eine valide Klassifikation über das entsprechende Referenzobjekt *IfcClassificationReference* zugeordnet ist.

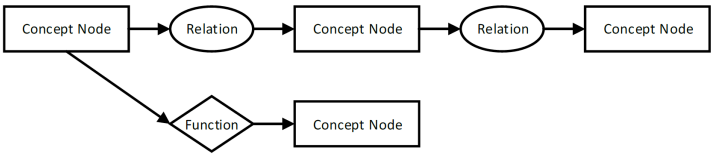


Abbildung 6.18: Schematische Darstellung eines *Concept Graph* (Solihin und Eastman, 2016)

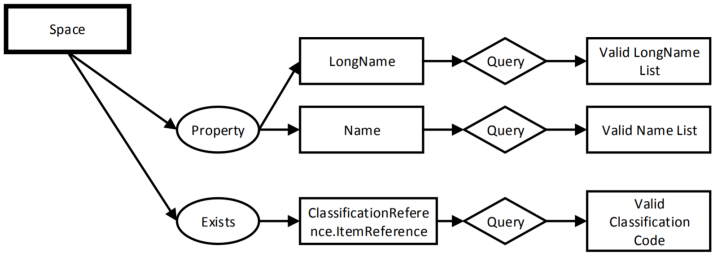


Abbildung 6.19: Darstellung der Prüfung der Attribute *Name* und *LongName* sowie der Zurodnung einer gültigen Klassifikation für die Entitäten *IfcSpace* als *Concept Graph* (Solihin und Eastman, 2016)

Der in Abbildung 6.20 dargestellte Graph kann nun auch mit Hilfe der FOL wie folgt dargestellt werden (Solihin und Eastman, 2016):

$$\forall a(Space(a)) \wedge \exists a((Space(a) \wedge Query(a, Name)) \wedge (Space(a) \wedge Query(a, LongName)) \wedge (\exists(Space(a) \wedge Query(a, IfcClassificationReference, ItemReference)))$$

Als zusätzliches Notation-Element führen Solihin und Eastman (2015b) das *Constraint* ein. Dieses kann auf jeden Knoten im Graphen angewendet werden, der angibt, für welches Konzept das *Constraint* gilt. Da *Constraints* die Spezifität der Konzepte, auf die sie sich beziehen, definiert haben, müssen sie sich auf die explizit definierten Eigenschaften oder Beziehungen in der IFC-Datei verlassen. Dazu gehören Element- und Typ-Eigenschaftsnamen, Eigenschaftswerte, Klassifizierungen oder Beziehungen (Solihin et al., 2016).

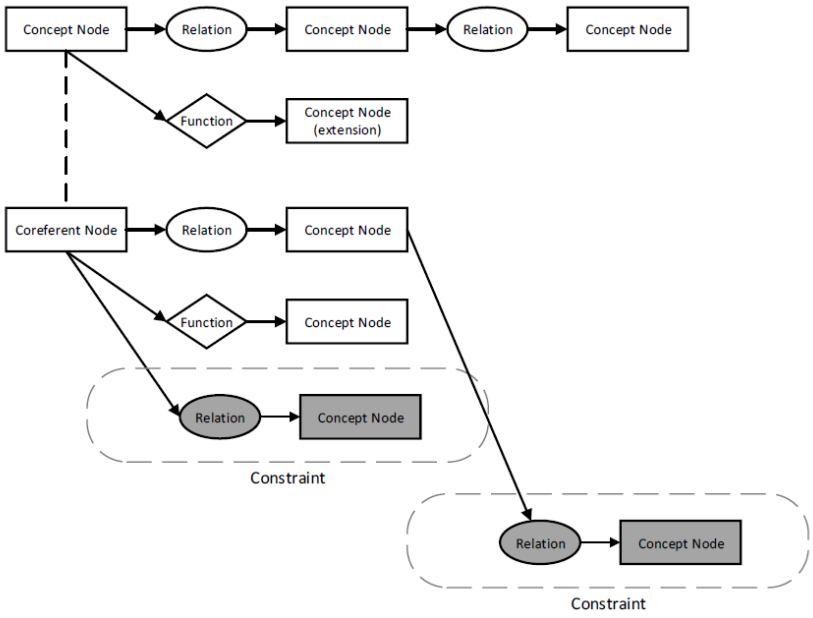


Abbildung 6.20: Schematische Darstellung eines Concept Graph mit Constraints (Solihin und Eastman, 2015b)

6.9 Park und Lee (2016): KBimCode

In ihrem Ansatz kombinieren Park und Lee (2016) Methoden des NLP mit der eigenen deklarativen Skriptsprache KBimCode für die Darstellung von Regeln. Mit Hilfe der NLP-Methoden wird zunächst die Struktur eines zu übersetzenden Satzes in drei Teile gegliedert: Substantiv- (Objekte/Eigenschaften) und Verbphrasen (Methoden) sowie das Verhältnis von Sätzen untereinander (Logik). Hierbei können Objekte physisch greifbare Elemente eines Bauwerksmodells (z.B. Gebäudeelemente) oder auch nicht-physisch greifbare Elemente (z.B. Geschoss oder Prozess) sein. Die Eigenschaften werden in drei Typen untergliedert: (1) explizite Eigenschaften (z.B. Objektname), (2) relationale Eigenschaften (z. B. Abstand zwischen zwei Objekten), (3) berechnete Eigenschaften (z.B. Bodenfläche).

Identifizierte Verbphrasen werden in sogenannte *high-level* Methoden übersetzt, welche die Objekte und Eigenschaften als Argumente verwenden. Die Methoden werden als *high-level* bezeichnet, weil insgesamt vier unterschiedliche Ebenen der Komplexität eingeführt werden. In einer ersten Ebene werden lediglich die Objektinstanz und die Eigenschaft der jeweiligen Objektinstanz behandelt. In einer zweiten Ebene können die vier unterschiedlichen Eigenschaften (Grundeigenschaft, geometrische Eigenschaft, komplexe Eigenschaft, relationale Eigenschaft) verarbeitet werden. Für spezifische Prüfungen sind in einer dritten Ebene Methoden definiert, welche Informationen, wie z.B. die Objektbreite in eine Unterkategorie der geometrischen Eigenschaft, verarbeiten können. In der vierten Ebene werden spezifisch definierte Methoden, wie etwa *getObjectWidth()*, angeboten.

Die Beziehung der Elemente untereinander wird mit Hilfe von logischen Ausdrücken wie *IF/THEN/ELSEIF/ELSE* abgebildet. So kann ein Satz in die beiden Teile *Zustand* und *Inhalt* aufgeteilt werden. Als Beispiel wird der Satz

Ein Gebäude mit sechs oder mehr Stockwerken und einer Gesamtfläche von 2.000 Quadratmetern oder mehr soll mit einem Aufzug ausgestattet sein.

folgendermaßen dargestellt: (1) "Ein Gebäude hat mehr als sechs Stockwerke" und (2) "Die Gesamtfläche eines Gebäudes beträgt mehr als 2.000 Quadratmeter" (Park und Lee, 2016). Inhalt der entsprechenden Regel ist, dass ein Gebäude einen Aufzug haben muss. Die entsprechende Übersetzung in die Skriptsprache *KBimCode* ist in Quellcode 6.18 dargestellt.

```
Check(BA_64_1){
  IF(getBuildingStoriesCount()>=6
  AND getGrossFloorArea()>=2000)
  THEN isExist(Elevator)=TRUE
  ENDIF}
```

Quellcode 6.18: Übersetzung einer Regel mit Hilfe von KBimCode (Park und Lee, 2016)

Im Rahmen des staatlich geförderten *KBim*-Projekts, ist die Softwareanwendung *KbimAssess-lite* entwickelt worden, in welcher eine erste praktische Anwendung von *KBimCode* möglich ist. Mit dieser wurden Auszüge des *Korean Building Act* (Korea Legislation Research Institute, 2008) als explizite berechenbare Regeln mit Hilfe der Skriptsprache übersetzt. Die Entwicklungen rund um das *KBimCode*-Projekt werden auch aktuell fortgeführt und weitere Validierungen der Sprache sind bereits angekündigt. Im Rahmen dieses Projekts wurde mittlerweile auch eine visuelle Notation entwickelt, mit welcher man die Inhalte einer Bauvorschrift abbilden kann (Kim et al., 2018, 2017a). Allerdings gibt es hierzu bisher noch keine weiterführende Literatur mit konkreten Umsetzungen.

6.10 Martins und Monteiro (2013): LicA

Martins und Monteiro (2013) stellen ihre Arbeiten zu einer neuartigen Methode und prototypischen Softwareanwendung *LicA* für die automatische Regelprüfung von (Haus-)Wassernetzwerken vor. Die entwickelten Methoden und die Anwendung beziehen sich zwar explizit auf die in Portugal geltenden Vorschriften, Richtlinien und Spezifikationen, können aber laut den Autoren auch für die Anwendung auf Richtlinien anderer Länder angepasst werden.

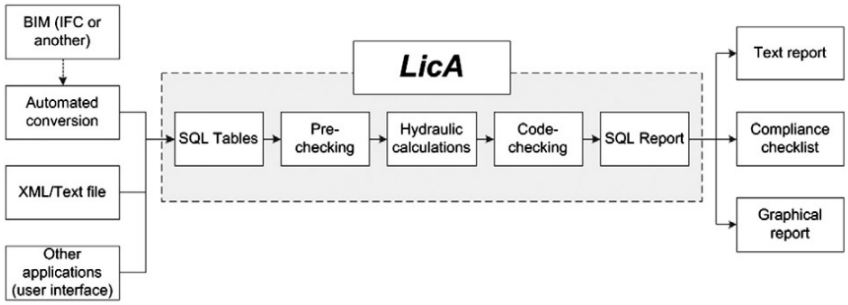


Abbildung 6.21: Aufbau des LicA-Systems (Martins und Monteiro, 2013)

Wie in dem Aufbau des Systems in Abbildung 6.21 dargestellt, wurde auf die Einführung einer eigenen Regel(-programmier)sprache verzichtet. Die Hauptkomponenten von *LicA* sind in einer SQL Server-Datenbank enthalten. Diese Datenbank verfügt über eine Reihe von Tabellen mit allen Objekten, die zur Darstellung des Netzwerks und der Berechnungen benötigt werden. Dazu gehören sowohl hydraulische Werkzeuge für die Analyse als auch die eigentlichen Prüfroutinen. Diese Methoden wurden in der Sprache *T-SQL* entwickelt. Martins und Monteiro (2013) führen an, dass die Sprache SQL gewählt wurde, da es sich hierbei um ein sehr stabiles und interoperables Format handelt. Die Ergebnisse von Berechnungen werden in Tabellen für die Nachbearbeitung gesammelt. Auf die Datenbank kann über die standardisierte Datenbankschnittstelle *Open Data Base Connectivity* zugegriffen werden, was die Integration von kundenspezifischen Anwendungen, einschließlich Webanwendungen, mit diesem System ermöglicht. Zusätzlich wurde die graphische 3D-Benutzeroberfläche *LiCAD* entwickelt, welche sowohl die Eingaben als auch die Ergebnisse anzeigt bzw. visualisiert. Mit *LiCAD* können die Ergebnisse direkt im Wassernetzmodell farbig und mit Textbeschreibungen angezeigt werden.Bisher wurde der vorgestellte Ansatz lediglich an dem Beispiel der portugiesischen Wassernetze angewendet; eine weitergehende Validierung steht noch aus (Martins und Monteiro, 2013).

6.11 BCA Singapore (2014): CORENET Plattform

Eine der bekanntesten praktischen Anwendungen im Bereich des ACCC wurde 1995 von der Building Construction Authority (BCA) in Singapur mit der CORENET-Plattform eingeführt (BCA Singapore, 2006, 2014). Die Construction and Real Estate Network (CORENET) Plattform stellt einen der ersten Vertreter einer nationalen, digitalen Plattform für die Einreichung von Bauplanungsdokumenten (engl. *e-Submission*) dar. Grundlegendes Ziel dieser Plattform ist es, alle Planungsinformationen eines Bauprojekts zentral zu sammeln, um die Prozesse für die Genehmigung der Planung mit Hilfe digitaler Methoden und Werkzeuge zu optimieren. Hierzu wurden mehrere Anwendungen als Bestandteile dieser Plattform entwickelt, welche den Regierungsbehörden für die Verwaltung bzw. Genehmigung sowie für die ausführenden Baufirmen für das Einreichen und Prüfen der Planungsdokumente zur Verfügung stehen.

Als ein erstes wesentliches Werkzeug wurde die Anwendung *CORENET BP-Expert* 1995 mit dem Ziel entwickelt, die Übereinstimmung von digitalen 2D-basierten Zeichnungen mit den Vorschriften in Bezug auf Barrierefreiheit und Brandschutz zu prüfen. 1998 wurde CORENET grundlegend neu ausgerichtet und auf das IFC-Datenmodell aufgesetzt. Damit wurde nicht nur ein herstellerneutrales und softwareunabhängiges Datenformat als Basis für die Plattform gewählt, sondern gleichzeitig auch ermöglicht, mit dreidimensionalen Gebäudemodellen zu arbeiten.

2002 wurde die Prüfanwendung *CORENET BP-Expert* schließlich unter ihrem auch heute noch aktuellen Namen *e-Plan Check* veröffentlicht. Die Anwendung besitzt unter anderem ACCC-Funktion für die Prüfung wesentlicher Teile der in Singapur geltenden Bauvorschriften für die Gebäudesteuerung, Barrierefreiheit, Brandschutz sowie Umweltmedizin (Dimyadi und Amor, 2013).

Aus technischer Sicht basieren die Prüfprozesse von CORENET auf fest implementierten (engl. *hard-coded*) Programmroutinen. Somit sind die Algorithmen, Prozessschritte und Methoden für den Anwender nicht einsehbar. Der Prüfprozess gliedert sich in drei grundlegende Phasen: In einem ersten Schritt werden die Modellinformationen hinsichtlich ihrer Verfügbarkeit geprüft. Falls Informationen nicht in

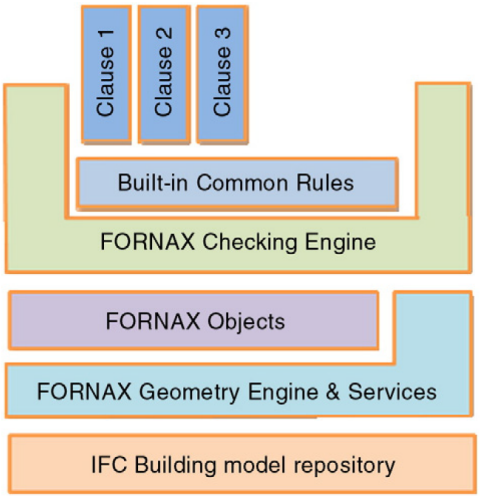


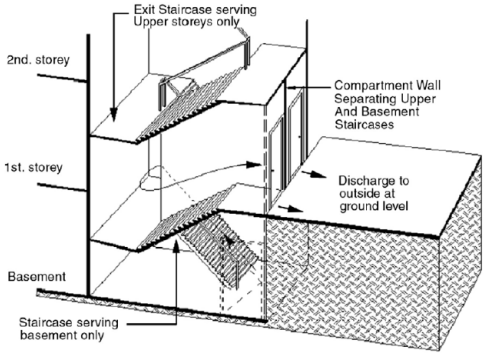
Abbildung 6.22: Aufbau der FORNAX-Biliothek (Eastman et al., 2009)

der erforderlichen Form vorliegen, wird in einem zweiten Schritt das Modell nach fehlenden Informationen auf den darunter liegenden Informationsschichten (z. B. eingebundene Bibliotheken oder externe Datenbanken) durchsucht (Eastman et al., 2009).

Um eine solche Aufbereitung des Datenmodells zu ermöglichen, entwickelte die Firma *novaCITYNETS* eine Bibliothek von fest implementierten Methoden, genannt *FORNAX*, mit der Programmiersprache *C++*. Die entwickelten Methoden enthalten eine Vielzahl von Routinen, welche die Informationen semantischer Objekte in gewünschter Form aus dem IFC-Datenschema extrahieren. Darüber hinaus sind in diesen Methoden nicht nur die Datenaufbereitung definiert, sondern auch Konformitätsprüfungen hinsichtlich der Bauvorschriften. Diese definierten Prüfprozesse sind direkt im Datenmodell von CORENET gespeichert (Eastman et al., 2009).

Die *FORNAX*-Bibliothek selbst hat eine mehrschichtige Architektur, welche eine Reihe von APIs zur Verfügung stellt, die eine Anpassung und Entwicklung neuer Regeln ermöglicht (Solihin et al., 2016). Diese APIs basieren auf dem *FORNAX*-Objektmodell, welches die Kapselung von Daten und Methoden in einem Objekt ermöglicht, was wiederum einen sauberen Zugriff auf viele der bereits implementierten Features und Geometriefunktionen erlaubt. Der Aufbau der *FORNAX* Bibliothek ist in Abbildung 6.22 dargestellt. Ein wesentlicher Bestandteil von *FORNAX* ist die Geometrie-Bibliothek, welche die entsprechenden räumlichen Operatoren integriert. Mit diesen Operatoren können die Objekte des IFC-Schemas hinsichtlich ihrer räumlichen Relationen effektiv analysiert werden, was für viele Anwendungsbereiche des ACCC hochrelevant ist.

In Abbildung 6.23 ist an dem Beispiel eines Treppenhausschachts dargestellt, wie ein *FORNAX*-Objekt für eine abgeleitete Entität definiert ist. Der Treppenschacht ist nicht nur eine Ansammlung von Räumen in einem vertikalen Stapel, sondern wird auch selbst als eigenes Objekt definiert, welches sowohl Attribute als auch eigene Methoden besitzt. Ausgehend von diesem Objekt können nun beispielsweise alle berührenden Randobjekte evaluiert werden, was unter anderem für eine Brandschutzprüfung relevant ist.



ExistStaircaseShaft	
Methods	Description
IsPressurized	Check if this exit staircase shaft requires pressurization
GetFireRating	Get the fire-rating of this exit staircase shaft.
GetDischargeLevels	Fetch all discharge levels from this shaft. Calculate the nearest discharge level through this shaft for a given storey.
MinStaircaseWidth	Calculate the minimum width of the staircase in this shaft between any two storeys which this shaft serves.
SwingDirection	Calculate the swing direction relative to this shaft for a given door.

Abbildung 6.23: Dokumentation einer festimplementierten *FORNAX*-Regel zur Brandschutzprüfung eines Treppenhauses (Eastman et al., 2009)

Die Entwicklung von CORENET und *FORNAX* stellt einen der frühesten, aber auch heute noch fortschrittlichsten Ansätze zur Automatisierung von ACCC dar. Ein wesentlicher Grund hierfür ist nicht zuletzt, dass die Plattform mit einer starken praktischen Relevanz entwickelt wurde.

CORENET konnte bisher bereits eine deutliche Beschleunigung der Prüfverfahren bewirken. So dauerte ein Prüfverfahren im Jahr 2007 durchschnittlich 103 Tage, im Jahr 2009 38 Tage und im Jahr 2014 lediglich noch 26 Tage (Doing Business, 2015; Fiedler, 2015). Die CORENET-Plattform wird von der Behörde selbst und ca. 2500 Unternehmen im AEC-Sektor genutzt (Eastman et al., 2009).

Der Aufwand für die Entwicklung der automatisierten Regelprüfung wurde durch den Einsatz der *FORNAX*-Bibliothek erheblich reduziert. Als Ergebnis wurden beispielsweise 281 Regeln der Richtlinie *Integraler Gebäudeservice (IBS)*, einer singapurischen Richtlinie im Bereich der Gebäudetechnik, mit Hilfe von *FORNAX* implementiert. Laut Eastman et al. (2009) deckte CORENET im Jahr 2008 die singapurischen Richtlinie *Integrale Bauplanung (IBP)* zu 92 % und *Integraler Gebäudeservice (IBS)* zu 77 % ab.

Die *FORNAX*-Objekte und -methoden werden auch als Grundlage für weitere Ansätze verwendet, wie beispielsweise bei (Rong Xu et al., 2004).

FORNAX ist grundsätzlich für den Umgang mit komplexen Regeln, d.h. Bauvorschriften, geeignet, die Anwendung im Einzelfall erfordert jedoch noch immer einen hohen Zeit- und Arbeitsaufwand sowie fundierte Programmierkenntnisse und Zugriffsrechte. Die Nutzung als Plattform reduziert den Zeit- und Arbeitsaufwand im Vergleich zur Neuentwicklung erheblich, aber sie verhindert auch, dass es für einfachere Regeln verwendet werden kann im Gegensatz zu dem SMC, mit welchen relativ einfach individuelle Regelwerke für die Qualitätsprüfung erstellt werden können. Weiterhin birgt *FORNAX* auch wesentliche Unzulänglichkeiten, da es extrem von der angenommenen guten Qualität der zugrunde liegenden Daten abhängig ist (Solihin et al., 2016). Auch wenn mit festimplementierten Abbildungsmethoden falsche Informationen aufgrund der schlechten Qualität eines Gebäudemodells aufgefangen werden können, ist dieses keine dauerhafte Lösung für das eigentliche Problem. Eine oder aber mehrere spezifische MVDs könnten an dieser Stelle Abhilfe schaffen, jedoch wäre dieser einmalige Entwicklungsaufwand ebenfalls hoch.

Die diversen Bemühungen der BCA rund um die CORENET Plattform werden in (Han, 1995; Khemlani, 2005, 2017; Solihin, 2004; Tien Foo Sing Qi Zhong, 2001) im Detail dargestellt.

6.12 Solibri Model Checker

Der Solibri Model Checker (SMC) ist eine Softwareanwendung für die Prüfung der Qualität digitaler Bauwerksmodellen und wurde in einer ersten Version im Jahr 2000 von der finnischen Firma Solibri Inc. (2017) auf den Markt gebracht. Die Prüfung der Modelle bezieht sich sowohl auf grundlegende datentechnische und inhaltliche Kriterien als auch auf spezifische Anforderungen, welche sich üblicherweise in diversen internationalen und nationalen Richtlinien finden. Dabei konzentriert sich die Prüfung mit dem SMC auf Modelle im IFC-Format. Das interne Datenmodell des SMC ist stark an den IFC-Standard angelehnt.

Trotz der Standardisierung exportieren die diversen BIM-Autorenwerkzeuge IFC-Daten mit leichten Eigenheiten und typischen Merkmalen. Daher basiert der Import des SMC mit Einschränkungen auf fest-programmierten Routinen, welche die Inhalte eines IFC-Modells anhand des angegebenen Autorenwerkzeugs passend aufbereiten. Aus diesem Grund findet der SMC auch als Visualisierungswerkzeug für den IFC-Standard weit verbreitete Anwendung (siehe auch 6.24).

Um die Daten verschiedener Disziplin-Modelle und Autorenwerkzeuge zu harmonisieren und zusammenzuführen, verwendet die SMC Klassifikationen. Die in Abschnitt 2.5.4 vorstellten Klassifikationen können im SMC verwendet und Modelle anhand dieser visualisiert werden. Allerdings beschränken sich diese Klassifikationen nicht ausschließlich auf Standards, sondern es können beliebige Klassfikationssysteme individuell oder als interner projektspezifischer Standard eingeführt und flexibel angepasst werden. Mit den Klassifikationen können Informationen aus verschiedenen Gebäudemodellen nicht nur gefiltert, sondern auch für den anschließenden Prüfprozess aufbereitet werden.

Kernstück des SMC ist der Ruleset Manager (RSM), welcher eine Basisbibliothek mit 50 einzelnen Regelvorlagen (Stand: 14. Juni 2020) beinhaltet. Eine vollständige Liste der aktuell vorhandenen Regelvorlagen ist im Anhang in Tabelle A.3 zu finden.

Eine Regelvorlage stellt die fest programmierte Standardprüfprozedur dar, die durch eine definierte Anzahl vorgegebener Parameter angepasst werden kann. Die Regelvorlagen reichen von einfachen Attribut- und Datentyp-Prüfungen bis hin zu Prüfungen von Freiräumen für die Barrierefreiheit oder die komplexe Berechnung von Entfluchtungsrouten. Die Regelvorlagen können vom Anwender nach

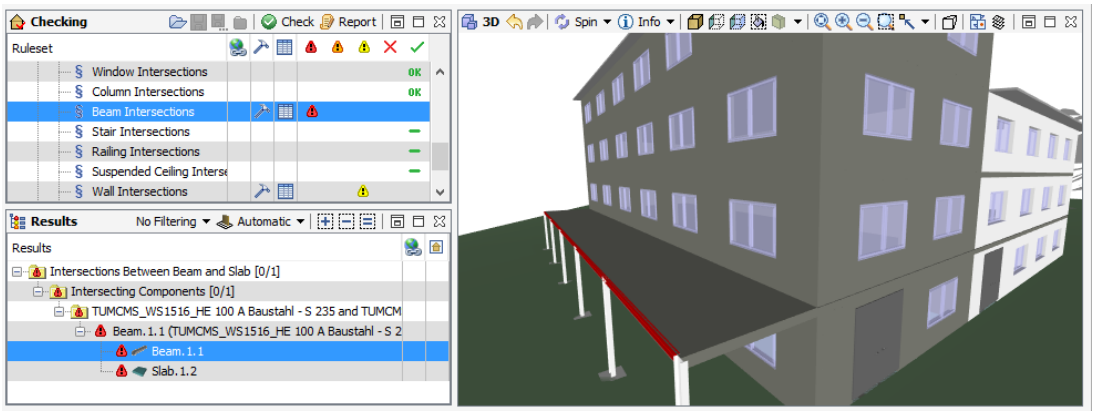


Abbildung 6.24: Nutzeroberfläche des SMC (Solibri Inc., 2017)

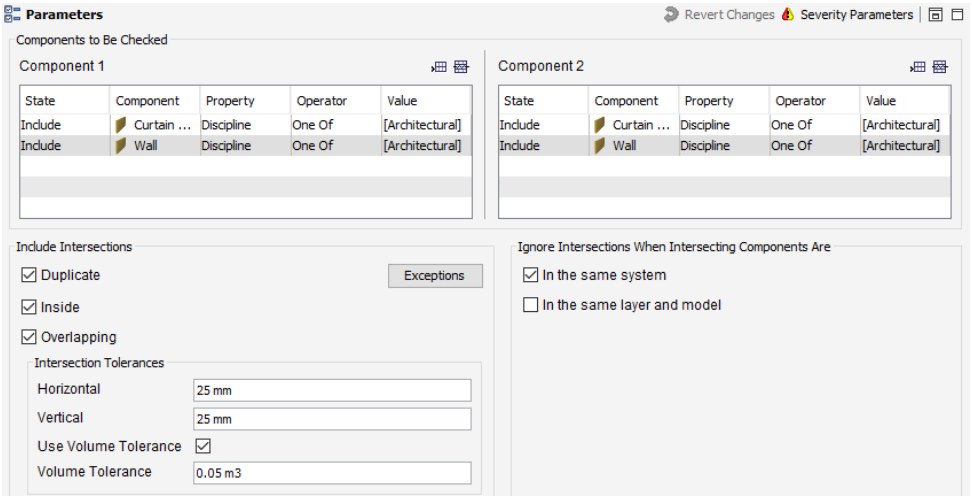


Abbildung 6.25: Nutzeroberfläche zum Anpassen einer Regelvorlage im SMC (Solibri Inc., 2017)

seinen individuellen Anforderungen zusammengestellt und angepasst werden. Eine individuelle Zusammenstellung wird als Regelsatz bezeichnet und kann auch konditionale Verzweigungen beinhalten. So können beispielsweise Regelvorlagen nur auf IFC-Komponenten angewendet werden, welche zuvor eine andere Regelvorlage bestanden oder aber nicht bestanden haben. Um die definierten Prüfprozesse auszutauschen und gemeinsam zu nutzen, kann der Regelsatz lokal gespeichert werden. Als Beispiel ist in Abbildung 6.25 die Schnittstelle zur Anpassung einer Regel zur Überprüfung von Schnittpunkten zwischen definierten architektonischen Bauteilen dargestellt.

Die Komposition der Standard-Regeln im *Ruleset Manager* ermöglicht dem Nutzer einen gewissen Grad an Transparenz für den Prüfprozess und dessen Ablauf. Darüber hinaus wird durch die Kommunikation der festgestellten Probleme, welche mit spezifischen Erläuterungen beschrieben sind, welcher Fehler vorliegt, dem Nutzer erläutert, welches das Endergebnis einer Prüfung ist. Daher kann in diesem Falle nicht von einem vollständigen *Black-Box*-Ansatz gesprochen werden. Die einzelnen Regeln und deren zugrunde liegende Algorithmen sind bis auf Angabe der Parameter jedoch nicht einsehbar und stellen daher durchaus *Black-Box*-Prozesse dar. Je nach Komplexität, Anwendungsbereich und Umfang einer Regelvorlage variiert die Einsehbarkeit und das Verständnis für die Prozesse allerdings sehr stark.

Da die Zusammenstellung solcher Regeln jedoch nicht nur ein hohes Maß an Wissen über die Regel selbst, sondern auch über die Datenmodellierung und die IFC-Struktur erfordert, wird diese Funktion meist von Fachanwendern und nicht von den Planern genutzt. Die meisten Benutzeroberflächen des SMC basieren auf einem zeilenweisen Vorgehen und der Zusammenstellung von Anforderungen und Anpassungen, was für Laien sehr kompliziert sein kann. So verwenden derzeit die meisten Anwender die vordefinierten Musterregeln, die von Solibri zur Verfügung gestellt werden. Diese konzentrieren sich zumeist auf grundlegende architektonische Prüfungen wie beispielsweise die Vollständigkeit oder Kollisionen von Bauteilen. Individuellere und spezifischere Regelsätze wie *COBie Compliance*, *ADA/ABA Accessibility* oder *Fire Escape Routing Checks* sind als kostenpflichtige Erweiterungen verfügbar und können über das *Solibri Solution Center*, ein Online-Portal, erworben werden.

Der SMC versteht sich als Werkzeug für die Qualitätsprüfung von Gebäudemodellen, und daher ist eine Modifikation und gar das Schreiben von IFC-Inhalten nicht möglich. Als Ausgabedaten werden im SMC ausschließlich die Ergebnisse einer Prüfung ausgegeben. Entsprechende Berichte können beispielsweise als *PDF*- oder BCF-Reports exportiert werden.

Für ausgewählte Projekte und Anwendungsfälle stellt Solibri eine API für die Entwicklung von Regelvorlagen zur Verfügung. Diese API war allerdings bisher nicht öffentlich zugänglich und nur für Forschungszwecke im Gebrauch. Aktuell arbeitet das Unternehmen daran, diese Schnittstelle als offiziellen Teil des Produkts zu veröffentlichen. Gleichwohl wurden auf Basis dieser API bereits diverse Erweiterungen für Forschungs- und Entwicklungsprojekte umgesetzt, welche im Folgenden vorgestellt werden sollen:

6.12.1 Design Assessment Tool

Anfang 2003 wurde in den USA von der Regierungsbehörde US General Services Administration (GSA), welche für das US-amerikanische Wirtschaftsministerium den größten Anteil der öffentlichen Gewerbeflächen verwaltet, im Rahmen eines staatlichen Programms zur Förderung von BIM ein Werkzeug zur Konformitätsüberprüfung der Gestaltungsplanung von öffentlichen Gebäuden entwickelt (Eastman, 2009; Eastman et al., 2009). Ziel dieser Forschungsarbeit, welche in enger Zusammenarbeit mit dem *Georgia Institute of Technology* durchgeführt wurde, war es, bereits in sehr frühen Phasen der Gestaltungsplanung die architektonischen Entwürfe auf deren Eignung hinsichtlich der geltenden Gestaltungsrichtlinien automatisiert zu prüfen und so die Planer zu unterstützen.

Die in dem Projekt betrachteten Gestaltungsvorschriften, die *US Courts Design Guide*, werden von dem *Administrative Office of the US* herausgegeben und beinhalten eine Vielzahl von Raum-, Umwelt-, Sicherheit- und Gebäudetechnikanforderungen für die gestalterische Planung von Justizgebäuden, welche aufgrund des US-amerikanischen Rechtssystems besonderen Auflagen unterliegen. Neben diesen Anforderungen fordern die Richtlinien der GSA von den Planern die Vorlage von mindestens drei unterschiedlichen Raumkonzepten, unter denen die Behörde einen Favoriten auswählen kann. Dieses führt dazu, dass sowohl aufseiten der Planer, als auch auf der Seite der Behörde ein hoher Aufwand dafür entsteht, die verschiedenen Konzepte im Hinblick auf Einhaltung der Vorschriften zu überprüfen.

Speziell für die Überprüfung dieser Richtlinien wurde das *Design Assessment Tool* als Erweiterung für den SMC entwickelt. Das Werkzeug konzentriert sich dabei maßgeblich auf die Einhaltung der räumlichen Vorschriften, erlaubt gleichzeitig aber auch eine vereinfachte Energie- und Kostenanalyse.

Für die Durchführung der jeweiligen Konformitätsprüfung wurden Minimalanforderungen an den Informationsgehalt eines Gebäudemodells gestellt. Eine entsprechende Auflistung dieser Anforderungen für die jeweiligen Überprüfungsarten wurde von der GSA und dem *Georgia Institute of Technology* in dem begleitenden Dokument, dem *GSA Preliminary Concept Design BIM Guide* (US General Services Administration, 2008), zusammengefasst. Damit die Ergebnisse der Überprüfung aussagekräftig bleiben, wird in einem ersten Schritt die Einhaltung von inhaltlichen Minimalanforderungen überprüft.

Da eine Vielzahl der Richtlinien räumlich und raumbasiert formuliert ist, ist der Ausgangspunkt dieses Ansatzes ein Raummodell, welches über die Bezeichnungen der einzelnen Räume eine jeweilige Funktion und Rolle im Gebäude zuordnet. Da diese Rolle jedoch je nach Überprüfungsart wechseln kann, muss das Raummodell jeweils auf die Anforderungen der spezifischen Kontrolle zugeschnitten werden. Hierzu

dient eine festgelegte Zuordnung, ein sogenanntes *Mapping*, welches die Räume je nach Anforderungen kategorisiert.

Mittels dieser Zuordnung der Räume kann das Gebäudemodell auf die einzelnen Anforderungen hin überprüft werden. Eine zentrale Regelung der *US Courts Design Guide* ist das Sicherheitskonzept für Justizgebäude, welches sich insbesondere im Raumprogramm widerspiegelt. Aufgrund des US-amerikanischen Rechtssystems unterliegt das Gebäude hinsichtlich Aufteilung und Zugänglichkeit der Räume besonderen Anforderungen, die in jedem einzelnen Gestaltungskonzept eingehalten werden muss. Ein Beispiel für eine solche Anforderung ist die Regelung, dass das Büro des Staatsanwalts mit dem Tagungsraum der Jury über eine Sicherheitszone zu erreichen sein muss.

Bei der Entwicklung des *Design Assessment Tool* wurden von dem Entwicklungsteam 216 solcher Regelungen in der *US Courts Design Guide* identifiziert und als feste Routinen innerhalb des SMC implementiert. Da viele der Vorschriften die Zugänglich- und Erreichbarkeit von Räumen untereinander betreffen, wurde ein relationales Modell entwickelt, in welchem die im Gebäudemodell gegebenen Verhältnisse mit Hilfe eines Graphen abgebildet werden. Mittels eines solchen Graphen können sowohl topologische als auch metrische Zusammenhänge im Raummodell dargestellt und so nicht nur die Zugänglichkeiten bestimmt, sondern auch Distanzen berechnet werden (Eastman, 2009).

Das im Rahmen dieser Forschungsarbeit entwickelte Plug-In für den SMC stellt eine gute Lösung dar, den aufwendigen Überprüfungsprozess hinsichtlich der spezifischen Anforderungen der GSA zu erleichtern. Das Werkzeug wurde bis heute bei mehreren Bauprojekten von Justizgebäuden erfolgreich eingesetzt und automatisiert die Konformitätsüberprüfung zu ca. 90%. Aber auch hier ergibt sich das Problem, dass der eigentliche Prozess aufgrund der geschlossenen Struktur des SMC für den Anwender nur teilweise sichtbar ist (Eastman et al., 2009).

6.12.2 HITOS

In einem anderen Forschungsprojekt wurde der SMC von der norwegischen Baubehörde *Statsbygg* als zentrales Prüfwerkzeug verwendet. Die *Statsbygg*, die für die Immobilienverwaltung des norwegischen Staates zuständig und damit der größte Kunde im norwegischen Bauwesen ist, rief 2005 das Projekt *HITOS* (lokales Akronym für die Universität Tromsø) in Zusammenarbeit mit der norwegischen Universität Tromsø ins Leben (Lê et al., 2006). Das *HITOS* ist ein Pilotprojekt, welches zum Ziel hat, die Forschung und Entwicklung einer digitalen Plattform sowie auch zeitgemäßer Planungsinstrumente für das Bauwesen und deren Interaktion untereinander voranzutreiben. Als Grundlage für das Projekt dient der Neubau der Gebäude der Fakultät für Ingenieur- und Erziehungswissenschaften der Universität Tromsø mit einer Investition von ca. 18 Mio. Euro und einer Nettogeschossfläche von ca. 5150 m^2 (Lê et al., 2006; Statsbygg, 2006).

In dem Projekt fungiert die Plattform als universelles Instrument für digitale Planung und dient als zentrales Speichermedium von IFC-Modellen für alle Projektbeteiligten mit sämtlichen Informationen über sämtliche Lebenszyklen des Bauwerkes hinweg. Zu den wesentlichen Anforderungen an die Plattform zählen die ACCC-Werkzeuge, welche sich insbesondere auf räumliche Analysen konzentrieren. Hierzu wurden im Rahmen des Projektes unter anderem die bereits vorgestellten Softwarelösungen *CORENET e-PlanCheck*, der *SMC* sowie der *EDM Model Checker* eingesetzt, getestet und im Vergleich zu herkömmlichen Planungsmethoden abschließend bewertet.

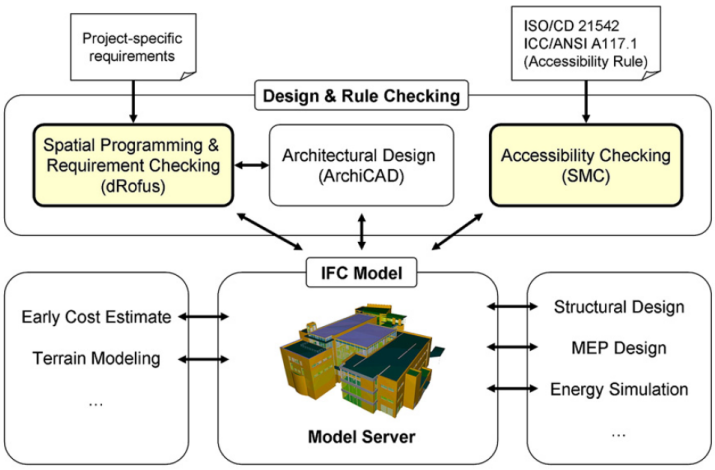


Abbildung 6.26: Aufbau des Hitos-Projekts (Eastman et al., 2009)

Die in Abbildung 6.26 dargestellte räumliche Struktur und Aufteilung des Gebäudemodells wurde mit Hilfe des Softwaretools *dRofus* erstellt und eine Prüfung der Barrierefreiheit mit Hilfe des *SMC* durchgeführt (Eastman et al., 2009).

Auf der HITOS-Plattform dient das Management- und Planungs-Tool *dRofus* dazu, Anforderungen an einzelne Räume hinsichtlich ihrer Lage und technischen Ausstattung für ein gesamtes Raumprogramm eines Bauwerkes zu formulieren und deren Einhaltung anschließend anhand des zentralen IFC-Datenmodells zu prüfen. Die Anforderungen müssen in diesem Falle nicht an ein spezifisches Regelwerk gebunden sein, sondern können je nach individuellem Anspruch an das Bauwerk definiert werden. Hierfür werden in einem Editor Vorgaben für unterschiedliche Raumklassen deklariert. Diese können den einzelnen Räumen eines Gebäudemodells zugewiesen werden. Über ein webbasiertes Interface können parallel mehrere Bauprojekte verwaltet werden. Daher eignet sich das Tool auch für den Einsatz bei mehrteiligen Bauprojekten, wie dem Pilotprojekt *HITOS* selbst. *dRofus* greift lediglich auf eine fest definierte Teilmenge des jeweiligen zentralen Datenmodells zurück, welche die geplante Struktur des Raummodells und die Beschaffenheit einzelner Räume beschreibt und diese schließlich mit einer hierarchischen Datenstruktur, die sich aus den Anforderungen und den zugewiesenen *Raum-Templates* des Anwenders ergibt, vergleicht (Eastman et al., 2009).

Weiterer wesentlicher Bestandteil der *HITOS*-Plattform ist die Prüfung des Gebäudemodells hinsichtlich der Barrierefreiheit. Hierzu wurden Teile der internationalen Normen *ISO 21542:2011 - Gebäude - Barrierefreiheit von Gebäuden und sonstigen Bauwerken* (ISO, 2011-12) und *ICC/ANSI A117.1 - Accessible and Usable Buildings and Facilites* (International Code Council, 2017a) im SMC mit Hilfe des *Ruleset Manager* sowie unter zu Hilfenahme der API als sogenanntes *Accessibility Ruleset* implementiert. Wie bereits beschrieben, können die einzelnen Regeln über ausgewählte Parameter (z.B. Dimensionen der erforderlichen Freiflächen) von dem Nutzer angepasst werden (Eastman et al., 2009).

Zur Überprüfung der Barrierefreiheit und Zugänglichkeit unterstützt der SMC verschiedene Funktionen, bei denen nicht nur die Geometriedaten eines einzelnen Bauobjekts, sondern auch andere zugehörige Objekte und deren Eigenschaften, wie z.B. Oberflächenmaterial, berücksichtigt werden können. Die Regeln der Barrierefreiheit beziehen sich im Wesentlichen auf Gebäudeobjekte wie Räume, Türen, Treppen,

Rampen und Fenster. Die meisten Regeln befassen sich mit der Beziehung zwischen diesen Objekten und daher arbeiten diese mit den Beziehungen zwischen den Teilobjekten, aus welchen sie zusammengesetzt sind. Beispielsweise spielen für die Überprüfung der lichten Weite von Fluren nicht nur Kriterien wie die Flurbreite als Eingabeparameter, sondern auch für andere zugehörige Objekte wie Säulen, Türen, Waschbecken usw. eine wesentliche Rolle.

6.12.3 AutoCodes Project

2011 wurde von der ICC das *Autocodes Project* in Zusammenarbeit mit den Softwareherstellern *Solibri* und *Fiatech* gestartet. Ziel des Projekts ist die Entwicklung von *Autocodes*, einem prototypischen System auf Basis des SMC, das eine integrierte Konformitätsprüfung hinsichtlich ausgewählter US-Gebäudevorschriften ermöglicht (Fiatech, 2011; Nawari, 2018). Dabei fokussieren sich diese Entwicklungen auf Richtlinien zur Barrierefreiheit und Zugänglichkeit von Gebäuden. Hintergrund sind die Bemühungen der ICC, alle manuellen, papierabhängigen in automatisierte Prozesse der Entwurfsprüfung zu überführen.

In einer ersten Phase des Projektes wurde zunächst die Inkonsistenzen zwischen den Konformitätsprüfungen verschiedener zuständiger US-Behörden festgestellt und dokumentiert. Gleichzeitig wurde die grundlegende Machbarkeit einer Prüfung für Zu- und Ausgänge von digitalen Gebäudemodellen erfolgreich nachgewiesen. In der zweiten Phase wurde von der Standards Development Organization (SDO) der Rahmen für die Validierung des digitalen Überprüfungsprozesses abgesteckt und Richtlinien für ein Modell, die *Minimum Modeling Matrix*, erstellt. In dieser Richtlinie ist der Arbeitsprozess für die Konformitätsprüfung festgelegt und bildet so die Basis für eine Weiterbildung für die staatlichen und lokalen Behörden, welche diesen Prozess schließlich anwenden sollen. In der letzten Phase des Projekts sollen alle Ergebnisse der vorangegangenen Phasen verfeinert und erweitert werden. Hier müssen unter anderem auch die vorhandenen oder aber auch eigens entwickelten Routinen für die Prüfung der Barrierefreiheit verbessert werden, damit diese auch für andere Richtlinien und Bereiche angewendet werden können (Nawari, 2018).

6.13 abimo Checker

Die Entwicklung des *abimo Checker* ist Teil eines von der Südkoreanischen Regierung geförderten Projekts zur Entwicklung einer offenen BIM-basierten Plattform für eine internationale IT-Umgebung für das Bauwesen (Cho und Choi, 2014). Das Projekt umfasst neben der Entwicklung eines BIM-Modellierungtools und eines BIM-Servers auch die Entwicklung einer Prüfsoftware, welche in der Lage ist, verschiedene Anforderungen zu prüfen. Der *Abimo Checker* basiert zwar auf dem Datenmodell, welches auch in dem Modellierungtool verwendet wird, verfügt jedoch auch über eine IFC-Schnittstelle.

In dem UI der Anwendung, dem sogenannten *ICON*-Formular, können Regelsätze konfiguriert werden. Dabei können in der Umgebung Interferenzen (Überlagerung zwischen *Element A* und *Element B*), Adjazenzen (*Space A* und *Space B* sind nebeneinander), Prüfungen der Breite (Breite des *Elementes A* ist größer als *x m*) und Existenzprüfungen (*Element X* muss in *Raum A* vorhanden sein) definiert werden. Mit Hilfe dieser Elemente können grundlegende Anforderungen, wie insbesondere die koreanischen Request for proposal, übersetzt werden. Fertiggestellte Regeln können vom Benutzer über Parameter angepasst und wiederverwendet werden.

Neben der Möglichkeit zur Komposition von Regelprüfungen im UI, besitzt der *abimo Checker* eine Entwicklungsumgebung, in welcher direkt mit der Skriptsprache *Python* gearbeitet werden kann (Cho und Choi, 2014). Diese wurde unter anderem bei dem Ansatz von (Uhm et al., 2015) verwendet.

6.14 EDMmodelChecker

Parallel zu den Entwicklungen rund um CORENET in Singapur wurde im Jahr 1998 von dem norwegischen Technologieunternehmen *Jotne EPM Technology* die Kollaborationsplattform Express Data Manager (EDM) eingeführt. Aus technischer Sicht basiert diese Plattform auf einer objektorientierten Datenbank, welche mit der Datenmodellierungssprache *EXPRESS* arbeitet und darauf ausgerichtet ist, Produktdatenmodellen mehrerer unterschiedlicher Ingenieurfachdisziplinen zu verwalten. Der Aufbau der Plattform ist in Abbildung 6.27 dargestellt.

Da *EXPRESS* ein weit verbreiteter Standard für die digitale Definition von Produktdatenmodellen im Ingenieurwesen ist (siehe auch Abschnitt 2.4.2), besitzt der EDM eine hohe Kompatibilität mit einer Vielzahl verschiedener Datenmodelle. Über ein integriertes Modul zur Konvertierung von Informationen, dem sogenannten *EDMmodelConverter*, können gespeicherte Datenmodelle mit einer verhältnismäßig geringen Fehleranfälligkeit auf jeweils andere Formate abgebildet werden. Da das IFC-Format ebenfalls als EXPRESS-Schema modelliert ist, ist das EDM-Datenbanksystem insbesondere mit diesem Format kompatibel (Ding et al., 2004, 2006).

Neben der zentralen Datenbank der EDM-Plattform selbst wird eine Vielzahl weiterer Module zur Verarbeitung der Informationen angeboten. Für Konformitätsprüfungen dient das Modul *EDMmodelChecker*, mit welchem Vorschriften basierend auf EXPRESS formuliert und anschließend auf die Informationen eines gespeicherten Gebäudemodells angewendet werden können (Ding et al., 2004; Jotne EPM Technology, 2004). Für externe Entwickler stellt eine Integration und Entwicklungsmöglichkeit mit EXPRESS als Regelsprache ein wichtiges Entscheidungskriterium dar, da sie Offenheit und Flexibilität bei der Formulierung der Regeln bietet. Auf Basis des EDM wurden bereits diverse Forschungsansätze im ACCC-Bereich durchgeführt.

In Australien wurde von dem *Cooperative Research Center for Construction Innovation* das Softwaretool *DesignCheck* auf Basis von EDM entwickelt. Zu Beginn der Entwicklungen wurden zwar zunächst mehrere Plattformen als Basis in Betracht gezogen, doch aufgrund der Offenheit und Flexibilität der bereits integrierten Datenbankverwaltung entschied man sich letztlich für die EDM-Umgebung (Eastman et al., 2009).

Ähnlich wie bereits bei dem *EDMmodelChecker* können in *DesignCheck* einzelne Regeln oder Regelwerke mit EXPRESS direkt als Objekte in das EDM-Datenmodell geschrieben werden. Für die Übersetzung eines Regelwerks wird von den Entwicklern empfohlen, Inhalt und Struktur des gewünschten Objektes zunächst in Pseudocode zu definieren und erst anschließend per Hand oder automatisiert in ein Regelobjekt zu transformieren (Ding et al., 2004). In Abbildung 6.28 ist dieser Übersetzungsprozess, in welchem aus dem Pseudocode letztlich echter Quellcode in der *EDM*-eigenen Sprache übersetzt wird, schematisch dargestellt.

Um ein Regelobjekt in *DesignCheck* zu definieren, müssen zunächst die relevanten Bauteilobjekte sowie deren zugehörige Parameter deklariert werden. Weiterhin müssen auch die Beziehungen (z. B. räumlich), welche die Objekte miteinander ins Verhältnis setzen, definiert werden. Abschließend können

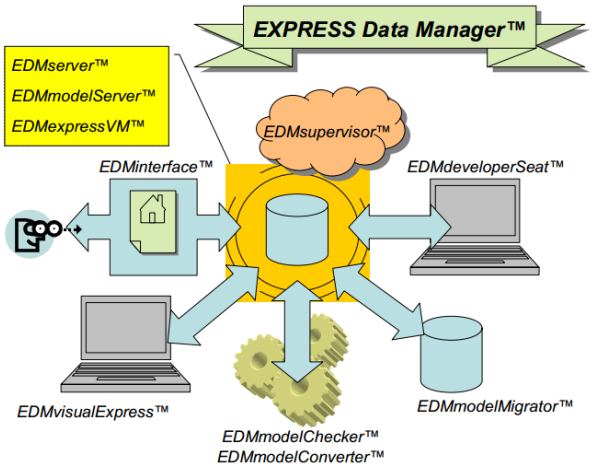


Abbildung 6.27: Aufbau der Plattform EDM (Jotne EPM Technology, 2004)

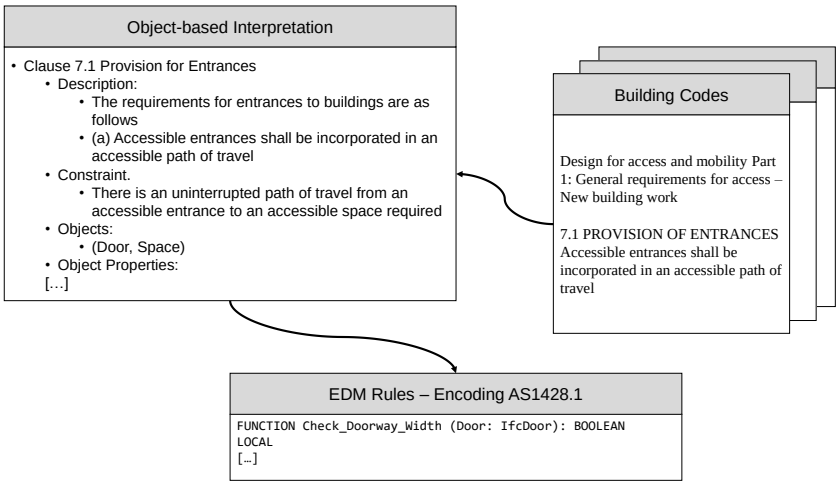
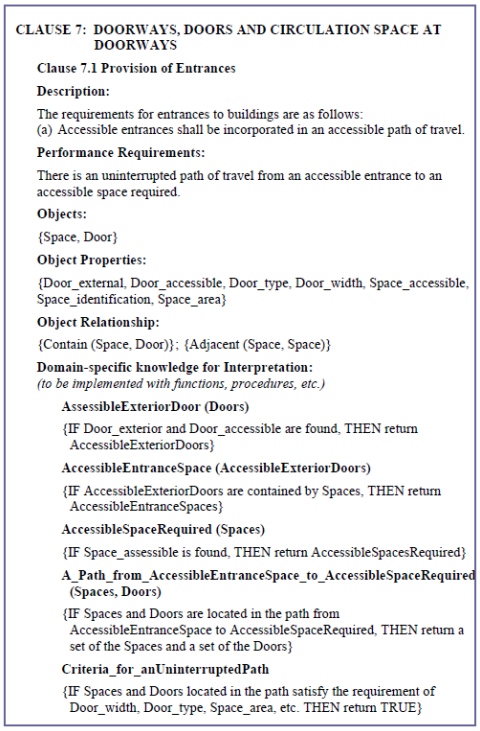


Abbildung 6.28: Darstellung des Übersetzungsprozesses einer Klausel in eine objektbasierte Interpretation und anschließend in die EDM-Regel (Ding et al., 2006)



CLAUSE 7: DOORWAYS, DOORS AND CIRCULATION SPACE AT DOORWAYS

Clause 7.1 Provision of Entrances

Description:

The requirements for entrances to buildings are as follows:
(a) Accessible entrances shall be incorporated in an accessible path of travel.

Performance Requirements:

There is an uninterrupted path of travel from an accessible entrance to an accessible space required.

Objects:

{Space, Door}

Object Properties:

{Door_external, Door_accessible, Door_type, Door_width, Space_accessible, Space_identification, Space_area}

Object Relationship:

{Contain (Space, Door)}; {Adjacent (Space, Space)}

Domain-specific knowledge for Interpretation:
(to be implemented with functions, procedures, etc.)

AssessibleExteriorDoor (Doors)

{IF Door_exterior and Door_accessible are found, THEN return AccessibleExteriorDoors}

AccessibleEntranceSpace (AccessibleExteriorDoors)

{IF AccessibleExteriorDoors are contained by Spaces, THEN return AccessibleEntranceSpaces}

AccessibleSpaceRequired (Spaces)

{IF Space_assessible is found, THEN return AccessibleSpacesRequired}

A_Path_from_AccessibleEntranceSpace_to_AccessibleSpaceRequired (Spaces, Doors)

{IF Spaces and Doors are located in the path from AccessibleEntranceSpace to AccessibleSpaceRequired, THEN return a set of the Spaces and a set of the Doors}

Criteria_for_anUninterruptedPath

{IF Spaces and Doors located in the path satisfy the requirement of Door_width, Door_type, Space_area, etc. THEN return TRUE}

Abbildung 6.29: Pseudocode-Darstellung einer Regel für die Implementierung im EDM (Eastman et al., 2009)

auf Basis dieser Deklarationen die eigentlichen Anweisungen implementiert werden, die sich auch aus mehreren Teilen zusammensetzen können. Jede dieser einzelnen Teilfunktionen beschreibt mit Hilfe einer Implikation und den deklarierten Objekten sowie Parametern die Anweisungen, welche den Überprüfungsprozess beschreiben.

Zur Veranschaulichung ist in Abbildung 6.29 ein Beispiel einer Regel in Pseudocode dargestellt. Hier werden zunächst die Datenobjekte *Raum* und *Tür* sowie deren zugehörige Parameter deklariert. Mit den Relationen wird angegeben, dass die beiden Objekte in einer bestimmten räumlichen Abhängigkeit stehen, da ein *Raum* eine *Tür* enthält und die *Räume* untereinander jeweils benachbart sind. Mit Hilfe dieser Angaben können schließlich mehrere Entscheidungsfälle formuliert werden, welche die Konformitätsüberprüfung bilden (Ding et al., 2004).

Zwar ist diese Methode zur Beschreibung eines Informationssystems sehr umfassend, jedoch kann es regulatorische Anforderungen geben, welche sich nur unzureichend mit dem Pseudocode beschreiben lassen. Für solche Fälle wurde eine Darstellung von Algorithmen mit Hilfe von Graphen entwickelt, welche im Gegensatz zu dem Pseudocode in der Lage ist, diese Inhalte abzubilden. In Abbildung 6.30 ist ein solcher Graph dargestellt, welcher beispielhaft den Algorithmus zur Ermittlung einer behindertengerechten Route durch ein Gebäude gemäß der australischen Norm AS 1428.1 darstellt. Innerhalb des Graphen ist beschrieben, wie sich die behindertengerechte Zugänglichkeit eines Raumes durch seine Nachbarschaft zu einem anderen angrenzenden Raumes auswirkt (Eastman et al., 2009).

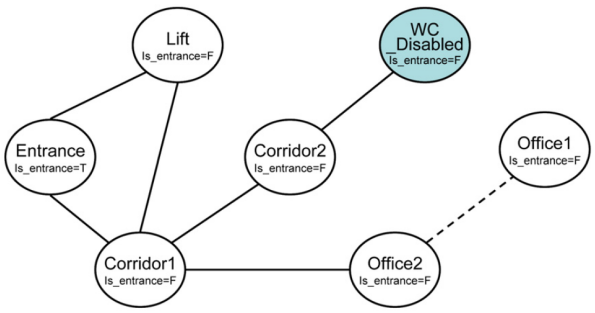


Abbildung 6.30: Graph-Darstellung einer räumlichen Regel in EDM (Eastman et al., 2009)

Die Ergebnisse einer Prüfung werden für den Anwender in Form von XML- bzw. HTML-Dokumenten aufbereitet und beinhalten detaillierte Informationen zu dem Prozess selbst sowie zu den einzelnen untersuchten Objekten. Mit einem interaktiven UI soll auch eine Plausibilitätsüberprüfung der Ergebnisse möglich sein. Eine graphische bzw. visuelle Analyse ist aufgrund der fehlenden direkten Visualisierung der Ergebnisse allerdings nicht möglich.

6.15 Zusammenfassung

Zusammenfassend kann festgehalten werden, dass bedeutende Forschungsarbeiten zur Automatisierung der Konformitätsprüfung für digitale Gebäudemodelle durchgeführt wurden. Allerdings weisen diese Forschungsansätze sowie bereits verfügbare Werkzeuge erhebliche Defizite auf:

In einigen Ansätzen sind die in den regulatorischen Dokumenten enthaltenen Regeln und Anforderungen jedoch fest in die Software integriert oder werden lediglich durch Programmcode festgehalten. Der eigentliche Prüfprozess bleibt für den Prüfingenieur, welcher über keine fundierten Programmierkenntnisse verfügt, verborgen und somit kann er produzierte Ergebnisse nicht validieren. Dieses führt letztlich zu einem gravierenden Mangel an Transparenz und Flexibilität. An dieser Stelle haben nur wenige Ansätze einen Einsatz einer maschinen- und gleichzeitig menschenlesbaren Zwischendarstellung der Anforderungen untersucht.

Eine Vielzahl der veröffentlichten Arbeiten verfolgt einen ontologiebasierten Ansatz zur Repräsentation von Regelwissen. Allerdings weisen diese Ansätze erhebliche Einschränkungen auf: Die Kodierung der Regeln in eine ontologische Repräsentation ist kompliziert und kann in dieser Form lediglich von Experten übernommen werden, welche sich mit Programmiersprachen auskennen. Allerdings verfügen Domäne-Experten, wie diese im Bauwesen üblicherweise für solche Prozesse verantwortlich sind, über keine fundierten Programmierkenntnisse. Überdies zeigen die resultierenden Systeme erhebliche Einschränkungen auf, wenn es darum geht, semantisch höhere Konstrukte und Anforderungen zu beschreiben. Die vorgestellten Beispiele sind in großen Teilen sehr einfach gehalten. Eine Betrachtung von Regelwissen und Anforderungen, wie diese in praxisnahen Szenarien üblich wären, steht aus. So können insbesondere abstrakte geometrische und topologische Randbedingungen weder dargestellt noch überprüft werden, obwohl diese einen essenziellen Bestandteil vieler Bauvorschriften ausmachen.

Andere Ansätze fokussieren sich sehr stark auf einzelne Aspekte der Konformitätsprüfung, vor allem aber auf eine Lösungsfindung für die Formalisierung des Regelwissens. Allerdings ist die Automatisierung der Konformitätsprüfung ein komplexes Problem, welches ganzheitlich angegangen werden muss und nicht ausschließlich aus einem einzelnen Prozessschritt besteht. Die Darstellung des Regelwissens selbst reicht allerdings nicht aus, auch die Aufbereitung der Informationen aus den Bauwerksmodellen muss betrachtet werden. Beispielsweise stellt die Extraktion von höherwertigen geometrisch-räumlich oder semantischen Informationen eine zentrale Herausforderung dar, welche direkt mit der Repräsentation des eigentlichen Regelwissens verknüpft ist. Somit erweisen sich diese Ansätze als zu unflexibel oder bieten zu wenig aussagekräftige Methoden in Bezug auf die Anpassung der Verfahren an spezifische (z.B. regionale oder nationale) Anforderungen oder das Hinzufügen neuer Verfahren.

Eine Forderung nach einem flexiblen und leistungsstarken Werkzeug für die Extraktion von Informationen ist eng mit dem zugrunde liegenden Datenmodell verknüpft. In Abschnitt 5.2.2 wurde erläutert, warum eine Modellprüfung insbesondere dann sinnvoll ist, wenn diese auf einem herstellerneutralen Standard erfolgt. Da IFC in der Industrie der weit verbreitete Standard ist, muss es einen effizienteren Weg geben, um einen direkten Zugriff auf die Daten zu ermöglichen, ohne dass die Belastung durch ein hoch strukturiertes IFC-Schema entsteht.

Es ergibt sich an dieser Stelle also der Bedarf, eine weitere Darstellungssprache zu entwickeln, welche in der Lage ist, sowohl geometrische als auch nicht-geometrische Randbedingungen und Anforderungen auszudrücken, für Domäne-Experten mit begrenztem Softwareentwicklungswissen anwendbar ist und schließlich auch unabhängig von bestimmten Softwareumgebungen eingesetzt werden kann.

7 Visual Code Compliance Checking

7.1 Einführung

Wie die Ausführungen in den vorigen Kapiteln gezeigt haben, konnten die Herausforderungen eines Automated Code Compliance Checking bislang noch nicht ganzheitlich gelöst werden. Die bisherigen Ansätze weisen unterschiedliche Schwächen und Unzulänglichkeiten auf, welche es erforderlich machen, sich der Thematik grundlegend neu anzunehmen. In den folgenden Abschnitten wird ein Ansatz eingeführt, der die bestehenden Problemstellungen adressiert und löst. Die vorgestellte Methode basiert auf der Einführung einer dediziert visuellen Programmiersprache, welche sich spezifisch für das Anwendungsgebiet der Konformitätsprüfungen im Bauwesen eignet. Für eine umfassende Vorstellung des Ansatzes wird folgend die zugrunde liegende Methodik und anschließend die Sprache sowie deren Komponenten und Syntax selbst erläutert werden.

7.2 Visuelle Sprachen und Programmierung

Allgemein wird eine visuelle Sprache als "*formale Sprache mit graphischer Notation*" (Erwig et al., 2017; Hils, 1992; Myers, 1990) definiert. Das bedeutet, dass diese Form der Sprache an Stelle textueller graphische Elemente für die Beschreibung von Inhalten verwendet. Schiffer (1998) beschreibt die visuelle Sprache als eine "*formale Sprache mit visueller Syntax und Semantik*". Wie auch eine textuelle führt eine visuelle Sprache ein System von Zeichen und Regeln auf der syntaktischen Ebene ein und weist diesen visuellen Elementen eine spezifische Bedeutung auf der semantischen Ebene zu. Kommt diese Form der Sprache im Bereich der Computerwissenschaften für die Definition von informationsverarbeitenden Systemen zum Einsatz, wird üblicherweise von Visual Programming Languages (VPLs) oder aber Visual Programming (VP) gesprochen (Schiffer, 1998; Shu, 1988).

Der Begriff *visuell* bezeichnet *das Sehen oder etwas, was den Gesichtssinn betrifft* (Duden, 2013) und beschreibt somit alles, was über die Sehfähigkeit des Menschen aufgenommen und verarbeitet werden kann. Zu diesen Wahrnehmungen gehören nicht nur rein visuelle Elemente, wie beispielsweise Bilder oder Piktogramme, sondern insbesondere auch die Schrift. Allerdings spielt hierbei nicht nur die Bedeutung der visuellen Elemente eine Rolle, sondern insbesondere die Art und Weise dieser Darstellung. Diese kann für den Menschen einen erheblichen Einfluss auf die Interpretation und Wahrnehmung haben.

So wird auch in textuellen Programmiersprachen mit unterschiedlichen visuellen Darstellungsweisen gearbeitet. Es ist weitverbreitete Praxis, Quellcode nach bestimmten Regeln einer jeweiligen Programmiersprache mittels farblicher und stilistischer Merkmalen zu formatieren. Sinn und Zweck einer solchen Formatierung und Darstellung sind, dass es dem Anwender/Programmierer deutlich leichter fällt, einzelne wiederkehrende Elemente, wie z. B. Signalwörter, und deren Zusammenhang untereinander zu

erkennen. Dadurch können der Kontext und die Bedeutung des Programmes beträchtlich schneller erkannt und verarbeitet werden.

Eine Antwort auf die Frage, warum Informationen in visueller Gestalt vom Menschen im Vergleich zu anderen Formen besser aufgenommen werden können, gibt die sogenannte Kognitionspsychologie (Schiffer, 1998). Mehrere Studien belegen, dass visuelle Elemente und Darstellungsweisen den Menschen bei dem Prozess des Aufnehmens und Verstehens unterstützen, und führen dies auf eine Spezialisierung von Prozessen innerhalb des menschlichen Gehirns zurück. Die verschiedenen Prozesse der Informationsaufnahme sprechen unterschiedliche Regionen des menschlichen Gehirns an. Die linke Hemisphäre ist überwiegend für logisches und analytisches Denken verantwortlich, während in der rechten Gehirnhälfte vorwiegend gefühlsbezogene und ästhetische Prozesse sowie auch die Verarbeitung bildlicher und räumlicher Informationen ablaufen. Es ist anerkannter Stand der Forschung, dass analytische Prozesse nur sequentiell, bildliche Prozesse hingegen parallel verarbeitet werden (Staufer, 1987). Die Aufnahme von visuellen Informationen bietet also ein sehr großes Potenzial mit Kapazitäten für die Aufnahme. Werden nun visuelle und analytische Elemente miteinander kombiniert in diesen Prozess eingegeben, fällt es dem Menschen deutlich leichter die Gesamtinformation aufzunehmen. Eine ausführliche Darstellung und Diskussion dieser Erkenntnisse ist bei (Schiffer, 1998) zu finden.

Visuelle Programmiersprachen werden oft auch als *flow-based* bezeichnet, da sie informationsverarbeitende Systeme als Fluss von Informationen von der Eingabe bis hin zur Ausgabe abbilden. Diese Darstellung wird oft mit einem hohen Grad an Benutzerfreundlichkeit in Verbindung gebracht. Insbesondere bei Endanwendern, die ihre ersten Schritte im Bereich der Programmierung bzw. Datenmodellierung machen, soll die visuelle Darstellung die Einstiegshürde senken, da durch die intuitive Interaktion Programme einfacher erstellt und verstanden werden können als durch das textuelle Gegenstück. In einer Studie belegen Catarci und Santucci (1995) die Benutzerfreundlichkeit von visuellen Sprachen mit Hilfe eines Beispiels auf Basis der Anfragesprache SQL. Dieser Beleg wird durch einen vermehrten Einsatz von VPLs in Softwareapplikationen in unterschiedlichen Industriezweigen verstärkt. Entsprechende Beispiele für das Bauwesen werden in dem nachfolgenden Abschnitt 7.4 vorgestellt.

Die Vor- und Nachteile visueller Programmiersprachen werden von Schiffer (1998) ausführlich diskutiert. Als Nachteil wird in der Regel von Kritikern angeführt, dass die mit einer VPL erstellten Programme oft nicht den hohen Anforderungen an eine professionelle Softwareentwicklung genügen. Insbesondere komplexe Kontrollstrukturen wie Schleifen oder Rekursionen sind entweder nicht oder nur schwer umsetzbar und können auch nur schwer vom Nutzer nachvollzogen werden. Als Schlussfolgerung wird angeführt, dass Anwender, die eine visuelle Sprache anwenden, den Informationsprozess auch mit einer herkömmlichen Programmiersprache beschreiben können sollten und dieses letztlich denselben Zeitaufwand mit sich bringt. Der Informatiker Tony Hoare führt als Beispiel zwei visuelle Systeme an, die sich durch nur eine kleine Erweiterung unterscheiden und gemäß einem mathematischen Beweis äquivalent sind. Abbildung 7.1 macht deutlich, dass die Komplexität und Größe eines Systems bereits durch die Erweiterung um eine Komplexitätsstufe beachtlich zunehmen kann (Schiffer, 1998).

Der Aufbau einer visuellen Sprache gleicht den Grundregeln einer herkömmlichen Sprache als "System von Zeichen und Regeln, das einer Sprachgemeinschaft als Verständigungsmittel dient" (Duden, 2013). In diesem Zusammenhang müssen drei unterschiedliche Ebenen des Systems Sprache betrachtet werden (Schiffer, 1998):

Syntax beschreibt die Verknüpfung von Zeichen, d.h. die Beziehung der Zeichen untereinander.

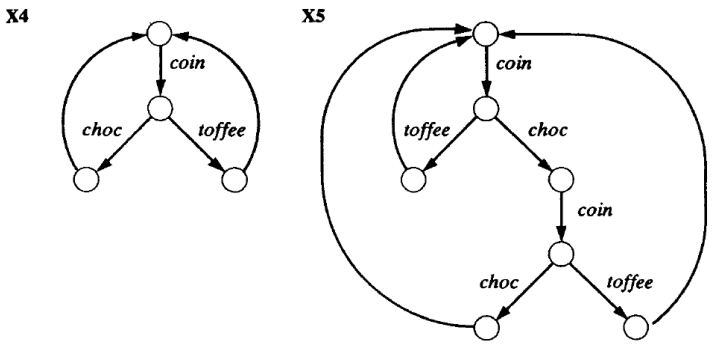


Abbildung 7.1: Veranschaulichung der Komplexität eines visuellen Systems nach Hoare (Schiffer, 1998)

Semantik: beschreibt die Bedeutung von Zeichen, d.h. die Beziehung der Zeichen zu den bezeichneten Dingen.

Pragmatik: beschreibt die Wirkung von Zeichen, d.h. Beziehung der Zeichen zu den betroffenen Personen.

Bei der Datenmodellierung und dem Erstellen von informationsverarbeitenden Systemen wird Sprache meist auf die beiden Ebenen Syntax und Semantik reduziert. Diese Ebenen können in Kombination auch als *formale Sprache* bezeichnet werden, mit welcher die mathematische Annäherung des Informationsgehaltes einer Aussage definiert werden können. Eine solche kann jedoch nur dann getroffen werden, wenn der Sprache eine syntaktische Struktur und jedem einzelnen Element eine wohldefinierte Bedeutung, wie z.B. eine Anweisung oder aber Methode, zugewiesen wurde.

Bei dem Einsatz von visuellen Sprachen muss bei der beabsichtigten Verwendung genau betrachtet werden, ob der maßgebliche Einsatz der Fixierung von Wissen oder dem Senden, Empfangen und Verarbeiten von Informationen dient. Schiffer (1998) unterscheidet zwei Erscheinungsformen hinsichtlich der Anwendungsgebiete: die statische und dynamische Zeichengebung. Ein statisches Informationssystem, welches beispielsweise mit einer formalen, textuellen Sprache formuliert wurde, dient vorrangig der Fixierung von Wissen und arbeitet unabhängig von Raum und Zeit. Der Anwender erhält für eine Eingabe einen Ausgabewert, er kann jedoch nicht mit den Elementen des Systems interagieren. Das Senden und Empfangen von Informationen in einem solchen System läuft zeitlich versetzt ab.

Ein dynamisches Informationssystem hingegen ist offen gestaltet und reagiert auf äußere Einflüsse, wie z.B. Signale. Dieses System beinhaltet ebenfalls Wissen, mit welchem es auf eine Eingabe antworten kann, jedoch lässt es selbst Interpretationsspielraum für den Anwender, welcher in direkte Interaktion mit dem System tritt. Das Ausmaß und die Freiheit dieser Interaktion zwischen System und Anwender werden mit Hilfe der Grammatik festgelegt. Diese spezifiziert für die einzelnen Elemente der Sprache Variablen, die in dem Informationssystem modifiziert werden können. Das Senden und Empfangen von Informationen in einem dynamischen System geschieht immer gleichzeitig.

Der Einsatz von graphischen Elementen für die Beschreibung visueller Programmiersprachen bzw. Entwicklungsumgebungen lässt einen großen Spielraum bei der Umsetzung offen. Es wurden mittlerweile diverse Formen visueller Sprachen entwickelt. Nachfolgend werden einige Arten und Formen visueller Sprachen aufgeführt und kurz erläutert (Burnett und Baker, 1994; Daum, 2018; Schiffer, 1998):

Kontrollfluss-Systeme übertragen das imperative Programmierparadigma auf das VP. Der Kontrollfluss innerhalb eines Programms wird durch explizite Befehle ausgedrückt. Für eine visuelle Darstellung können *Nassi-Shneiderman*-Diagramme oder Petri-Netze verwendet werden.

Datenflusssysteme sind mit funktionalen Systemen verwandt und weisen keine strenge Reihenfolge der Operationen zu. Stattdessen löst die Verfügbarkeit von Daten die Ausführung von Operationen aus. Diese Eigenschaft kann genutzt werden, um ein paralleles Ausführungsmodell für entsprechende Programme zu realisieren.

Funktionale Systeme setzen auf dem Paradigma der funktionalen Programmierung auf. Sie basieren auf höherwertigen und polymorphen Funktionen sowie einem impliziten Typensystem. Für die visuelle Darstellung können Funktionsdiagramme verwendet werden.

Objekt-orientierte Systeme führen eine Klassenbibliothek ein, anhand welcher Instanzen erzeugt werden. Nach dem objektorientierten Paradigma kommunizieren diese Instanzen untereinander, indem sie Nachrichten senden und empfangen. Typischerweise bieten diese Systeme lediglich eine Kapselung/Gruppierung, allerdings keine weiteren objektorientierten Eigenschaften (z. B. Vererbung, Polymorphismus).

Constraint-basierte Systeme beschreiben ein System über die Definition von Randbedingungen (engl. *Constraints*) zwischen Variablen und Konstanten. Das formale Konzept wird als *Constraint Logic Programming* bezeichnet. Variablenzuordnungen, Prozeduren oder Kontrollstrukturen können in diesen Systemen nicht verwendet werden. Ein *Constraint-Solver* findet alle gültigen Kombinationen der eingeführten Variablen.

Formular-basierte Systeme beschreiben Programme in Form von Formularen und Tabellen. In den Zellen der Formulare können Skalare und strukturierte Daten sowie Formeln gespeichert werden. Dabei kann eine Formel auf eine jeweils andere Zellen referenzieren. Auf diese Weise dienen Inhalte anderer Zellen als Eingangsdatum. Ändert sich der Inhalt einer Zelle, stößt das System automatisch eine Neuberechnung abhängiger Zellen an. Bei dieser Systemform werden keine Kontrollstrukturen verwendet werden, handelt es sich um einen deklarativen Programmierstil.

Ergänzend soll an dieser Stelle für den Begriff *visuell* eine wichtige Unterscheidung im Zusammenhang mit Programmiersprachen getroffen werden, da in den vergangenen Jahren diverse Programmiersprachen eingeführt wurden, welche in der englischen Sprache ebenfalls mit dem Adjektiv visuell (engl. *visual*) ausgestattet wurden. Zu diesen gehören insbesondere Sprachen wie *Visual Basic* oder aber *Visual C++* von *Microsoft*. Diese Bezeichnung bezieht sich allerdings auf die Funktionalität zum Bau von Benutzerschnittstellen (UI), welche mit den jeweiligen Programmiersprachen erstellt werden können. Die Sprachen selber sind jedoch keine visuellen Programmiersprachen.

7.3 Historische Entwicklung der VPLs

Erste Forschungen zu visuellen Programmiersprachen lassen sich bis in die 1950er Jahre zurückführen. Sie basieren auf der Idee, dass ein Anwender auch ohne fundierte Programmierkenntnisse in der Lage sein sollte, Verarbeitungssysteme zu erstellen, zu steuern und zu bedienen. Myers (1990) beschreibt, dass der Bedarf für ein solches Steuerungs- bzw. Bedienungswerkzeug mit der Zahl der Computer und deren Anwender rasant anstieg.

Daher wurden Forschungen zu innovativen Bedienungskonzepten, die letztlich zu den VPLs führten, international intensiviert.

Als frühzeitiger markanter Forschungsansatz gilt der *Graphical Program Editor*, der von Sutherland (1966) am *Massachusetts Institute of Technology* im Rahmen seiner Dissertation entwickelt wurde. Mit dieser Anwendung können Schaltpläne für Hardwarekomponenten visualisiert und anschließend auch wieder interpretiert, also eingelesen, werden. Diese Entwicklung gilt als das erste visuelle Programmiersystem.

Durch den technischen Fortschritt in den folgenden Jahrzehnten und die damit verbundene Entwicklung von Soft- und Hardwarekomponenten konnten auch im Bereich der visuellen Sprachen große Fortschritte gemacht werden. Ein weiterer Meilenstein ist die Entwicklung der Programmierumgebung *Pygmalion* von Smith (1975). Das grundlegende Ziel von *Pygmalion* ist es, das kreative Denken eines Anwenders mit Hilfe einer visuellen Umgebung anzuregen, in welcher mit Bildern und Piktogrammen anstelle von Fließ- oder Quellcodebeschreibungen gearbeitet wird. Das Ergebnis wurde im Rückblick auch als *Executable Electronic Blackboard* bezeichnet. Eine wesentliche Grundlage für diesen Ansatz war die zeitgleiche Entwicklung von Benutzerschnittstellen bzw. graphischen Benutzeroberflächen, sogenannte User Interface (UI), in den Computerwissenschaften.

Eine erste kommerzielle VPL wurde in den 1990er Jahren von der US-amerikanischen Firma National Instruments (2018) mit der Einführung von *LabVIEW* auf den Markt gebracht. Hierbei handelt es sich um eine visuelle Programmierumgebung, die auch noch heute auf dem Markt erhältlich ist. Mit dieser können elektrotechnische Schaltungen- und Messeinrichtungen mittels Blockdiagrammen zusammengestellt und anschließend Simulationen auf diesen Systemen durchgeführt werden.

Pau und Olason (1991) stellen das *Visual Logic Programing* für den Bereich der Prädikatenlogik vor. Mit dieser visuellen Sprache ist es möglich, prädikatenlogische Aussagen mit Hilfe graphischer Elemente zu formulieren, diese anschließend automatisch in die Programmiersprache PROLOG zu übersetzen und auszuführen. Für die Formulierung wird in einem Editor ein Graph mit Knoten- und Kantenobjekten gezeichnet, welcher die logische Aussage repräsentiert.

Ein wesentlich neuerer Ansatz kommt aus dem Bereich der Erziehungswissenschaften. 2007 wurde die Programmiersprache *Scratch* am *Massachusetts Institute of Technology* vorgestellt. Diese Anwendung verfolgt das Ziel, jungen Anwendern bei dem Erlernen von Programmiersprachen zu helfen (Maloney et al., 2010). Daher ist auch die Oberfläche von *Scratch* auf eine junge Klientel ausgelegt und arbeitet mit starken visuellen Anreizen und Elementen. Einzelne graphische Objekte sind ähnlich wie Puzzle-Teile dargestellt und geben so die Entwicklungsmöglichkeiten vor. Aus diesen Möglichkeiten erschließt sich die Grammatik der Sprache intuitiv. Gleichzeitig wird jeder Schritt der Entwicklung in einem separaten Fenster direkt visualisiert, damit nicht nur das Ergebnis kontrolliert werden kann, sondern auch die Motivation des Anwenders steigt.

Auch heutzutage finden sich visuelle Sprachen in vielen verschiedenen Anwendungsbereichen wieder, werden fortwährend weiterentwickelt und sind daher hoch aktuell. So findet *Simulink* für *MatLab* insbesondere als Simulationswerkzeug für ingenieurtechnische Anwendungen breite Anwendung, da es dem Anwender eine hierarchische Modellierung mit Hilfe graphischer Blöcken ermöglicht (Angermann, 2009). Im Anwendungsbereich von Roboter- oder aber *IoT*-Programmierung wurden in den letzten Jahren *VIPLE* (Chen und de Luca, 2016) oder aber *Open Roberta* (Jost et al., 2014) vorgestellt.

7.4 Anwendung von VPLs im Bauwesen

Auch das Bauwesen bietet diverse Anwendungsbereiche für visuelle Programmiersprachen. Diese kommen maßgeblich in zwei Bereichen zum Einsatz: (1) zur Generierung von geometrischen und semantischen Informationen und (2) als Abfrage- oder Analysewerkzeug bestehender Modelle (Ritter et al., 2015).

Ein erster Vertreter einer visuellen Sprache im Bauwesen ist die visuelle Sprache für die *Generative Components*, die 2007 von *Bentley Systems* in den Markt eingeführt wurden (Bentley Systems, 2019; Mueller und Smith, 2013). Das Ziel hinter dieser Entwicklung war es, dass Planer mehrere Designvarianten in kürzerer Zeit ausloten, bessere Designs erstellen und komplexe geometrische Beziehungen effizient erstellen sowie verwalten können.

Heutzutage ist das Plug-In *Grasshopper* für die 3D-Modellierungssoftware *Rhinoceros3D* (Grasshopper, 2017; Robert McNeel Associates, 2016) ein prominenter Vertreter einer VPL. *Grasshopper* erweitert das Modellierungswerkzeug um eine visuelle Programmierumgebung und bietet diverse Möglichkeiten, geometrische Operationen mit Parametern unterschiedlicher Datentypen in einem Graphen als visuellen Informationsfluss zu definieren. Der Fluss repräsentiert die Schritte, welche zur Erstellung einer Geometrie erforderlich sind. Darüber hinaus können durch Modifikation und Manipulation der Parameter auch komplexe Geometrien einfach kontrolliert und gesteuert werden. Diese Anpassungsfähigkeit ermöglicht dem Anwender heuristisch nach der bestmöglichen Lösung zu suchen (engl. *"trial and error"*), was insbesondere bei kreativen Tätigkeiten, wie beispielsweise der Findung einer ästhetisch ansprechenden Form, von großem Wert und Interesse ist.

Daher hat sich *Grasshopper* insbesondere im Bereich der Modellerstellung, also besonders in der Architektur und Gestaltungsplanung, etabliert und erfährt einen starken Rückhalt aus einer großen *Community* sowie von Drittanbietern, die das Projekt mit Bibliotheks-Erweiterungen, also neuen visuellen Operatoren, unterstützen. Viele der VPL-Umgebungen, wie insbesondere auch *Grasshopper*, bieten Entwicklern die Möglichkeit, die Bibliotheken um benutzerdefinierte Funktionen zu erweitern und so den Anwendungsbereich zu erweitern. Dabei können das Design und die Funktionalität eines VPL durch seinen Anwendungsbereich entscheidend beeinflusst werden. In der jüngeren Vergangenheit haben diverse BIM-Autorenwerkzeuge eine (Interaktions-)Schnittstelle zu *Grasshopper* und *Rhinoceros3D* veröffentlicht (z. B. *Trimble Tekla* oder *Graphisoft ArchiCAD*), damit Anwender die Vorteile der VPL in ihrer gewohnten Umgebung nutzen können (Graphisoft, 2018; Trimble, 2018).

Aufgrund des Erfolgs von Werkzeugen wie *Grasshopper* bieten einige Softwarehersteller von BIM-Autorenwerkzeugen mittlerweile eigene VPL-Umgebungen, die sich für den Einsatz in spezifischen Anwendungen eigenen. Als Beispiel sind an dieser Stelle insbesondere *Dynamo* für Autodesk Revit (Autodesk Inc., 2018) sowie *Marionette* für *Vectorworks* zu nennen (Computerworks, 2018). Diese Tools nutzen

die in den jeweiligen Softwareprodukten gegebenen Methoden zur Manipulation und Erstellung von Bauwerksinformationen und bieten den Anwendern die Möglichkeit, Informationsverarbeitungsprozesse mit Hilfe einer VPL zu definieren.

Kommerzielle VPL-Systeme konzentrieren sich in der AEC-Industrie hauptsächlich auf gestaltungsplanerische Aufgabenfelder, wie insbesondere die Geometriemodellierung. Daher sind die Sprachen in der Regel so konzipiert, dass sie bei intuitiven Designaufgaben unterstützen. Um dem Anwender möglichst viel Freiraum bei diesen Aufgaben einzuräumen, weisen die VPL-Systeme in der Regel eine hohe Fehlertoleranz auf. Daher fehlt es diesen in der Regel an einer formalen Stringenz und Strenge. So erlauben diese beispielsweise die Definition von generischen (*untypisierten*) Parametern, was zu vielfältigen Fehlern führen kann.

Mit zunehmender Attraktivität der kommerziellen Anwendungen von VPLs im Bauwesen sind auch im Bereich der Forschung diverse Ansätze entwickelt worden, die sich auf verschiedene Einsatzbereiche konzentrieren.

Dieckmann und Russell (2014) stellen die Anwendung der visuellen Programmiersprache *Dynamo* für die Modellprüfung im Rahmen der Lehre an ihrem Institut vor. In der Forschungsarbeit wird ein Ansatz zur Kategorisierung und Organisation von Modellprüfungen im pädagogischen Kontext vorgestellt. Für die Automatisierung des Bewertungs- und Feedbackprozesses kommt die visuelle Programmiersprache zur Anwendung. Ziel dieses Ansatzes ist es, die Bewertungsprozesse in BIM-Softwarekursen zu unterstützen, indem die Hauptziele des Projektes objektiv beurteilt werden. In der Studie wird gezeigt, dass der Zeitaufwand für die Bearbeitung einer großen Anzahl von eingereichten Modellen erheblich reduziert werden kann.

Seifert und Mühlhaus (2013) stellen Methoden und Werkzeuge vor, mit welchen Nachverdichtungspotentiale auf innerstädtischen Planungsflächen erkannt und Nachverdichtungsstrategien abgeleitet und überprüft werden können. In der Anwendung *Urban Strategic Playground* wird eine VPL-Schnittstelle zur Steuerung und Modifikation verwendet. Dieses soll insbesondere Architekten den Umgang mit dem Werkzeug erleichtern. Die Szenarien zur Nachverdichtung werden mit Hilfe eines parametrischen 3D-Stadtmodells durchgeführt, in welchem die Gebäuderepräsentationen nicht mit statischen geometrischen Objekten, sondern durch die generischen geometrischen Beziehungen im Sinne der semantischen und geometrischen Randbedingungen verschiedener Gebäudetypologien und deren topologische Abhängigkeiten dargestellt sind. Die Parameter des Modells können mit Hilfe der VPL gesteuert sowie Analysen durchgeführt werden (Seifert et al., 2014; Seifert und Petzold, 2016).

Kensek (2015) beschreibt in einer Forschungsarbeit drei Fallstudien: die Erweiterbarkeit der VPL-Umgebung Dynamo für den Einsatz zur Gebäudeenergie-Simulation, die Steuerung der Reaktion eines virtuellen Modells durch Lichtsensoren und die interaktive Aktualisierung von Beschattungskomponenten für eine Gebäudefassade in Abhängigkeit des jeweiligen Sonneneinfallswinkels. In der Arbeit wird gezeigt, dass man mit Hilfe einer visuellen Programmiersprache die Komponentenparameter fortwährend aktualisieren kann. So können die Simulationsergebnisse direkt in das BIM einfließen, ohne dass die Arbeitsumgebung verlassen werden muss. Die Fallstudien zeigen, dass die vorgeschlagenen Arbeitsprozesse für nachhaltige Entwurfssimulationen mit Hilfe einer VPL viel einfacher durchgeführt werden können als in dem herkömmlichen Prozess, in welchem Entwurfs- und Analysemodelle mit Hilfe eines neutralen Datenformates wie IFC oder gbXML für den Datentransfer getrennt gehalten werden.

Zusammenfassend sind in Tabelle 7.1 wesentliche ausgewählte Vertreter von VPL-Anwendungen aufgelistet.

Tabelle 7.1: Ausgewählte VPL-Anwendungen

Name	Typ	Hersteller	URL
Dynamo	Standalone; Plug-In für die BIM-Autorensoftware Autodesk Revit	Autodesk	`http://dynamobim.org/`
Google Blockly	Web-basiert	Google	`https://developers.google.com/blockly/`
Grasshopper 3D	Rhinoceros3D/ArchiCAD Plug-In	open source	`http://www.grasshopper3d.com/`
Marionette	Vectorworks Plug-In	Vectorworks	`http://www.vectorworks.net/training/marionette`
Scratch	Web-basiert	MIT	`https://scratch.mit.edu/`
TUM.CMS.VplControl	Standalone Entwicklungs-Bibliothek	Lehrstuhl CMS	`https://github.com/tumcms/TUM.CMS.VPLControl`

7.5 Visual Code Checking Language

In der vorliegenden Arbeit soll gezeigt werden, dass die zentralen Herausforderungen, die in Kapitel 5 vorgestellt wurden, mit Hilfe einer spezifisch auf die Anforderungen von Konformitätsprüfungen im Bauwesen zugeschneiderten visuellen Sprache gelöst werden können. Wesentliche Gründe für die Einführung einer visuellen Sprache in diesem Anwendungsgebiet insbesondere im Hinblick auf die Unzulänglichkeiten und Schwächen der in Kapitel 6 vorgestellten Ansätze sind:

Fachspezifische Anwendbarkeit. Die eingeführte VPL adressiert maßgeblich die fachspezifischen Anwender der Konformitätsprüfung, also Fachplaner, wie Architekten und Ingenieure, sowie Ingenieure von Seiten der prüfenden Institution. In den bisherigen Forschungsarbeiten und Ansätzen wird insbesondere die Formalisierung von Regelwissen fokussiert, hierbei jedoch häufig außer Acht gelassen, dass Architekten und Ingenieure üblicherweise nicht über fundierte Programmierkenntnisse verfügen. Viele der vorgestellten Methoden anderer Ansätze richten sich offensichtlich an Softwareexperten bzw. Programmierer. Eine eigenständige Anwendung und Umsetzung vieler Methoden der vorgestellten Ansätze sind vonseiten der Fachanwender selbst schwierig und bedürfen der Unterstützung durch Softwareentwicklern.

Wenn die Formalisierung des Regelwissens jedoch von einem Programmierer durchgeführt wird, dann hat dieses einen sehr hohen einseitigen Entwicklungsaufwand zur Folge, da jede Abwandlung oder Ausprägung einer Regulierung externen Aufwands bedarf. Zum anderen ergibt sich dadurch eine hohe Abhängigkeit von dieser Entwicklungsleistung. Eine visuelle Programmiersprache schließt diese Lücke, da sie eine Anwendung ohne fundierte Programmierkenntnisse ermöglicht und so die Fachexperten die Übersetzungsleistung selbst übernehmen können.

Fachspezifische Ausdruckskraft. VPLs lassen sich über das Sprachdesign, die verfügbaren Elemente und die Grammatik sehr spezifisch auf ein Anwendungsgebiet anpassen. Wie in Abschnitt 5.4 beschrieben, stellen die komplexen Bauwerksstrukturen sowie die Konformitätsprüfung selbst sehr spezifische Anforderungen an die Ausdruckskraft einer Regelsprache.

Eine visuelle Sprache ermöglicht es, die komplexe Struktur von Bauvorschriften zu erfassen, diese gleichzeitig für Experten zugänglich und einfach handhabbar zu machen. Mit der Einführung von

passenden Operatoren einer visuellen Sprache kann diese Ausdruckskraft entsprechend bereitgestellt werden. Eine VPL lässt sich darüber hinaus individuell erweitern und räumt somit die notwendige Flexibilität ein, um die Ausdruckskraft in den spezifischen Anwendungsbereichen zu erhöhen.

So lassen sich mit Hilfe einer VPL nicht nur Methoden zur Aufbereitung von Informationen aus dem Gebäudemodell, sondern auch geometrisch-topologische Analysewerkzeuge implementieren.

Eine hohe Spezifität, welche bei einer solchen Sprache angestrebt wird, bedingt allerdings auch, dass diese von Grund auf neu entwickelt wird und nicht auf einem bestehenden System aufsetzt. Bestehende VPL-Umgebungen und -systeme wie beispielsweise *Dynamo* (Autodesk) oder *Marionette* (Vectorworks) basieren maßgeblich auf nativen Datenmodellen und sind auf einen Einsatz innerhalb der jeweiligen Autorenwerkzeuge ausgerichtet.

Gleichzeitig ist es nur bedingt sinnvoll, ein Werkzeug für die Qualitätssicherung ausschließlich auf einer nativen Umgebung aufzubauen. Vielmehr sollte sich eine Umgebung zur Prüfung von Modellinhalten hinsichtlich mannigfaltiger Anforderungen so nah wie möglich an offenen Datenstandards, wie insbesondere die IFC, anlehnen, um so eine möglichst weitreichende Anwendbarkeit zu erreichen. Wie bereits in Kapitel 2 beschrieben, gehen die Modellinhalte bei der Lieferung aus der nativen Autorenumgebung in eine gemeinsame Umgebung über. An dieser Schnittstelle werden entweder nicht alle Modellinhalte, welche in dem Autorenwerkzeug erstellt und gepflegt wurden, übergeben oder aber diese werden in ein anderes Datenschema, z. B. die IFC, überführt. Bei der Überführung werden Informationen gezielt auf die Anforderungen des Datenschemas von IFC sowie nach den jeweiligen spezifischen Anforderungen abgebildet, was bedeuten kann, dass diese angepasst, transformiert oder aber verändert werden. Daher ist eine Prüfung in der Ausgangsumgebung zwar sinnvoll, aber weder hinreichend noch maßgebend. Darüber hinaus handelt es sich bei der Prüfung von Bauwerksmodellen um eine koordinative Tätigkeit, welche erfordert, dass auch mehrere Modelle unterschiedlicher Disziplinen gleichzeitig betrachtet werden. Eine solche Betrachtung kann nur durchgeführt werden, wenn die Informationen aus den verschiedenen Disziplin-Modellen und Autorenwerkzeugen zusammengeführt werden können. Ein Schlüssel hierfür sind offene Standards, wie insbesondere IFC. Daher ist es sinnvoll, diesen bei der Qualitätssicherung im Allgemeinen zu folgen.

Darüber hinaus werden etablierte VPL-Tools im Bauwesen vermehrt für die Erzeugung von Informationen, wie z.B. das intuitive Generieren und Steuern komplexer dreidimensionaler Geometrien, verwendet. Bei einem analytischen Einsatz spielt jedoch die Verlässlichkeit und Stringenz des Systems eine wesentliche Rolle. Nisbet et al. (2008) zeigen auf, dass die Akzeptanz als Schlüsselkriterium für die erfolgreiche Implementierung eines ACCC-Systems nur dann gegeben ist, wenn Anwender Vertrauen in eine Softwareanwendung hegen können. Ist dieses nicht der Fall, verringern sich die Akzeptanz und somit auch die Möglichkeiten zur Automatisierung deutlich. Eine hohe Fehlertoleranz, wie diese insbesondere bei intuitiven Aufgaben zum Einsatz kommt, wäre an dieser Stelle nachteilig. Eine Eigenentwicklung der Sprache ermöglicht die Festlegung grundlegender Eigenschaften, wie insbesondere die Fehlertoleranz.

Transparenz und Prüfbarkeit. Wie bereits in Kapitel 4 und 5 dargestellt, übernimmt der menschliche Anwender die Verantwortung für die Ergebnisse der Überprüfung und muss als Konsequenz die

(Zwischen-)Ergebnisse regelmäßig prüfen. Durch den Einsatz einer VPL kann eine notwendige Transparenz geboten und so die Überprüfbarkeit für den Fachanwender gewährleistet werden. Anhand der offengelegten Prozessstruktur kann zu bestimmten Prozess- und Zeitpunkten ein Zwischenergebnis abgerufen und eine Plausibilitätsüberprüfung durchgeführt werden.

Die visuelle Sprache unterstützt somit maßgeblich die Einrichtung einer Mensch-Maschine-Schnittstelle, welche eine zwingende Voraussetzung für den Erfolg der Automatisierung der Konformitätsprüfung darstellt. Ziel ist es, dass der Anwender zu jedem Zeitpunkt und Grad der Fertigstellung des visuellen Verarbeitungssystems in der Lage ist, die einzelnen Verarbeitungsschritt nachzuvollziehen und zu kontrollieren. So kann er die Genauigkeit der Ergebnisse prüfen und die Verantwortung des Prüfprozesses trotz Automatisierung übernehmen.

Aufgrund der aufgeführten Vorteile einer visuellen Sprache im Bezug auf das Anwendungsgebiet ACCC soll in der vorliegenden Arbeit die VCCL, eine visuelle Sprache für die Beschreibung von Modellprüfungsprozessen vorgestellt werden. Diese ermöglichen es, Konformitätsprüfungen hinsichtlich geltender Normen und Richtlinien durchzuführen. Grundsätzlich verfolgt die VCCL dabei die folgenden Prinzipien bzw. Eigenschaften, welche zusammengenommen ein Höchstmaß an Flexibilität bei gleichzeitiger Übersicht und Transparenz für den Anwender bewirken:

Generizität beschreibt die Eigenschaft der VCCL, dass alle Elemente unabhängig vom Grad der Komplexität möglichst generisch beschrieben sind. Dadurch kann jedes Element in jeder Situation und an jedem Punkt des gewünschten Prüfprozesses eingesetzt werden.

Feinste Granularität gewährleistet die Zerlegbarkeit eines Gesamtprozesses in seine Einzelteile und somit die gesamtheitliche Transparenz. Jeder mit Hilfe der VCCL formulierte Informationsprozess ermöglicht dem Anwender, diesen bis auf die unterste Ebene und Einzelteile zu zerlegen, um so eine Einsicht zu erhalten.

Semi-Automatik bezeichnet die Eigenschaft der VCCL, dass diese keine Vollautomatisierung anstrebt. Wie in Abschnitt 5.3 beschrieben sind vollautomatische Systeme mit Vorsicht zu behandeln, da der Mensch weitestgehend als Kontrollinstanz herausgenommen wird. Die VCCL ist daher als Werkzeug zu verstehen, welches es dem Anwender ermöglicht, Regeln formal und intuitiv zu formulieren. Diese Formalisierung und Standardisierung erhöht in letzter Konsequenz auch die Effizienz, allerdings stehen an erster Stelle die Ausdruckskraft, Transparenz, Überprüfbarkeit sowie die Anwendbarkeit im Fokus.

7.5.1 Aufbau der VCCL

Wie bereits in Abschnitt 7.2 beschrieben, müssen auch für eine visuelle Sprache die grundlegenden Ebenen Syntax und Semantik definiert werden. Daher werden in diesem Abschnitt die einzelnen Elementgruppen bzw. -typen der VCCL sowie deren graphische Darstellung vorgestellt.

Die VCCL ist eine stark typisierte, objektorientierte Sprache. Auf der semantischen Ebene bietet sie zwei verschiedene Elemente: *Methoden* und *Schnittstellen* (*Ports* genannt).

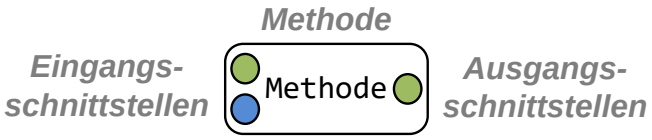


Abbildung 7.2: Graphische Darstellung einer VCCL-Methode

```
1  Process (INPUT input_1, INPUT input_2, [...])
2  {
3    OUTPUT result;
4    result = [...];
5    return result;
6  }
```

Quellcode 7.1: Darstellung einer generischen VCCL-Methode in Pseudocode

Datenverarbeitungsprozesse werden innerhalb eines VCCL-Systems als Methoden dargestellt. Eine Methode beschreibt eine wohldefinierte Operation für eine bestimmte Anzahl von Eingangsgrößen, die sogenannten *Operanden*, und erzeugt ein entsprechendes Ergebnis. Da eine Methode mehrere Informationsflüsse bzw. Eingangsparameter zusammenführt und diese zu einem Ergebnis hin verarbeitet, werden diese Objekte auch als *Knoten* bezeichnet. Als graphische Notation eines Methoden-Knoten verwendet die VCCL ein Rechteck mit abgerundeten Ecken. Ein generisches Beispiel ist in Abbildung 7.2 dargestellt.

Sowohl die eingehenden als auch die ausgehenden Informationen sind für den Nutzer transparent und sichtbar, die Verarbeitung selbst jedoch nicht. Technisch wird die Verarbeitungsprozedur, die innerhalb einer Methode durchgeführt wird, mit Hilfe von einem textuellem Programmiercode definiert. Um das Prinzip der Implementierung eines solchen Verarbeitungsprozesses aufzuzeigen, lässt sich dieser als Pseudo-Code darstellen. Die Darstellung einer generischen Methode, wie diese in Abbildung 7.2 aufgeführt ist, ist in Quellcode 7.1 dargestellt.

Bei der Beschreibung von Informationsprozessen kann es hilfreich oder auch erforderlich sein, dass eine Methode nicht nur einen, sondern mehrere Rückgabewerte adressieren kann. Üblicherweise erlauben herkömmliche, textuelle Programmiersprachen zwar die Definition von mehreren Eingabe-, aber nicht von mehreren Rückgabewerten. Diese Einschränkung kann zwar mittels der Deklaration einer Ergebnisdatenstruktur umgangen werden, allerdings erfordert dieses einen Mehraufwand. Die graphische Darstellung der VCCL-Methoden hingegen erlaubt eine visuelle Abgrenzung mehrerer Rückgabewerte, da die jeweiligen Ausgabewerte über die einzelnen Schnittstellen eindeutig zugeordnet werden können. Die visuelle Darstellung der Methoden wird hier also zu einem Vorteil und macht die Deklaration einer zusätzlichen Ergebnisstruktur unnötig. Eine entsprechende graphische sowie Pseudo-Code Darstellung ist in Abbildung 7.3 und Quellcode 7.2 aufgeführt.

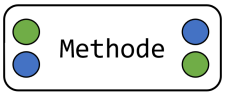


Abbildung 7.3: Graphische Darstellung einer VCCL-Methode mit mehreren Rückgabewerten

```
Process (INPUT input_1, INPUT input_2, [...])
{
  OUTPUT result1, result2;
  result1 = [...];
  result2 = [...];
  return result1, result2;
}
```

Quellcode 7.2: Darstellung einer generischen VCCL-Methode in Pseudocode mit mehreren Rückgabewerten

Für die Übertragung der Informationen besitzt ein Methoden-Knoten Eingangs- sowie Ausgangsschnittstellen, über die Informationsflüsse zu der Methode eingehen und das Ergebnis in der Folge weitergegeben werden kann. Angelehnt an die europäische Schriftrichtung fließen Informationen von links nach rechts durch das Verarbeitungssystem. Daher befinden sich die Eingangs-Schnittstellen auf der linken, die Ausgangs-Schnittstellen hingegen auf der rechten Seite einer Methode.

Die Schnittstellen werden als *Ports* bezeichnet und repräsentieren Datenobjekte bestimmter Datentypen, welche innerhalb eines VCCL-Programms verarbeitet werden können. Zur Kennzeichnung der jeweiligen Datentypen werden die Ports als Kreise mit einer farbigen Füllung dargestellt, , die den Datentyp der Eingang- oder Ausgangsgröße repräsentiert. Um die Übersichtlichkeit des Gesamtprogramms zu verbessern und den Anwender beim Verständnis der Verarbeitung der Informationen zu unterstützen, können die Ports zusätzlich mit einem Label versehen werden. Dieses kann optional für einen jeweiligen Port neben dem Datentyp eine benutzerdefinierte Zeichenkette enthalten, die beschreibt, welche Informationen an diesem Port übertragen werden. Eine Auswahl der verfügbaren Datentypen sowie die entsprechende graphische Notation ist in Abbildung 7.4 dargestellt.

Um die notwendige Flexibilität zu gewährleisten, ist das Typensystem der VCCL nicht limitiert, sondern kann erweitert werden. In einem ersten Schritt stellt das System eine Anzahl vordefinierter Datentypen zur Verfügung, wie diese aus gängigen Programmiersprachen bekannt sind. Zu diesen zählen grundlegende Typen wie etwa $String$, $Boolean$, $Integer$, $Double$, $Float$. Darüber hinaus wir die Relation als primärer Datentyp unterstützt. Weiterführende Details zu der Anwendung dieser Relationen in der VCCL-Umgebung sind in Abschnitt 7.5.3 zu finden. Um die Entitäten eines Bauwerkmodells sowie dieses selbst darzustellen, bietet die VCCL überdies die Datentypen *BuildingElement* sowie *BuildingModel*.

In regulatorischen Anforderungen werden oftmals Datentabellen beispielsweise für die Darstellung von Randbedingungen in Abhängigkeit von weiteren Parametern verwendet. Daher wird zusätzlich der Datentyp *DataTable* eingeführt, welcher die Inhalte solcher Datentabellen darstellen und übertragen kann. Ein Beispiel für die Verwendung dieses Datentyps wird in Abschnitt 8.3 vorgestellt.

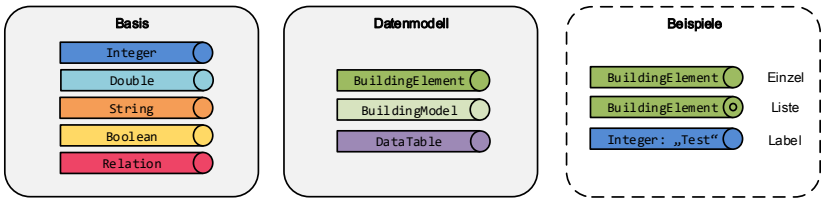


Abbildung 7.4: Graphische Darstellung der Ports mit der jeweiligen farblichen Darstellung unterschiedlicher Datentypen

Auf der Basis dieser Datentypen lassen sich über die Ports nicht nur einzelne, sondern auch Listen mit Elementen des gleichen Datentyps übertragen. Mehrdimensionale Elemente eines bestimmten Datentyps werden in der graphischen Notation als Doppelkreis dargestellt. Abbildung 7.4 gibt eine Übersicht der definierten VCCL-Datentypen.

Um mit den gegebenen Elementen eine Verarbeitungskette aufzubauen, werden die Ports der Methoden mittels gerichteter Kanten verbunden. Die Verbindung der Elemente untereinander unterliegt spezifischer Regeln, welche die syntaktische Ebene der VCCL darstellen und als wesentlicher Kern der VCCL-Grammatik verstanden werden können:

Eindeutige Festlegung des Informationsflusses. Eine Kante leitet Informationen vom Ausgangsport eines Quellknotens an den Eingangsport des Zielknotens weiter. Dieses bedeutet, dass der Fluss der Informationen stets von links nach rechts zeigt. Die weitergeleiteten Informationen können nur in diese Richtung übertragen werden, damit diese eindeutig festgelegt ist.

Strikte Typisierung. Da jedem Port ein klar definierter Datentyp zugeordnet ist, können nur Informationen des definierten Typs über diesen fließen und somit übertragen werden. Auf diese Weise wird die Datenverarbeitung innerhalb eines VCCL-Programms streng kontrolliert. Da Informationen somit nicht irregeführt werden können, werden die Präzision und Genauigkeit erhöht.

Definition von Start- und Endgrößen. Für die Erstellung eines VCCL-Graphen steht dem Anwender eine Oberfläche zur Verfügung, welche in eine Eingangs-, eine Ausgangs- sowie eine Prozessumgebung unterteilt ist. Um einen eindeutigen Start- und Endpunkt eines jeden VCCL-Programms zu definieren, können vom Nutzer in dem globalen Ein- und Ausgabebereich benutzerdefinierte Ports angegeben werden, welche als initiale Informationsquelle oder als resultierender Endpunkt für die definierte Routine fungieren. Bei der Erstellung eines VCCL-Programms muss der Anwender also festlegen, welche Art von Informationen als initiale Eingabe in das System eingeht und welche Informationen als Endergebnis von dem Programm erzeugt werden.

Die verschiedenen Teile eines VCCL-Programms sowie eine beispielhafte Erstellung eines generischen Programms sind in Abbildung 7.5 dargestellt.

Mit Hilfe der globalen Ports können Informationen in das jeweilige VCCL-Programm, welches in sich generisch formuliert ist, eingegeben werden. Die Eingangsports fungieren daher als Nutzereingabe oder als Zugriff auf das Bauwerksmodell, welche für das jeweilige VCCL-Programm als Ausgangspunkt verwendet werden soll. Dem Nutzer stehen entsprechende Möglichkeiten zur Verfügung, über einen

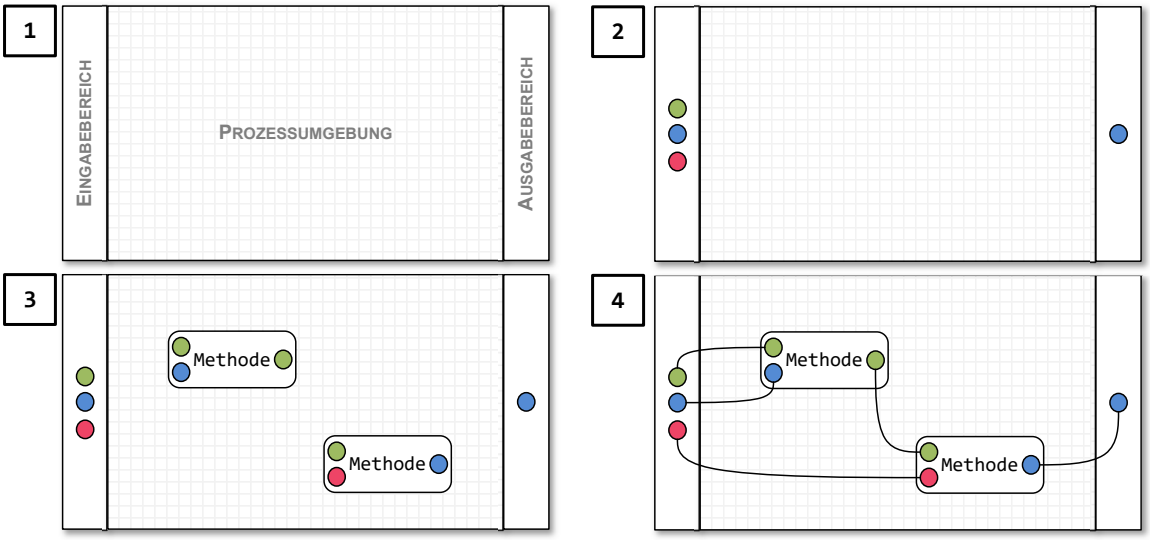


Abbildung 7.5: Darstellung der VCCL-Nutzeroberfläche mit der Eingangs-, Ausgangs- sowie Prozessumgebung

Port vom Typ *double* eine Konstante in das System oder durch einen mehrdimensionalen Port vom Typ *BuildingElement* die Bauteile aus einem spezifischen IFC-Modell in das System einzugeben.

Um festzustellen, ob ein VCCL-Programm korrekt durchgeführt werden kann, muss dieses validiert werden können. Als wesentliches Ziel eines Programms muss die Erzeugung des beabsichtigen Endergebnisses betrachtet werden. Ein wesentlicher Kern der Validierung ist die Feststellung, ob das Programm selbst ausgeführt und ein entsprechendes Ergebnis erzeugt werden kann. Jedes VCCL-Programm muss somit hinsichtlich seiner prozesstechnischen Durchführungsfähigkeit überprüfbar sein. Ausgehend von den anfänglich gegebenen Informationen, repräsentiert durch die globalen Eingangsports, fließen die Informationen durch das VCCL-Programm, indem diese über die Kanten von Port zu Port geleitet werden. Es gilt, dass ein VCCL-System nur dann gültig ist, wenn (1) alle Kanten korrekt typisiert zugeordnet sind und (2) alle erforderlichen Ports verbunden sind, sodass das Ergebnis herbeigeführt werden kann. Dieses bedeutet also nicht, dass alle Ports besetzt sein müssen. Das Prinzip des Informationsflusses ist für ein generisch gültiges Programm in Abbildung 7.6 schematisch dargestellt.

Wie bereits in Abschnitt 7.5 beschrieben, folgt die VCCL dem Prinzip der feinsten Verarbeitungsgranularität. Dies bedeutet, dass der Anwender einen Einblick in jeden einzelnen Teil der Verarbeitungskette eines VCCL-Programms haben soll. Um dieses zu ermöglichen und gleichzeitig große VCCL-Programme zu vermeiden, besitzt die VCCL die Möglichkeit zur Verschachtelung von Teilmengen eines Programms. Dieses Konzept ist vergleichbar mit dem Konzept der Funktionen oder Unterprogrammen in textuellen Programmiersprachen.

Basis für diese Schachtelung sind der globale Ein- und Ausgabebereich. Jede VCCL-Methode beschreibt einen Verarbeitungsprozess und kann als separates VCCL-Programm dargestellt werden. Ein verschachteltes VCCL-Programm besteht aus einer Reihe wohl definierter Verarbeitungsschritte, die auf einer oberen Ebene ausgeblendet werden können, um die Komplexität des Gesamtprogramms zu

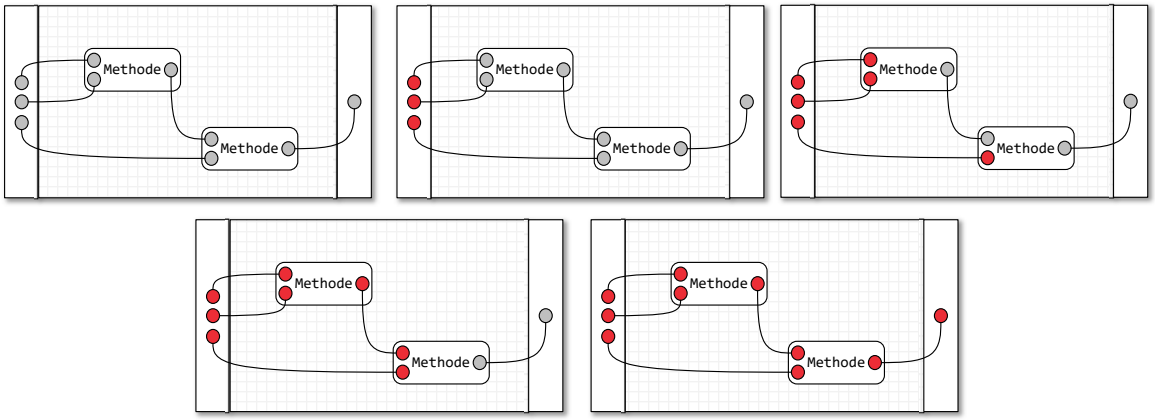


Abbildung 7.6: Darstellung des Informationsflusses durch ein VCCL-Programm. Das Programm wird ausgehend von den globalen Eingangsports über die Kanten und Methoden hinweg mit Informationen *geflutet*.

reduzieren. Das verschachtelte Programm hat klar definierte Eingänge und Ausgänge und kann in jedem beliebigen VCCL-Programm wiederverwendet werden. Die Ein- und Ausgabeports des betroffenen Methodenknotens definieren die Ports des globalen Ein- und Ausgabebereichs. In Abbildung 7.7 ist das Grundprinzip eines solchen geschachtelten VCCL-Programms dargestellt. Zur Verdeutlichung, dass es sich bei der Methode um ein verschachteltes Programm handelt, wird diese zusätzlich eingefärbt, damit der Unterschied eindeutig sichtbar ist.

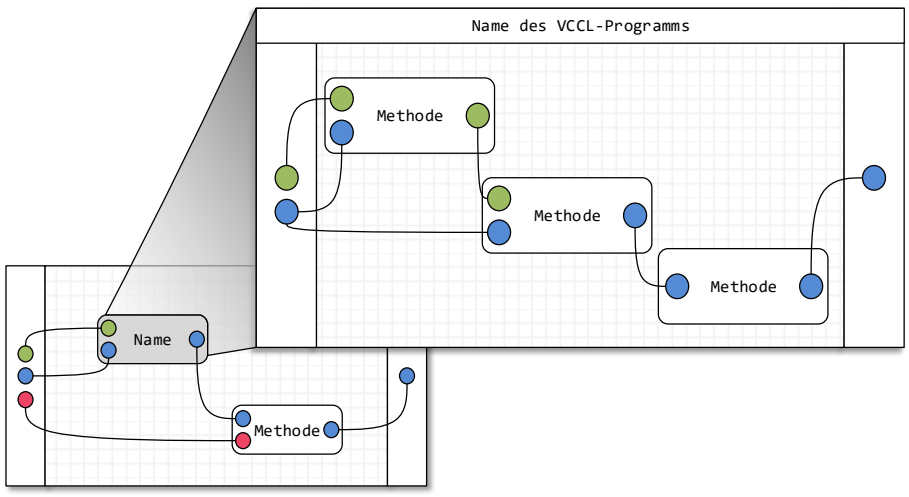


Abbildung 7.7: Einbettung eines VCCL-Programms als Methode in einem VCCL-Programm höherer Ordnung

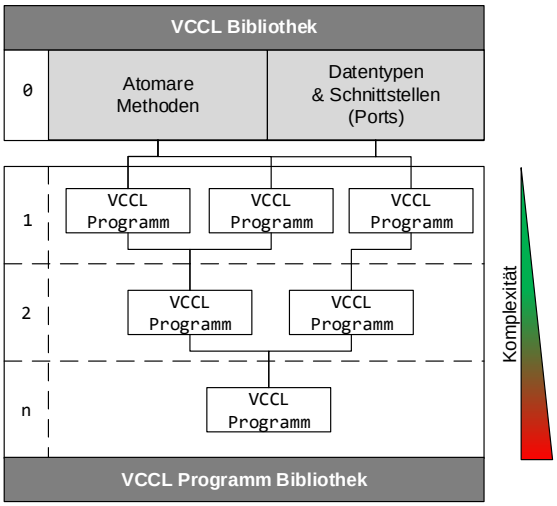


Abbildung 7.8: Aufeinander aufbauende VCCL-Programme ergeben eine Bibliothek von Routinen steigender Komplexität, die bei der Erstellung neuer Programme wieder verwendet werden können

Auf diese Weise kann der Anwender eine hierarchische VCCL-Bibliothek aufbauen, die den besonderen Anforderungen einer jeweiligen Prüfung oder eines Prüfbereichs gerecht wird. So können beispielsweise spezifische, höherwertige Methoden aufgebaut werden, welche sich insbesondere für die Auswertung geometrischer Randbedingungen eignen.

Basis für die Verschachtlung ist eine Basis-Bibliothek, welche grundlegende Basismethoden und Datentypen definiert. Diese Elemente können dann vom Nutzer für die Zusammenstellung von höherwertigen Strukturen verwendet werden. Die daraus resultierende hierarchische Struktur der VCCL-Elemente ist in Abbildung 7.8 dargestellt.

Für die VCCL-Bibliothek müssen grundlegende Methoden definiert werden, welche als Ausgangspunkt und Basisroutinen für den Aufbau von komplexeren Programmen vorgegeben sind. Sie beschreiben Operationen eines VCCL-Programms, deren Semantik eineindeutig definiert ist. Diese Methoden werden daher als *atomare Methoden* bezeichnet. Da die interne Struktur der atomaren Methoden für den Anwender unsichtbar ist, stellen sie *Black-Boxes* dar. An diesem Punkt muss man allerdings davon ausgehen, dass ein bestimmtes *Black-Box-Niveau* für einen Anwendungsbereich akzeptabel ist, in dem eine tiefer gehende Kontrolle und Einsicht weder erwünscht noch notwendig ist, da sonst ein zu hoher Aufwand für die Implementierung der bereitgestellten Funktionalität mit VCCL anstelle einer Standard-Programmiersprache entsteht.

Ein Beispiel ist die Auswertung von geometrischen Informationen, wie z.B. die Berechnung einer kürzesten Wegstrecke für einen gegebenen Grundriss. An dieser Stelle kann angenommen werden, dass für den Endanwender das Ergebnis der Operation, also die erhaltene Wegstrecke, eine deutlich höhere Relevanz hat als die dahinter liegende, komplexe Berechnung. In diesem Falle kann also abgewägt werden, dass für den Ingenieur als Endanwender einen größeren Mehrwert und eine akzeptable Barriere darstellt, keinen weiteren Einblick in die verarbeitete Routine zu bekommen.

So kann diese Berechnung als atomare Methode vorgegeben und für verschiedene Anwendungen wie z. B. Fluchtwegberechnungen oder Prozessabstandsberechnungen wiederverwendet werden. Ein wichtiges Merkmal solcher atomarer Methoden ist die Generizität, sodass die Methoden für möglichst viele verschiedene Zwecke eingesetzt werden können. Allerdings müssen unterschiedliche *Black-Box*-Beschränkungen für verschiedene Anwendungen und Erfahrungsniveaus berücksichtigt werden.

Die atomaren Methoden der VCCL werden im nachfolgenden Abschnitt 7.5.3 im Detail erläutert.

7.5.2 Das Datenmodell der VCCL

Ein weiterer wesentlicher Aspekt bei der Anwendung der VCCL ist das Datenmodell, welches als Grundlage herangezogen wird und auf welchem die Methoden der VCCL operieren. Auch bei visuellen Programmiersprachen hat die Komplexität des zugrunde liegenden Datenmodells einen wesentlichen Einfluss auf die Komplexität der Sprache selbst.

Wie bereits in Abschnitt 2.4.2 ausgeführt wurde, gibt es für das digitale Bauwesen den offenen Datenstandard IFC, welcher sich für die Übergabe von bauwerkbezogenen Informationen eignet. Da dieses Datenformat von den meisten Autorenwerkzeugen unterstützt wird, bietet es sich an, auf diesen Datenstandard zu setzen. Durch die hohe Anzahl der definierten Entitäten und nicht zuletzt durch die Verwendung von inversen Attributen sowie objektifizierten Beziehungen handelt es sich bei den IFC um ein verhältnismäßig komplexes Datenmodell (siehe auch Abschnitt 2.4.2). Daher wurden in den vergangenen Jahren diverse Anfragesprachen entwickelt, welche sich für eine einfache Abfrage der in einem IFC-Modell enthaltenen Informationen eignen. Entsprechende Ansätze, wie etwa bei Daum (2018); Mazairac (2015), wurden bereits in Kapitel 6 vorgestellt.

Der Zugriff auf die Informationen eines Datenmodells über eine Programmiersprache kann einfach gestaltet werden, wenn das Modell dieses auch zulässt. Bei einer hohen Anzahl von Entitäten und Datentypen eines Datenmodells müssen diese auch in Form von Methoden oder Operatoren anwendbar und zugreifbar zur Verfügung stehen, was schnell zu einem Überangebot für den Nutzer führen kann.

Für die VCCL wird auf eine direkte Verwendung des IFC-Datenschemas verzichtet. Vielmehr wird eine Zwischenstufe in Form eines Modells eingeführt, welche sich stark an das IFC-Datenmodell anlehnt. Die hierarchische Struktur der Bauteil-Entitäten, die Speicherung der Attribute in Attributsätzen sowie die objektifizierten Beziehungen (vgl. *IfcPropertySet*, *IfcProperty* und *IfcRelationship* in Abschnitt 2.4.2.2) werden aus dem Datenmodell der IFC übernommen. Lediglich auf die Verwendung von inversen Attributen wird verzichtet, um die Komplexität des Datenmodells gering zu halten.

Diese Informationen werden stattdessen direkt mit den betroffenen Entitäten verknüpft und als Referenzen gespeichert. Das Referenzieren erfolgt über die Speicherung der *GUID* der jeweilig betroffenen Elemente untereinander. Dieser Mechanismus erlaubt die Definition von Attributsätzen, wie diese auch in IFC definiert sind, darüber hinaus allerdings auch die Definition von Beziehungen, welche für die VCCL wesentliche Werkzeuge für die Ableitung von Informationen ist und in den folgenden Abschnitten im Näheren vorgestellt wird.

Wesentliches Ziel ist es, die zu verarbeitenden Daten ohne Verluste aus dem IFC-Modell extrahiert und in eine vereinfachte Form gebracht werden können, die welche direkt von der VCCL verarbeitet werden kann. Auf diese Weise müssen vom Nutzer keine zusätzlichen Prozessschritte für die Extraktion von Modelldaten definiert werden.

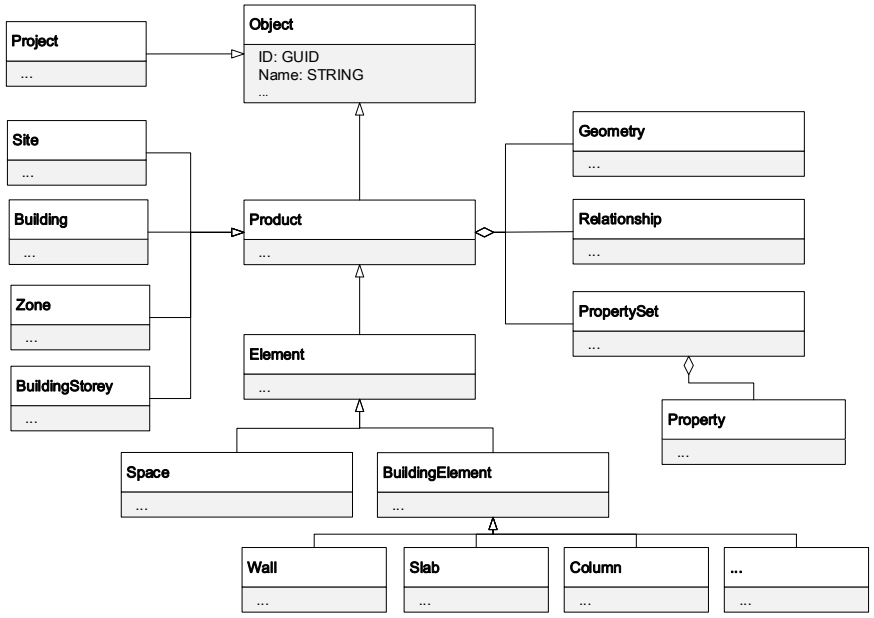


Abbildung 7.9: Vereinfachte Darstellung des VCCL-Datenmodells in UML

Wichtig ist, dass das resultierende Datenmodell der VCCL als Zwischenspeicher der Bauwerksinformationen angesehen werden muss, denn die Bauwerksinformationen werden lediglich intern für die VCCL verarbeitet und so für den Zugriff der einzelnen Methoden und die jeweilige Prüfung entsprechend aufbereitet. Das Ziel hinter dem Datenmodell besteht also ausschließlich darin, die Bauwerksinformationen so zur Verfügung zu stellen, dass die Methoden der VCCL direkt auf die notwendigen Daten Zugriff haben. Eine auch nur annähernd weite Anwendungsvielfalt, wie diese bei der IFC gegeben ist, wird damit also explizit nicht verfolgt. Das resultierende VCCL-Datenmodell ist schematisch in Abbildung 7.9 dargestellt.

7.5.3 Atomare Methoden der VCCL

Wie bereits im vorigen Abschnitt beschrieben, bietet die VCCL eine wohldefinierte Menge atomarer Methoden. Diese Methoden stellen die Basisebene der VCCL-Bibliothek dar. Die Verarbeitungsprozesse, welche innerhalb dieser Methoden durchgeführt werden, können von dem Nutzer nicht eingesehen werden. Alle VCCL-Programme lassen sich auf Verknüpfungen dieser atomaren Methoden zurückführen.

Um übermäßig komplexe VCCL-Programme zu vermeiden, können in einem Methodenknoten der VCCL mehrere Operatoren zusammengefasst und zusätzliche Optionen für Nutzereingaben angeboten werden. Hierzu werden in einem Knoten UI-Felder eingebettet, die dem Nutzer die Möglichkeit bieten, gewünschte Einstellungen und Nutzereingaben vorzunehmen. Auf diese Weise können mehrere Operationen bzw. Operatoren in einem einzelnen VCCL-Knoten zusammengefasst werden. Zusätzliche Benutzereingaben beispielsweise über globale Eingangsports können so vermieden werden, wodurch sich die Anzahl der Knoten und Kanten deutlich reduziert. Als Folge sinkt die Komplexität des VCCL-Programms.

Um dieses zu ermöglichen, ist die Spezifikation der Eingabe- und Ausgabeports einer VCCL-Methode abhängig von der jeweils gewählten Einstellung. Hierbei gilt strikt, dass in der VCCL für jeden Knoten und für jede Benutzereinstellung wohl definiert ist, welche Datentypen über die Ports weiterverarbeitet werden können. In Abhängigkeit der gewählten Einstellungen in der Methode kann die Auswahl der Eingangs- und Ausgangsports und somit der Datentypen variieren. Ist eine VCCL-Methode bereits als Teil eines Programms mit einer Kante verbunden und wird die Einstellung der Methode durch Benutzereingabe so verändert, dass sich der spezifizierte Datentyp des verknüpften Ports verändert, wird die ungültige Verbindung automatisch aufgelöst.

Der aktuelle Katalog (siehe Abbildung 7.19) wurde im Rahmen von mehreren Fallstudien und Anwendungsbeispielen (Preidel und Borrmann, 2015, 2016; Preidel et al., 2017) sowie auf Basis von Referenzliteratur entwickelt (Kim et al., 2017b; Lee et al., 2016; Solihin, 2016; Uhm et al., 2015). Der aktuelle Stand stellt keinen finalen Stand dar, sondern gilt vielmehr als Ausgangsbasis und Status quo, welcher sich an den adressierten Anwendungsfällen aus Kapitel 8 orientiert. Die Entwicklung und Einführung weiterer atomarer Methoden ist ausdrücklich erwünscht und beabsichtigt, da nur so die Ausdruckskraft der Sprache für weitere Anwendungsfälle vergrößert werden kann.

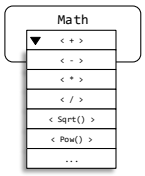
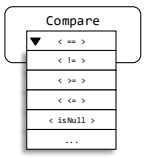
Es muss allerdings bei jeder Erweiterung darauf geachtet werden, dass die hinzugefügte Methode für die Ausdruckskraft der Sprache einen einzigartigen Mehrwert darstellt. Für eine lang anhaltende, positive Nutzererfahrung ist es essenziell, dass die Übersicht über und das Verständnis der einzelnen Methoden auf lange Sicht gewahrt werden kann. Dieses kann nur erreicht werden, wenn die Bibliothek der VCCL sich auf eine minimal notwendige Anzahl an Methoden beschränkt. Lässt sich eine Methode durch einzelne oder durch die Komposition mehrerer bereits vorhandener atomarer Methoden darstellen, so ist die Einführung redundant und nicht gerechtfertigt. In diesem Falle kann die neue Methode, wie bereits in Abschnitt 7.5.1 vorgestellt, als Verschachtelung abgebildet und in einem VCCL-Programm wiederverwendet werden.

Im Folgenden werden die atomaren VCCL-Methoden zusammengefasst und in entsprechenden Kategorien vorgestellt. Hierfür werden die Methoden jeweils in Tabellen mit einer Beschreibung, den Eingangs- als auch Ausgangsdatentypen sowie den Auswahlmöglichkeiten, welche vom Nutzer über die Eingabefelder in dem Knoten ausgewählt werden können, aufgeführt. Zusätzlich sind die graphischen Notationen der Methoden in den Tabellen dargestellt.

Methoden für die Anwendung von arithmetischen Operatoren und Vergleichsoperatoren

In regulatorischen Dokumenten finden sich oftmals Anforderungen, die mit Hilfe von Formeln oder Gleichungen beschrieben werden. Daher sind grundlegende Rechen- und Vergleichsoperationen für die Abbildung der regulatorischen Inhalte und für eine Aufbereitung bzw. Weiterverarbeitung von Bauwerksinformationen essenziell. Für diese Operationen bietet die VCCL jeweils eigene Methoden (siehe Tabelle 7.2). Um die Anwendung dieser Methoden zu verdeutlichen, findet sich in Abbildung 7.10 beispielhaft ein VCCL-Programm.

Tabelle 7.2: VCCL-Methoden für die Anwendung arithmetischer Operatoren und Vergleichsoperatoren

Notation	Beschreibung	Eingang	Auswahlmöglichkeiten	Ausgang
	Wendet einen arithmetischen Operator an	mehrere Zahlenwerte [$double$, $integer$]	Auswahl des arithmetischen Operators: Addition [+], Subtraktion [-], Multiplikation [$\times$], Division [$\div$], Radizieren [$\sqrt{}$], Potenzieren [*Pow()*], Minimum [*Min()*], Maximum [*Max()*]	ein Zahlenwert [$double$, $integer$]
	Wendet einen logischen Vergleichsoperator an: Zwei Werte werden miteinander verglichen und als Ergebnis ein Boolescher Wert ausgegeben	zwei Zahlenwerte [$double$, $integer$]; (Un-)Gleichheit kann für alle Datentypen geprüft werden	Auswahl des Vergleichsoperators: Gleich [=], Ungleich [$\neq$], Größer als [>], Kleiner als [<], Größer oder gleich [$\geq$], Kleiner oder gleich [$\leq$]	ein Boolescher Wert [$bool$]

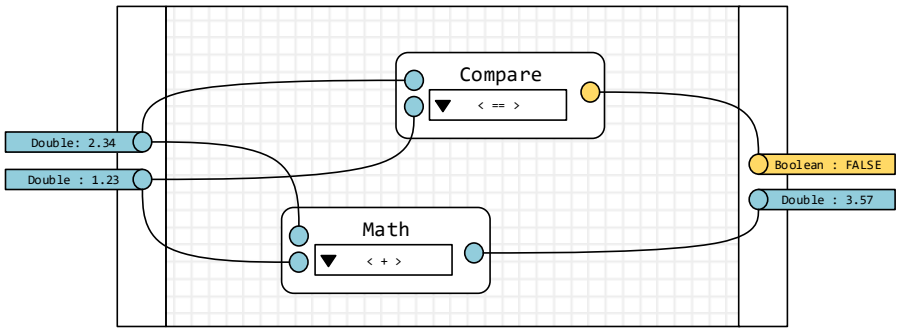


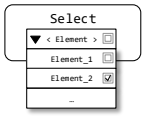
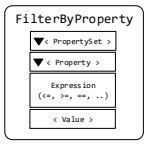
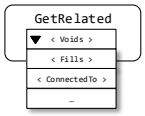
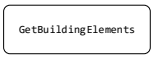
Abbildung 7.10: Beispielhafte Anwendung der VCCL-Methoden für arithmetische und Vergleichsoperatoren

Methoden für das Selektieren/Filtern von Bauteilen und den Informationszugriff

Zu jedem Zeitpunkt und Prozessschritt soll die VCCL dem Nutzer die Möglichkeit bieten, Objekte nach spezifischen Kriterien zu filtern sowie zu selektieren und auf gewünschte Informationen des Bauwerkmodells zuzugreifen. Hierfür bietet die VCCL Methoden, die den Zugriff auf und Umgang mit dem bereits in Abschnitt 7.5.1 beschriebenen Datenmodell erlauben (siehe Tabelle 7.3). Wie bereits

bei der Vorstellung des Datenmodells der VCCL erläutert wurde, wird dabei auf die von den IFC bekannte Spezifikation der Attribute zurückgegriffen (siehe auch Abschnitt 2.4.2.2), nach welcher jedes Element mehrere Attributsätze (*PropertySet*) beinhalten kann, welche wiederum diverse thematisch zusammengehörige Attribute (*Property*) mit jeweils spezifizierten Datentypen zusammenfasst. Eine beispielhafte Anwendung der vorgestellten VCCL-Methoden ist in Abbildung 7.11 zu finden.

Tabelle 7.3: VCCL-Methoden für das Selektieren und Filtern von Bauteilen

Notation	Beschreibung	Eingang	Auswahlmöglichkeiten	Ausgang
	Filtert die Bauteilliste nach einem oder mehreren Bauteiltypen	Liste vom Datentyp [$BuildingElement$]	Selektion eines oder mehrerer Bauteiltypen (z.B. Wand [*Wall*], Stütze [*Column*], Raum [*Space*])	Liste vom Datentyp [$BuildingElement$]
	Selektiert Bauteile aus einer Liste gemäß einer manuellen Eingabe	Liste vom Datentyp [$BuildingElement$]	Auswahl von Bauteilen	Liste vom Datentyp [$BuildingElement$]
	Filtert nach Bauteilen, die die spezifizierten Anforderungen hinsichtlich derer Attribute erfüllen	Liste vom Datentyp [$BuildingElement$]	• *PropertySet*: Name des Attributsatzes • *Property*: Name des Attributs • *Expression*: In diesem Feld kann der Nutzer einen Vergleichsoperator auswählen, um eine Randbedingung für das ausgewählte Attribut zu spezifizieren. • *Value*: Wird ein Vergleichsoperator genutzt, dann kann in diesem Feld ein Wert eingegeben werden, welcher sich auf den Vergleichsoperator bezieht	Liste vom Datentyp [$BuildingElement$]
	Greift auf Bauteile, die mit den eingegebenen Bauteilen in einer spezifizierten Beziehung stehen, zu	Liste vom Datentyp [$BuildingElement$]	Auswahl des Beziehungstyps in Anlehnung an die *IfcRelationships*, welche in Abschnitt 2.4.2.2 vorgestellt wurden; beispielsweise *Voids*, *Fills*, *Decomposes*, *ConnectedTo*	Liste vom Datentyp [$BuildingElement$]
	Greift auf ein spezifiziertes Attribut eines Objekts zu	eine Entität oder eine Liste des Datentyp [$BuildingElement$]	• *PropertySet*: Name des Attributsatzes • *Property*: Name des Attributs • *BuildingElement*: Falls der Eingangswert eine Liste vom Datentyp [$BuildingElement$] ist, kann über dieses Feld das Element ausgewählt werden, für welches das Attribut bezogen werden soll.	ein Wert vom Datentyp des jeweiligen, spezifizierten Attributs
	Greift auf die Bauteilelemente zu, welche in einem Bauwerksmodell gespeichert sind	ein Element vom Datentyp [$BuildingModel$]	-	Liste vom Datentyp [$BuildingElement$]

Hilfsmethoden

Eine zentrale Stärke von visuellen Programmiersprachen ist, dass der eigentliche Verarbeitungsprozess und dessen Zwischenschritte sichtbar gemacht werden können. Im Zusammenhang mit den Bauwerksinformationen, die in einem VCCL-Programm typischerweise verarbeitet werden, wird diese Stärke optimal genutzt.

Für eine unmittelbare Visualisierung von Zwischenergebnissen bietet die VCCL Methoden (siehe Abbildung 7.4), die sowohl alphanumerische als auch geometrische Informationen sichtbar machen. Hierfür kann jeder Ausgabe-Port eines VCCL-Programms mit einem Eingabe-Port einer Hilfsmethode verknüpft werden, um die Informationen, die an diesem Prozesspunkt verarbeitet wurden, einzusehen. So können Eingabedaten, Zwischenergebnisse oder Endergebnisse präsentiert und vom Benutzer überprüft werden. Ein entsprechendes Beispiel ist in Abbildung 7.11 dargestellt.

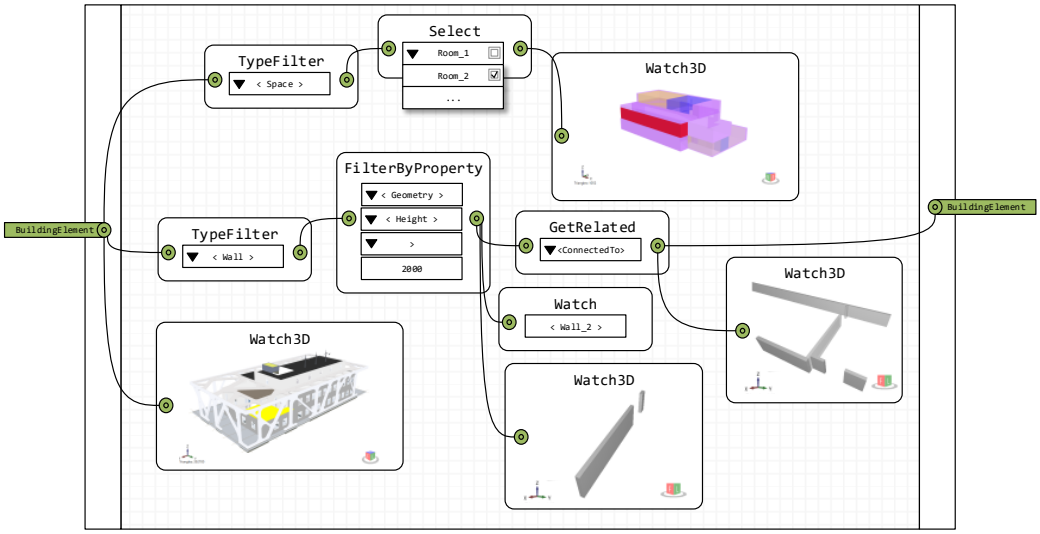
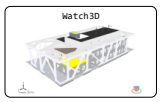


Abbildung 7.11: Beispielhafte Anwendung von Methoden zum Filtern und Selektieren von Bauteilen sowie Hilfsmethoden für die Visualisierung von geometrischen und alphanumerischen Informationen

Neben der Visualisierung können Zwischenergebnisse mit Hilfe der VCCL als BCF-Objekt (siehe Abschnitt 2.5.3) kommuniziert werden. Hierfür kann der Nutzer die betroffenen Bauteile an die *Communication*-Methode als Eingangswert weitergeben und in der Methode selbst zusätzlich einen Blickwinkel sowie weitere Angaben wie etwa einen Namen und eine Beschreibung des *Issue* spezifizieren. Wird das VCCL-Programm bis zu dieser Methode ausgewertet, so wird ein *BCF*-Objekt mit den entsprechenden Angaben automatisch auf einem *BCF*-Server, welcher für die Umgebung vorkonfiguriert sein muss, gespeichert. Ein Beispiel für einen solchen Server wird entsprechend bei der Vorstellung des Prototyps genannt (siehe Abschnitt 7.6). Ein Beispiel für die Anwendung dieser Methode ist in Abbildung 7.12 dargestellt.

Da diese Methoden lediglich ein Hilfsmittel für die Visualisierung der Zwischenergebnisse sind und somit nicht zwingender Teil des Programms, werden diese als Hilfsmethoden bezeichnet.

Tabelle 7.4: VCCL-Hilfsmethoden für die Anzeige und Visualisierung von Zwischenergebnissen

Notation	Beschreibung	Eingang	Auswahlmöglichkeiten	Ausgang
	Zeigt den Informationsgehalt der jeweiligen Eingangsgröße als Text an.	alle Datentypen	-	-
	Visualisiert den Informationsgehalt der Eingangsgröße, wenn diese geometrische Information enthält	Liste vom Datentyp [*BuildingElement*]	-	-
	Erstellt eine BCF-*Issue*	Liste vom Datentyp [*BuildingElement*]	• *3D Ansicht*: In der Ansicht kann der Nutzer einen Blickwinkel einstellen, welcher dann für das BCF-Kommunikationsobjekt übernommen wird • *Issue Name*: In der Textbox kann der Nutzer den Namen des *Issue* eingeben • *Description*: Textbox für weiteren Beschreibungstext des *Issue*	-

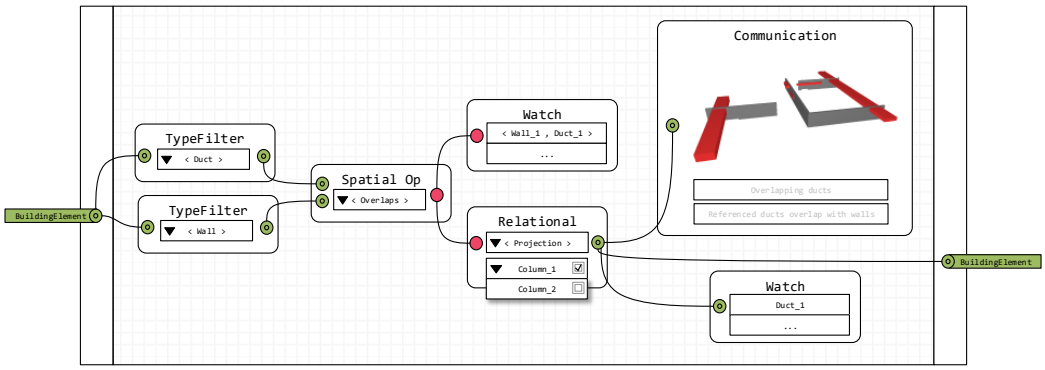


Abbildung 7.12: Anwendungsbeispiel für die *Communication*-Methode> Alle Leitungsobjekte (*Duct*) für die eine Überschneidung mit Wandbauteilen (*Wall*) identifiziert werden konnte, werden an die Methode weitergegeben. In der Oberfläche der Methode kann der Anwender zusätzlich einen Blickwinkel sowie weitere Informationen wie Name und Beschreibung angeben

Methode für den Umgang mit Listen

Im Kontext der Bauwerksmodelle, ist es sehr oft erforderlich, mit mehrdimensionalen Werten, wie insbesondere Listen von Bauteilen, umzugehen. Konsequenterweise bietet die VCCL die Methode *Set*, welche mehrere Operationen für die Weiterverarbeitung von mehrdimensionalen Werten und Informationen zusammenfasst und in Tabelle 7.5 dargestellt ist. Des Weiteren findet sich eine beispielhafte Anwendung dieser Methode als Teil eines VCCL-Programms in Abbildung 7.13.

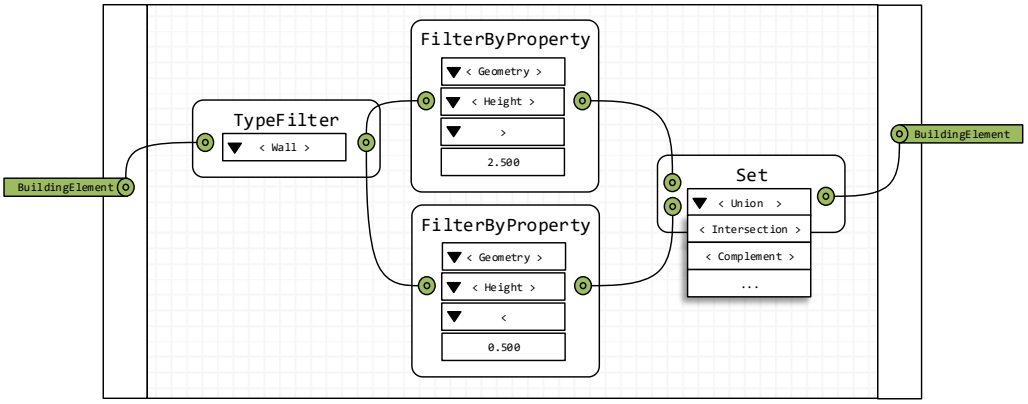
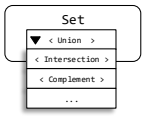


Abbildung 7.13: Beispielhafte Anwendung von Methoden für die Weiterverarbeitung von Listen

Tabelle 7.5: VCCL-Methode für die Verarbeitung von Listen

Notation	Beschreibung	Eingang	Auswahlmöglichkeiten	Ausgang
	Wendet einen Mengenoperator an	zwei Listen vom gleichen Datentyp; z. B. Listen vom Datentyp [$BuildingElement$]	Auswahl des Mengenoperators: *Vereinigung* [$\cup$], *Schnitt* [$\cap$], *Differenz* [$\setminus$]	eine Liste vom Datentyp der Eingangswerte

Relationale Methoden

Das mathematische Konzept einer Relation hat sich in den Computerwissenschaften als nützlich erwiesen und spielt auch bei der Darstellung und Abfrage von hochstrukturierten Daten, wie dieses typischerweise bei BIM-Modellen der Fall ist, eine wesentliche Rolle. Aktuell basieren viele, weitverbreitete Datenbankmanagementsysteme (DBMS) auf Relationen und werden daher auch als Relationale DBMS bezeichnet. Diese Datenbanktechnologie stellt derzeit den de-facto-Standard dar. Auch semantische Web-Technologien wie RDF und OWL, welche bereits in Abschnitt 6.4 vorgestellt wurden, basieren in erster Linie auf dem mathematischen Konzept der Relation.

Grundlage für die Abfrage relationaler Datensätze ist die relationale Algebra (Codd, 1970). Formal wird eine Relation R in der Mathematik als eine Teilmenge des kartesischen Produkts von n Domänen definiert (Kemper und Eickler, 2011):

$$R \subseteq D_1 \times D_2 \times ... \times D_n$$

Anschaulicher kann eine Beziehung als eine Menge von n-Tupeln beschrieben werden, bei denen jedes Tupel die Objekte, die in einer bestimmten Beziehung zueinander stehen, miteinander kombiniert. Das bedeutet, dass eine Relation Elementen einer Menge jeweils Elemente einer anderen Menge zuweist und dadurch eine Beziehung zwischen den Mengen spezifiziert. Für das bessere Verständnis kann eine Relation auch schematisch wie in Abbildung 7.14 dargestellt werden. Kern der relationalen Algebra sind die Operationen, die auf Relation definiert sind. Dabei sind insbesondere die Selektion, die Projektion und der Verbund von großer praktischer Bedeutung.

	1	2	3	4	5	6
A	1	0	0	1	0	1
B	0	1	0	0	1	0
C	0	0	1	0	0	1
D	1	0	0	1	0	0
E	0	1	0	0	1	0
F	1	0	1	0	0	1

1	{A,D,F}
2	{B,E}
3	{C,F}
4	{A,D}
5	{B,E}
6	{A,C,F}

	1	2	3	4	5	6
A	X			X		X
B		X			X	
C			X			X
D	X			X		
E		X			X	
F	X		X			X

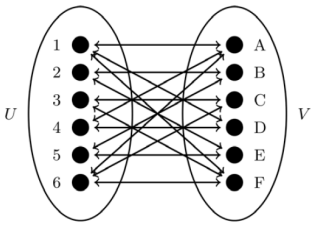


Abbildung 7.14: Vier unterschiedliche schematische Darstellungen der gleichen Relation (Kemper und Eickler, 2011; Preidel et al., 2017)

In der VCCL kommen Relationen auf diverse Art und Weise zur Anwendung (siehe Tabelle 7.6):

Mit Hilfe von Relationen kann die Beziehung zwischen Objekten oder Werten als Ergebnis abgebildet werden. Der wesentliche Vorteil ist, dass das Wissen über die Beziehung von mehrdimensionalen Werten auf diese Weise nicht verloren geht, sondern gesichert wird und anschließend auch weiterverarbeitet werden kann. Daher verwenden diverse Methoden der VCCL Relationen als Ausgabewert.

Hierbei wird in der Regel eine Relation aus mindestens zwei Eingabewerten erzeugt, für welche jeweils geprüft wird, ob ein oder mehrere spezifische Kriterien erfüllt sind. Wenn die Kriterien erfüllt sind, stehen die Elemente in einer definierten Beziehung zueinander, und folglich wird die Beziehung als n-dimensionales Tupel (zum Beispiel als Paar oder Tripel) der Ergebnisrelation hinzugefügt.

Zu den Methoden, welche Relationen erzeugen, zählen beispielsweise die geometrisch-räumlichen Methoden, welche im folgenden Abschnitt vorgestellt werden. Ein weiteres Beispiel ist die VCCL-Methode *PropertyRelation*, welche jedem Bauteil einer Liste deren Attribut in einer Relation zuordnet. Das Attribut kann dabei vom Nutzer selbst spezifiziert werden. Ein möglicher Anwendungsfall für diese Methode ist in Abbildung 7.16 dargestellt worden.

Um die erzeugten Relationen weiterzuverarbeiten, bietet die VCCL eine Methode für die Anwendung der Operatoren der relationalen Algebra, welche auch in der Abfragesprache SQL zur Anwendung kommen. Diese Operatoren sind mächtige Werkzeuge, da sie auf der strikten mathematischen Grundlage der relationalen Algebra basieren und eine fehlerfreie Abfrage bzw. Verarbeitung auch sehr großer Mengen von Informationen garantieren. Schematische Darstellungen der einzelnen Operationen für die Manipulation und Abfrage von Relationen sind in Abbildung 7.15 dargestellt (Codd, 1970, 1991).

Viele Abfragesprachen, die mit Relationen und relationalen Operatoren arbeiten, folgen dem deklarativen Programmierparadigma, wie beispielsweise SQL. Im Gegensatz hierzu arbeiten VPLs in einzelnen Prozessschritten und verfolgen somit ein imperatives Paradigma mit der sequentiellen Ausführung einzelner

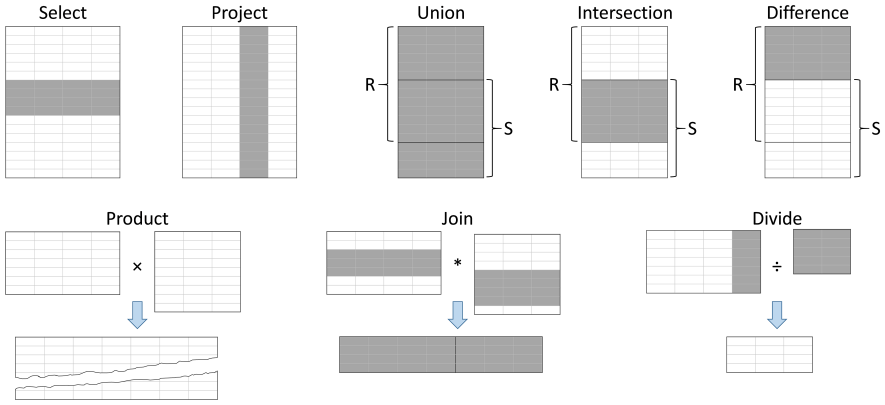


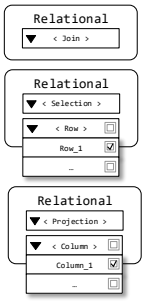
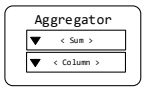
Abbildung 7.15: Schematische Darstellung der Operatoren der relationalen Algebra (Preidel et al., 2017)

Informationsverarbeitungsbefehle. Es ist allerdings ein allgemeines Missverständnis, dass die relationale Algebra ausschließlich in einem deklarativen Programmierkontext angewendet werden kann. Die Operatoren können als einzelne Verarbeitungsschritte eines sequentiellen Prozesses ausgeführt werden und stellen somit auch für die VCCL einen enormen Mehrwert dar (Preidel et al., 2017). Die Operatoren werden in der VCCL in einer Methode zusammengefasst, um so Relationen weiterzuverarbeiten (Kemper und Eickler, 2011). Ein Anwendungsbeispiel der Methode ist in Abbildung 7.18 dargestellt.

In einer weiteren Methode der VCCL werden Aggregationsoperatoren bzw. Aggregationsfunktionen, welche eine weitere Klasse von Operatoren der relationalen Algebra bilden (Grabisch et al., 2009) und auf dem Prinzip der relationalen Vollständigkeit basieren, zusammengefasst. Im Prinzip stellt ein solcher Operator eine Funktion f mit einer Beziehung R als Argument, die einen einzelnen Wert in einem definierten Bereich zurückgibt, dar. Für Relationen können diverse Funktionen spezifiziert werden, um beispielsweise die Summe, das Maximum, das Minimum, das arithmetische Mittel oder die Anzahl der Elemente einer Relation zu bestimmen.

Ein Anwendungsbeispiel für einen Aggregationsoperator ist in Abbildung 7.16 dargestellt. In dem Beispiel sind in der Relation für alle Wandbauteile die entsprechenden Höhen an zweiter Stelle des Tupel gespeichert. Der Aggregationsoperator greift nun für jedes Element der Relation auf die zweite Stelle des Tupel [*BuildingElement*; *double*] zu und summiert die Werte auf. Auf diese Weise werden alle Höhenwerte der Bauteile aufsummiert und als Ergebnis ausgegeben.

Tabelle 7.6: VCCL-Methoden für den Umgang und die Weiterverarbeitung von Relationen

Notation	Beschreibung	Eingang	Auswahlmöglichkeiten	Ausgang
	Erzeugt eine Relation für eine Liste von Bauteilen mit dem zugehörigen, selektierten Attribut	Liste vom Datentyp [*BuildingElement*]	• *PropertySet*: Name des Attributsatzes • *Property*: Name des Attributs	eine Relation; [*Relation*]
	Wendet einen Operator der relationalen Algebra an	zwei Relationen; [*Relation*]	Auswahl des relationalen Operators: • *Join* • *Projection*; hier muss zusätzlich vom Nutzer selektiert werden, auf welche Zeilenelemente die Operation angewendet wird • *Selection*; hier muss zusätzlich vom Nutzer selektiert werden, auf welche Spaltenelemente die Operation angewendet wird	eine Relation; [*Relation*]
	Wendet einen Aggregationsoperator an	eine Relation; [*Relation*]	Auswahl der Aggregationsfunktionen: Aufsummieren [*Sum*], Durchschnitt [*Avg*], Minimum [*Min*], Maximum [*Max*], Anzahl [*Count*]	Datentyp hängt von der Aggregationsfunktion ab

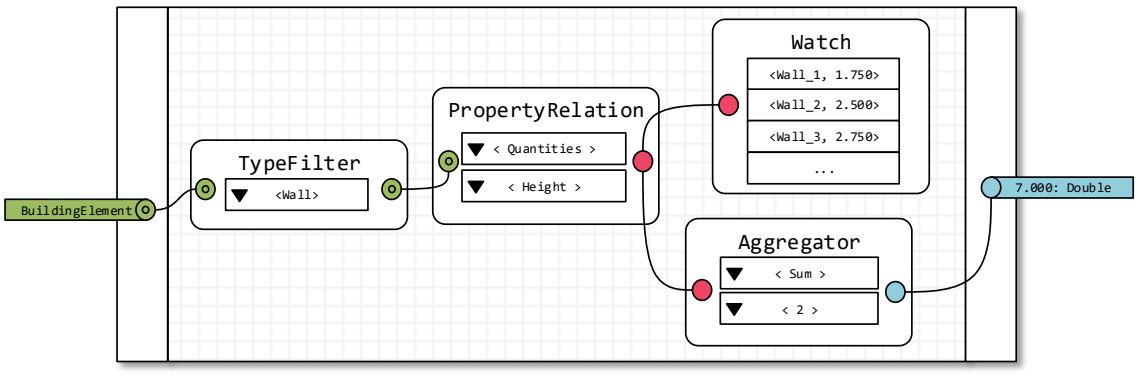


Abbildung 7.16: Beispielhafte Anwendung einer Aggregationsoperation mit Hilfe der VCCL-Methode *Aggregator*

Geometrisch-Räumliche Methoden

Diverse regulatorische Anforderungen und ingenieurtechnische Fragestellungen beziehen sich auf die geometrische Form oder räumliche Positionierung physischer Objekte eines Bauwerks. Um geometrisch-räumliche Verhältnisse formal abbilden zu können, ist es notwendig, den Zugriff auf und die Weiterverarbeitung von geometrischen Informationsgehalt von Bauwerksmodellen mit Hilfe von entsprechenden Operatoren zu ermöglichen. Borrmann (2007) unterteilt räumliche Operatoren grundlegend in drei Kategorien. Im Folgenden sind diese gemeinsam mit Operatoren, die in Anlehnung an (Borrmann, 2007) von den Methoden der VCCL abgedeckt werden, aufgeführt:

Metrische Operatoren werden verwendet, um die Abmessungen eines Objekts oder den Abstand zwischen zwei Objekten **A** & **B** zu bestimmen.

Anwendung: Bestimme die { kürzeste Distanz im dreidimensionalen Raum [*3D ShortestDistance*] | kürzeste horizontale Distanz [*Horizontal Distance*] | kürzeste vertikale Distanz [*Vertical Distance*] } zwischen **A** und **B**.

Direktionale Operatoren werden verwendet, um die direktionale Beziehung zweier Objekte **A** & **B**, d.h. deren relative Position zueinander, zu bestimmen.

Anwendung: **A** befindet sich { östlich [*EastOf*] | westlich [*WestOf*] | nördlich [*NorthOf*] | südlich [*SouthOf*] | oberhalb [*Above*] | unterhalb [*Below*] } von **B**.

Topologische Operatoren werden verwendet, um die topologische Beziehung zweier Objekte **A** & **B** zu bestimmen. Eine Auswahl topologischer Operatoren ist in Abbildung 7.17 dargestellt.

Anwendung: **A** und **B** { sind identisch [*Equal*] | haben keinen Zusammenhang [*Disjoint*] | berühren sich [*Touch*] | überlappen sich [*Overlap*] }. **A** { beinhaltet **B** [*Contains*] | befindet sich innherhalb von **B** [*Within*] }.

Bei der Definition der entsprechenden räumlichen Operatoren lehnt sich die VCCL stark an die zuvor genannten Kategorien an. Allerdings werden der Einfachheit halber alle Operatoren, deren Prüfung einen Booleschen Wert als Ergebnis erzeugt, in der Methode *Spatial* zusammengefasst. Die Auswahl des

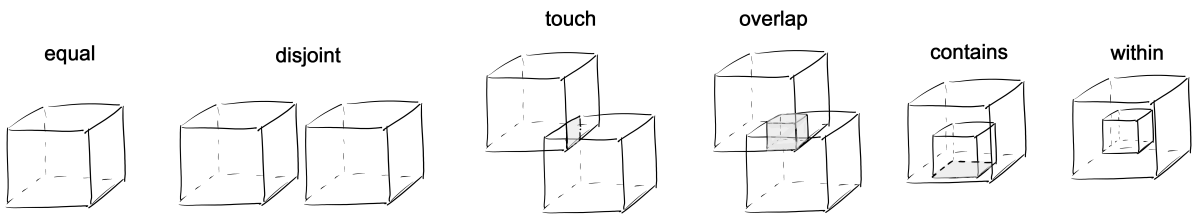
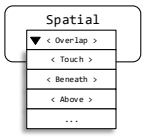
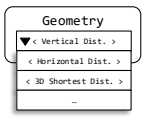


Abbildung 7.17: Schematische Darstellung topologische Operatoren; angelehnt an (Borrmann, 2007; Borrmann et al., 2009; Daum, 2018)

jeweiligen Operators wird, wie bereits bekannt, entsprechend dem Nutzer in einem Eingabefeld zur Wahl gestellt. Als Ergebnis produziert die Methode eine Relation, die alle Bauteilpaare enthält, welche hinsichtlich des gewählten direktionalen bzw. topologischen Operators positiv getestet wurden.

Entsprechend sind die metrischen Operatoren in der VCCL-Methode *Geometry* getrennt. Als Ergebnis produziert die Methode ein dreidimensionales Tupel, welche für jedes geprüfte Bauteilpaar die berechnete Entfernung entsprechend dem gewählten metrischen Operators vorhält.

Tabelle 7.7: VCCL-Methode für die Anwendung von geometrisch-räumlichen Methoden

Notation	Beschreibung	Eingang	Auswahlmöglichkeiten	Ausgang
	Wendet einen topologischen oder direktionalen Operator an	zwei Listen vom Datentyp [*BuildingElement*]	Auswahl eines topologischen bzw. direktionalen Operators: [*EastOf*], [*WestOf*], [*NorthOf*], [*SouthOf*], [*Above*], [*Below*], [*Equal*], [*Disjoint*], [*Touch*], [*Overlap*], [*Contains*], [*Within*]	eine Relation [*Relation*], die alle Bauteilpaare enthält, für welche die Prüfung des gewählten räumlichen Operators *WAHR* ergeben hat
	Wendet einen metrischen Operator an	zwei Listen vom Datentyp [*BuildingElement*]	Auswahl eines metrischen Operators: [*3D ShortestDistance*], [*Horizontal Distance*], [*Vertical Distance*]	eine Relation [*Relation*], die für alle geprüften Bauteilpaare, die Bauteile selbst sowie das resultierende metrische Ergebnis hält

Ein Beispiel für die Anwendung räumlicher Operatoren der VCCL ist in Abbildung 7.18 dargestellt. Ziel dieses VCCL-Programms ist es, die Beziehung zwischen Einbau- und Wandbauteilen zu bestimmen. Hierfür werden zunächst die Öffnungselemente, die Wandbauteile sowie sämtliche Bauteile generisch über Typenfilter-Methoden aufbereitet. Mit Hilfe von topologischen Operatoren kann anschließend geprüft werden, (1) welches Öffnungselement innerhalb welches Wandbauteils und (2) welches Einbauteil innerhalb welches Öffnungselements liegen. Die Ergebnisse werden entsprechend in Relationen festgehalten und können anschließend zunächst mit einer *Join* Operation zusammengeführt werden, um dann mit einer *Projektion* auf die Auswahl der gewünschten Bauteilpaare als Relation [*Einbauteil-Wandbauteil*] reduziert zu werden, welche die Beziehung zwischen den Einbauteilen zu den Wandbauteilen beschreibt.

Bei der Implementierung der räumlichen Operatoren wurde innerhalb der VCCL mit Genehmigung der Autoren auf die Vorarbeiten von Daum (2018); Daum und Borrmann (2014) zurückgegriffen (siehe auch Abschnitt 7.6). Quellcode 7.3 zeigt in Pseudocode-Darstellung, wie die interne Verarbeitung der räumlichen VCCL-Operatoren abläuft.

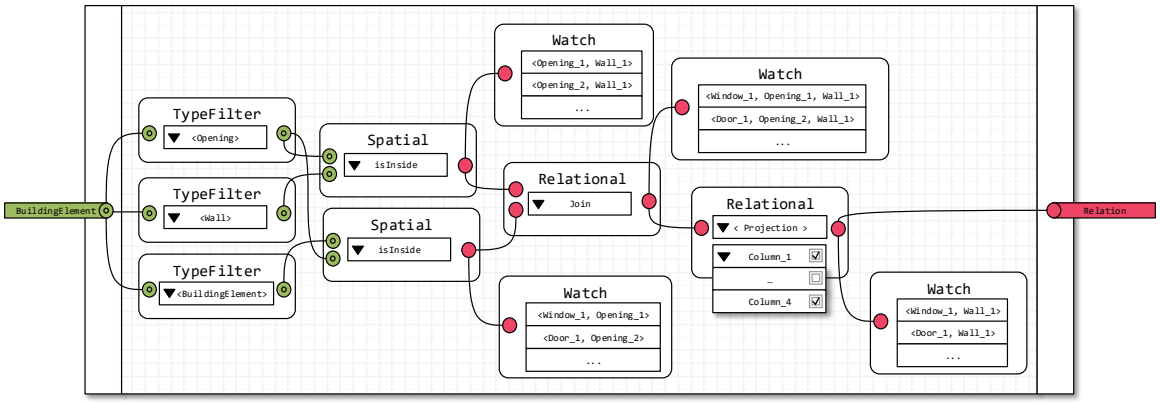


Abbildung 7.18: Beispielhafte Anwendung räumlicher Operatoren in einem VCCL-Programm

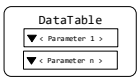
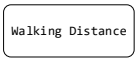
```
Process (INPUT[] Bauteile_1, INPUT[] Bauteile_2)
{
  RELATION[] result;
  foreach(bauteil_1 IN Bauteile_1)
  {
    foreach(bauteil_2 IN Bauteile_2)
    {
      if (IsInside(bauteil_1.Geometry, bauteil_2.Geometry) == TRUE)
      {
        result.ADD([bauteil1.ID,bauteil2.ID])
      }
    }
  }
  return result;
}

```

Quellcode 7.3: Interne Verarbeitung des topologischen Operator *IsInside* der VCCL in Pseudocode

Methoden für spezifische Anwendungsfälle

Bedingt durch die Vielfalt der Darstellungsweisen regulatorischer Anforderungen, muss die Ausdruckskraft der VCCL sukzessive durch die Einführung weiterer atomarer Methoden erweitert werden. Die in Tabelle 7.8 aufgeführten atomaren Methoden ergeben sich aus den in Kapitel 8 im Detail vorgestellten Fallstudien. Diese sollen an dieser Stelle zur Vollständigkeit aufgeführt werden. Für eine detaillierte Darstellung der Funktionsweise sei an dieser Stelle entsprechend auf das folgende Kapitel verwiesen.

Tabelle 7.8: VCCL-Methoden für den Daten- und Informationszugriff

Notation	Beschreibung	Eingang	Auswahlmöglichkeiten	Ausgang
	Erzeugt Freiflächen-Körper vor und hinter beweglichen Bauteilen, wie insbesondere Fenster und Türen	Liste vom Typ [*BuildingElement*]	Dimensionen der Freiflächenkörper (Breite, Höhe, Tiefe)	Liste vom Typ [*BuildingElement*] (*Die erzeugten Freiflächenkörper werden als Raumobjekt behandelt*)
	Fügt einem Bauteil ein gewähltes Attribut hinzu	Liste vom Typ [*Building Element*]	• *PropertySet*: Name des Attributsatzes • *Property*: Name des Attribut • *Value*: Wert, der eingetragen werden soll	-
	Zugriff auf eine Datentabelle, die beispielsweise Teil einer regulatorischen Anforderung ist	Parameter, die für die Selektion des jeweiligen Wertes, der aus der Datentabelle als Anforderung entnommen werden soll, erforderlich sind	Eingabe der Parameter, welche für das Auslesen erforderlich sind	Datentyp des jeweiligen Ausgabewerts
	Berechnung der Laufwegdistanz zwischen mehreren Objekten auf einer Geschossebene	• *Quellobjekte*: Liste vom Typ [*Building Element*] • *Zielobjekte*: Liste vom Typ [*Building Element*]	-	eine Relation [*Relation*], die das jeweilige End- und Zielobjekt sowie die errechnete Distanz enthält

Die in den vorigen Abschnitten vorgestellten VCCL-Knoten sind zusammenfassend in Abbildung 7.19 dargestellt.

7.5.4 Kontrollstrukturen

Die Anforderungen komplexer Konformitätsprüfverfahren erfordern, dass die VCCL auch über die Möglichkeit verfügt, Kontrollflüsse zu definieren. Die meisten imperativen Programmiersprachen bieten zumindest einen grundlegenden Einsatz von Kontrollflusselementen, wie etwa konditionale Verzweigungen oder Iterationen. In textuellen Programmiersprachen wird die konditionale Verzweigung typischerweise durch eine *Wenn-Dann*-Anweisung (engl. *If-Else*) umgesetzt, während Iterationen über Schleifen implementiert werden. Diese Konstruktionen haben sich als sehr effektiv, leicht verständlich und ausreichend für viele Anwendungsszenarien erwiesen (Böhm und Jacopini, 1966). Entsprechend stellt VCCL diese beiden Kontrollflusselemente in einer graphischen Notation zur Verfügung.

Die Ablaufsteuerungen für die Kontrollflusselemente sind grundsätzlich als Ableitungen geschachtelter Methoden implementiert. Auf diese Weise kann jedes VCCL-Programm einzeln für sich auf seine Gültigkeit hin geprüft werden. Spezielle Kontrollmethoden würden zur Verwendung von mehreren Ausgabeports und damit mehrdeutigen Programmen führen, was ungültige VCCL-Programme zur Folge haben kann.

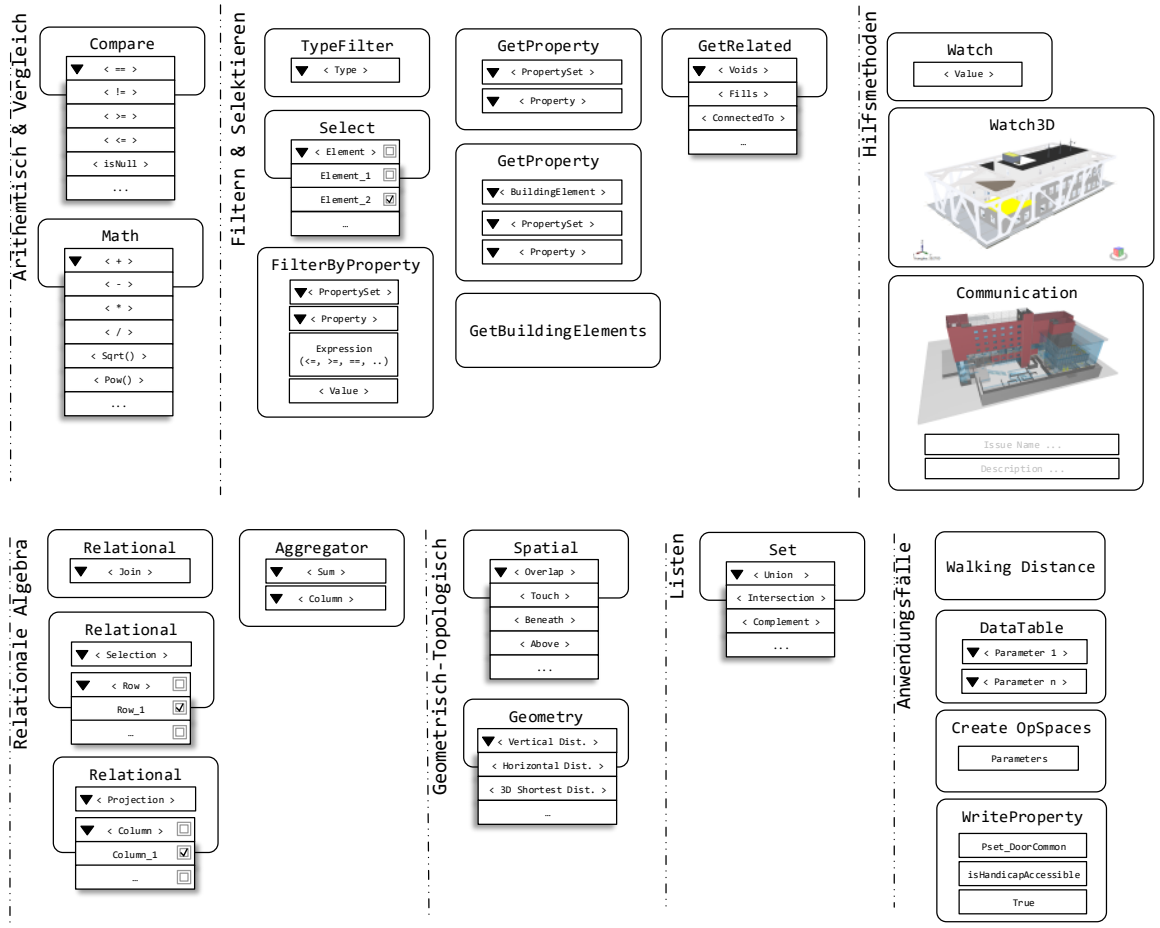


Abbildung 7.19: Übersicht über die atomaren VCCL-Methoden

If-Else Verzweigung

Abbildung 7.20 zeigt die Notation einer *If-Else*-Verzweigung. Für diese werden zwei geschachtelte VCCL-Unterprogramme definiert, von welchen letztlich - abhängig vom Ergebnis eines Testfalls - nur ein einziges entscheidend ist. Voraussetzung für eine solche Verzweigung ist also das Boolesche Ergebnis (*True* oder *False*) einer zuvor ausgewerteten Prüfung. Wird der Boolesche Wert als *True* ausgewertet, wird der entsprechende Bereich, der für diesen Fall definiert ist, ausgeführt. Wird er als *False* ausgewertet, wird das jeweils andere Unterprogramm ausgeführt.

Grundsätzlich muss mindestens ein Anwendungsfall (*True*- oder *False*-Zweig) definiert sein - die Definition des zweiten Falles ist optional und kann auch leer gelassen werden. Ist eines der Unterprogramme nicht definiert, wird entsprechend keine weitere Aktion ausgeführt, sobald die Routine in diesen Bereich führt. Der Informationsfluss wird somit unterbrochen und das Programm nicht weiter ausgeführt. Damit der Kontrollfluss für alle Fälle nach Ausführung des jeweiligen Unterprogramm,

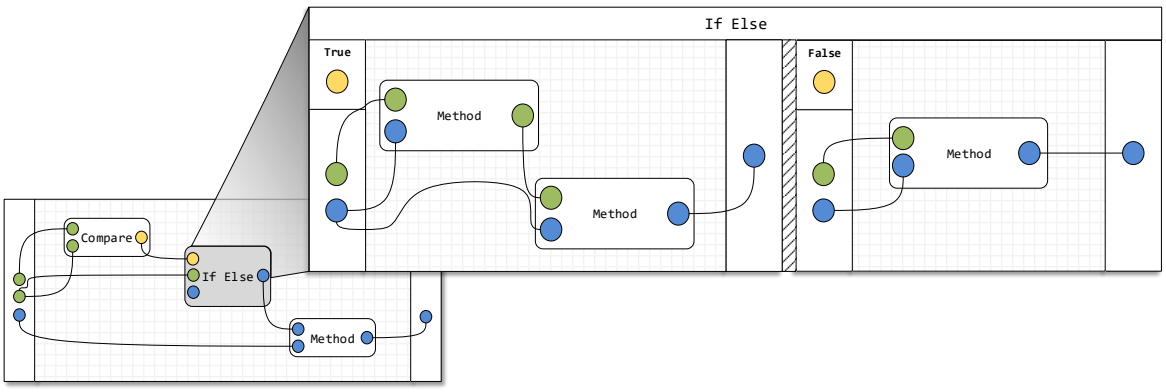


Abbildung 7.20: Schematische Darstellung einer *If-Else*-Verzweigung in VCCL-Notation

im Hauptprogramm weitergeführt werden kann, müssen die globalen Ein- und Ausgänge für beide Unterprogramme zwingend vom gleichen Datentyp sein.

```
Process (INPUT Case, [...])
{
  OUTPUT result;
  IF(Case == TRUE)
  {
    result = [...];
  }
  ELSE(Case == FALSE)
  {
    result = [...];
  }
  RETURN result;
}
```

Quellcode 7.4: Darstellung der *If-Else*-Verzweigung in Pseudocode

Foreach Schleife

Abbildung 7.21 zeigt die Notation einer Iteration. Diese kann angewendet werden, wenn eine bestimmte Operation bzw. eine Prozedur auf jedes einzelne Element einer Liste angewendet werden soll. Die Eingangsgröße muss in diesem Falle folglich eine Liste sein. Das geschachtelte Unterprogramm wiederum definiert einen Prozess ein einzelnes Element der Liste des gleichen Datentyps, sodass dieser Prozess für jedes Element, das Teil der Eingabemenge ist, durchgeführt werden kann. Da das geschachtelte Unterprogramm für jedes Element ein Ergebnis liefert, ist das Ergebnis dieses Knotens eine Relation, die sich aus Tupeln zusammensetzt, bei dem jedem Element der Eingangsliste der Ergebniswert zugeordnet wird.

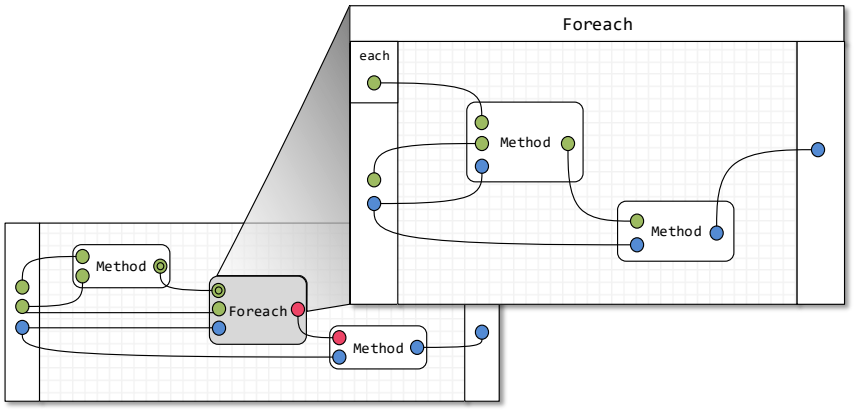


Abbildung 7.21: Schematische Darstellung einer *Foreach*-Schleife in VCCL-Notation

```
Process (INPUT[] Elements, [...])
{
  OUTPUT[] Result;
  FOREACH(element IN Elements)
  {
    result = [...];
    Result.ADD([element,result])
  }
  RETURN Result;
}
```

Quellcode 7.5: Darstellung der *Foreach*-Schleife in Pseudocode

Eine einfache beispielhafte Anwendung beider VCCL-Kontrollstrukturelemente ist in Abbildung 7.22 dargestellt. In diesem Beispiel wird jedes Element innerhalb einer Gruppe von Wänden auf eine bestimmte Höhe geprüft. Erfüllt die Wand das Kriterium, das durch den logischen Ausdruck innerhalb der geschachtelten Iteration beschrieben wird, wird sie der resultierenden Beziehung hinzugefügt. In diesem Fall ist das Unterprogramm *Else* nicht definiert, da das Objekt nicht weitergeleitet wird, wenn das Kriterium nicht erfüllt wird.

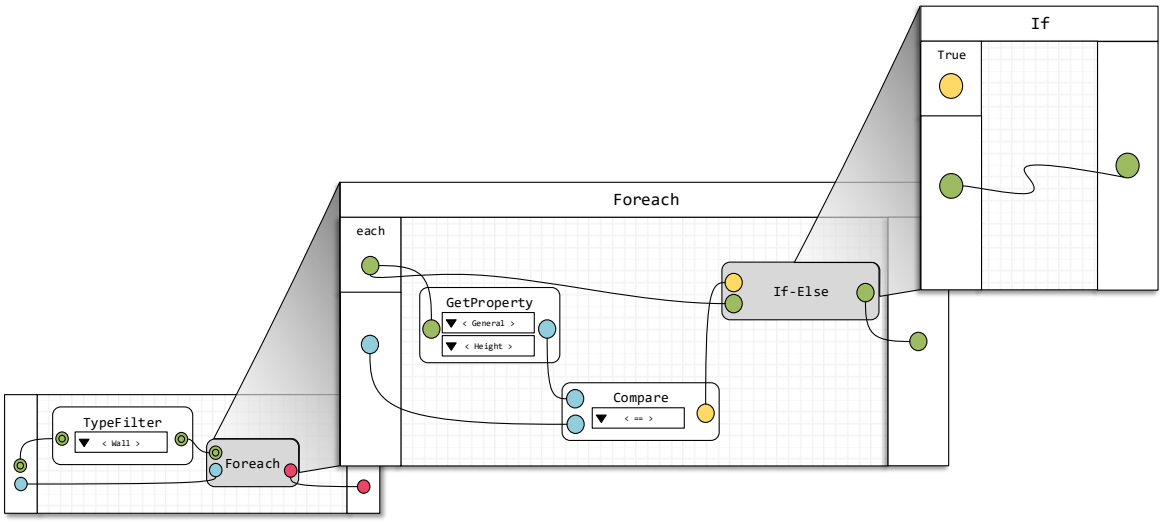


Abbildung 7.22: Beispielhaftes VCCL-Programm, welches zwei Kontrollstrukturen ineinander kombiniert

7.6 Softwareprototyp

Für den Nachweis der Tragfähigkeit des vorgestellten Ansatzes wurde im Rahmen der vorliegenden Arbeit ein Softwareprototyp implementiert. Die Entwicklung wurde in enger Zusammenarbeit mit dem Softwareunternehmen *Nemetschek Group* & *ALLPLAN* durchgeführt.

Den Kern des Demonstrators, einer *.NET*-basierten *Standalone*-Anwendung, bildet die VCCL. Der Prototyp stellt dem Nutzer zentral die in Abschnitt 7.5.1 (siehe insbesondere Abbildung 7.5) vorgestellte Benutzeroberfläche zur Verfügung, welche es dem Benutzer ermöglicht, ein VCCL-Programm mit Hilfe einer Bibliothek von atomaren Methoden zu erstellen.

Für die Implementierung der VCCL selbst wurde die *.NET*-Bibliothek *TUM.CMS.VplControl* (Chair of Computational Modeling and Simulation, 2016) verwendet. Diese wurde im Zuge diverser Forschungsansätze, die mit visuellen Sprachen arbeiten (insbesondere auch bei (Singer, 2015; Singer und Borrmann, 2015)), als Entwicklungsgrundlage erarbeitet (siehe auch Tabelle 7.1 aus Abschnitt 7.4).

Der Prototyp interagiert mit der Datenplattform *ALLPLAN Bimplus* (Allplan GmbH, 2016), einer zentralen Plattform zur Verwaltung von Gebäudeinformationsmodellen (vgl. Abschnitt 2.5.6). *Bimplus* dient in diesem Zusammenhang als Bibliothek, aus welcher die Applikation nicht nur die Bauwerksmodelle beziehen kann, sondern in welcher auch die erstellten VCCL-Programme verwaltet werden können. Überdies fungiert die Datenplattform auch als BCF-Server und kann somit auch von der VCCL als Kommunikationswerkzeug genutzt werden. BCF-Objekte, die mit Hilfe der *Communication*-Methode (siehe Abschnitt 7.5.3) erstellt wurden, werden automatisch in den Projektraum gespeichert, welcher von dem Anwender genutzt wurde, um auf das jeweilige Bauwerkmodell zuzugreifen.

Für die Integration wurde auf das in (Preidel et al., 2016) vorgestellte BIM Integration Framework (BIF) zurückgegriffen. Bei diesem handelt es sich um eine Schnittstellen-Erweiterung, welche sich verhältnismäßig einfach in BIM-basierte Softwareprodukte integrieren lässt (Preidel und Tretheway, 2016, 2017)

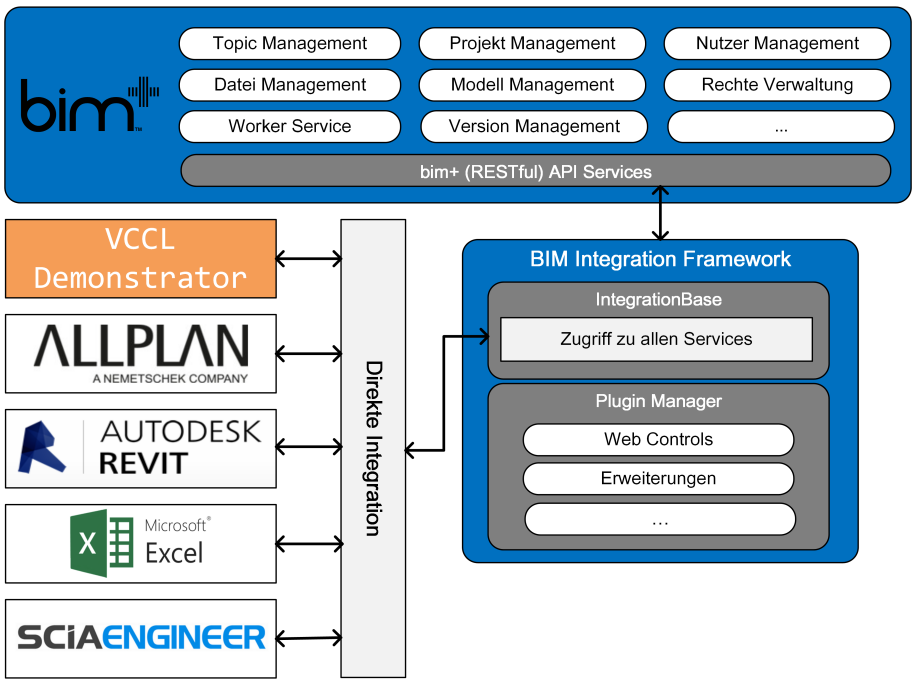


Abbildung 7.23: Anbindung des VCCL-Demonstrators (sowie weitere BIM-Werkzeuge) an die Datenplattform *ALLPLAN Bimplus* über das BIF; angelehnt an (Preidel et al., 2016)

und dort die Funktionalitäten einer Datenplattform, wie etwa den direkten Zugriff auf Ressourcen, verfügbar macht. In Abbildung 7.23 ist die enge Anbindung der VCCL-Umgebung an *Bimplus* schematisch dargestellt.

Das Datenmodell der VCCL (siehe auch Abschnitt 7.5.1) erlaubt, dass sowohl die nativen Datenmodelle der Datenplattform *Bimplus* als auch IFC-Modelle importiert und so der VCCL direkt verfügbar gemacht werden können. Für das Importieren und Auslesen der IFC-Modelle wurde in dem Demonstrator auf das akademische Framework *xBim Toolkit (eXtensible Building Information Modeling)* (Northumbria University, 2018) zurückgegriffen. Das *xBIM Toolkit* ist ein *open-source .NET* Entwicklungs-Toolkit, welches diverse Datenmodelle von buildingSMART, wie insbesondere IFC (siehe Abschnitt 2.4.2.2) und BCF (siehe Abschnitt 2.5.3), unterstützt.

Native sowie IFC-Bauwerksmodelle können in dem Demonstrator somit direkt aus der Datenplattform *Bimplus* oder aber lokal als IFC-Modell aus dem Dateisystem bezogen werden. Auf diese Weise kommt an dieser Stelle auch noch einmal das Prinzip der Allgemeingültigkeit der VCCL zum Tragen. Jedes erstellte Programm kann auf ein beliebiges Bauwerksmodell angewendet werden. Auf diese Weise kann das Modell als Eingangsinformation des VCCL-Programms im laufenden Betrieb geändert werden. Eine Veränderung der Eingangsdaten und somit der Informationen, die mit einem globalen Eingangsport verknüpft sind, stößt lediglich eine Neuausführung des VCCL-Gesamtprogramms an: die Methoden, welche von der Veränderung betroffen sind, werden neu verarbeitet und die Ergebnisse entsprechend aktualisiert.

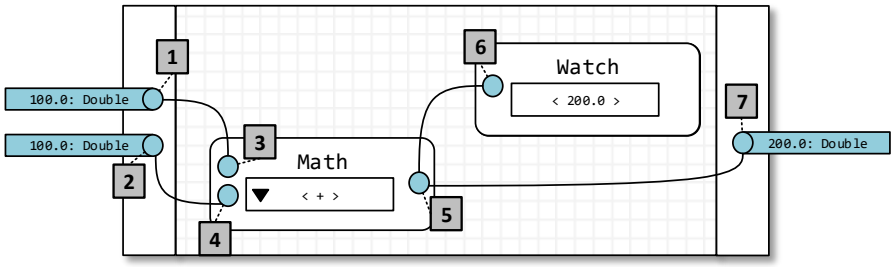


Abbildung 7.24: Graphische Darstellung des VCCL-Programms in Quellcode 7.6; zusätzlich sind die Ports in der Reihenfolge ihrer Entstehung nummeriert, so dass diese bei der Speicherung referenziert werden können.

Für die Umsetzung der atomaren, räumlich-geometrischen VCCL-Methoden (siehe Abschnitt 7.5.3) konnte mit Einverständnis von Daum (2018) auf die Forschungs- und Implementierungsarbeiten der *QL4BIM* zurückgegriffen werden.

Für die Transparenz und Prüfbarkeit der VCCL-Programme wurden auch die Hilfsmethoden für die Visualisierung der Schnittstelleninformationen implementiert. Um eine höhere Flexibilität bei der Darstellung von geometrischen Inhalten zu erreichen, wurde anstatt der vorhandenen Möglichkeiten zur Visualisierung des *xBim Toolkit* für den Demonstrator das *Helix Toolkit* herangezogen, welches umfassende Visualiserungsmöglichkeiten für eine *.NET* & *WPF*-Entwicklungsumgebung bietet.

So können in dem Demonstrator die Inhalte der Schnittstellen visualisiert und Zwischenergebnisse des Verarbeitungsprozesses angezeigt werden. Dieses ermöglicht es dem Anwender die Überprüfung, ob das verarbeitete Ergebnis seinen Erwartungen und Anforderungen entspricht.

Für die Handhabung der VCCL spielt die Austauschbarkeit eines erstellten VCCL-Programms eine große Rolle. Zum einen müssen die erstellten Prüfroutinen als Projekte angesehen werden, die permanent von einem oder aber mehreren Nutzern verändert werden. Gleichzeitig entwickeln sich die regulatorischen Anforderungen beständig weiter und verändern sich somit auch inhaltlich. Zum anderen bedingt der Verschachtelungsmechanismus der VCCL, dass eine Laufzeitumgebung in der Lage sein muss, Programme auch auf Basis einer nicht-graphischen Darstellung auszuführen.

Daher wurde für die VCCL ein eigenes Format auf Basis der XML entworfen, mit dessen Hilfe ein erstelltes Programm gesichert werden kann. Bei der Speicherung werden lediglich die enthaltenen Methodenknoten und die Verbindungen über die Schnittstellen der Knoten untereinander gespeichert. Quellcode 7.6 zeigt die XML-Übersetzung des in Abbildung 7.24 dargestellten Beispiels.

Wird ein VCCL als Unterprogramm verschachtelt, so wird für dieses automatisch eine *XML*-Instanz gesichert, welche von einem VCCL-Interpreter ausgeführt werden kann. Auf diese Weise sind Unterprogramme bei Bedarf sichtbar, können allerdings auch ohne graphische Darstellung des VCCL-Programms im Hintergrund ausgeführt werden.

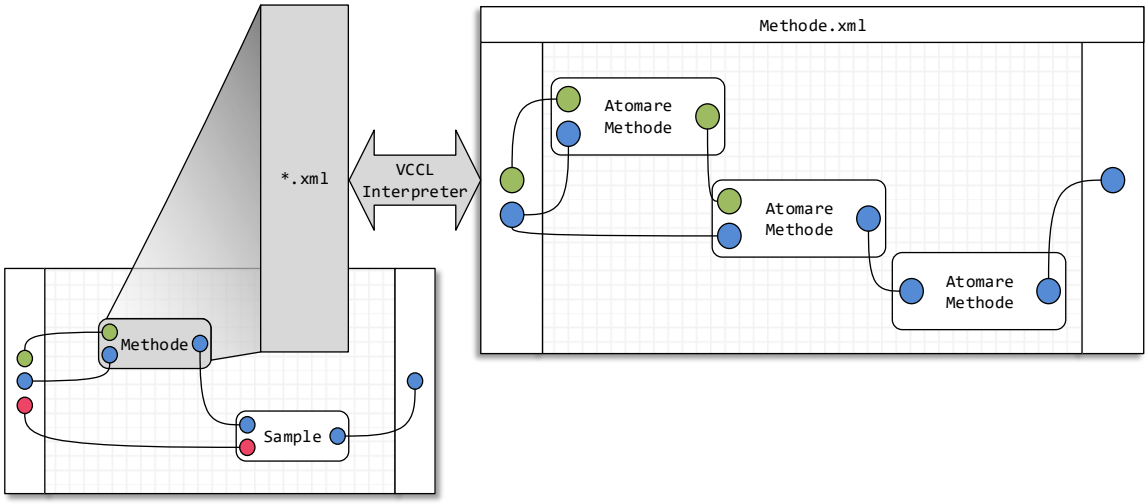


Abbildung 7.25: Verarbeitung von verschachtelten VCCL-Unterprogrammen, die als XML-Instanzen gesichert werden und bei Bedarf vom VCCL-Interpreter graphisch dargestellt werden können

```
<NodeGraphControl>
  <NodeGraphControl.GlobalInput>
    <NodeGraphLibrary.DoublePort Name="Double" Value ="100.0"/>
    <NodeGraphLibrary.DoublePort Name="Double" Value ="100.0"/>
  </NodeGraphControl.GlobalInput>
  <NodeGraphControl.GlobalOutput>
    <NodeGraphLibrary.DoublePort Name="Result"/>
  </NodeGraphControl.GlobalOutput>
  <NodeGraphControl.NodeGraphView>
    <NodeGraphNodeCollection>
      <NodeGraphLibrary.MathNode Name="A+B" Position="..."/>
      <NodeGraphLibrary.WatchNode Name="Result" Position="..."/>
    </NodeGraphNodeCollection>
    <NodeGraphLinkCollection>
      <NodeGraphControl.NodeGraphLink InputNodeId="3" OutputNodeId="1"/>
      <NodeGraphControl.NodeGraphLink InputNodeId="4" OutputNodeId="2"/>
      <NodeGraphControl.NodeGraphLink InputNodeId="6" OutputNodeId="5"/>
      <NodeGraphControl.NodeGraphLink InputNodeId="7" OutputNodeId="5"/>
    </NodeGraphLinkCollection>
  </NodeGraphControl.NodeGraphView>
</NodeGraphControl>
```

Quellcode 7.6: XML-Darstellung des VCCL-Programms in Abbildung 7.24

Die erstellten Programme können als **.xml* gespeichert und entsprechend geladen werden . Dieses bildet auch die Basis für die Verschachtelungs- und Kontrollstrukturmechanismen, die in Abschnitt 7.5.1 & 7.5.4 vorgestellt wurden.

Der Vollständigkeit halber sind sämtliche verwendete Bibliotheken und Systeme in Tabelle 7.9 übersichtlich dargestellt.

Tabelle 7.9: Systeme und Entwicklungsbibliotheken, welche für die Umsetzung des VCCL-Demonstrator eingesetzt wurden

Name	Funktion	Quelle
ALLPLAN Bimplus	zentrale Datenplattform, über welche native sowie dezentrale IFC-Modelle bezogen werden können. Überdies können VCCL-Programme als **.xml* über die Datenplattform verwaltet und ausgetauscht werden	`https://www.bimplus.net/de/` `https://www.allplan.com/de/`
TUM.CMS.VPLControl	*WPF*-basierte *.NET*-Entwicklungsbibliothek für visuelle Programmiersprachen	`https://github.com/tumcms/TUM.CMS.VPLControl`
xBim Toolkit	BIM Entwicklungsbibliothek für das Lesen und Verarbeiten von IFC-Modellen	`http://docs.xbim.net/index.html`
Helix Toolkit	Entwicklungsbibliothek für die Visualisierung von geometrischen Informationen	`http://www.helix-toolkit.org/`
QL4BIM	Implementierung der internen Verarbeitungsprozesse der geometrisch-räumlichen Operatoren der VCCL	Daum (2018)

In Abbildung 7.26 und 7.27 ist die Nutzeroberfläche des Demonstrators mit jeweiligen Anwendungsbeispielen zu dargestellt.

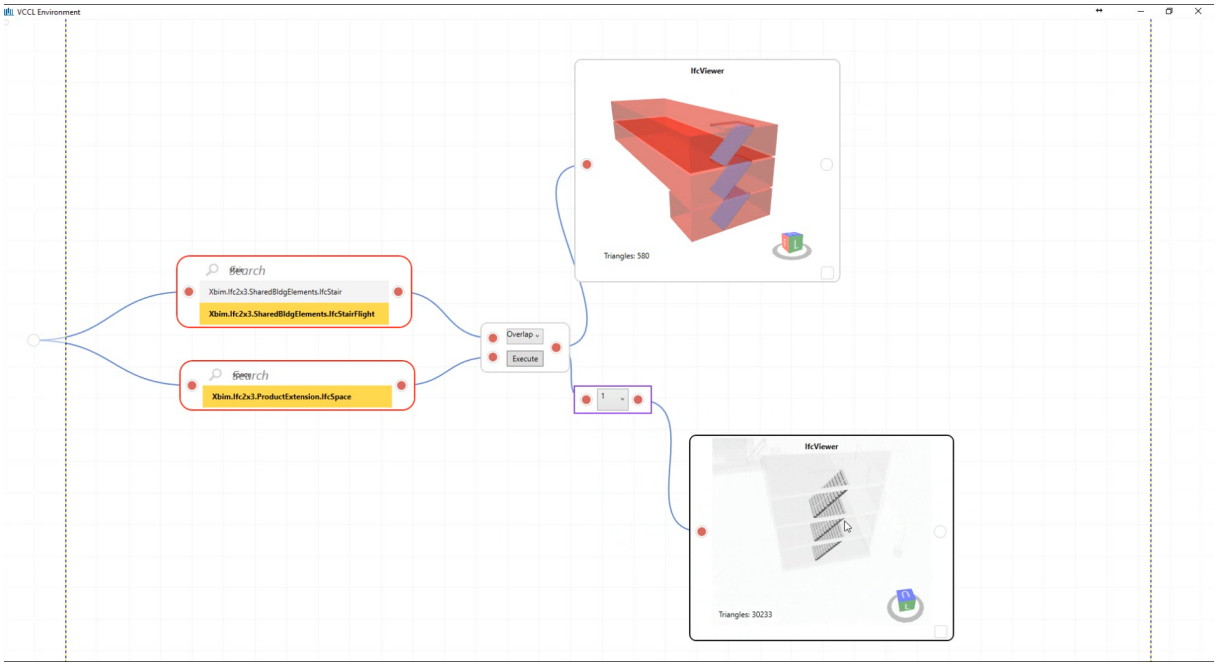


Abbildung 7.26: VCCL-Programm im VCCL-Demonstrator für die Auswertung und Identifikation von Treppenhäusern in einem IFC-Modell, die für die Fluchtwegroute berücksichtigt werden sollen

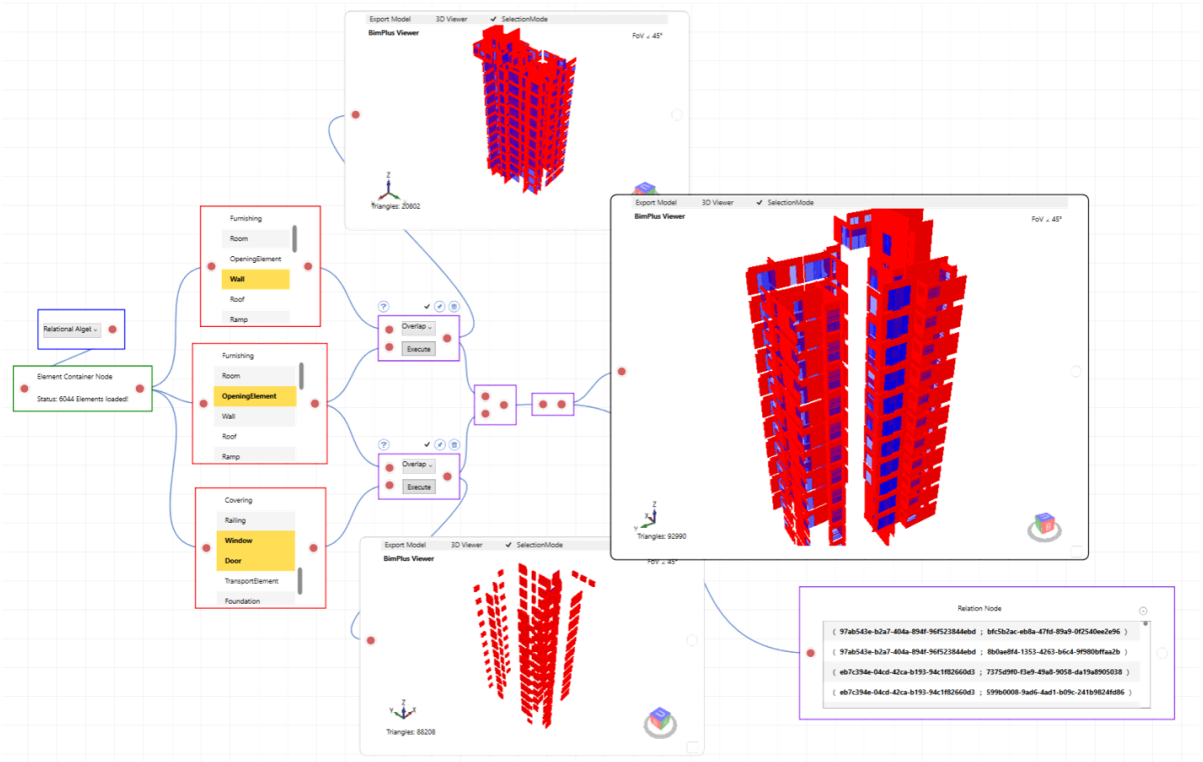


Abbildung 7.27: VCCL-Programm im VCCL-Demonstrator für die Abfrage der Beziehung zwischen Wand- und Einbauteilen in einem nativen *Bimplus*-Modell mit Hilfe von relationalen Operatoren

7.7 Zusammenfassung

Das Kapitel führt die Visual Code Checking Language (VCCL) als Ansatz ein, welcher zum einen die in Kapitel 5.2 vorgestellten Einzelkomponenten der allgemeinen Struktur eines ACCC abdeckt und zum anderen dabei den zentralen Herausforderungen des Automated Code Compliance Checking (ACCC) (siehe Abschnitt 5.4) gerecht wird. Bisherige Ansätze adressieren entweder nur einzelne Teilschritte oder aber werden den Herausforderungen nur in Teilen gerecht (siehe Kapitel 6).

Neben der für den beabsichtigen Anwendungszweck geeigneten Ausdruckskraft spielen insbesondere die Anwendbarkeit für typische Anwender sowie die Transparenz der mit Hilfe der VCCL beschriebenen Verarbeitungsprozesse bei Struktur und Gestaltung der Sprache eine wesentliche Rolle. Ausgehend von diesen Aspekten zeichnet sich die VCCL als visuelle, imperative, strikt-typisierte Programmiersprache aus, welche sich explizit für die formale Beschreibung von Konformitätsprüfungsprozessen im Bauwesen eignet. Visuelle Programmiersprachen stellen im Gegensatz zu textuellen Programmiersprachen die beschriebenen Verarbeitungssysteme graphisch und somit besser verständlich für Anwender, die über keine oder aber geringe Programmierkenntnisse verfügen, dar. Der Mehrwert visueller Programmiersprachen zeigt sich unter anderem in einer steigenden Zahl von ingenieurtechnischen Anwendungen. Aktuell werden die Sprachen im AEC-Bereich bisher weniger für Prüf- oder Analysezwecke, sondern vorwiegend für gestalterische Aufgaben herangezogen.

In der VCCL bilden Methodenknoten, Kanten und Schnittstellen die zentralen Elemente, welche auf einer dreiteiligen Benutzeroberfläche zu einem visuellen Informationsfluss zusammengestellt werden. Erstellte VCCL-Programme können gespeichert und zusammengefasst als eigener Methodenknoten in einem anderen VCCL-Programm zugreifbar gemacht werden. Auf diese Weise lassen sich beliebig komplexe Programme erstellen. Überdies kann dieser Mechanismus auch für die Einführung von Kontrollstrukturen wie einer *Foreach*-Iteration und einer konditionalen *If-Else*-Verzweigung genutzt werden.

Auch wenn ein mit der VCCL formuliertes Programm voll automatisiert ausgeführt werden kann, bietet die visuelle Sprache den zentralen Vorteil, dass ein Anwender die Möglichkeit hat, bei jedem Prozessschritt, die erhaltenen Ergebnisse mit Hilfe von Methoden zu visualisieren und so den Prüfprozess auf dessen Integrität hin zu untersuchen.

Den Ausgangspunkt der Ausdruckskraft der VCCL bilden atomare Methodenknoten, welche jeweils wohl definierte Operationen beschreiben und nicht zerlegt werden können. Die atomaren Methoden stellen die unterste Ebene der VCCL dar und bestimmen maßgeblich die Ausdruckskraft der Sprache. Als Basiselemente der VCCL können die Prozesse, welche innerhalb der atomaren Methoden verarbeitet werden, vom Anwender nicht weiter eingesehen werden und somit handelt es sich hierbei um eine Barriere hinsichtlich der Transparenz. Bei der Einführung einer atomaren Methode muss daher sehr genau abgewägt werden, ob es sich hierbei um eine akzeptable Barriere handelt, die für den Anwender keine grundlegende Einschränkung darstellt.

Zu den wesentlichen Kategorien zählen insbesondere die *relationalen* sowie *räumlich-geometrischen* atomaren Methoden, welche es erlauben, komplexe Anforderungen zu formulieren. Die Gruppe der Hilfsmethoden hingegen verleiht der Sprache und somit dem Anwender die Möglichkeit, Inhalte von Prozess- bzw. Verarbeitungsschritten zu visualisieren und Inhalte als Referenz im Modell zu kommunizieren. Der aktuelle Entwicklungsstand der Methodenbibliothek stellt keinen finalen Stand dar, sondern kann bei Bedarf um weitere atomare Methoden erweitert werden.

Auf Basis des beschriebenen Aufbaus wurde die VCCL prototypisch als Demonstrator umgesetzt, mit welcher sich VCCL-Programme für eine praktische Anwendung erstellen lassen.

8 Validierung der VCCL

8.1 Einführung

In diesem Kapitel soll die in Kapitel 7 vorgestellte Visual Code Checking Language (VCCL) anhand ausgewählter Anwendungsbeispiele hinsichtlich der Tragfähigkeit und Eignung für den beabsichtigen Einsatzzweck gezeigt werden. Die Validierung gliedert sich in drei Teile.

In den ersten beiden Abschnitten 8.2 und 8.3 werden jeweils spezifische Beispielpassagen aus nationalen Normen vorgestellt. Diese werden anschließend mit Hilfe der VCCL formalisiert. In dem anschließenden Abschnitt 8.4 wird in einem Fallbeispiel ebenfalls eine Passage aus einer nationalen Norm untersucht. Hier soll die VCCL allerdings hinsichtlich ihrer Eignung als Werkzeug für eine Bauwerksmodellaufbereitung als Vorbereitungsschritt für eine darauffolgende Modellprüfung untersucht werden.

8.2 Korean Building Act: Article 34 Clause 1

In seinem *Report on Open Standards for Regulations, Requirements and Recommendations Content* (BuildingSmart, 2017d) wurde von *buildingSMART* ein Abschnitt des koreanischen Baurechts als Grundlage für einen Vergleich verschiedener Ansätze zum ACCC gewählt. In dem Bericht heißt es, dass man sich für diesen Ausschnitt entschieden hat, da dieser in ausreichender und angemessener Art und Weise typische regulatorische Inhalte darstellt und somit eine gute Basis für die Vergleichbarkeit der verschiedenen Ansätze bietet. Daher soll an dieser Stelle eben dieser Abschnitt mit Hilfe der VCCL untersucht werden. Bei der ausgewählten regulatorischen Anforderung handelt es sich um Artikel 34 Absatz 1 des *Korean Building Act* (Korea Legislation Research Institute, 2008), welcher im Originaltext folgendermaßen lautet:

> *On each floor of a building, direct stairs leading to the shelter floor or the ground other than the shelter floor shall be installed in the way that the walking distance from each part of the living room to the stairs is not more than 30 meters: Provided, that in cases of a building of which main structural part is made of a fireproof structure or non-combustible materials, the walking distance of not more than 50 meters may be established, and in cases of a factory prescribed by Ordinance of the Ministry of Land, Infrastructure and Transport, which is equipped with automatic fire extinguishers, such as sprinklers, in an automated production facility, the walking distance of not more than 75 meters may be established.*

Der Abschnitt beschreibt Anforderungen für die Wegstrecken von den Aufenthaltsräumen in jedem Stockwerk eines Bauwerks zu den Treppen, die als Ausgang dienen. Dabei hängt der Anforderungswert für die Länge der Wegstrecke davon ab, ob das Bauwerk selbst aus einer feuerfesten Konstruktion bzw. nicht brennbaren Materialien besteht oder aber mit automatischen Feuerlöschern, wie beispielsweise Sprinklern, ausgestattet ist.

Die Möglichkeit mit Hilfe der VCCL Unterprogramme zu definieren, erlaubt es, den Abschnitt zunächst zu unterteilen. Dieses ermöglicht es, die verhältnismäßig lange und komplexe Anforderung in Teile zu gliedern und so die spätere visuelle Übersetzung auf ein notwendiges Maß zu reduzieren.

In einem ersten VCCL-Unterprogramm *Exit Stairs* werden die Treppen, die als Ausgang dienen, für ein definiertes Stockwerk identifiziert. Hierfür werden zunächst die Treppenhäuser (*"Finde alle Räume, welche Treppen beinhalten"*) daraufhin geprüft, ob diese als Ausgänge genutzt werden können (Anfrage: *"Finde alle Räume, die Türen beinhalten, die als Ausgang dienen"*). Anschließend wird geprüft, ob die identifizierte Treppe in dem Stockwerk selbst liegt. Schließlich werden die identifizierten Ausgangstreppen von dem Programm ausgegeben, welches in Abbildung 8.1 dargestellt ist.

In einem zweiten Schritt kann nun die eigentliche Vorschrift abgebildet werden. Hierfür wird das zuvor definierte Unterprogramm wiederverwendet. Für die Abbildung der Vorschrift ist es zudem notwendig, die Laufwegstrecke zwischen den Wohnräumen (*LivingRoom*) und den Treppen zu berechnen. An dieser Stelle wird die Annahme getroffen, dass es sich hierbei um einen Berechnungsschritt handelt, bei welchem der Nutzer in erster Linie an der Laufdistanz selbst, nicht aber deren Berechnung interessiert ist. Hierbei handelt es sich also um eine akzeptable *Black-Box*-Barriere, die die Einführung einer neuen atomaren Methode für die Berechnung der Laufdistanz rechtfertigt. Daher wird nun an dieser Stelle die atomare Methode *Walking Distance* eingeführt (siehe auch Tabelle 7.8), welche in einem Stockwerk die Laufdistanz zwischen Quell- und Endobjekten berechnet und die errechneten Distanzen in einer Relation ausgibt.

In dem VCCL-Programm werden zunächst alle Räume, die als *LivingRoom* gekennzeichnet sind als Quellobjekte und die identifizierten Ausgangstreppen als Endobjekte in diese Methode eingegeben. Als Resultat ergibt sich eine Relation, die für alle Paare dieser Objekte die entsprechende Distanz auf dem vom Nutzer selektierten Stockwerk berechnet. Die maximale und somit maßgebende Distanz kann anschließend über die *Aggregator*-Methode aus der Relation abgefragt werden.

Um nun zu prüfen, ob das vorliegende Bauwerksmodell den Anforderungen entspricht, muss ausgewertet werden, ob die Hauptstruktur des Gebäudes als feuerfest eingestuft werden kann und ob diese zusätzlich mit einer automatisierten Sprinkleranlage ausgestattet ist. Bei beiden Auswertungen handelt es sich entweder um eine Benutzereingabe, die von dem Anwender in die Prüfroutine eingegeben werden muss, oder um eine Auswertung der entsprechenden Information aus dem Bauwerksmodell. So könnte die Angabe zu der Feuerfestigkeit mittels der VCCL abgefragt werden, wenn diese beispielsweise als Attribut für das Bauwerksmodell [*BuildingModel*] selbst hinterlegt ist.

Eine Prüfung, ob das Bauwerk mit einer Sprinkleranlage ausgestattet ist, könnte umgesetzt werden, indem geprüft wird, ob das Bauwerk entsprechende Bauteiltypen, wie beispielsweise ein *IfcFireSuppressionTerminal*, enthält. Hierzu ist in Abbildung 8.3 entsprechend die Prüfung als VCCL-Unterprogramm dargestellt.

In dem vorliegenden VCCL-Programm wird an dieser Stelle davon ausgegangen, dass es sich hierbei um Benutzereingaben handelt, welche als Eingangsgröße in das Programm eingegeben werden. Für die Auswertung der relevanten Anforderungsgröße zu der maximal erlaubten Wegstrecke werden jeweils *If-Else*-Verzweigung verwendet, welche den jeweiligen Anforderungswert weitergibt.

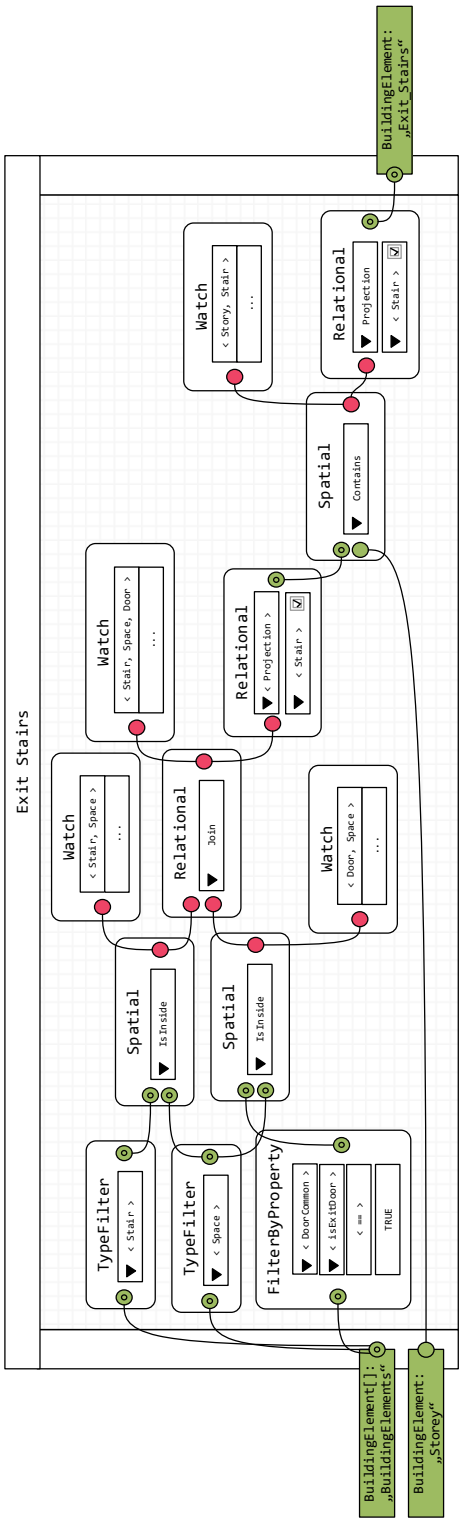


Abbildung 8.1: VCCL-Unterprogramm, welches alle Treppen identifiziert, die in einem Stockwerk als Ausgang dienen

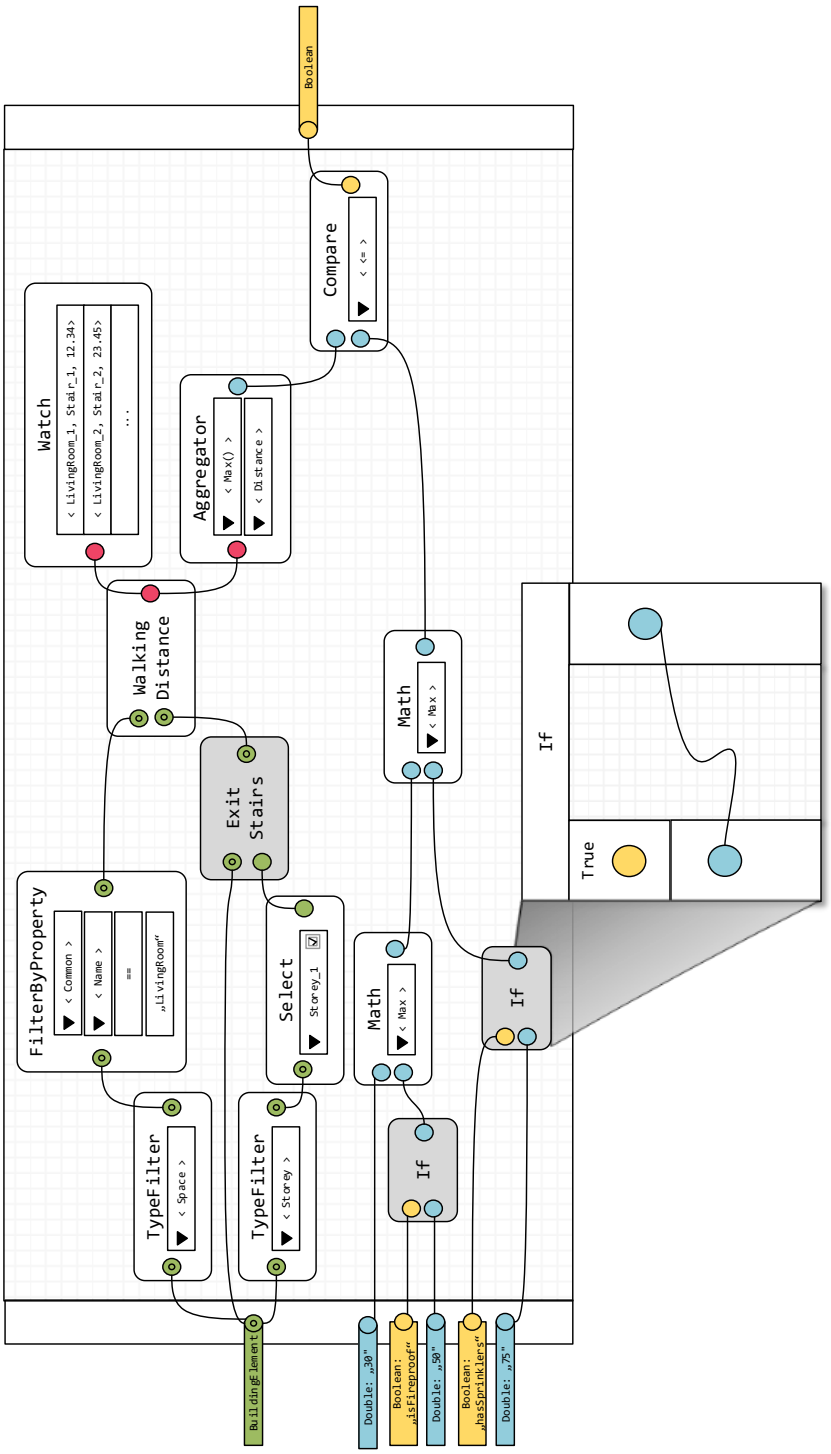


Abbildung 8.2: VCCL-Kodierung des *Korean Building Act: Article 34 Clause 1*

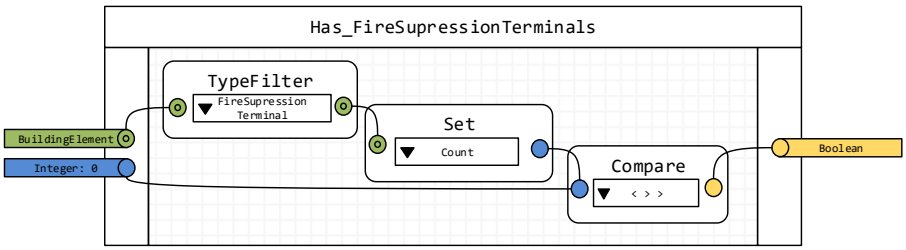


Abbildung 8.3: VCCL-Programm, mit welchem untersucht wird, ob ein Gebäudemodell mit Sprinkler-Objekten (*IfcFireSuppressionTerminal*) ausgestattet ist

8.3 Anforderungen zur Rauch- und Wärmefreihaltung: DIN 18232-2:2007-1

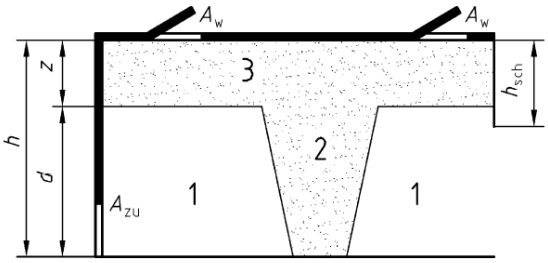
Die Norm DIN 18232-2:2007-11 (DIN, 2007-11) beinhaltet Vorschriften zur Bemessung und zum Einbau von natürlichen Rauchabzugsanlagen zur Rauch- und Wärmefreihaltung. Ziel der Norm ist es, informative Hinweise für die Bemessung und den Einbau von natürlichen Rauchabzugsanlagen für Räume zu geben. Hierfür sind in der Norm Tabellen und Berechnungsverfahren für die Dimensionierung von Öffnungsflächen in Räumen enthalten.

Für die Darstellung der Prüfung bezieht sich die Norm auf einen raumbezogenen Brandfall, welcher in Abbildung 8.4 zu finden ist. Durch den Brand kommt es zur Rauchentwicklung, dargestellt durch eine Plume (2) und die Rauchschicht (3). Als Schutz gegen den Rauch fungieren in dem Raum Zu- und Abluftflächen sowie Rauchschürzen, welche das Raummodell in Rauchabschnittsflächen untergliedern und maßgeblich für die Höhe der raucharmen Schicht (1) sind. Bei der Prüfung hinsichtlich der Norm soll nun gezeigt werden, dass ein Raum durch die vorhandenen Zu- und Abluftflächen an der oberen Seite ohne mechanische Unterstützung einen ausreichenden natürlichen Rauchabzug besitzt und somit die gesetzlichen Regelungen einhält.

Um dieses zu garantieren, sind in der Norm mehrere Vorschriften auf unterschiedliche Art und Weise beschrieben. Eine erste Darstellungsweise einer solchen Anforderung sind Fließtext-Passagen, welche spezielle Vorschriften an die Beschaffenheit des Raums formulieren. Als Beispiel hierfür soll eine Anforderung dienen, deren Einhaltung direkt zu Beginn der Norm eingefordert wird:

> *Die Bemessung der notwendigen Rauchabzugsfläche nach dieser Norm setzt voraus, dass die Rauchabschnittsflächen entweder $\leq$ 1 600 m^2 groß sind oder durch Rauchschürzen in maximal 1 600 m^2 große Rauchabschnittsflächen A_R unterteilt werden [...].*

Da Rauchschürzen als solche bisher noch nicht in dem IFC-Datenschema enthalten sind, muss an dieser Stelle davon ausgegangen werden, dass diese über Attribute beschrieben werden und nur auf diese Art und Weise identifiziert werden können.



Legende

A_w	Rauchabzugsfläche in m^2	z	Höhe der Rauchschicht ($h - d$) in m
A_{zu}	Größe der Zuluftfläche in m^2	1	raucharme Schicht
d	Höhe der raucharmen Schicht in m	2	Plume
h	Höhe des zu schützenden Raumes in m	3	Rauchschicht
h_{sch}	Höhe der Rauchschürze in m		

Abbildung 8.4: Schematische Darstellung eines raumbezogenen Brandfalls für die Dimensionierung der minimal erforderlichen Rauchabzugsflächen (DIN, 2007-11)

Weiterhin definiert die Norm in Abhängigkeit von der Raumhöhe, von der Höhe der durch das Feuer verursachten Rauchschicht und von dem Brandbemessungsgrad eine minimal erforderliche Rauchabzugsfläche A_w. Der Wert ergibt sich hierbei in Abhängigkeit von den genannten Parametern aus einer Datentabelle, welche bereits in Kapitel 4 in Abbildung 4.5 dargestellt ist.

Zur Abbildung einer solchen Tabelle mit Hilfe der VCCL dient zum einen der Datentyp [*DataTable*], welcher es erlaubt den Informationsgehalt von Tabellen über Schnittstellen zu übertragen. Zum anderen wird an dieser Stelle die weitere atomare *DataTable*-Methode eingeführt (siehe auch Tabelle 7.8), welche ermöglicht, auf einen spezifischen Wert der Datentabelle mit Hilfe von Parametern zuzugreifen.

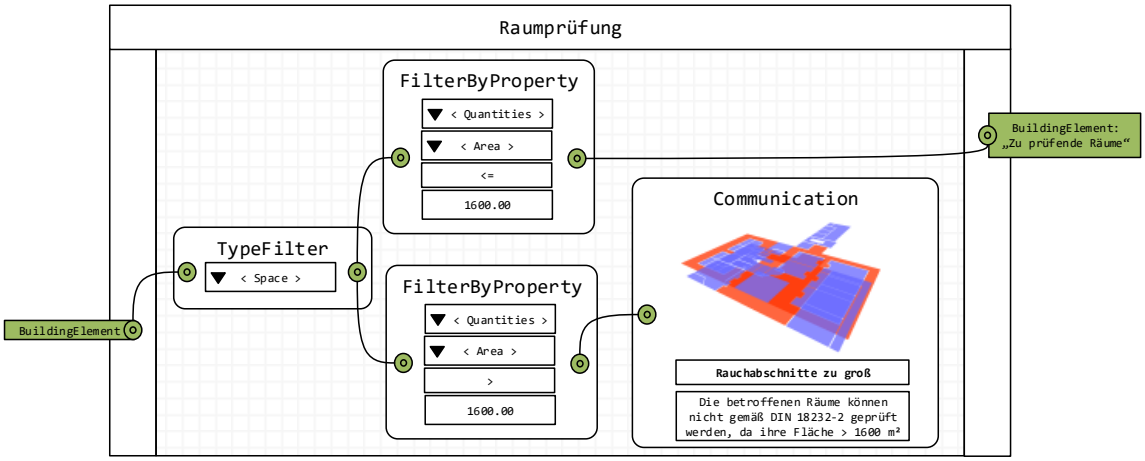


Abbildung 8.5: Prüfung, welche Brandabschnittsflächen (dargestellt als *Space*) für die Untersuchung gemäß DIN 18232-2 relevant sind. Nicht zu prüfende Raumobjekte werden entsprechend mittels der *Communication*-Methode kommuniziert und festgehalten.

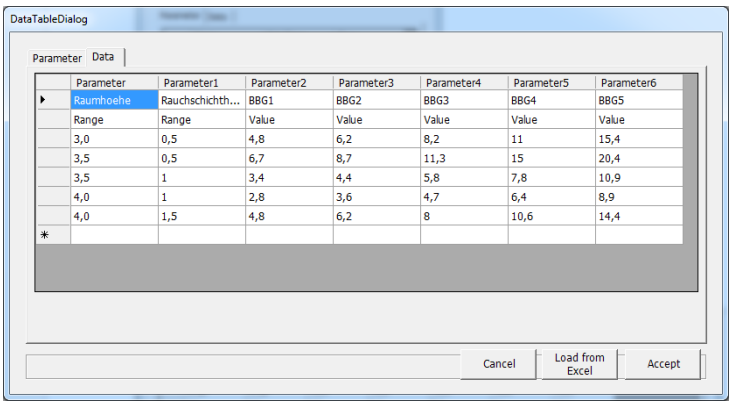


Abbildung 8.6: Dialogfenster der VCCL-Applikation, mit dessen Hilfe Parameter sowie Werte von Datentabellen vom Nutzer für den *DataTable*-Knoten eingegeben werden können.

In dem vorliegenden Fall müssen also in die Datentabelle die Raumhöhe ([*Double*]), die Rauchschichthöhe ([*Double*]) sowie die Brandbemessungsgruppe ([*Integer*]) eingegeben werden, um so den minimalen Soll-Wert für die Rauchabzugsfläche ([*Double*]) zu erhalten. In Abbildung 8.6 ist für das vorliegende Beispiel das Dialogfenster in der VCCL-Applikation zu sehen, mit dessen Hilfe die Parameter sowie die entsprechenden Werte der Datentabelle vom Nutzer eingegeben werden können. Anhand dieser Methode kann nun die Anforderung mit Hilfe der VCCL, wie in Abbildung 8.7 dargestellt, kodiert werden.

In dem resultierenden VCCL-Programm wird zunächst ein einzelner Raum, der hinsichtlich der Norm geprüft werden soll, selektiert und anschließend untersucht, welche Öffnungsobjekte sich oberhalb von diesem befinden und den Raum gleichzeitig berühren. Daraus ergibt sich eine Liste der Öffnungsobjekte, deren aufsummierte Fläche für den Bemessungsfall als Ist-Wert maßgebend ist. Die Aufsummierung kann aus der resultierenden Liste der Öffnungsobjekte durch die *Aggregator*-Methode bezogen werden.

Wie bereits zuvor beschrieben, wird der Soll-Wert zu der Rauchabzugsfläche der Datentabelle der Norm entnommen und hierzu durch den *DataTable*-Knoten abgebildet. Bei der Höhe des Raumes handelt es sich um das Attribut des Raumobjekts, dessen Wert direkt aus dem Gebäudedatenmodell bezogen werden kann. Die Höhe der Rauchschicht und die Brandbemessungsgruppe hingegen beziehen sich auf die Brandschutzklasse und die Intensität des Brandes, welche für den Raum überprüft werden sollen. An dieser Stelle wird angenommen, dass es sich hierbei um eine Nutzereingabe handelt. Diese wird somit als Eingangswert für das Programm angesetzt.

Der letzte Schritt dieses Prüfverfahrens ist der Vergleich der beiden erhaltenen Werte, um so zu überprüfen, ob der Grenzwert eingehalten wird. Das Ergebnis ist ein Wahrheitswert, der anschließend für die weitere Verarbeitung der Prüfergebnisse verwendet werden kann, wie zum Beispiel die Zuordnung der nicht erfüllten Anforderungen zu einem verantwortlichen Projektbeteiligten über den *Communication*-Knoten.

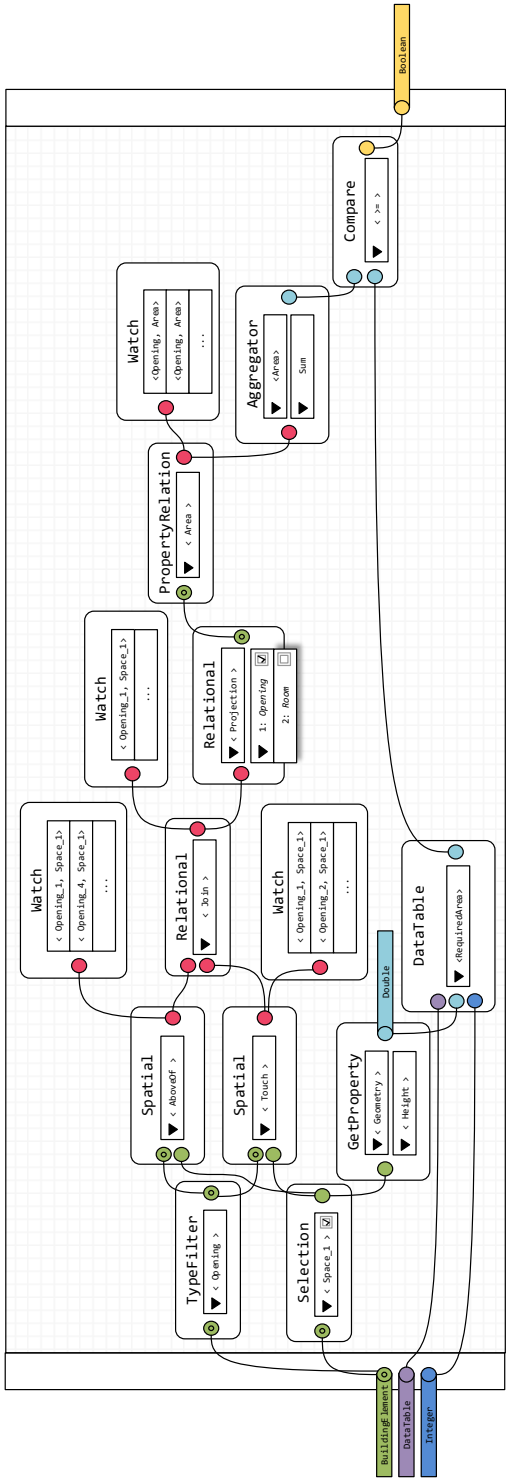


Abbildung 8.7: VCCL-Kodierung der Anforderung aus DIN 18232-2 für die Einhaltung der minimal erforderlichen Rauchabzugsfläche für den raumbezogenen Brandfall

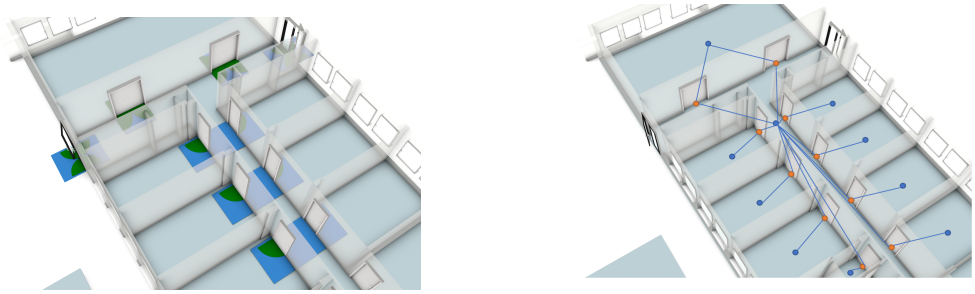
8.4 VCCL als Werkzeug für die Aufbereitung von Bauwerksinformationen

In einem weiteren Anwendungsfall soll die VCCL hinsichtlich ihrer Eignung als Werkzeug für die Aufbereitung von Bauwerksinformationen untersucht werden, wie dieser auch als Teilschritt des ACCC in Kapitel 5 beschrieben wurde. Grundlage für die folgenden Ausführungen ist eine Zusammenarbeit mit dem CSTB (siehe hierzu auch 6.4.3), welche in (Preidel et al., 2018) festgehalten wurde. Ansatzpunkt für diesen Anwendungsbereich ist, dass verschiedene Ansätze für die automatisierte Modellprüfung, wie diese in 6 vorgestellt wurden, oftmals nicht über eine ausreichende Ausdruckskraft oder über die erforderlichen Methoden verfügen, um aus Bauwerksmodellen höherwertige Informationen für Prüfungen abzuleiten. Weiterhin sehen standardisierte Datenschemata, wie beispielsweise die IFC, die Abbildung dieser für die Modellprüfung relevanten Informationen nicht vor.

Als Beispiel sei an dieser Stelle die Berechnung von Bewegungs- bzw. Freiflächen genannt, wie diese für eine Prüfung der Barrierefreiheit, erforderlich ist. In Kapitel 4 wurde in Abbildung 4.2 hierzu bereits ein nationales Beispiel einer solchen regulatorischen Anforderung graphisch dargestellt.

Für das bessere Verständnis sind in Abbildung 8.8 exemplarisch die Freiflächen (*blau*) sowie der Türschwenkbereich (*grün*). Im Gegensatz zu den Schwenkbereichen beschreiben Freiflächen den minimalen Platzbedarf vor oder hinter beweglichen Bauteilen (insbesondere Türen), sodass beim Öffnen und Schließen genügend Platz für ein Drehen und Wenden von Rollstühlen garantiert werden kann. Grundsätzlich sind die Freiflächen direkt von der jeweiligen Tür abhängig. Allerdings ergeben sich die erforderlichen Dimensionen aus den jeweils geltenden internationalen bzw. nationalen Richtlinien. Eine Abbildung dieser Informationen ist in Gebäudemodellen in der Regel nicht vorgesehen.

Ein weiteres Beispiel ist die Abbildung der Zugänglichkeit zwischen den einzelnen Räumen eines Bauwerks. Diese Information ist für die Prüfung der Barrierefreiheit notwendig. Geprüft wird, ob mindestens eine Route in dem Gebäude existiert, welche barrierefrei begehbar ist. Die Barrierefreiheit der Route ist maßgeblich durch die einzelnen Bauteile und Räume bestimmt, welche entlang der Route passiert werden müssen. Nur wenn beispielsweise eine Tür eine ausreichende Höhe und Breite sowie die erforderlichen Freiflächen besitzt, kann diese als gültiger Teil einer barrierefreien Route gelten. In Abbildung 8.8 ist exemplarisch ein Graph dargestellt, welcher die Zugänglichkeit abbildet.



(a) Bewegungs- und Freiflächen für bewegliche Bauteile

(b) Zugänglichkeit zwischen Räumen über bewegliche Bauteile

Abbildung 8.8: Beispiele für Bauwerksinformationen, die typischerweise nicht direkt in einem Gebäudemodell gespeichert sind

An dieser Stelle kann die VCCL mit ihrer Ausdruckskraft als ein Vorverarbeitungssystem für die Aufbereitung bzw. Ableitung fehlender Informationen helfen. Die visuelle Sprache bringt auch hier den wesentlichen Vorteil mit sich, dass dem Benutzer die Zwischenschritte der Aufbereitung transparent und nachvollziehbar dargestellt werden. Selbst wenn ausgewählte Ansätze, wie diese in Kapitel 6 vorgestellt wurden, in der Lage sind, diese Informationen aufzubereiten, sind die Zwischenschritte in der Regel nicht einsehbar oder aber nicht parametrisiert.

Für den Anwendungsfall wurde ein Vorverarbeitungsprozess für IFC-Modelle konzipiert, dessen Hauptzweck darin besteht, aus dem Gebäudemodell Informationen abzuleiten und diese anschließend für eine Verifizierung der Entwurfsplanung heranzuziehen. In Abbildung 8.9 sind die wesentlichen drei Komponenten dargestellt: (1) die Identifizierung von Informationen, die abgeleitet werden müssen, (2) die entsprechenden VCCL-Programme, welche die Ableitung dieser Informationen parametrisch beschreibt und in dem IFC-Modell speichert und (3) eine abschließende MVD-basierte Prüfung der in dem Bauwerksmodell gespeicherten Informationen.

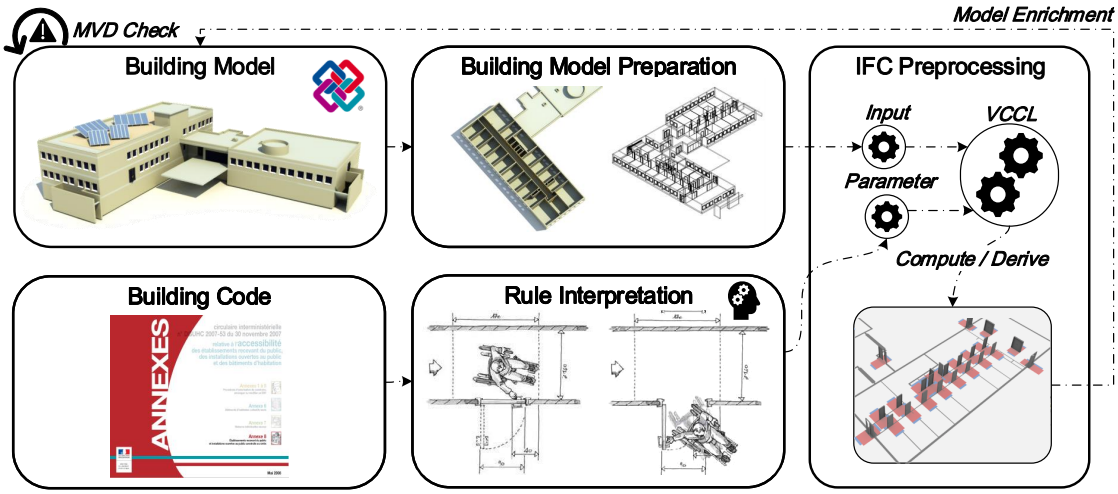


Abbildung 8.9: Schematischer Ablauf der Modellprüfung mit Hilfe der VCCL als Vorverarbeitungswerkzeug der Bauwerksinformationen (Preidel et al., 2018)

Der vorgestellte Ablauf wurde anhand der Freiflächenprüfung, wie diese im *Annex 8* (Ministere du logement et de la ville, 2008-05) eingefordert wird, auf seine Tragfähigkeit hin geprüft.

Dabei spezifiziert die gewählte französische Verordnung verschiedene Möglichkeiten, wie Rollstuhlfahrer einen Raum betreten können müssen. Die erforderlichen Freiflächen werden durch eine bestimmte Breite und Länge beschrieben und können entweder direkt vor oder leicht versetzt neben der Tür platziert werden, sodass ausreichend Freiraum vorhanden ist. Die Raumsituation unterscheidet sich auch darin, von welcher Seite sich der Rollstuhlfahrer nähert und in welche Richtung sich die Tür öffnet. Aus den Anforderungen lässt sich ableiten, dass eine Kombination von Freiflächen vor und hinter einer Tür eingehalten sein muss, damit die Tür als barrierefrei und behindertengerecht betrachtet werden kann. Die Raumkombinationen sowie die Bemaßung müssen manuell extrahiert werden, wie in Abbildung 8.10 dargestellt.

Die VCCL bietet bisher keine Methode, welche sich für die Ableitung von neuen Raum- bzw. Flächenobjekten eignet. Daher wird an dieser Stelle eine neue atomare Methode *Create OpSpaces* eingeführt (siehe auch Tabelle 7.8), mit welcher Freiflächen für Bauteile, wie etwa Türen oder Fenster, erstellt werden können. Als Parameter können in die Methode die Abmessungen der erforderlichen Freiflächen eingegeben werden. Auf diese Weise wird auch sicher gestellt, dass anderweitige regulatorische Anforderungen, welche abweichende erforderliche Dimensionen spezifizieren, ebenfalls abgedeckt werden können.

Überdies ist für das Speichern der Resultate als Attribut ein weiteres Hilfsmittel für die VCCL erforderlich. Daher wird die atomare Methode *Write Property* (siehe ebenfalls Tabelle 7.8) eingeführt, welche einen Wert als Attribut für ausgewählte Bauteile in das IFC-Modell schreiben kann. Aus den gegebenen Anforderungen ergibt sich somit das vollständige VCCL-System, welches in Abbildung 8.11 dargestellt ist.

In dem Programm wird nach dem Erstellen der erforderlichen Freiflächenkombinationen geprüft, mit welchen vorhandenen Bauteilen keine Überlappungen bzw. Kollisionen vorliegen. Wenn mindestens eine Variante für eine Tür gültig ist, kann diese als behindertengerecht erachtet werden. Um das Ergebnis der Prüfung in dem Modell zu speichern, wird allen Türen, die als barrierefrei identifiziert wurden, das IFC-Standardattribut *HandicapAccessible* [*Boolean*] in dem Attributsatz *Pset_DoorCommon* entsprechend auf $True$ gesetzt. Analog ergeben sich die Türen, welche nicht als barrierefrei identifiziert wurden, durch eine Mengenoperation und können mit einem $False$-Wert belegt werden. Das resultierende IFC-Modell kann nun abschließend mit Hilfe einer MVD-Spezifikation daraufhin geprüft werden, welche *IfcDoor*-Instanzen einen positiven Wert für die *HandicapAccessible*-Attributdefinition besitzt. Auf diese Weise kann die eigentliche Prüfung gemäß der ursprünglichen Anforderung beschrieben werden.

In Abbildung 8.12 ist als Ergebnis eine Visualisierung der mit Hilfe der VCCL erzeugten Freiflächen, die vor beweglichen Bauteilen (in diesem Falle Türen) eingehalten werden sollen, abgebildet.

In der Folge kann nun auch auf ähnliche Art und Weise die Untersuchung folgen, ob ein Gebäudemodell hinsichtlich der Zugänglichkeit für Rollstuhlfahrer gesamtheitlich geeignet ist. An dieser Stelle sollen die Möglichkeiten aufgezeigt werden, welche sich mittels der VCCL ergeben, nicht aber die Prüfung selbst bis vollständig abgebildet werden. Um die Zugänglichkeit (siehe auch Abbildung 8.8) zwischen den Räumen und die Barrierefreiheit zu untersuchen, ist in Abbildung 8.13 ein entsprechendes VCCL-Programm dargestellt, in welchem mit Hilfe von räumlichen und relationalen Methoden Räume

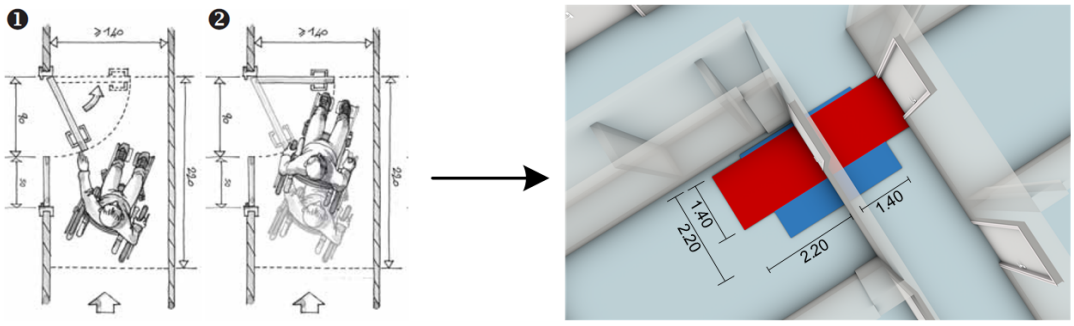


Abbildung 8.10: Erforderliche Kombination von Freiflächen für eine Tür. Mindestens ein Raum (*blau* oder *rot*) vor oder hinter der Tür muss frei sein, damit die Tür als behindertengerecht gelten kann (Ministere du logement et de la ville, 2008-05)

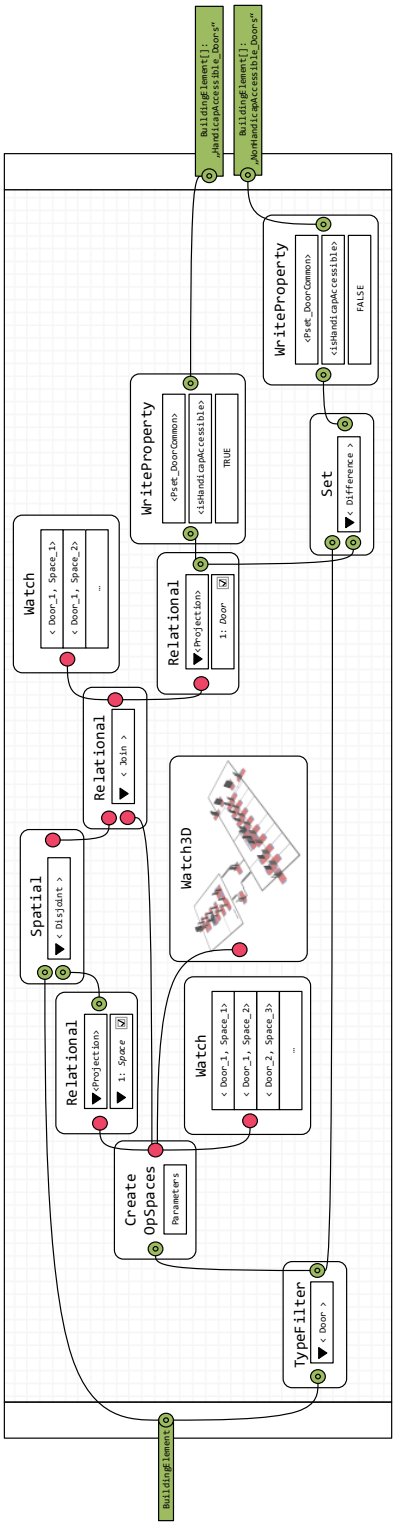


Abbildung 8.11: VCCL-Programm für die Untersuchung, ob die erforderlichen Freiflächen vor Türen eingehalten werden

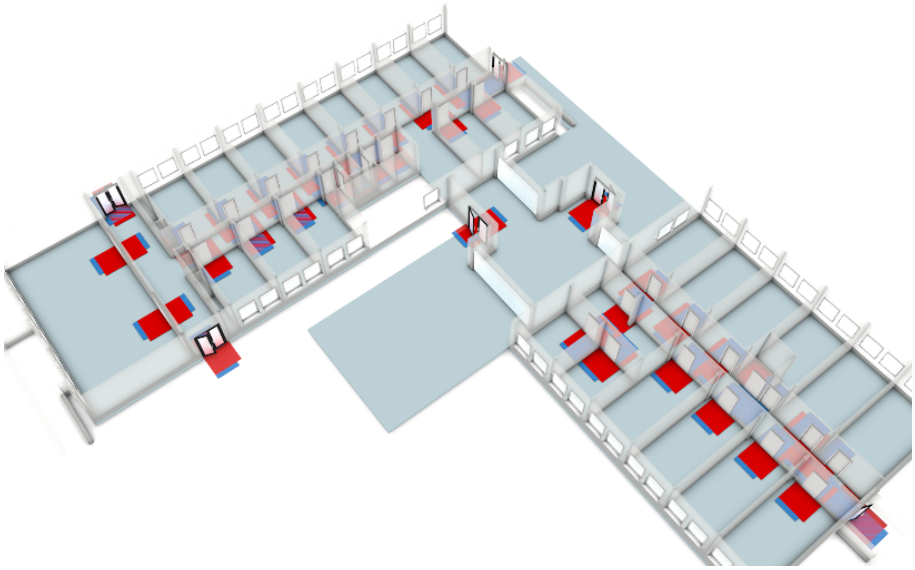


Abbildung 8.12: Visualisierung der mit Hilfe der VCCL erzeugten Freiflächen, die vor beweglichen Bauteilen (in diesem Falle Türen) eingehalten werden sollen

identifiziert werden, welche über Türen miteinander verbunden und zugänglich sind. Die Ergebnisse beschreiben die Zugänglichkeit in Relationen und sind für das bessere Verständnis in Abbildung 8.14 anhand eines Grundrissbeispiels graphisch aufbereitet.

Mit der Vorstellung des Vorverarbeitungssystems wird aufgezeigt, wie die VCCL als hilfreiches Werkzeug die Verarbeitung von für Prüfungen benötigten Informationen transparent beschreibt, sodass ein Anwender gezielt bestimmen kann, welche Informationen als IFC-Modellinhalt generiert werden sollen. Der eigentliche Prüfprozess wird mittels MVD durchgeführt und damit effektiv von der Aufbereitung der Informationen getrennt. Mit der Implementierung der IFC-Vorverarbeitung am Beispiel der Freiflächen wird gezeigt, dass der Ansatz anwendbar ist. Auch wenn in diesem System der eigentliche Anwendungszweck der VCCL nur indirekt erfüllt wird, trägt die Sprache in diesem Szenario wesentlich zu einer Automatisierung der Konformitätsprüfung bei und zeichnet sich durch Flexibilität aus.

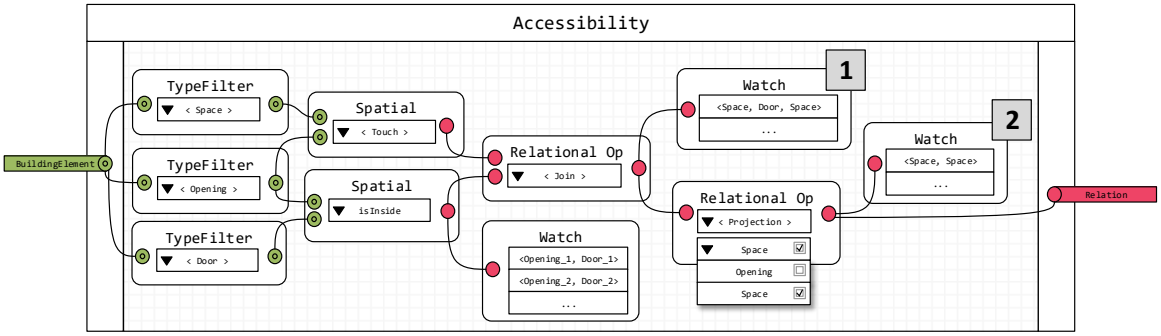


Abbildung 8.13: VCCL-Programm, welches die Erreichbar- und Zugänglichkeit zwischen Räumen untersucht. Die Ergebnisse werden entsprechend in Relationen festgehalten und deren Inhalt ist beispielhaft in Abbildung 8.14 visuell dargestellt.

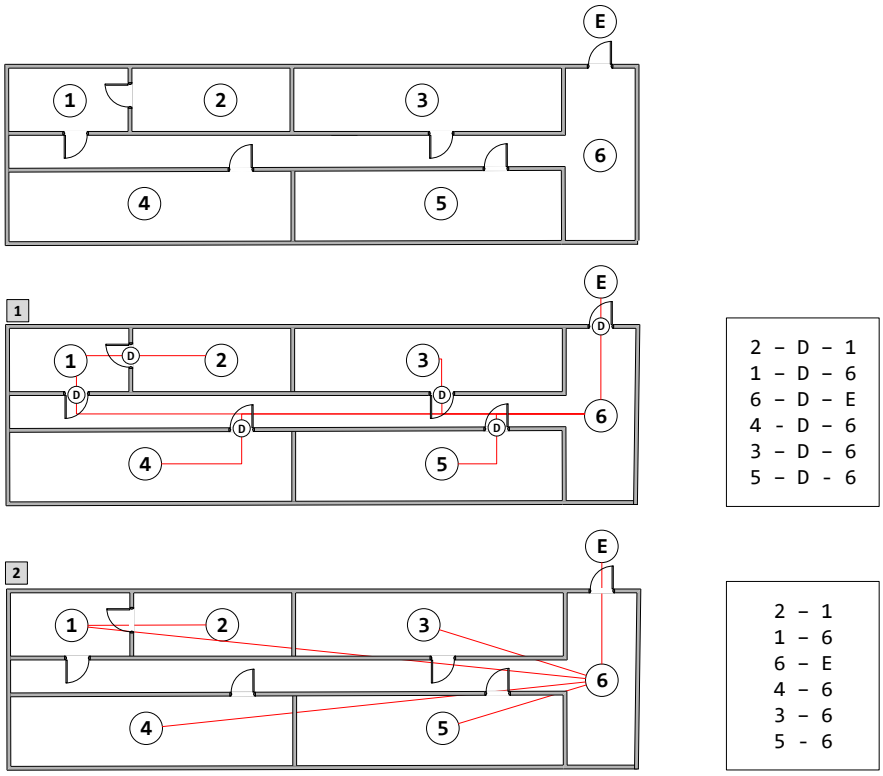


Abbildung 8.14: Visuelle Darstellung der Zugänglichkeitsgraphen, die mit dem Programm in Abbildung 8.13 erstellt wurden, für einen Beispielgrundriss

8.5 Grenzen der VCCL

Nach der Vorstellung diverser Anwendungsfälle der VCCL, welche die Überwindung der Unzulänglichkeiten bisheriger Ansätze (siehe Kapitel 6 und Abschnitt 5.4) adressiert, sollen im Folgenden auch die Grenzen und Einschränkungen für das Konzept der VCCL benannt werden:

(1) Die Ausdruckskraft der VCCL basiert auf dem Katalog der atomaren Methoden. Wie bei den spezifischen Anwendungsfällen in diesem Kapitel zu sehen ist, steht und fällt die Anwendung mit dem Vorhandensein der passenden Methoden. Daraus ergibt sich die Notwendigkeit, dass der Katalog der VCCL wachsen muss, um so den erforderlichen Grad der Ausdruckskraft zu erreichen. Die aktuelle Bibliothek kann nur ein erster Ausgangspunkt sein, da dieser aus allgemeinen sowie ausgewählten spezifischen Anwendungsfällen entstanden ist. Wie bereits in Abschnitt 7.5.1 beschrieben, birgt die Definition von neuen atomaren Methoden hierbei die Gefahr, dass die Anzahl der zur Verfügung stehenden Methoden die Übersichtlichkeit enorm leiden lässt und so die Transparenz der VCCL, die ihre Stärke ist, verloren geht.

(2) Der allgemeine Kritikpunkt an visuellen Sprachen, einfache Abfragen unnötigerweise übermäßig komplex darzustellen, bleibt auch für die VCCL erhalten und kann nicht ohne Weiteres überwunden werden. Zwar helfen Mechanismen wie die Verschachtelung dabei, große Programme auf ein Mindestmaß zu reduzieren, allerdings kann der Ansatz auch hier an Grenzen stoßen.

(3) Die VCCL adressiert Normen und Vorschriften, wie diese heutzutage weit verbreitet sind. Das bedeutet, dass mit dem Ansatz insbesondere beabsichtigt wird, präskriptive Anforderungen zu formalisieren. Wie allerdings bereits in Abschnitt 4.3 ausgeführt wurde, gibt es diverse Gründe, warum der Gebrauch von leistungsbasierten Normen vorzuziehen ist. Nicht zuletzt aus dieser Diskussion heraus gibt es Anzeichen dafür, dass künftige regulatorische Anforderungen vermehrt in diese Richtung formuliert werden. Allerdings hat sich bereits in dem Anwendungsfall in Abschnitt 8.4 gezeigt, dass die VCCL Flexibilität aufweist und auch anderweitig genutzt werden kann.

8.6 Zusammenfassung

Das Kapitel untersucht die in Kapitel 7 vorgestellte VCCL hinsichtlich ihrer Ausdruckskraft und Eignung für die Formalisierung von regulatorischen Anforderungen, wie diese in Normen und Richtlinien des Bauwesens zu finden sind. Hierfür werden regulatorische Anforderungen ausgewählter nationaler Normen vorgestellt und mit Hilfe der VCCL kodiert. Dabei werden unter anderem auch spezifische Anwendungsfälle erläutert, die die Einführung weiterer atomarer Methoden für die VCCL erfordern, um die notwendige Ausdruckskraft für die Formalisierung zu erreichen. In einem letzten Anwendungsfall wird die VCCL zudem hinsichtlich ihrer Eignung als Vorverarbeitungswerkzeug für die transparente Aufbereitung von fehlenden, höherwertigen Bauwerksinformationen untersucht. Auf Grundlage der vorgestellten Anwendungsfälle werden schließlich die Grenzen und Limitationen der VCCL zusammengefasst.

9 Zusammenfassung und Ausblick

In den beiden folgenden Abschnitten sollen zunächst ein Überblick über die Inhalte der vorliegenden Arbeit gegeben werden. Dabei werden die zentralen Kernaussagen herausgearbeitet und zusammengefasst. Die Arbeit schließt mit einem Ausblick zu nachfolgenden möglichen Forschungsfragen, die sich aus der vorliegenden Arbeit ergeben oder aber auf dieser aufbauen.

9.1 Zusammenfassung

Mit Einführung digitaler Methoden, wie insbesondere dem Building Information Model, durchläuft auch die Baubranche eine Transformation. Die BIM-Methode sieht den konsequenten Einsatz digitaler Bauwerksmodelle über den gesamten Lebenszyklus einer gebauten Anlage hinweg vor. War die Baubranche bislang in der Nutzung der Potenziale, die sich durch eine Digitalisierung ergeben, eher zurückhaltend, wird dieses in zunehmender Weise aufgeholt. Dieses spiegelt sich auch in einem voraussichtlichen Marktwachstum von 4,9 Mrd. US-$ im Jahr 2019 auf 8,9 Mrd. US-$ im Jahr 2024 für die Modellierung von Gebäudeinformationen bei einer durchschnittlichen jährlichen Wachstumsrate von 12,7% wider (MarketsandMarkets, 2019). Als Hauptfaktoren für das Wachstum gelten globale Herausforderungen wie etwa eine wachsende Bevölkerung und eine fortschreitende Urbanisierung sowie die Entwicklung neuer Technologien und die zunehmende Akzeptanz von BIM selbst.

Die vorliegende Arbeit leistet in diesem Zusammenhang einen Beitrag, indem sie diverse Konzepte für die Sicherung der daten- und informationstechnischen Qualität digitaler Bauwerksmodelle auf Basis formaler Konformitätsüberprüfungen in unterschiedlicher Detailtiefe vorstellt.

Ausgangspunkt für diese Betrachtung ist, dass mit zunehmender Reife und Etablierung der BIM-Methodik neben dem Erzeugungsprozess der Modellinhalte selbst konsequenterweise die Qualität der erzeugten Inhalte in den Fokus rückt. Jeder beabsichtigte Anwendungsfall erfordert einen spezifischen Grad der geometrischen Detaillierung sowie des Informationsgehalts eines Modells. Nicht erkannte bzw. behobene Fehler führen unweigerlich zu weiteren Fehlern oder gar zu einer Verkettung derselben. Daher müssen nicht nur der Austausch der Modelle zwischen den Projektpartnern organisiert, sondern alle enthaltenen Informationen in regelmäßigen Abständen hinsichtlich mannigfaltiger qualitativer Kriterien geprüft werden. Daraus folgt, dass Qualitäts- und Konformitätsprüfungsprozesse einen essenziellen Baustein bei der digitalen Bauplanung darstellen, um die Modellqualität auf einem konstant hohen Niveau zu halten.

Bei der Prüfung müssen unterschiedlichen Komplexitäts- und Detailstufen, auf welche sich der Inhalt eines Bauwerksmodells hinsichtlich seiner Qualität beziehen kann, unterschieden werden. Die vorliegende Arbeit stellt in diesem Zusammenhang ein Ebenenmodell vor, welches die allgemeine Modellqualität in die Hauptgruppen datentechnische, inhaltliche und gestaltungsplanerische Qualität untergliedert. In jeder dieser Ebenen kann eine Vielzahl von qualitativen Aspekten identifiziert werden.

Die Konformität hinsichtlich regulatorischer Anforderungen, wie diese in Normen und Richtlinien des Bauwesens zu finden sind, wird in diesem Modell der gestaltungsplanerischen Qualität zugeordnet und bildet somit die oberste Ebene. Diese kann erst sinnvoll geprüft werden, wenn die Aspekte der vorhergehenden Ebenen zuvor adressiert wurden. Bauvorschriften dienen nicht nur dazu, sicherzustellen, dass das zu errichtende Gebäude einwandfrei funktioniert, sondern vor allem, dass die Sicherheit der Benutzer gewährleistet ist. Die Vorschriften formulieren den anerkannten Stand der Technik und ergeben sich in erster Linie aus dem geltenden Recht und der Gesetzgebung. Die meisten geltenden regulatorischen Dokumente folgen bei der Formulierung der enthaltenen Anforderungen einem präskriptiven Ansatz. Das bedeutet, dass klar definierte Eigenschaften und Randbedingungen eingefordert werden, nicht aber die Beschaffenheit oder Leistungsfähigkeit des Ergebnisses beschrieben wird. Die Darstellung der Anforderungen kann sehr unterschiedlich erfolgen. Daher finden sich in regulatorischen Dokumenten Fließtextpassagen, graphische Darstellungen, Gleichungen oder aber Datentabellen.

Die Konformitätsprüfung der Gestaltungsplanung hinsichtlich geltender Normen und Richtlinien spielt im Bauwesen eine Schlüsselrolle. Die Qualität der Planungsgrundlagen muss fortwährend auf ihre Richtigkeit und Einhaltung der geltenden Anforderungen überprüft werden. Insbesondere bei der Planung, aber auch bei der Durchführung von Bauprojekten, muss eine Vielzahl von Bauvorschriften beachtet werden. Aktuell ist dieser Kontrollprozess mühsam, umständlich und fehleranfällig, da dieser zumeist manuell auf Basis der zweidimensionalen Planung und iterativ bei jeder Planungsänderung durch den zuständigen Planungsingenieur und der Baugenehmigungsbehörde durchgeführt werden muss. Eine Automatisierung dieses hochrelevanten Prozesses mit Hilfe digitaler Methoden, wie insbesondere dem Building Information Modeling, ermöglicht es, den Arbeitsaufwand und somit die Kosten zu reduzieren und gleichzeitig den Planungsingenieur von monotoner, iterativer Arbeit zu befreien, damit dieser mehr Zeit für zentrale, kreative Arbeiten hat.

Die wesentlichen Herausforderungen bei der technischen Umsetzung einer automatisierten Konformitätsprüfung können an der schematischen Grundstruktur des Gesamtprozesses festgemacht werden, welche sich in vier einzelne Prozessschritte gliedert: die Übersetzung des Regelwissens aus den regulatorischen Anforderungen in ein formales und maschinenlesbares Format, die Aufbereitung der Gebäudemodelldaten für die Prüfung, der eigentliche Prüfprozess und schließlich die Aufbereitung der erhaltenen Ergebnisse für die erforderliche Kommunikation mit den Projektpartnern.

Mit dem Überblick zu den bisherigen Ansätzen des Automated Code Compliance Checking (ACCC) wird in der vorliegenden Arbeit die außerordentliche Relevanz und Bedeutung der Automatisierung der Konformitätsüberprüfung von Regelwerk und Gebäudedatenmodell für die Bauindustrie aufgezeigt. Allerdings weisen diese Forschungsansätze sowie bereits verfügbare Werkzeuge erhebliche Defizite auf: Viele Ansätze stellen die Vorschriften ausschließlich *hard-coded*, also als fest implementierten Programmcode, dar. Auf diese Weise bleibt der eigentliche Prüfprozess für den Prüfingenieur, welche den Programmcode nicht lesen kann, verborgen und produzierte Ergebnisse können von ihm nicht validiert werden. Dieses führt letztlich zu einem gravierenden Mangel an Transparenz und Flexibilität. Andere Ansätze kodieren die Vorschriften übermäßig kompliziert, sodass die Übersetzung ausschließlich von Programmierern übernommen werden kann. Allerdings verfügen Domänenexperten, die im Bauwesen üblicherweise für solche Prozesse verantwortlich sind, über keine fundierten Programmierkenntnisse.

Weitere Ansätze adressieren nur einzelne Aspekte der Konformitätsprüfung, wie zum Beispiel die Formalisierung des Regelwissens. Allerdings ist die Automatisierung der Konformitätsprüfung ein komplexes Problem, welches ganzheitlich angegangen werden sollte.

Es liegt aus diesen Gründen nahe, einen neuen Ansatz zu entwickeln, welcher es ermöglicht, sowohl geometrische als auch nicht-geometrische Randbedingungen und Anforderungen auszudrücken, für Domänenexperten mit begrenztem Softwareentwicklungswissen anwendbar ist und schließlich auch unabhängig von bestimmten Softwareumgebungen eingesetzt werden kann.

Um diesen Herausforderungen zu begegnen, wird in der vorliegenden Arbeit die Visual Code Checking Language (VCCL) eingeführt. Neben der für den beabsichtigen Anwendungszweck geeigneten Ausdruckskraft spielen insbesondere die Anwendbarkeit für typische Anwender sowie die Transparenz der mit Hilfe der VCCL beschriebenen Verarbeitungsprozesse bei Struktur und Gestaltung der Sprache eine wesentliche Rolle. Ausgehend von diesen Aspekten zeichnet sich die VCCL als visuelle, imperative, strikt-typisierte und objektorientierte Programmiersprache aus, welche sich explizit für die formale Beschreibung von Konformitätsprüfungsprozessen im Bauwesen eignet. Visuelle Programmiersprachen stellen im Gegensatz zu textuellen Programmiersprachen die beschriebenen Verarbeitungssysteme graphisch dar und sind somit für Anwender besser verständlich, die über keine oder aber geringe Programmierkenntnisse verfügen. Der Mehrwert visueller Programmiersprachen zeigt sich unter anderem in einer steigenden Zahl von ingenieurtechnischen Anwendungen.

In der VCCL bilden Methodenknoten, Kanten und Schnittstellen die zentralen Elemente, welche auf einer dreiteiligen Benutzeroberfläche zu einem visuellen Informationsfluss zusammengestellt werden. Erstellte VCCL-Programme können gespeichert und zusammengefasst als eigener Methodenknoten in einem anderen VCCL-Programm zugreifbar gemacht werden. Auf diese Weise lassen sich beliebig komplexe Programme erstellen. Überdies kann dieses auch für die Einführung von Kontrollstrukturen wie einer *Foreach*-Iteration und einer konditionalen *If-Else*-Verzweigung genutzt werden.

Auch wenn ein mit der VCCL beschriebenes Programm voll automatisiert ausgeführt werden kann, bietet die visuelle Sprache den zentralen Vorteil, dass ein Anwender die Möglichkeit hat, bei jedem Prozessschritt die erhaltenen Ergebnisse mit Hilfe von Methoden zu visualisieren und so den Prüfprozess auf dessen Integrität hin zu untersuchen.

Den Ausgangspunkt der Ausdruckskraft der VCCL bilden atomare Methodenknoten, welche jeweils wohl definierte Operationen beschreiben und nicht zerlegt werden können. Da allerdings eine atomare Methode eine klar definierte grundsätzliche Operation beschreibt, kann davon ausgegangen werden, dass der Nutzer kein Interesse daran hat, eine tief greifende Einsicht in die jeweilige Verarbeitung zu erhalten. Die atomaren Methoden bilden die Basis der VCCL-Bibliothek, die mit zusammengefassten Programmen beliebig ausgebaut werden kann.

Zu den wesentlichen Kategorien zählen insbesondere die *relationalen* sowie *räumlich-geometrischen* atomaren Methoden, welche es erlauben, komplexe Anforderungen zu formulieren. Die Gruppe der Hilfsmethoden hingegen verleiht der Sprache und somit dem Anwender die Möglichkeit, Inhalte von Prozess- bzw. Verarbeitungsschritten zu visualisieren und Inhalte als Referenz im Modell zu kommunizieren. Der aktuelle Entwicklungsstand der Methodenbibliothek stellt keinen finalen Stand dar, sondern kann bei Bedarf um weitere atomare Methoden erweitert werden.

Auf Basis des beschriebenen Aufbaus wurde die VCCL prototypisch als Demonstrator umgesetzt, mit welchem sich VCCL-Programme für eine praktische Anwendung erstellen lassen.

Um die VCCL hinsichtlich ihrer Ausdruckskraft und Eignung für die Formalisierung von regulatorischen Anforderungen zu untersuchen, wurden Ausschnitte ausgewählter nationaler Normen mit Hilfe der VCCL kodiert. In einem weiteren Anwendungsfall wurde die VCCL zudem hinsichtlich ihrer Eignung als Vorverarbeitungswerkzeug für die transparente Aufbereitung von fehlenden, höherwertigen Bauwerksinformationen untersucht. Hierbei wurden unter anderem auch Fälle erläutert, die die Einführung weiterer atomarer Methoden für die VCCL erfordern, um die notwendige Ausdruckskraft für die Formalisierung zu erreichen. An dieser Stelle zeigt sich, dass die Ausdruckskraft der VCCL maßgeblich auf dem Katalog der atomaren Methoden basiert. Um den für den Anwendungsfall erforderlichen Grad der Ausdruckskraft zu erreichen, muss der Katalog der atomaren VCCL-Methoden also erweitert werden. Dabei birgt die Einführung neuer atomarer Methoden die Gefahr, dass die Anzahl der zur Verfügung stehenden Methoden die Übersichtlichkeit enorm leiden lässt und so die Transparenz der VCCL, die ihre Stärke ist, verloren geht. Und hier liegt gleichzeitig auch ein allgemeiner Kritikpunkt an visuellen Sprachen, dass mit Hilfe dieser einfache Abfragen unnötigerweise übermäßig komplex dargestellt werden. Zwar helfen an dieser Stelle der VCCL Mechanismen wie die Verschachtelung dabei, große Programme auf ein Mindestmaß zu reduzieren, allerdings kann der Ansatz auch hier an Grenzen stoßen.

Mit der Darstellung der Anwendungsfälle konnte die Eignungsfähigkeit der VCCL gezeigt werden die Inhalte von Normen und Vorschriften zu formalisieren, welche vorrangig präskriptive Anforderungen enthalten. Wie allerdings bereits in Abschnitt 4.3 ausgeführt wurde, gibt es gute Gründe, warum der Gebrauch von leistungsbasierten Normen vorzuziehen ist. Nicht zuletzt aus dieser Diskussion heraus gibt es Anzeichen dafür, dass künftige regulatorische Anforderungen vermehrt in diese Richtung formuliert werden.

9.2 Ausblick

In der vorliegenden Arbeit wurde hinreichend dargestellt, dass die Modellprüfung eine zunehmend essenzielle Rolle im Kontext des Building Information Modeling einnimmt. Folglich rückt auch die Automatisierung der Konformitätsprüfung hinsichtlich geltender regulatorischer Anforderungen zunehmend in den Fokus.

In diesem Zusammenhang wird hierbei allerdings lediglich eine Auswahl nationaler regulatorischer Anforderungen mit Hilfe der VCCL untersucht. Die enorme Vielzahl geltender regionaler, nationaler sowie internationaler Vorschriften kann im Rahmen dieser Arbeit nicht abgedeckt werden und bedingt, dass an dieser Stelle weitere Anwendungsfälle anderer Fachbereiche des Bauwesens untersucht werden. In dem Konzept der VCCL sind allerdings Maßnahmen verankert, damit sich diese flexibel erweitern lässt. Durch die Einführung weiterer atomarer Methoden kann die Ausdruckskraft der Sprache ausgebaut werden und so für die Anwendung auf weitere Bereiche erweitert werden. Wenn es darum geht, den Nutzer in seiner kreativen Tätigkeit zu unterstützen, muss darüber nachgedacht werden, inwiefern hier die VCCL behilflich sein kann.

Weiterhin konzentrieren sich die Anwendungsfälle maßgeblich auf die Vorschriften des Hochbaus. Ein anderer wesentlicher Teil des Bauwesens ist der Infrastrukturbau, welcher eigene Anforderungen hinsichtlich geltender Vorschriften, aber insbesondere auch hinsichtlich der Modelle und Modellierungsprozesse selbst, mit sich bringt. An dieser Stelle sind weitergehende Untersuchungen erforderlich und es muss geprüft werden, inwiefern das Konzept zu der VCCL auch auf diesen Bereich anwendbar ist.

Nach der ersten Veröffentlichung des grundlegenden Konzepts zu der VCCL in (Preidel und Borrmann, 2015), wurde der Ansatz, die Herausforderungen des ACCC mit einer visuellen Sprache anzugehen, von anderen Forschern bzw. -gruppen, wie etwa bei Ghannad et al. (2019) oder Kim et al. (2019, 2017a), aufgegriffen. Daraus ergibt sich die Fragestellung, inwiefern sich die verschiedenen Ansätze miteinander verbinden lassen, um so übergreifende Synergieeffekte zu erzielen. Ergebnis einer Zusammenarbeit könnte sein, eine gemeinsame visuelle Sprache so zu vereinheitlichen, dass diese zu einem diszplin- und länderübergreifenden Werkzeug für die Formalisierung regulatorischer Anforderungen wird. Dieses könnte ein Ausgangspunkt zu einem *Esperanto* für die Formulierung regulatorischer Anforderungen darstellen.

Ein zentraler Gedanke hinter der Einführung der VCCL ist es, sich nicht ausschließlich an dem regulatorischen Wissen aus Normen zu orientieren, sondern insbesondere auch den Ingenieuren die Möglichkeit zu geben, ihre Erfahrung und ihr Wissen in den Formalisierungsprozess mit einzubeziehen. Auf Basis der so abgebildeten Erfahrung ließe sich ein Prüfsystem dahingehend erweitern, dass es nicht nur Fehler erkennt, sondern darüber hinaus auch Handlungsempfehlungen ausspricht. Dieses würde also aus dem reinen Prüfsystem ein Entscheidungsunterstützungssystem machen, wie dieses in Abschnitt 3.3 vorgestellt wurde. In diesem Zusammenhang bietet es sich überdies an zu untersuchen, inwiefern Methoden der künstlichen Intelligenz auf das abgebildete Wissen angewendet werden können. Ziel eines solchen Systems könnte sein, auf die Frage eines Anwenders auf Grundlage formalisierten Fachwissens und daraus gezogener logischer Schlüsse entsprechende Antworten zu liefern.

Allgemein stellt sich auch die Frage, inwiefern sich Normen und Richtlinien in Zukunft weiterentwickeln werden. In der Arbeit wurde in diesem Zusammenhang der Unterschied zwischen dem leistungs-basierten und präskriptiven Ansatz bei der Formulierung regulatorischer Anforderungen hinlänglich ausgeführt. In diesem Kontext müsste auch für die VCCL untersucht werden, inwiefern es möglich ist, mit Hilfe einer visuellen Sprache eine gewünschte Leistungsbeschreibung eines Bauwerks zu formalisieren.

A Anhang

Tabelle A.1: Klassifizierung von in Deutschland gültigen Normen (Hudeczek, 2017)

Originalnorm	Erklärung	Bezeichnung
DIN	Nationale deutsche Norm (DIN). DIN Normen existieren auch weiterhin nach der Harmonisierung für Produkte und Dienstleistungen, für die es auf ISO- und EN-Ebene kein notwendiges Äquivalent gibt.	DIN
ISO	Internationale Norm der ISO	ISO
DIN ISO	Nationale deutsche Ausgabe einer unverändert übernommenen ISO-Norm.	ISO
EN	Europäische Norm des CEN. Gilt als eigenständige Norm, wenn eine unveränderte Übernahme einer ISO-Norm als EN ISO-Norm nicht gelingt. EN-Nummern weichen von den ISO-Nummern ab.	EN
DIN EN	Nationale deutsche Ausgabe einer unverändert übernommenen EN-Norm.	EN
EN ISO	Europäische Norm, die unverändert von ISO übernommen wurde. Hier stimmt die EN-Nummer mit der ISO-Nummer überein.	ISO
DIN EN ISO	Nationale deutsche Ausgabe einer unverändert von der ISO übernommenen EN-Norm.	ISO

Tabelle A.2: Auswahl zentraler Normen für das Deutsche Bauwesen (Beuth Verlag, 2018)

Struktur	Beispiele	Titel
01.000 Grundlagen		
01.100 Allgemeines		
01.110 Zeichnungen, Begriffe, Maßgrundlagen	DIN 1356-1 Norm, 1995-02	Bauzeichnungen - Teil 1: Arten, Inhalte und Grundregeln der Darstellung
	DIN EN ISO 4157-1 Norm, 1999-03	Zeichnungen für das Bauwesen - Bezeichnungssysteme - Teil 1: Gebäude und Gebäudeteile (ISO 4157-1:1998); Deutsche Fassung EN ISO 4157-1-199
01.130 Lastannahmen	DIN 1055-2 Norm, 2010-11	Einwirkungen auf Tragwerke - Teil 2: Bodenkenngrößen
	DIN EN 1990 Norm, 2010-12	Eurocode: Grundlagen der Tragwerksplanung; Deutsche Fassung EN 1990:2002 + A1:2005 + A1:2005/AC:2010
	DIN EN 1991-1-1 Norm, 2010-12	Eurocode 1: Einwirkungen auf Tragwerke - Teil 1-1: Allgemeine Einwirkungen auf Tragwerke - Wichten, Eigengewicht und Nutzlasten im Hochbau; [...]
01.140 Toleranzen	DIN 18202 Norm, 2013-04	Toleranzen im Hochbau - Bauwerke
	DIN 18203-3 Norm, 2008-08	Toleranzen im Hochbau - Teil 3: Bauteile aus Holz und Holzwerkstoffen
01.150 Güteüberwachung, Werksbescheinigungen	DIN 488-6 Norm, 2010-01	Betonstahl - Teil 6: Übereinstimmungsnachweis
	DIN EN 197-1 Norm, 2011-11	Zement - Teil 1: Zusammensetzung, Anforderungen und Konformitätskriterien von Normalzement; Deutsche Fassung EN 197-1:2011
01.160 Qualitäts- management, Zertifizierung	DIN 1319-1 Norm, 1995-01	Grundlagen der Meßtechnik - Teil 1: Grundbegriffe
	DIN 1319-4 Norm, 1999-02	Grundlagen der Meßtechnik - Teil 4: Auswertung von Messungen; Meßunsicherheit
	DIN 55350-11 Norm, 2008-05	Begriffe zum Qualitätsmanagement - Teil 11: Ergänzung zu DIN EN ISO 9000:2005
01.200 Vermessung und Geoinformation	DIN 18709-1 Norm, 1995-10	Begriffe, Kurzzeichen und Formelzeichen im Vermessungswesen - Teil 1: Allgemeines
	DIN 18710-1 Berichtigung 1 Norm, 2011-01	Ingenieurvermessung - Teil 1: Allgemeine Anforderungen, Berichtigung zu DIN 18710-1:2010-09
01.220 Photogrammetrie und Fernerkundung	DIN 18716 Norm, 2012-08	Photogrammetrie und Fernerkundung - Begriffe
01.230 Geodätische Instrumente und Geräte	DIN 18703 Norm, 1996-11	Nivellierlatten
01.300 Bauwerksplanung		

01.310 Flächen- und Raumberechnungen, Bau- und Nutzungskosten	DIN 276-1 Norm, 2008-12	Kosten im Bauwesen - Teil 1: Hochbau
	DIN 276-4 Norm, 2009-08	Kosten im Bauwesen - Teil 4: Ingenieurbau
	DIN 277-3 Norm, 2005-04	Grundflächen und Rauminhalte von Bauwerken im Hochbau - Teil 3: Mengen und Bezugseinheiten
01.330 Industriebau- planung	DIN 18225 Norm, 1988-06	Industriebau; Verkehrswege in Industriebauten
01.340 Bauwerksplanung, Sonstiges	DIN 5035-3 Norm, 2006-07	Beleuchtung mit künstlichem Licht - Teil 3: Beleuchtung im Gesundheitswesen
	DIN 13080 Beiblatt 1 Norm, 2003-07	Gliederung des Krankenhauses in Funktionsbereiche und Funktionsstellen - Hinweise zur Anwendung für Allgemeine Krankenhäuser
	DIN 18040-1 Norm, 2010-10	Barrierefreies Bauen - Planungsgrundlagen - Teil 1: Öffentlich zugängliche Gebäude
	DIN 18040-2 Norm, 2011-09	Barrierefreies Bauen - Planungsgrundlagen - Teil 2: Wohnungen
	DIN 18205 Norm, 1996-04	Bedarfsplanung im Bauwesen
01.400 Standsicherheit		
01.410 Mauerwerks-konstruktionen	DIN 1053-4 Norm, 2013-04	Mauerwerk - Teil 4: Fertigbauteile
	DIN V 18580 Vornorm, 2007-03	Mauermörtel mit besonderen Eigenschaften
	DIN EN 1996-1-1 Norm, 2013-02	Eurocode 6: Bemessung und Konstruktion von Mauerwerksbauten - Teil 1-1: Allgemeine Regeln für bewehrtes und unbewehrtes Mauerwerk [...]
01.400 Standsicherheit		
01.420 Stahlbeton- und Spannbetonkonstruktionen	DIN 1045-2 Norm, 2008-08	Tragwerke aus Beton, Stahlbeton und Spannbeton - Teil 2: Beton - Festlegung, Eigenschaften, Herstellung und Konformität - Anwendungsregeln zu DIN EN 206-1
	DIN 1045-100 Norm, 2011-12	Bemessung und Konstruktion von Stahlbeton- und Spannbetontragwerken - Teil 100: Ziegeldecken
	DIN EN 1992-1-1 Norm, 2011-01	Eurocode 2: Bemessung und Konstruktion von Stahlbeton- und Spannbetontragwerken - Teil 1-1: Allgemeine Bemessungsregeln und Regeln für den Hochbau [...]
01.430 Stahl- und Aluminium-konstruktionen	DIN EN 1993-1-1 Norm, 2010-12	Eurocode 3: Bemessung und Konstruktion von Stahlbauten - Teil 1-1: Allgemeine Bemessungsregeln und Regeln für den Hochbau [...]
01.440 Holzkonstruktionen	DIN EN 383 Norm, 2007-03	Holzbauwerke - Prüfverfahren - Bestimmung der Lochleibungsfestigkeit und Bettungswerte für stiftförmige Verbindungsmittel [...]
	DIN 1052-10 Norm, 2012-05	Herstellung und Ausführung von Holzbauwerken - Teil 10: Ergänzende Bestimmungen
01.450 Verbundkonstruktionen	DIN 18168-2 Norm, 2008-05	Gipsplatten-Deckenbekleidungen und Unterdecken - Teil 2: Nachweis der Tragfähigkeit von Unterkonstruktionen und Abhängern aus Metall
01.460 Gründungen	DIN 1054 Norm, 2010-12	Baugrund - Sicherheitsnachweise im Erd- und Grundbau - Ergänzende Regelungen zu DIN EN 1997-1
	DIN 4085 Norm, 2011-05	Baugrund - Berechnung des Erddrucks

01.500 Nutzungssicherheit		
01.510 Unfallverhütung	DIN 33404-3 Norm, 1982-05	Gefahrensignale für Arbeitsstätten; Akustische Gefahrensignale; Einheitliches Notsignal; Sicherheitstechnische Anforderungen, Prüfung
	DIN 58125 Norm, 2002-07	Schulbau - Bautechnische Anforderungen zur Verhütung von Unfällen
	ETB Absturzsicherung Technische Regel, 1985-06	ETB-Richtlinie "Bauteile die gegen Absturz sichern"Fassung 1985-06
01.500 Nutzungssicherheit		
01.520 Instandhaltung	DIN 31051 Norm, 2012-09	Grundlagen der Instandhaltung
	DIN 32736 Norm, 2000-08	Gebäudemanagement - Begriffe und Leistungen
	DIN EN 15221-1 Norm, 2007-01	Facility Management - Teil 1: Begriffe; Deutsche Fassung EN 15221-1:2006
01.600 Schutzmaßnahmen		
01.610 Brandschutz	DIN 4102-16 Norm, 2015-09	Brandverhalten von Baustoffen und Bauteilen - Teil 16: Durchführung von Brandschachtprüfungen [...]
	DIN 18232-5 Norm, 2012-11	Rauch- und Wärmefreihaltung - Teil 5: Maschinelle Rauchabzugsanlagen (MRA); Anforderungen, Bemessung [...]
01.620 Wärmeschutz	DIN 4108-2 Norm, 2013-02	Wärmeschutz und Energie-Einsparung in Gebäuden - Teil 2: Mindestanforderungen an den Wärmeschutz [...]
	DIN EN 16012 Norm, 2015-05	Wärmedämmstoffe für Gebäude - Reflektierende Wärmedämm-Produkte - Bestimmung der Nennwerte der wärmetechnischen Eigenschaften; Deutsche Fassung EN 16012:2012+A1:2015 [...]
01.630 Gesamtenergie- effizienz von Bauwerken	DIN V 18599 Beiblatt 2 Vornorm, 2012-06	Energetische Bewertung von Gebäuden - Berechnung des Nutz-, End- und Primärenergiebedarfs für Heizung, Kühlung, Lüftung, Trinkwarmwasser und Beleuchtung [...]
01.640 Feuchteschutz (Abdichtung)	DIN 4108-3 Norm, 2014-11	Wärmeschutz und Energie-Einsparung in Gebäuden - Teil 3: Klimabedingter Feuchteschutz - Anforderungen, Berechnungsverfahren und Hinweise für Planung und Ausführung [...]
01.660 Erschütterungs-, Erdbebenschutz	DIN EN 1998-1/A1 Norm, 2013-05	Eurocode 8: Auslegung von Bauwerken gegen Erdbeben - Teil 1: Grundlagen, Erdbebeneinwirkungen und Regeln für Hochbauten; Deutsche Fassung EN 1998-1:2004/A1:2013
01.670 Schutz von Baustoffen und Bauteilen	DIN EN 1504-5 Norm, 2013-06	Produkte und Systeme für den Schutz und die Instandsetzung von Betontragwerken - Definitionen, Anforderungen, Qualitätsüberwachung und Beurteilung der Konformität - Teil 5: Injektion von Betonbauteilen; [...]
	DIN EN 16242 Norm, 2013-03	Erhaltung des kulturellen Erbes - Verfahren und Geräte zur Messung der Luftfeuchte und des Austausches von Feuchtigkeit zwischen Luft und Kulturgut; Deutsche Fassung EN 16242:2012 [...]
01.680 Hygiene, Gesundheit, Umweltschutz	DIN 4102-16 Norm, 2015-09	Brandverhalten von Baustoffen und Bauteilen - Teil 16: Durchführung von Brandschachtprüfungen [...]
	DIN 18232-5 Norm, 2012-11	Rauch- und Wärmefreihaltung - Teil 5: Maschinelle Rauchabzugsanlagen (MRA); Anforderungen, Bemessung [...]

Tabelle A.3: Regelvorlagen, welche dem Nutzer im Solibri Model Checker zur Verfügung stehen, mit entsprechender Kurzbeschreibung (Solibri Inc., 2017)

SOL	Name	Beschreibung
1	General Intersection Rule	This rule checks intersections of components. The user can configure what components this rule is checking, and how.
9	Property Values Must Be from Agreed List	This rule checks that only property values that have been agreed upon are used in the model.
11	Model Should Have Components	This rule checks that the model contains components of a selected types. It also checks that components have a construction type.
17	Layer of Component Must Be from Agreed List	This rule checks that components are located on the correct layers.
19	Spaces Must Have Enough Window Area	This rule checks that ratio between window area and area of the space is within limits.
21	Components Must Have Unique Identifier	This rule checks that every component has a unique identifier (in the whole model, in the building storey or in the same space group). It also checks that identifiers are correct (when this is required).
23	Components Must Touch Other Components	This rule checks that components touch surfaces of other components above or below them.
25	Components Must Be Connected to Spaces	This rule checks that exterior components are connected to one and interior components to two spaces. Checks doors, windows and openings.
36	Space Requirements	This rule checks that the model contains a given number of spaces with a given space type, name or number and area, e.g. 10 office spaces with an area between 10m2 +/- 5%.
37	Total Spaces Area On Each Floor	This rule checks that the total area of spaces on each floor is inside given limits.
38	Space Count On Each Floor	This rule checks that each floor has a given number of spaces of a given type, e.g that there are 10 office spaces on the ground floor. Only the given space types are checked; unlisted space types are ignored.
111	Floor and Gross Area Analysis	This rule checks and reports various figures related to floors. The rule requires either Gross Area Compartments (defined in Compartmentation View), or existence of a 'gross area space' in each floor. 'Gross area space' is a space component, which represents the gross area of the floor and it groups all other spaces in the floor.
132	Space Area	This rule checks that area of specified spaces is inside set limits.
161	Distances Between Spaces	This rule checks that distances between spaces follow the given requirements.
162	Spaces Must Be Included in Space Groups	This rule checks that all spaces in the model are included in some space group.
171	Component Property Values Must Be Consistent	This rule checks that the components in the model or in the same floor have the same property values.
172	Fire Walls Must Have Correct Wall, Door, and Window Type	This rule checks that all walls between different fire zones have a correct type and that doors and windows in these walls are fire resisting. It also checks that fire resisting wall, door and window types are not used elsewhere.
175	Space Group Containment	This rule checks that all space groups, which contains spaces, has required amount of required type of spaces.

176	Model Structure	This rule checks that the model includes a building and floors with unique names. It also checks, that all components are contained in a floor and all floors have components. It also checks that windows and doors are connected to walls.
179	Escape Route Analysis	This rule checks that it is possible to exit safely from the building in case of fire or other emergency. The building must have sufficient amount of suitable located exit passage ways that have sufficient capacity, so that exit time is not dangerously long.
190	Fire Compartment Are Must Be within Limits	This rule checks that area of fire compartments is within limits.
191	Spaces Must be Included in Fire Compartments	This rule checks that all spaces in the model are included in fire compartments.
202	Space Validation	This rule checks that space geometry and location are correct. It checks that boundaries are near walls, columns or other objects, and space is touching a slab surface above and below itself. It also checks space height and intersections with other components.
203	Required Property Sets	This rule checks that the model contains required property sets and properties. It can also checks that the properties have (or don't have) a value and the type of the value is acceptable.
206	Model Comparison	This rule compares two models and shows the differences of them.
207	Accessible Ramp Rule	This rule checks the accessibility of ramps from different perspectives. It checks the ramp slope, length, width, and free spaces at the beginning and at the end of a ramp. It also checks the dimensions of intermediate landings.
208	Accessible Door Rule	This rule checks the accessibility of door from different perspectives. It checks the dimensions, glazing ratios, opening directions, and free spaces of the door. To use this rule the spaces you must classify spaces by usage.
209	Free Floor Space	This rule checks that the spaces have enough free floor space.
210	Accessible Stair Rule	This rule checks the accessibility of stairs from different perspectives. It checks the number of steps, dimensions of steps, dimensions of intermediate landings, free space at the beginning, and at the end of the stair. Head clearance above and under the stair are also checked.
211	Accessible Window Rule	This rule is designed to check the accessibility of windows from different perspectives. Currently it checks only the height of sill. To use this rule the spaces you must classify spaces by usage.
212	Building Envelope Validation	This rule checks that building envelope defined in the model (ref. Building Envelope property in the wall Info view) is same as building envelope surrounding gross area spaces and/or spaces in the model.
213	Shelf Running Metre Rule	This rule checks that running metres of shelves. Rule also creates a report all storage spaces with their shelf running metres.
215	Allowed Profiles	This rule checks that only listed beam and column profiles are used.
216	Wall Validation	This rule checks wall geometry and dimensioning. The rule has dimensioning requirements for distances of windows, doors, openings, and wall edges. There can be also limitations to accepted wall geometry type. The direction of extrusion geometry can be limited.
218	Element Hole Validation Rule	This rule checks that element holes are invalid locations.
220	Floor Distance	This rule checks vertical distances between components.

221	Wall Distance	This rule checks that distances between walls are in acceptable range.
222	Component Distance	This rule checks components distance between each other.
223	Structural Components Fire in Architectural Ones	This rule checks that components in structural model fit inside architectural model components.
224	Architectural Components Are Filled	This rule checks that specific Architectural components are filled with structural components.
225	Number of Components in Space	This rule checks that there are required number of components in space.
226	Free Are in Front of Components	This rule checks there is no blocking components in front of certain components.
228	Floor Names Must be Consecutive	This rule checks that floor names are numeric and consecutive.
230	Property Rule Template with Component Filters	This rule checks only components that pass the filters in the "Components to Check"table. The "Requirements"table lists the requirements for the components. Both of these tables can contain at least one filter.
231	Comparison Between Property Values	This rule is used to compare the values of two properties attached to a component.
232	Manual Checking Rule	This rule creates issues defined in rule parameters. This is rule can be used if there exist cases that cannot be checked yet using existing rules.
233	Allowed Beam Intersections	This rule checks intersections with beams and other components so, that it is allowed to go through a beam in specified area. Components (typically pipes and ducts) which go through the beam in allowed area will not create any issue.
234	Component Inside Component	This rule checks distances between component surfaces, when a components are inside each other's.
235	Relative Number	This rule checks relative amounts of components in a specific location. For example, check that for each ten parking lots in a garage, there has to be at least one accessible parking lot.
236	Horizontal Structures must be Guarded against Falling	This rule checks that it is not possible to fall from horizontal components, such as slabs. This rule checks that horizontal components are surrounded by vertical components, such as walls, or railings. If no vertical component exists on the edge of a horizontal component, another horizontal component needs to continue and the drop to the other horizontal component must not be more than specified.
237	Parking Rule	This rule checks the size of the parking spaces. The checking can be limited to check only parking spaces with a specific orientation to the parking aisle or with specific obstructions nearby.
238	Accessible Route Rule	This rule checks the accessible routes. The checking requires the accessible route modeled as a component. The rule can check the clear width of the route and components such as doors, stairs, and ramps within the route. The rule checks also the connections from an accessible route to the accessible spaces and accessible elevators.

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

AAMODT, A. und NYGÅRD, M. (1995). Different roles and mutual dependencies of data, information, and knowledge — An AI perspective on their integration. In: *Data & Knowledge Engineering*, 16(3):191–222.

ACHARYA, N. K., LEE, Y.-D. und KIM, J.-S. (2006). Design Errors: Inefficiency or Carelessness of Designer? In: *Journal of Performance of Constructed Facilities*, 20(2):192–195.

AEC3 GMBH (2017). BIM Anforderungs- und Qualitätsmanagement mit BIMQ. URL: `http://www.aec3.com/de/kompetenzen/BIM-Q.htm` (Letzter Zugriff am: 15.01.2018).

AEC3 UK (2017). International Code Council. URL: `http://www.aec3.com/en/5/5_013_ICC.htm` (Letzter Zugriff am: 01.10.2017).

AHO, A. V., SETHI, R. und ULLMAN, J. D. (1999). *Compilerbau*. Internationale Computer-Bibliothek. Oldenbourg, München, 2. durchges. Auflage.

ALBERT, A. und SCHNEIDER, K.-J. (2016). *Bautabellen für Ingenieure: Mit Berechnungshinweisen und Beispielen*. 22 Auflage.

ALLPLAN GMBH (2016). Bimplus. URL: `https://www.bimplus.net/` (Letzter Zugriff am: 08.03.2016).

AMERICAN INSTITUTE OF STEEL CONSTRUCTION (2015). AISC Homepage. URL: `www.aisc.org` (Letzter Zugriff am: 10.01.2018).

ANGERMANN, A. (2009). *MATLAB - Simulink - Stateflow: Grundlagen, Toolboxen, Beispiele*. Oldenbourg, München, 6. aktualisierte Auflage. URL: `http://www.oldenbourg-link.com/doi/book/10.1524/9783486595468`.

ARNOLD, P. (2009). Information und Wissen. URL: `http://www.informatik.uni-leipzig.de/~graebe/Texte/Arnold-09.pdf` (Letzter Zugriff am: 01.12.2017).

AUTODESK INC. (2018). Dynamo BIM. URL: `http://dynamobim.org/` (Letzter Zugriff am: 8.5.2018).

BAINBRIDGE, L. (1983). Ironies of automation. In: *Automatica*, 19(6):775–779.

BAUMANNS, T., FREBER, P.-S., SCHOBER, K.-S. und KIRCHNER, F. (2016). Bauwirtschaft im Wandel: Trends und Potenziale bis 2020. URL: `https://www.rolandberger.com/publications/publication_pdf/roland_berger_hvb_studie_bauwirtschaft_20160415_1_.pdf` (Letzter Zugriff am: 15.12.2017).

BCA SINGAPORE (2006). CORENET e-Plan Check System: The Importance of IFC for Implementation in Singapore. URL: `http://www.jacic.or.jp/acit/round2-teo2.pdf` (Letzter Zugriff am: 15.02.2018).

BCA SINGAPORE (2013). Singapore BIM Guide. URL: `https://www.corenet.gov.sg/media/586132/Singapore-BIM-Guide_V2.pdf` (Letzter Zugriff am: 04.10.2016).

BCA SINGAPORE (2014). CORENET. URL: `http://www.corenet.gov.sg/` (Letzter Zugriff am: 01.01.2014).

BEACH, T. H., REZGUI, Y., LI, H. und KASIM, T. (2015). A rule-based semantic approach for automated regulatory compliance in the construction sector. In: *Expert Systems with Applications*, 42(12):5219–5231.

BECKER, J. (2013). *Die Digitalisierung von Medien und Kultur*. Springer Fachmedien Wiesbaden, Wiesbaden.

BECKER, R. (2008). Fundamentals of performance-based building design. In: *Building Simulation*, 1(4):356–371.

BEETZ, J., BORRMANN, A. und WEISE, M. (2015). Prozessgestützte Defintion von Modellinhalten. In: BORRMANN, A., KÖNIG, M., KOCH, C. und BEETZ, J. (Herausgeber) *Building Information Modeling*, VDI-Buch, 129–147. Springer Vieweg, Wiesbaden.

BEETZ, J., DE LAAT, R., VAN BERLO, L. und VAN DEN HELM, P. (2010). bimserver.org – An Open Source IFC Model Server. In: *CIB W78 2010: 27th International Conference*.

BEETZ, J., DE VRIES, B. und VAN LEEUWEN, J. P. (2007). RDF-based distributed functional part specifications for the facilitation of service-based architecture. In: REBOLJ, D. (Herausgeber) *Bringing ITC knowledge to work*. Faculty of Civil Engineering, Maribor. URL: http://itc.scix.net/data/works/att/w78-2007-028-087-beetz.pdf.

BEETZ, J., VAN LEEUWEN, J. und DE VRIES, B. (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(01):89.

BENTLEY SYSTEMS (2019). Generative Components. URL: https://www.bentley.com/de/products/product-line/modeling-and-visualization-software/generativecomponents (Letzter Zugriff am: 01.04.2020).

BEUTH VERLAG (2018). Baurelevante DIN-Normen. URL: https://www.vob-online.de/de/service/baunormen-katalog/baurelevante-din-normen (Letzter Zugriff am: 15.01.2019).

BEW, M. und RICHARDS, M. (2008). Bew-Richards BIM maturity model: BuildingSMARTConstruct IT Autumn Members Meeting, Brighton. URL: http://www.ukbimalliance.org/media/1050/ukbima_bimreview_past_present_future_20161019-1.pdf (Letzter Zugriff am: 15.02.2018).

BIMFORUM (2017). Level of Development Sepcification Guide. URL: http://bimforum.org/wp-content/uploads/2017/11/LOD-Spec-2017-Guide_2017-11-06-1.pdf (Letzter Zugriff am: 10.11.2017).

BIZER, C., HEATH, T. und BERNERS-LEE, T. (2009). Linked data-the story so far. URL: http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf (Letzter Zugriff am: 10.12.2016).

BLANKENBACH, J. (2015). Bauwerksvermessung für BIM. In: BORRMANN, A., KÖNIG, M., KOCH, C. und BEETZ, J. (Herausgeber) *Building Information Modeling*, VDI-Buch, 343–362. Springer Vieweg, Wiesbaden.

BLOCH, T. und SACKS, R. (2018). Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models. In: *Automation in Construction*, 91:256–272.

BMVI (2015a). Reformkommission Bau von Großprojekten: Komplexität beherrschen – kostengerecht, termintreu und effizient: Endbericht. URL: https://www.bmvi.de/SharedDocs/DE/Publikationen/G/reformkommission-bau-grossprojekte-endbericht.pdf?__blob=publicationFile (Letzter Zugriff am: 12.12.2017).

BMVI (2015b). Stufenplan Digitales Planen und Bauen. URL: https://www.bmvi.de/SharedDocs/DE/Publikationen/DG/stufenplan-digitales-bauen.html (Letzter Zugriff am: 15.01.2018).

BMWI (2013). Mensch-Technik-Interaktion: Leitfaden für Hersteller und Anwender. URL: http://www.autonomik.de/documents/AN_Band_3_MTI_bf_130325.pdf (Letzter Zugriff am: 17.11.2017).

BÖHM, C. und JACOPINI, G. (1966). Flow diagrams, turing machines and languages with only two formation rules. In: *Communications of the ACM*, 9(5):366–371.

BOPALGNI, M. (2016). The many faces of LOD. URL: http://www.bimthinkspace.com/2016/07/the-many-faces-of-lod.html (Letzter Zugriff am: 21.02.2018).

BORRMANN, A. (2007). *Computerunterstützung verteilt-kooperativer Bauplanung durch Integration interaktiver Simulationen und räumlicher Datenbanken*. Dissertation, Technische Universität München, München. URL: https://mediatum.ub.tum.de/618188 (Letzter Zugriff am: 18.01.2018).

BORRMANN, A., BEETZ, J., KOCH, C. und LIEBICH, T. (2015a). Industry Foundation Classes – Ein herstellerunabhängiges Datenmodell für den gesamten Lebenszyklus eines Bauwerks. In: BORRMANN, A., KÖNIG, M., KOCH, C. und BEETZ, J. (Herausgeber) *Building Information Modeling*, VDI-Buch, 83–127. Springer Vieweg, Wiesbaden.

BORRMANN, A., KÖNIG, M., KOCH, C. und BEETZ, J. (Herausgeber) (2015b). *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*. VDI-Buch. Springer Vieweg, Wiesbaden.

BORRMANN, A., KÖNIG, M., KOCH, C. und BEETZ, J. (Herausgeber) (2018). *Building Information Modeling: Technology Foundations and Industry Practice*. Springer International Publishing, Cham. URL: http://dx.doi.org/10.1007/978-3-319-92862-3.

BORRMANN, A. und RANK, E. (2009). Specification and implementation of directional operators in a 3D spatial query language for building information models. In: *Advanced Engineering Informatics*, 23(1):32–44.

BORRMANN, A., SCHRAUFSTETTER, S. und RANK, E. (2009). Implementing Metric Operators of a Spatial Query Language for 3D Building Models: Octree and B-Rep Approaches. In: *Journal of Computing in Civil Engineering*, 23(1):34–46.

BOUZIDI, K. R., FIES, B., FARON-ZUCKER, C., ZARLI, A. und LE THANH, N. (2012). Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry. In: *Future Internet*, 4(4):830–851.

BRAMER, M. (2013). *Logic Programming with Prolog*. Springer London, London, 2. Auflage. URL: http://dx.doi.org/10.1007/978-1-4471-5487-7.

BRICH, S. (Herausgeber) (2014). *Gabler Wirtschaftslexikon*. Springer Gabler, Wiesbaden, 18. aktualisierte und erw. Auflage.

BRITISH STANDARDS INSTITUTION (2013). *PAS 1192-2:2014: Specification for information management for the capital/delivery phase of construction projects using building information modelling*. URL: https://shop.bsigroup.com/ProductDetail?pid=000000000030281435.

BRITISH STANDARDS INSTITUTION (2014). *PAS 1192-3:2014: Specification for information management for the operational phase of assets using building information modelling*. URL: https://shop.bsigroup.com/ProductDetail?pid=000000000030311237.

BUILDINGSMART (2007). IFC2x3 TC1: FC Specification - IFC2x Edition 3 Technical Corrigendum 1. URL: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/, (Letzter Zugriff am: 01.10.2017).

BUILDINGSMART (2016a). BCF introduction. URL: http://www.buildingsmart-tech.org/specifications/bcf-releases (Letzter Zugriff am: 14.03.2016).

BUILDINGSMART (2016b). MVD Documentation. URL: http://www.buildingsmart-tech.org/specifications/ifc-view-definition (Letzter Zugriff am: 13.05.2016).

BUILDINGSMART (2016c). Regulatory Room. URL: http://buildingsmart.org/standards/standards-organization/rooms/regulatory-room/ (Letzter Zugriff am: 25.08.2016).

BUILDINGSMART (2017a). IFC Releases. URL: http://www.buildingsmart-tech.org/specifications/ifc-releases (Letzter Zugriff am: 15.02.2018).

BUILDINGSMART (2017b). ifcXML Overview. URL: http://www.buildingsmart-tech.org/specifications/ifcxml-releases (Letzter Zugriff am: 01.07.2017).

BUILDINGSMART (2017c). Information Delivery Manuals. URL: http://iug.buildingsmart.org/idms/ (Letzter Zugriff am: 10.01.2017).

BUILDINGSMART (2017d). Report on Open Standards for Regulations, Requirements and Recommendations Content. URL: https://buildingsmart-1xbd3ajdayi.netdna-ssl.com/wp-content/uploads/2017/11/17-11-08-Open-Standards-for-Regulation.pdf (Letzter Zugriff am: 13.11.2017).

BUILDINGSMART (2017e). simple ifcXML. URL: http://www.buildingsmart-tech.org/specifications/ifcxml-releases/simple-ifcxml (Letzter Zugriff am: 15.02.2018).

BURNETT, M. M. und BAKER, M. J. (1994). A Classification System for Visual Programming Languages. In: *Journal of Visual Languages & Computing*, 5(3):287–300.

CATARCI, T. und SANTUCCI, G. (1995). Are Visual Query Languages Easier to use than traditional ones? : an Experimental Proof. In: KIRBY, M. A. R. (Herausgeber) *People and computers X*, Cambridge programme on human-computer interaction. Cambridge Univ. Pr, Cambridge.

CHAIR OF COMPUTATIONAL MODELING AND SIMULATION (2016). TUM.CMS.VplControl. URL: https://github.com/tumcms/TUM.CMS.VPLControl (Letzter Zugriff am: 20.12.2018).

CHEN, Y. und DE LUCA, G. (2016). VIPLE: Visual IoT/Robotics Programming Language Environment for Computer Science Education. In: *2016 IEEE 30th International Parallel and Distributed Processing Symposium workshops*, 963–971. Ieee, Piscataway, NJ.

CHENG, C. P., LAU, G. T. und LAW, K. H. (2007). Mapping regulations to industry-specific taxonomies. In: WINKELS, R. (Herausgeber) *Proceedings of the 11th international conference on Artificial intelligence and law*, 59. ACM, New York, NY.

CHO, Y. und CHOI, J. (2014). A Study on Utilization 'ABIMO CHECKER' for BIM and Indoor Spatial Information Built to Integrated Platform. In: *Journal of Industrial and Intelligent Information*, 3(3).

CODD, E. F. (1970). A relational model of data for large shared data banks. In: *Communications of the ACM*, 13(6):377–387.

CODD, E. F. (1991). *The relational model for database management: Version 2*. Addison-Wesley, Reading, Mass., 2. kor. Auflage.

COMPUTERWORKS (2018). Vectorworks Marionette - Python Scripting. URL: https://www.computerworks.de/produkte/vectorworks/vectorworks-architektur/marionette.html (Letzter Zugriff am: 8/5/2018).

CONSTRUCTION SPECIFICATIONS INSTITUTE (2016a). MasterFormat. URL: https://www.csiresources.org/practice/standards/masterformat (Letzter Zugriff am: 15.12.2016).

CONSTRUCTION SPECIFICATIONS INSTITUTE (2016b). UniFormat. URL: https://www.csiresources.org/practice/standards/uniformat (Letzter Zugriff am: 15.02.2018).

COSTA, G. und PAUWELS, P. (2015). Building product suggestions for a BIM model based on rule sets and a semantic reasoning engine. 98–107. URL: https://biblio.ugent.be/publication/6890587/file/6974113.pdf.

DASSAULT SYSTEMES (2016). End-to-end collaboration enabled by BIM level 3: An industry approach based on best practices from manufacturing: White Paper. URL: https://www.3ds.com/industries/architecture-engineering-construction/resource-center/white-papers/end-to-end-collaboration-enabled-by-bim-level-3/ (Letzter Zugriff am: 17.11.2017).

DAUM, S. (2018). *Konzeption einer raum-zeitlichen Anfragesprache für die Analyse und Prüfung von 4D-Gebäudeinformationsmodellen*. Dissertation. URL: https://mediatum.ub.tum.de/doc/1381857/1381857.pdf (Letzter Zugriff am: 15.06.2018).

DAUM, S. und BORRMANN, A. (2013). Checking Spatio-Semantic Consistency of Building Information Models by means of a Query Language. In: DAWOOD, N. und KASSEM, M. (Herausgeber) *Proceedings of the 13th International Conference on Construction Applications of Virtual Reality, 30-31 October 2013, London, UK*. Teesside University. URL: http://itc.scix.net/data/works/att/convr-2013-49.pdf.

DAUM, S. und BORRMANN, A. (2014). Processing of Topological BIM Queries using Boundary Representation Based Methods. In: *Advanced Engineering Informatics*, 28(4):272–286.

DE FARIAS, T. M., ROXIN, A. und NICOLLE, C. (2015). Semantic Web Technologies For Implementing Cost-effective and Interoperable Building Information Modeling.

DELIS, E. A. und DELIS, A. (1995). Automatic Fire-Code Checking Using Expert-System Technology. In: *Journal of Computing in Civil Engineering*, 9(2):141–156.

DEUTSCHE BAHN (2016). BIM–Vorgaben: BIM-Methodik - Digitales Planen und Bauen. URL: https://www1.deutschebahn.com/file/sus-infoplattform/7760216/pUljufd9SywWtTUKMzNblbSkhcE/10443952/data/Vorgaben_zur_Anwendung_der_BIM_Methodik_V2.2.pdf (Letzter Zugriff am: 10.01.2018).

DEUTSCHE BAHN (2017). Regelwerke. URL: http://fahrweg.dbnetze.com/fahrweg-de/kunden/nutzungsbedingungen/regelwerke/betrieblich-technisch_regelwerke/14420616/regelwerke.html?start=0 (Letzter Zugriff am: 15.02.2018).

DEUTSCHES INSTITUT FÜR BAUNORMUNG (2017). Was ist die Musterbauordnung? URL: https://www.dibt.de/de/Zulassungen/abZ-FAQ-Frage-5.html (Letzter Zugriff am: 15.02.2018).

DIECKMANN, A. und RUSSELL, P. (2014). The Truth Is In The Model: Utilizing Model Checking to Rate Learning Success in BIM Software Courses: Proceedings of the 32nd eCAADe Conference. URL: http://publications.rwth-aachen.de/record/464000 (Letzter Zugriff am: 10.01.2016).

DIMYADI, J. und AMOR, R. (2013). Automated Building Code Compliance Checking – Where is it at ? In: *Proceedings of CIB WBC 2013*, 172–185.

DIMYADI, J., AMOR, R. und SPEARPOINT, M. (2016a). Using BIM to Support Simulation of Compliant Building Evacuation. In: CHRISTODOULOU, S. (Herausgeber) *eWork and eBusiness in Architecture: ECPPM 2016 : Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016), Limassol, Cyprus, 7-9 September 2016*. CRC Press.

DIMYADI, J., CLIFTON, G. C., SPEARPOINT, M. und AMOR, R. (2014). Computer-aided Compliance Audit to Support Performance-based Fire Engineering Design. In: *SFPE 10th International Conference on Performance-based Codes and Fire Safety Design Method, Gold Coast, Australia*. URL: https://www.cs.auckland.ac.nz/~trebor/papers/DIMY14C.pdf.

DIMYADI, J., GOVERNATORI, G. und AMOR, R. (2017). Evaluating LegalDocML and LegalRuleML as a Standard for Sharing Normative Information in the AEC/FM Domain. In: BOSCHÉ, F., BRILAKIS, I. und SACKS, R. (Herausgeber) *2017 Lean and Computing in Construction Congress (LC3)*, 637–644. Heriot-Watt University, Edinburgh (UK).

DIMYADI, J., SOLIHIN, W., EASTMAN, C. M. und AMOR, R. (2016b). Integrating the BIM Rule Language into Compliant Design Audit Processes. In: *Proceedings of the 33rd CIB W78 2016*, Band 33. Brisbane, Australia.

DIN (2007-03). Normung und damit zusammenhängende Tätigkeiten; Allgemeine Begriffe (ISO/IEC Guide 2:2004); Dreisprachige Fassung EN 45020:2006. URL: https://www.beuth.de/de/norm/din-en-45020/95609390 (Letzter Zugriff am: 15.01.2018).

DIN (2007-11). 18232:2007-11 - Smoke and heat control systems - Part 2: Natural smoke and heat exhaust ventilators; design, requirements and installation. URL: https://www.beuth.de/de/norm/din-18232-2/101330298 (Letzter Zugriff am: 15.01.2018).

DIN (2010-10). DIN 18040-1: Barrierefreies Bauen - Planungsgrundlagen – Teil 1: Öffentlich zugängliche Gebäude. URL: https://www.beuth.de/de/norm/din-18040-1/133692028 (Letzter Zugriff am: 15.01.2018).

DIN (2012-12). DIN 820-2: Normungsarbeit - Teil 2: Gestaltung von Dokumenten. URL: https://www.beuth.de/de/norm/din-820-2/165417565 (Letzter Zugriff am: 15.01.2018).

DIN (2016). Rechtsverbindlichkeit von Normen. URL: http://www.din.de/de/ueber-normen-und-standards/normen-und-recht/rechtsverbindlichkeit-durch-normen (Letzter Zugriff am: 10.04.2016).

DIN (2017). DIN-Normenausschuss Bauwesen (NABau). URL: https://www.din.de/de/mitwirken/normenausschuesse/nabau (Letzter Zugriff am: 15.02.2018).

DIN (2017-09). DIN 4108-3: Wärmeschutz und Energie-Einsparung in Gebäuden – Teil 3: Klimabedingter Feuchteschutz – Anforderungen, Berechnungsverfahren und Hinweise für Planung und Ausführung. URL: https://www.beuth.de/de/norm/din-4108-3/222022641 (Letzter Zugriff am: 15.01.2018).

DING, L., DROGEMULLER, R., JUPP, J., ROSENMAN, M. A. und GERO, J. S. (2004). Automated Code Checking. In: *CRC for Construction Innovation, Clients Driving Innovation International Conference*, 1–17.

DING, L., DROGEMULLER, R., ROSENMAN, M. und MARCHANT, D. (2006). Automating code checking for building designs - DesignCheck. In: *Cooperative Research Centre (CRC) for Construction Innovation*, 1–16.

DOING BUSINESS (2015). Dealing with Building Permit. URL: http://www.doingbusiness.org/data/exploretopics/dealing-with-construction-permits (Letzter Zugriff am: 15.01.2017).

DUDEN (2013). *Duden - Die deutsche Rechtschreibung: Auf der Grundlage der aktuellen amtlichen Rechtschreibregeln ; [das umfassende Standardwerk auf der Grundlage der amtlichen Regeln]*. 26. völlig neu bearb. u. erw. Auflage.

EASTMAN, C. (2009). Automated Assessment of Early Concept Designs. In: *Architectural Design*, 79(2):52–57.

EASTMAN, C., FISHER, D., LAFUE, G., LIVIDINI, J., STOKER, D. und YESSIOS, C. (1974). An outline of the building description system: Institute of Physical Planning, Carnegie-Mellon University. URL: `http://eric.ed.gov/?id=ED113833` (Letzter Zugriff am: 01.07.2019).

EASTMAN, C., LEE, J.-M., JEONG, Y.-S. und LEE, J.-K. (2009). Automatic rule-based checking of building designs. In: *Automation in Construction*, 18(8):1011–1033.

EASTMAN, C. M. (1999). *Building product models: Computer environments supporting design and construction*. CRC Press, Boca Raton, Fla. URL: `http://www.loc.gov/catdir/enhancements/fy0646/99029602-d.html`.

EASTMAN, C. M., TEICHOLZ, P., SACKS, R. und LISTON, K. (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors*. Wiley, Hoboken, N.J, 2. bearb. Auflage. URL: `https://books.google.de/books?id=aCi7Ozwkoj0C`.

EBERTSHÄUSER, S. und VON BOTH, P. (2013). ifcModelCheck - A tool for configurable rule-based model checking. In: STOUFFS, R. und SARIYILDIZ, S. (Herausgeber) *eCAADe 2013 / Computation and performance*, Band 2. eCAADe and Faculty of Architecture Delft University of Technology, Brussels and Delft.

ENBAUSA.DE (2018). Projekt "BIM-basierter Bauantrag" gestartet: Digitale Modelle sollen Baugenehmigungen beschleunigen. URL: `https://www.enbausa.de/finanzierung/aktuelles/artikel/projekt-bim-basierter-bauantrag-gestartet-5669.html` (Letzter Zugriff am: 10.01.2018).

ERNST, H., SCHMIDT, J. und BENEKEN, G. H. (2015). *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis - eine umfassende, praxisorientierte Einführung*. Springer Vieweg, Wiesbaden, 5. vollst. überarb. Auflage. URL: `http://dx.doi.org/10.1007/978-3-658-01628-9`.

ERWIG, M., SMELTZER, K. und WANG, X. (2017). What is a visual language? In: *Journal of Visual Languages & Computing*, 38:9–17.

ESCHENBRUCH, K. und LEUPERTZ, S. (2016). *BIM und Recht*. Werner, Köln.

FAHAD, M., BUS, N. und ANDRIEUX, F. (2016). Towards Mapping Certification Rules over BIM. In: *Proceesings of the CIB W78 2016, Brisbane, Australia*.

FENVES, S. J. (1966). Tabular Decision Logic for Structural Design. In: *Journal of the Structural Division*, 92(6):473–490.

FENVES, S. J., GARRETT, J. H., KILICCOTE, H., LAW, K. H. und REED, K. A. (1995). Computer representations of design standards and building codes: US perspective. In: *The International Journal of Construction Information Technology*, 3(1):13–34.

FENVES, S. J., WRIGTH, R., STAHL, F. und REED, K. (1987). Introduction to SASE: Standards Analysis, Synthesis, and Expression. URL: `http://nvlpubs.nist.gov/nistpubs/Legacy/IR/nbsir87-3513.pdf` (Letzter Zugriff am: 12.08.2016).

FIATECH (2011). The Autocodes Project. URL: `http://fiatech.org/the-autocodes-project` (Letzter Zugriff am: 15.02.2018).

FIATECH (2012). AutoCodes Project: Phase 1, Proof-of-Concept: Final Report. URL: `https://constructech.com/wp-content/uploads/2014/09/Whitepaper_FIATECH_AutoCodes.pdf` (Letzter Zugriff am: 15.01.2018).

FIEDLER, J. N. (2015). *Modernisierungszenarien des Baubewilligungsverfahrens unter Berücksichtigung neuer technologischer Hilfsmittel*. Dissertation, Technische Universität Wien. URL: http://repositum.tuwien.ac.at/obvutwhs/content/titleinfo/1641705 (Letzter Zugriff am: 15.01.2018).

FOLIENTE, G. C. (2000). Developments in performance-based building codes and standards. In: *Forest Products Journal*, 50(7/8):12. URL: https://www.researchgate.net/publication/285970302_Developments_in_performance-based_building_codes_and_standards.

FUCHS-KITTOWSKI, K. (2001). Wissens-Ko-Produktion – Organisationsinformatik – Verbreitung, Verteilung und Entstehung von Informationen in kreativ-lernenden Organisationen. In: FUCHS-KITTOWSKI, K. (Herausgeber) *Organisationsinformatik und Digitale Bibliothek in der Wissenschaft*, Wissenschaftsforschung, 9–88.

FUCHS-KITTOWSKI, K., KAISER, H., TSCHIRSCHWITZ, R. und WENZLAFF, B. (1976). *Informatik und Automatisierung Band l: Theorie und Praxis der Struktur und Organisation der Informationsverarbeitung*. Humboldt-Universität zu Berlin, Zentraleinrichtung Computer- und Medienservice (Rechenzentrum). URL: https://edoc.hu-berlin.de/handle/18452/14301.

FURRER, F. J. (2012). Eine kurze Geschichte der Ontologie. In: *Informatik-Spektrum*, 37(4):308–317.

GALLAHER, M. P., O'CONNOR, A. C., DETTBARN, J. L., JR und GILDAY, L. T. (2004). Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry. In: *National Institute of Standards & Technology*, 1–210.

GARRETT, J. H., PALMER, M. E. und DEMIR, S. (2014). Delivering the Infrastructure for Digital Building Regulations. In: *Journal of Computing in Civil Engineering*, 28(2):167–169.

GENESERETH, M. R. und NILSSON, N. J. (2012). *Logical Foundations of Artificial Intelligence*. Elsevier Science, Burlington. URL: http://gbv.eblib.com/patron/FullRecord.aspx?p=1757719.

GHANNAD, P., LEE, Y.-C., DIMYADI, J. und SOLIHIN, W. (2019). Automated BIM data validation integrating open-standard schema with visual programming language. In: *Advanced Engineering Informatics*, 40:14–28.

GIJEZEN, S. und HARTMANN, T. (2010). Organizing 3D Building Information Models with the help of Work Breakdown Structures to improve the Clash Detection process: Final Report: University of Twente. URL: http://essay.utwente.nl/59401/1/scriptie_S_Gijezen.pdf (Letzter Zugriff am: 19.02.2018).

GONÇAL, C. (2017). *Integration of building product data with BIM modelling: A semantic-based product catalogue and rule checking system*. Dissertation, Universitat Ramon Llull, Barcelona. URL: https://www.tdx.cat/bitstream/handle/10803/450865/Tesi_Goncal_Costa.pdf (Letzter Zugriff am: 02.01.2019).

GRÄBER-SEISSINGER, U. (2015). *Duden, Recht A - Z: Fachlexikon für Studium, Ausbildung und Beruf ; [die ideale Ergänzung zu den gängigen Gesetzestexten]*. Dudenverl., Berlin, 3. aktualisierte Auflage.

GRABISCH, M., MARICHAL, J.-L., MESIAR, R. und PAP, E. (2009). Aggregation functions. In: GRABISCH, M., MARICHAL, J.-L., MESIAR, R. und PAP, E. (Herausgeber) *Aggregation Functions*, Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge.

GRAPHISOFT (2018). ARCHICAD – Rhinoceros – Grasshopper Connection. URL: http://www.graphisoft.com/archicad/rhino-grasshopper/ (Letzter Zugriff am: 8/5/2018).

GRASSHOPPER (2017). Grasshopper3D: Algorithmic Modeling for Rhino. URL: https://www.grasshopper3d.com/ (Letzter Zugriff am: 15.01.2018).

GROSS, M. D. (1996). Why can't CAD be more like Lego? CKB, a program for building construction kits. In: *Automation in Construction*, 5(4):285–300.

GRUBER, T. R. (1993). A translation approach to portable ontology specifications. In: *Knowledge Acquisition*, 5(2):199–220.

GSA (2007). BIM Guide for Spatial Program Validation—GSA BIM Guide Series 02. URL: https://www.gsa.gov/real-estate/design-construction/3d4d-building-information-modeling/bim-guides/bim-guide-02-spatial-program-validation (Letzter Zugriff am: 26.08.2016).

GUARINO, N. und GIARETTA, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. In: *Towards very Large Knowledge bases: Knowledge Building and Knowledge sharing*, 25(32):307–317. URL: https://www.researchgate.net/publication/220041941_Ontologies_and_knowledge_bases_towards_a_terminological_clarification.

HAHMANN, M. (2015). Normen-Dschungel. Ein Umdenken ist nötig! URL: https://www.industr.com/de/A-und-D-Magazin/auftakt/normen-dschungel-ein-umdenken-ist-noetig-259789 (Letzter Zugriff am: 11.01.2018).

HAN, T. L. B. (1995). The CORENET project in Singapore. URL: https://buildingsmart.no/sites/buildingsmart.no/files/Case_studie_Singapore.pdf (Letzter Zugriff am: 10.02.2018).

HARPER, R. F. und GODBEY, A. H. (1903). Text of the Code of Hammurabi, King of Babylon (About 2250 B. C.). In: *The American Journal of Semitic Languages and Literatures*, 20(1):1–84.

HARTLIEB, B. (Herausgeber) (2000). *Gesamtwirtschaftlicher Nutzen der Normung: Zusammenfassung der Ergebnisse ; wissenschaftlicher Endbericht mit praktischen Beispielen*. Beuth, Berlin u. a.

HAUSKNECHT, K. und LIEBICH, T. (2016). *BIM-Kompendium: Building Information Modeling als neue Planungsmethode*. Fraunhofer IRB Verlag, Stuttgart.

HAZEWINKEL, M. (2002). *Encyclopaedia of mathematics*. Springer-Verlag, Berlin and New York.

HEINDORF, V. (2010). *Der Einsatz moderner Informationstechnologien in der Automobilproduktentwicklung: Produktivitätspotenziale und Systemkomplementaritäten*. Gabler Verlag / Springer Fachmedien Wiesbaden GmbH Wiesbaden, Wiesbaden. URL: http://dx.doi.org/10.1007/978-3-8349-8939-0.

HEINLEIN, K., HILKA, M. und HILKA, M. (2013). Honorarordnung für Architekten und Ingenieure. URL: http://www.hoai.de/online/HOAI_2013/HOAI_2013.php (Letzter Zugriff am: 10.01.2017).

HIETANEN, J. (2006). IFC Model View Definition Format: IAI- International Alliance for Interoperability. URL: http://www.buildingsmart-tech.org/downloads/accompanying-documents/formats/mvdxml-documentation/MVD_Format_V2_Proposal_080128.pdf (Letzter Zugriff am: 10.01.2017).

HILDEBRAND, K., GEBAUER, M., HINRICHS, H. und MIELKE, M. (2015). *Daten- und Informationsqualität*. Springer Fachmedien Wiesbaden, Wiesbaden.

HILS, D. D. (1992). Visual languages and computing survey: Data flow visual programming languages. In: *Journal of Visual Languages & Computing*, 3(1):69–101.

HJELSETH, E. (2009a). Foundation for development of computable rules. In: *Proceedings of the CIBW78, 1-3 Octobre, Istanbul.*

HJELSETH, E. (2009b). Foundation for development of computable rules. In: VAN ECK, P., GORDIJN, J. und WIERINGA, R. (Herausgeber) *Advanced information systems engineering*, Lecture notes in computer science. Springer, Berlin.

HJELSETH, E. (2010a). Exploring Semantic based Model Checking. In: *CIB W78 2010: 27th International Conference*. URL: http://itc.scix.net/paper/w78-2010-54.

HJELSETH, E. (2010b). Overview of Concepts for Model Checking. In: *CIB W78 2010: 27th International Conference*. URL: https://www.academia.edu/873824/Overview_of_concepts_for_model_checking.

HJELSETH, E. (2015a). BIM-based Model Checking (BMC). In: ISSA, R. und OLBINA, S. (Herausgeber) *Building information modeling*, 33–61. American Society of Civil Engineers, Reston, Virginia.

HJELSETH, E. (2015b). Public BIM-based model checking solutions: lessons learned from Singapore and Norway: International Conference on Building Information Modelling (BIM) in Design, Construction and Operations, 9 - 11 September, Bristol, UK. In: . URL: https://www.researchgate.net/publication/300634090_Public_BIM-based_model_checking_solutions_lessons_learned_from_Singapore_and_Norway.

HJELSETH, E. und NISBET, N. (2010). Overview of concepts for model checking. In: *Proceedings of the CIB W78 2010: 27th International Conference –Cairo, Egypt, 16-18 November*.

HJELSETH, E. und NISBET, N. (2011). Capturing normative constraints by use of the semantic mark-up RASE methodology. In: *Proceedings of the 28th International Conference of CIB W78*, 26–28.

HOCHTIEF VICON GMBH (2016). Die fünf BIM-Komponenten. URL: https://www.hochtief-vicon.de/vicon/BIM-Welt/Fuenf-BIM-Elemente-43.jhtml (Letzter Zugriff am: 10.02.2017).

HUDECZEK, D. (2017). *Formalisierung von Normen mithilfe von Auszeichnungssprachen für die automatisierte Konformitätsüberprüfung*. Master thesis, Technische Universität München, München. URL: https://www.cms.bgu.tum.de/publications/theses/2017_Hudeczek.pdf (Letzter Zugriff am: 04.01.2018).

INTERNATIONAL CODE COUNCIL (2006). International Building Code 2006. URL: https://www.worldcat.org/title/international-building-code-2006/oclc/156982831 (Letzter Zugriff am: 28.01.2018).

INTERNATIONAL CODE COUNCIL (2017a). ICC/ANSI A117.1: Accessible and Usable Buildings and Facilites. URL: https://codes.iccsafe.org/content/ICCA117_12017 (Letzter Zugriff am: 18.01.2018).

INTERNATIONAL CODE COUNCIL (2017b). Overview of the International Energy Conservation Code. URL: https://www.iccsafe.org/codes-tech-support/codes/2018-i-codes/iecc/ (Letzter Zugriff am: 10.01.2018).

ISO (2011-12). ISO 21542:2011-12: Gebäude - Barrierefreiheit von Gebäuden und sonstigen Bauwerken. URL: https://www.beuth.de/de/norm/iso-21542/149085681 (Letzter Zugriff am: 18.01.2018).

ISO (2013-04). ISO 16739:2013-04 - Industry Foundation Classes (IFC) für den Datenaustausch in der Bauindustrie und dem Anlagen-Management. URL: https://www.beuth.de/de/norm/iso-16739/186171696 (Letzter Zugriff am: 15.01.2018).

ISO (2016). ISO 29481-1:2016 Building information models - Information delivery manual: Part 1: Methodology and format. URL: https://www.iso.org/standard/60553.html (Letzter Zugriff am: 15.03.2018).

JEONG, J. und LEE, G. (2009). Requirements for automated code checking for fire resistance and egress rule using BIM. In: *International Conference on Construction Enginerrung and Project Management (ICCEM/ICCPM)*, 316–322.

JØRGENSEN, K. A. (2011). Classification of building object types-Misconceptions, challenges and opportunities. In: *Proceedings of CIB W78-W102 International Conference. Sophia Antipolis, France*, 26–28.

JOST, B., KETTERL, M., BUDDE, R. und LEIMBACH, T. (2014). Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One? In: *2014 IEEE International Symposium on Multimedia (ISM)*, 381–386. Ieee, Piscataway, NJ.

JOTNE EPM TECHNOLOGY (2004). Express Data Manager: Vol. 1, Issue 6. URL: `http://www2.epmtech.jotne.com/download/EDM47.pdf` (Letzter Zugriff am: 08.07.2014).

JUBIERRE, J. J. (2017). *Consistency preservation methods for multi-scale design of subway infrastructure facilities*. Dissertation, Technische Universität München, München. URL: `https://mediatum.ub.tum.de/doc/1328419/1328419.pdf` (Letzter Zugriff am: 10.01.2018).

KAMARUDDIN, S. S., MOHAMMAD, M. F. und MAHBUB, R. (2016). Barriers and Impact of Mechanisation and Automation in Construction to Achieve Better Quality Products. In: *Procedia - Social and Behavioral Sciences*, 222:111–120.

KAMMHOLZ, K. (2014). So werden Baudesaster wie der BER künftig vermieden. URL: `https://www.welt.de/politik/deutschland/article127986675/So-werden-Baudesaster-wie-der-BER-kuenftig-vermieden.html` (Letzter Zugriff am: 10.02.2015).

KEMMERER, S. J. (1999). STEP: The Grand Experience. In: *Special Publication (NIST SP) - 939*. URL: `http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=821224`.

KEMPER, A. und EICKLER, A. (2011). *Datenbanksysteme: Eine Einführung*. Oldenbourg, München, 8. aktualisierte und erw. Auflage.

KENSEK, K. (2015). Visual Programming for Building Information Modeling: Energy and Shading Analysis Case Studies. In: *Journal of Green Building*, 10(4):28–43.

KERESHMEH, A. und EASTMAN, C. M. (2016). A Comparison of Construction Classification Systems Used for Classifying Building Product Models. URL: `https://www.researchgate.net/publication/303484920_A_Comparison_of_Construction_Classification_Systems_Used_for_Classifying_Building_Product_Models` (Letzter Zugriff am: 15.02.2018).

KERRIGAN, S. und LAW, K. H. (2003). Logic-based regulation compliance-assistance. In: ZELEZNIKOW, J. (Herausgeber) *Proceedings of the 9th international conference on Artificial intelligence and law*, 126–135. ACM, New York, NY. URL: `http://dl.acm.org/citation.cfm?id=1047820`.

KHEMLANI, L. (2005). CORENET e-PlanCheck: Singapore's Automated Code Checking System. URL: `http://www.aecbytes.com/feature/2005/CORENETePlanCheck.html` (Letzter Zugriff am: 30.05.2017).

KHEMLANI, L. (2017). Automated Code Compliance Updates. URL: `http://www.aecbytes.com/feature/2017/CodeCheckingUpdates.html` (Letzter Zugriff am: 30.05.2017).

KIM, H., KIM, J., SONG, J. und LEE, J.-K. (2018). Demonstration of Visual Language-based Representation of Korea Fire Code Regulations. Advanced Science and Technology Letters, 170–173. Science & Engineering Research Support soCiety.

KIM, H., LEE, J.-K., SHIN, J. und CHOI, J. (2019). Visual language approach to representing KBimCode-based Korea building code sentences for automated rule checking. In: *Journal of Computational Design and Engineering*, 6(2):143–148. URL: `http://www.sciencedirect.com/science/article/pii/S2288430018300678`.

KIM, H., LEE, J.-K., SHIN, J. und KIM, J. (2017a). Visual Language-based approach for the Definition of Building Permit Related Rules. In: CHENG, M.-Y., CHEN, H.-M. und CHIU, K. C. (Herausgeber) *34th International Symposium on Automation and Robotics in Construction and Mining (ISARC 2017), Taipei, Taiwan, 28 June-1 July 2017*, Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC). Faculty of Engineering, Slovak University of Technology in Bratislava and Printed by Curran Associates, Inc, Bratislava, Slovakia.

KIM, J., LEE, J.-K., SHIN, J. und KIM, H. (2017b). Classification of Objects and their Properties Defined in Korea Building Act to Translate into KBimCode and their Application. Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC). International Association for Automation and Robotics in Construction (IAARC).

KNOPP-TRENDAFILOVA, A. (2010). Link between a structural model of buildings and classification systems in construction. URL: https://aaltodoc.aalto.fi:443/bitstream/123456789/3311/1/urn100259.pdf (Letzter Zugriff am: 15.02.2018).

KÖHLER, H. (Herausgeber) (1995). *Bürgerliches Gesetzbuch*, Band 5001 von *Dtv*. Dt. Taschenbuch-Verl., München, sonderausg., 37. neubearb. Auflage.

KOREA LEGISLATION RESEARCH INSTITUTE (2008). Article 34 Clause 1. URL: https://elaw.klri.re.kr/kor_service/jomunPrint.do?hseq=33006&cseq=926869 (Letzter Zugriff am: 15.01.2018).

KOST, M. (2013). Grundlagen des Semantic Web: Web Ontology Language (OWL). URL: http://www.is.informatik.uni-duisburg.de/courses/db_ws04/folien/owl.pdf (Letzter Zugriff am: 10.01.2016).

KROMREY, H. (Herausgeber) (1981). *Die gebaute Umwelt: Wohngebietsplanung im Bewohnerurteil*, Band 2 von *Forschungstexte Wirtschafts- und Sozialwissenschaften*. VS Verlag für Sozialwissenschaften, Wiesbaden.

KULUSJÄRVI, H. (2012). Common BIM Requirements (COBIM) - Part 6 Quality Assurance. URL: https://asiakas.kotisivukone.com/files/en.buildingsmart.kotisivukone.com/COBIM2012/cobim_6_quality_assurance_v1.pdf (Letzter Zugriff am: 10.02.2018).

KUNZ, J. und FISCHER, M. (2012). Virtual Design and Construction: Themes, Case Studies and Implementation Suggestions. URL: https://stacks.stanford.edu/file/druid:gg301vb3551/WP097_0.pdf (Letzter Zugriff am: 15.02.2018).

LAAKSO, M. und KIVINIEMI, A. (2012). The IFC standard - a review of history, development, and standardization. In: *ITcon*, 17(9):134–161. URL: https://helda.helsinki.fi/dhanken/bitstream/10138/36870/2/Laakso_Kiviniemi_2012_The_IFC_Standard_A_Review_Of_History_Development_And_Standardization.pdf.

LÊ, M., MOHUS, F., KVARSVIK, O. K. und LIE, M. (2006). The HITOS Project - A Full Scale IFC Test. In: MARTÍNEZ, M. und SCHERER, R. (Herausgeber) *eWork and eBusiness in architecture, engineering and construction*, Balkema - proceedings and monographs in engineering, water and earth sciences. Taylor& Francis, London.

LEE, H., LEE, J.-K., PARK, S. und KIM, I. (2016). Translating building legislation into a computer-executable format for evaluating building permit requirements. In: *Automation in Construction*, 71:49–61.

LEE, J. K. (2011). *Building Environment Rule and Analysis (BERA) Language*. Dissertation, Georgia Institute of Technology. URL: https://smartech.gatech.edu/handle/1853/39482 (Letzter Zugriff am: 15.01.2018).

LEE, J.-S., MIN, K.-M., LEE, Y.-S., KIM, J.-H. und KIM, J.-J. (2008). Building ontology to implement the BIM (Building Information Modeling) focused on pre-design stage. In: ZAVADSKAS, E. K. (Herausgeber) *Selected papers / The 25th International Symposium on Automation and Robotics in Construction, ISARC-2008, June 26 - 29, 2008, Vilnius, Lithuania*, 350–354. Technika, Vilnius.

LEE, Y. C., EASTMAN, C. M. und LEE, J. K. (2015). Automated Rule-Based Checking for the Validation of Accessibility and Visibility of a Building Information Model. In: PONTICELLI, S. und O'BRIEN, W. J. (Herausgeber) *Computing in civil engineering 2015*, 572–579. American Society of Civil Engineers, Reston, Virginia.

LIU, L. und ÖZSU, M. T. (Herausgeber) (2009). *Encyclopedia of database systems*. Springer reference. Springer, New York, NY.

LOD PLANNER (2018). URL: `https://www.lodplanner.com/about/` (Letzter Zugriff am: 25.01.2018).

LOPEZ, L. und ELAM, S. (1985). Mapping Principles for the Standards Interface for Computer Aided Design. URL: `https://www.gpo.gov/fdsys/pkg/GOVPUB-C13-dbcc99bf2332a11e563536141196ae67/pdf/GOVPUB-C13-dbcc99bf2332a11e563536141196ae67.pdf` (Letzter Zugriff am: 01.12.2017).

LOPEZ, R. und LOVE, P. E. D. (2012). Design Error Costs in Construction Projects. In: *Journal of Construction Engineering and Management*, 138(5):585–593.

LOVE, P. E., LOPEZ, R. und EDWARDS, D. J. (2013). Reviewing the past to learn in the future: Making sense of design errors and failures in construction. In: *Structure and Infrastructure Engineering*, 9(7):675–688.

LOVE, P. E. D., EDWARDS, D. J., HAN, S. und GOH, Y. M. (2011). Design error reduction: Toward the effective utilization of building information modeling. In: *Research in Engineering Design*, 22(3):173–187.

LUO, L., HE, Q., JASELSKIS, E. J. und XIE, J. (2017). Construction Project Complexity: Research Trends and Implications. In: *Journal of Construction Engineering and Management*, 143(7):04017019.

MACIT İLAL, S. und GÜNAYDIN, H. M. (2017). Computer representation of building codes for automated compliance checking. In: *Automation in Construction*, 82:43–58.

MAHBUB, R. (2009). *An investigation into the barriers to the implementation of automation and robotics technologies in the construction industry*. Dissertation, Queensland University of Technology. URL: `https://eprints.qut.edu.au/26377/` (Letzter Zugriff am: 15.01.2018).

MALONEY, J., RESNICK, M., RUSK, N., SILVERMAN, B. und EASTMOND, E. (2010). The Scratch Programming Language and Environment. In: *ACM Transactions on Computing Education*, 10(4):1–15.

MANSFELD, I. (2017). Building Information Modeling: 10 Reasons why you need BIM. URL: `https://info.allplan.com/hubfs/07_Guides/Whitepaper_10_good_reasons_for_BIM_EN.pdf?t=1515678139952` (Letzter Zugriff am: 10.01.2018).

MARKETSANDMARKETS (2019). Building Information Modeling Market by Type (Software, Services), Application (Buildings, Civil Infrastructure, Industrial, Oil & gas), End-User (AEC, Contractors and Facility Managers), Project Lifecycle, and Region - Global Forecast to 2024: Market Research Report. URL: `https://www.marketsandmarkets.com/Market-Reports/building-information-modeling-market-95037387.html` (Letzter Zugriff am: 01.10.2019).

MARTINS, J. P. und MONTEIRO, A. (2013). LicA: A BIM based automated code-checking application for water distribution systems. In: *Automation in Construction*, 29:12–23.

MATTIUZZO, C. und MIESNER, S. (2016). Genormte Normensprache. URL: https://www.kan.de/publikationen/kanbrief/normatives-und-informatives/genormte-normensprache/ (Letzter Zugriff am: 15.02.2018).

MATZIG, G. (2018). Bauen mit Vollkasko-Mentalität. URL: http://www.sueddeutsche.de/kultur/jahre-din-bauen-mit-vollkasko-mentalitaet-1.3819419 (Letzter Zugriff am: 11.01.2018).

MAZAIRAC, W. (2015). BimQL. URL: http://bimql.org/ (Letzter Zugriff am: 14.03.2016).

MCGRAWHILL CONSTRUCTION (2007). Interoperability in the construction industry: SmartMarket Report, Interoperability Issue. URL: http://www.1stpricing.com/pdf/MGH_Interoperability_SmartMarket_Report.pdf (Letzter Zugriff am: 07.11.2017).

MEACHAM, B. J. (2010). Performance-Based Building Regulatory Systems: Principles and Experiences: A Report of the Inter-jurisdictional Regulatory Collaboration Committee. URL: http://ircc.info/Doc/A1163909.pdf (Letzter Zugriff am: 15.01.2018).

MESSER, P. und AMMON, B. (2015). *Musterbauordnung (MBO): Mit ergänzenden Musterbestimmungen und Begründungen*. Kulturbuch-Verl., Berlin, 7. Auflage.

MINISTERE DU LOGEMENT ET DE LA VILLE (2008-05). Annexe 8 : Accessibilité des établissments recevant du public, des installations ouvertes au public et des bâtiments d habitation. URL: http://www.cohesion-territoires.gouv.fr/IMG/pdf/annexe_8_de_la_circulaire_du_30_novembre_2007_illustree_1_.pdf (Letzter Zugriff am: 15.01.2018).

MISSION NUMÉRIQUE BÂTIMENT (2015). Plan Transition Numérique dans le Bâtiment. URL: http://www.batiment-numerique.fr/ (Letzter Zugriff am: 10.01.2018).

MOLODOVSKY, P. (2016). *Bayerische Bauordnung: Mit ergänzenden Rechts- und Verwaltungsvorschriften : Textausgabe*. rehm, Heidelberg, 24. aktualisierte Auflage.

MUELLER, V. und SMITH, M. (2013). Generative Components and Smartgeometry: Situated Software Development. In: PETERS, B. und PETERS, T. (Herausgeber) *Inside Smartgeometry*, AD Smart, 142–153. Wiley, Chichester, West Sussex, UK.

MYERS, B. A. (1990). Taxonomies of visual programming and program visualization. In: *Journal of Visual Languages & Computing*, 1(1):97–123.

NATIONAL FIRE PROTECTION ASSOCIATION (2017). List of NFPA Codes & Standards. URL: https://www.nfpa.org/Codes-and-Standards/All-Codes-and-Standards (Letzter Zugriff am: 01.10.2017).

NATIONAL INSTRUMENTS (2018). LabVIEW. URL: http://www.ni.com/de-de/shop/labview.html (Letzter Zugriff am: 15.02.2018).

NATSPEC (2013). BIM and LOD – Building Information Modelling and Level of Development. URL: https://bim.natspec.org/images/NATSPEC_Documents/NATSPEC_BIM_LOD_Paper_131115.pdf (Letzter Zugriff am: 01.10.2016).

NAWARI, N. (2012a). Automating Codes Conformance. In: *Journal of Architectural Engineering*, 18(4):315–323.

NAWARI, N. (2012b). The Challenge of Computerizing Building Codes in a BIM Environment. In: *Computing in Civil Engineering*, 285–292.

NAWARI, N. O. (2012c). Automating Codes Conformance. In: *Journal of Architectural Engineering*, 18(4):315–323.

NAWARI, N. O. (2018). *Building Information Modeling: Automated Code Checking and Compliance Processes*. CRC Press, Boca Raton, FL, 1. Auflage.

NAWARI, N. O. und ALSAFFAR, A. (2015). Methods for Computable Building Codes. In: *Civil Engineering and Architecture*, 3(6):163–171.

NBIMS (2017). National BIM Standard - United States ® Version 3. URL: https://www.nationalbimstandard.org/nbims-us (Letzter Zugriff am: 10.02.2018).

NBS (2007). UK Fire Code Part B4: Fire safety. URL: https://www.gov.uk/government/publications/fire-safety-approved-document-b (Letzter Zugriff am: 15.01.2018).

NBS (2015). Uniclass 2015. URL: https://www.thenbs.com/services/our-tools/introducing-uniclass-2015 (Letzter Zugriff am: 15.02.2018).

NESTLE, H. und FREY, H. (2003). *Fachkunde für Maurer/Maurerinnen, Beton- und Stahlbetonbauer/Beton- und Stahlbetonbauerinnen, Zimmerer/Zimmerinnen und Bauzeichner/Bauzeichnerinnen*. Europa-Fachbuchreihe für Bautechnik. Europa-Lehrmittel Nourney Vollmer, Haan-Gruiten, 10. überarb. Auflage.

NEUFERT, E. und KISTER, J. (2016). *Bauentwurfslehre: Grundlagen, Normen, Vorschriften über Anlage, Bau, Gestaltung, Raumbedarf, Raumbeziehungen, Maße für Gebäude, Räume, Einrichtungen, Geräte mit dem Menschen als Maß und Ziel : Handbuch für den Baufachmann, Bauherrn, Lehrenden und Lernenden*. 41., überarbeitete und aktualisierte Auflage.

NIBS (2012). National BIM Standard United States: National Institute of Building Sciences. URL: http://www.nationalbimstandard.org/ (Letzter Zugriff am: 01.10.2017).

NIEDERMEIER, A. und BÄCK, R. (2015). BIM-Kompendium: Theorie und Praxis: Allplan GmbH. URL: http://www.allplan-architektur.ch/_Resources/Persistent/2b93ec176bdbcb4f4399f03393dc7b22f21b4c1c/BIM%20Handbuch.pdf (Letzter Zugriff am: 10.01.2018).

NIEMEIJER, R. A., DE VRIES, B. und BEETZ, J. (2009). Check-mate: Automatic constraint checking of IFC models. In: DIKBAŞ, A. (Herausgeber) *Managing IT in construction / managing construction for tomorrow*. CRC, Boca Raton.

NIEMEIJER, R. A., DE VRIES, B. und BEETZ, J. (2014). Freedom through constraints: User-oriented architectural design. In: *Advanced Engineering Informatics*, 28(1):28–36.

NISBET, N., WIX, J. und CONOVER, D. (2008). The Future of Virtual Construction and Regulation Checking. In: BRANDON, P. S. und KOCATÜRK, T. (Herausgeber) *Virtual futures for design, construction & procurement*, 241–250. Blackwell Pub, Oxford and Malden, MA.

NORTHUMBRIA UNIVERSITY (2018). xBIM Toolkit: eXtensible Building Information Modelling. URL: http://docs.xbim.net/ (Letzter Zugriff am: 20.12.2018).

NS (2009). NS 11001-1 2009: Universal design of building works - Part 1: Buildings open to the public. URL: https://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=405200 (Letzter Zugriff am: 15.02.2018).

NZBC (2017). Building Code compliance. URL: https://www.building.govt.nz/building-code-compliance/ (Letzter Zugriff am: 21.01.2018).

OASIS (2017). LegalRuleML Core Specification: Version 1.0: Committee Specification Draft 02 / Public Review Draft 02. URL: http://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/legalruleml-core-spec-v1.0.pdf (Letzter Zugriff am: 15.01.2018).

OBERSTE BAUBEHÖRDE IM BAYERISCHEN STAATSMINISTERIUM DES INNERN (2012). *Barrierefreies Wohnen*. Oberste Baubehörde im Bayerischen Staatsministerium des Innern. URL: https://www.agsv.bayern.de/wp-content/uploads/2018/07/obb_faltblatt_barrierefr_wohnen.pdf.

OCCS (2017). Omniclass. URL: http://www.omniclass.org/ (Letzter Zugriff am: 15.02.2018).

OLKEN, F., PALMIRANI, M., SOTTARA, D., GOVERNATORI, G., ROTOLO, A., TABET, S., BOLEY, H. und PASCHKE, A. (Herausgeber) (2011). *LegalRuleML: XML-Based Rules and Norms: Rule-Based Modeling and Computing on the Semantic Web*. Springer Berlin Heidelberg.

OXFORD ECONOMICS (2017). Global Infrastructure Outlook: Infrastrcutre investment needs: 50 countries, 7 sectors to 2040. URL: http://www.oxfordeconomics.com/Global-Infrastructure-Outlook (Letzter Zugriff am: 10.01.2018).

PAN, J. Z. (2009). Resource Description Framework. In: STAAB, S. und STUDER, R. (Herausgeber) *Handbook on Ontologies*, International Handbooks on Information Systems, 71–90. Springer, Dordrecht.

PARK, S. und LEE, J.-K. (2016). KBimCode-based Applications for the Representation, Definition and Evaluation of Building Permit Rules. In: *2016 Proceedings of the 33rd ISARC*, 720–728. Auburn, USA. URL: http://www.iaarc.org/publications/fulltext/ISARC2016-Paper128.pdf.

PASSARGE, G. (2010). Der Dschungel ist übersichtlicher. URL: http://www.sueddeutsche.de/geld/bauvorschriften-in-muenchen-der-dschungel-ist-uebersichtlicher-1.570614 (Letzter Zugriff am: 10.01.2017).

PATIL, L., DUTTA, D. und SRIRAM, R. (2005). Ontology-Based Exchange of Product Data Semantics. In: *IEEE Transactions on Automation Science and Engineering*, 2(3):213–225.

PAU, L. F. und OLASON, H. (1991). Visual Logic Programming. In: *Journal of Visual Languages and Computing*, 2:3–15.

PAUWELS, P., DE FARIAS, T. M., ZHANG, C., ROXIN, A., BEETZ, J. und DE ROO, J. (2016). Querying and reasoning over large scale building data sets. In: GROPPE, S. und LE GRUENWALD (Herausgeber) *Proceedings of the International Workshop on Semantic Big Data (SBD 2016)*, 1–6. The Association for Computing Machinery, New York, New York.

PAUWELS, P. und ORASKARI, J. (2016). IFC-to-RDF-converter. URL: https://github.com/IDLabResearch/IFC-to-RDF-converter (Letzter Zugriff am: 24.03.2016).

PAUWELS, P. und ROXIN, A. (2016). SimpleBIM : from full ifcOWL graphs to simplified building graphs. In: CHRISTODOULOU, S. (Herausgeber) *EWork and eBusiness in Architecture*, 11–18. CRC Press. URL: https://biblio.ugent.be/publication/8041826/file/8070159.pdf.

PAUWELS, P., VAN DEURSEN, D., VERSTRAETEN, R., DE ROO, J., DE MEYER, R., VAN DE WALLE, R. und VAN CAMPENHOUT, J. (2011). A semantic rule checking environment for building performance checking. In: *Automation in Construction*, 20(5):506–518.

PAUWELS, P. und ZHANG, S. (2015a). Semantic rule-checking for regulation compliance checking: An overview of strategies and approaches. URL: https://biblio.ugent.be/publication/6890589/file/6974128.pdf.

PAUWELS, P. und ZHANG, S. (2015b). Semantic Rule-checking for Regulation Compliance Checking: An Overview of Strategies and Approaches. In: *Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands*, 619–628.

PILLING, A. (2016). *BIM - das digitale Miteinander*. Beuth Innovation. Beuth Verlag GmbH, Berlin and Wien and Zürich, 1. Auflage. URL: http://gbv.eblib.com/patron/FullRecord.aspx?p=4718282.

PINHEIRO, S., WIMMER, R., O'DONNELL, J., MUHIC, S., BAZJANAC, V., MAILE, T., FRISCH, J. und VAN TREECK, C. (2018). MVD based information exchange between BIM and building energy performance simulation. In: *Automation in Construction*, 90:91–103.

PORWAL, A. und HEWAGE, K. N. (2013). Building Information Modeling (BIM) partnering framework for public construction projects. In: *Automation in Construction*, 31:204–214.

PREIDEL, C. und BORRMANN, A. (2015). Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling. In: *Connected to the future - 32nd International Symposium on Automation and Robotics in Construction and Mining (ISARC 2015) : Oulu, Finland, 15-18 June 2015*. Curran Associates Inc, Red Hook, NY.

PREIDEL, C. und BORRMANN, A. (2016). Towards code compliance checking on the basis of a visual programming language. In: *Journal of Information Technology in Construction: Special Issue CIB W78 2015 Special track on Compliance Checking*, 402–421. URL: http://www.itcon.org/data/works/att/2016_25.content.01707.pdf.

PREIDEL, C., BORRMANN, A., OBERENDER, C.-H. und TRETHEWAY, M. (2016). Seamless Integration of Common Data Environment Access into BIM Authoring Applications: the BIM Integration Framework. In: CHRISTODOULOU, S. (Herausgeber) *eWork and eBusiness in Architecture: ECPPM 2016 : Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016), Limassol, Cyprus, 7-9 September 2016*, Band 11. CRC Press.

PREIDEL, C., BUS, N., BORRMANN, A. und FIES, B. (2018). Pre-Processing IFC Building Models for Code Compliance Checking based on Visual Programming: 17th International Conference on Computing in Civil and Building Engineering (ICCCBE) 2018, Tampere, Finland. In: . URL: https://icccbe2018.exordo.com/files/papers/7/final_draft/ICCCBE2018_Preidel_final.pdf.

PREIDEL, C., DAUM, S. und BORRMANN, A. (2017). Data retrieval from building information models based on visual programming. In: *Visualization in Engineering*, 5(1):89.

PREIDEL, C. und TRETHEWAY, M. (2016). Das BIM Integration Framework. In: *BIM - Building Information Modeling, Ernst & Sohn Special 2016*, 89–91.

PREIDEL, C. und TRETHEWAY, M. (2017). Das BIM Integration Framework: Bindeglied zwischen BIM-Tools und CDE. URL: https://info.allplan.com/de/gc/bim-integration-framework.html (Letzter Zugriff am: 25.09.2017).

PRIGGE, W. und NEUFERT, E. (1999). *Ernst Neufert: Normierte Baukultur im 20. Jahrhundert*, Band 5 von *Edition Bauhaus*. Campus-Verl., Frankfurt/Main.

RITTER, F., PREIDEL, C. und SINGER, D. (2015). Visuelle Programmiersprachen im Bauwesen - Stand der Technik und aktuelle Entwicklungen. In: *Proceedings of the 27th Forum Bauinformatik, Aachen, Germany,*.

ROBERT MCNEEL ASSOCIATES (2016). Rhinoceros3D. URL: https://www.rhino3d.com (Letzter Zugriff am: 15.01.2018).

RÖHL, K. F. und RÖHL, H. C. (2008). *Allgemeine Rechtslehre: Ein Lehrbuch*. Heymanns, Köln, 3., neu bearb. Auflage.

RONG XU, SOLIHIN, W. und ZHIYONG HUANG (2004). Code Checking and Visualization of an Architecture Design. In: *IEEE Visualization 2004*.

ROSCHELLE, J. und TEASLEY, S. D. (1995). The Construction of Shared Knowledge in Collaborative Problem Solving. In: O'MALLEY, C. (Herausgeber) *Computer Supported Collaborative Learning*, NATO ASI Series, Series F, 69–97. Springer, Berlin and Heidelberg.

RULEML INC. (2017). RuleML Home. URL: `http://wiki.ruleml.org/` (Letzter Zugriff am: 15.01.2018).

SÄCHSISCHE STAATSKANZLEI (2017). Verwaltungsvorschrift des Sächsischen Staatsministeriums des Innern zur Sächsischen Bauordnung. URL: `https://www.revosax.sachsen.de/vorschrift/2374-VwVSaechsBO#vwv6` (Letzter Zugriff am: 18.01.2018).

SALAMA, D. A. und EL-GOHARY, N. M. (2013). Automated Compliance Checking of Construction Operation Plans Using a Deontology for the Construction Domain. In: *Journal of Computing in Civil Engineering*, 27(6):681–698.

SAMOFALOV, M., PAPINIGIS, V. und JUOCEVIČIUS, V. (2015). Design revisions of complex building structures: Work experience in Lithuania, legislative, organizational and technical aspects. In: *Engineering Structures and Technologies*, 7(1):39–49.

SCHENCK, D. A. und WILSON, P. R. (1994). *Information modeling: The EXPRESS way*. Oxford University Press, New York. URL: `http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=e000xat&AN=151210`.

SCHENK, J. und RIGOLL, G. (2010). *Mensch-Maschine-Kommunikation: Grundlagen von sprach- und bildbasierten Benutzerschnittstellen*. Springer, Heidelberg. URL: `http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10372373`.

SCHENKE, M. (2013). *Logikkalküle in der Informatik*. Springer Vieweg, Wiesbaden.

SCHIFFER, S. (1998). *Visuelle Programmierung: Grundlagen und Einsatzmöglichkeiten*. Addison-Wesley, Bonn.

SCHNEIDER, J. und SCHLATTER, H. P. (1996). *Sicherheit und Zuverlässigkeit im Bauwesen: Grundwissen für Ingenieure*. vdf Hochschulverlag AG an der ETH and B.G. Teubner, Zürich and Stuttgart, 2., überarb. Auflage.

SCHNEIDER, K.-J. und RJASANOWA, K. (2018). *Schneider - Bautabellen für Architekten: Entwurf - Planung - Ausführung*. Bundesanzeiger, Köln, 23., überarbeitete Auflage.

SCHOBER, K.-S., NÖLLING, K. und HOFF, P. (2016). Digitalisierung der Bauwirtschaft: Der europäische Weg zu "Construction 4.0". URL: `https://www.rolandberger.com/publications/publication_pdf/roland_berger_digitalisierung_bauwirtschaft_final.pdf` (Letzter Zugriff am: 10.01.2018).

SCHONSCHEK, O. (2017). *IT-Lexikon*. WEKA Praxislösungen. WEKA MEDIA GmbH et Co. KG, Kissing.

SCHUMAN, S. (2006). *Creating a culture of collaboration: The International Association of Facilitators handbook*. The Jossey-Bass business & management series. Jossey-Bass, San Francisco, Calif., 1. Auflage.

SEE, R. (2008). SMARTcodes: Enabling BIM Based Automated Code Compliance Checking. In: *AEC-ST Conference Presentation*. URL: `https://portal.nibs.org/files/wl/?id=O9WUH1OENChzJWsuVelOIwakeWgUyW6N`.

SEIFERT, N. und MÜHLHAUS, M. (2013). *Digitales Werkzeug zur Steuerung und Überprüfung von Nachverdichtungsstrategien*. Diplomarbeit, Technische Universität München, München. URL: `http://wp.usp.ai.ar.tum.de/dwn/` (Letzter Zugriff am: 15.02.2018).

SEIFERT, N., MÜHLHAUS, M., SCHUBERT, G., FINK, D. und PETZOLD, F. (2014). Decision support for inner-city development: An interactive customizable environment for decision-making processes in urban planning. In: THOMPSON, E. M. (Herausgeber) *Fusion*, Band 32, 43–52.

SEIFERT, N. und PETZOLD, F. (2016). A Parametric 3d City Model: Basis for Decision Support in Inner-City Development. In: YABUKI, N. und MAKANAE, K. (Herausgeber) *Proceedings of the 16th International Conference on Computing in Civil and Building Engineering. Osaka, Japan: ICCCBE*, 1285–1292. Osaka, Japan.

SHIH, S.-Y., SHER, W. und GIGGINS, H. (2013). Assessment of the Building Code of Australia to inform the development of BIM-enabled code checking systems. URL: `https://nova.newcastle.edu.au/vital/access/%20/manager/Repository/uon:15195/ATTACHMENT01` (Letzter Zugriff am: 15.01.2018).

SHU, N. C. (1988). *Visual programming*. Van Nostrand Reinhold, New York.

SINGER, D. (2015). Einsatz wissensbasierter Methoden in frühen Phasen des Brückenentwurfs. In: REAL EHRLICH, C. M. und BLUT, C. (Herausgeber) *Proceedings of the 27th Forum Bauinformatik // Bauinformatik 2015*. Wichmann, Aachen, Germany.

SINGER, D. und BORRMANN, A. (2015). A Novel Knowledge-Based Engineering Approach for Infrastructure Design. In: TSOMPANAKIS, Y., KRUIS, J. und TOPPING, B. (Herausgeber) *Proceesings of the Fourth International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering*, Civil-Comp Proceedings. Civil-Comp PressStirlingshire, UK. URL: `https://www.researchgate.net/publication/286052616_A_Novel_Knowledge-Based_Engineering_Approach_for_Infrastructure_Design`.

SMITH, D. C. (1975). Pygmalion: A creative programming environment. URL: `http://worrydream.com/refs/Smith%20-%20Pygmalion.pdf` (Letzter Zugriff am: 15.02.2018).

SOLIBRI INC. (2017). Solibri Model Checker. URL: `https://www.solibri.com/` (Letzter Zugriff am: 08.032016).

SOLIHIN, W. (2004). Lessons learned from experience of code-checking implementation in Singapore. URL: `https://www.researchgate.net/publication/280599027_Lessons_learned_from_experience_of_code-checking_implementation_in_Singapore` (Letzter Zugriff am: 06.10.2016).

SOLIHIN, W. (2016). *A simplified BIM Data Representation Using a Relational Database Schema for an Efficient Rule Checking System and its Associated Rule Checking Language*. Dissertation, Georgia Institute of Technology. URL: `https://smartech.gatech.edu/handle/1853/54831` (Letzter Zugriff am: 15.01.2018).

SOLIHIN, W., DIMYADI, J., LEE, Y.-C., EASTMAN, C. und AMOR, R. (2017). The Critical Role of Accessible Data for BIM-Based Automated Rule Checking Systems. In: *Lean and Computing in Construction Congress - Volume 1: Proceedings of the Joint Conference on Computing in Construction*, 53–60. Heriot-Watt University, Edinburgh.

SOLIHIN, W. und EASTMAN, C. (2015a). Classification of rules for automated BIM rule checking development. In: *Automation in Construction*, 53:69–82.

SOLIHIN, W. und EASTMAN, C. (2015b). A Knowledge Representation Approach to Capturing BIM Based Rule Checking Requirements Using Conceptual Graph. In: *Connected to the future - 32nd International Symposium on Automation and Robotics in Construction and Mining (ISARC 2015) : Oulu, Finland, 15-18 June 2015*. Curran Associates Inc, Red Hook, NY.

SOLIHIN, W. und EASTMAN, C. (2016). A knowledge representation approach in BIM rule requirement analysis using the conceptual graph. In: *Journal of Information Technology in Construction: Special Issue CIB W78 2015 Special track on Compliance Checking*, 370–401. URL: `http://www.itcon.org/data/works/att/2016_24.content.02464.pdf`.

SOLIHIN, W., EASTMAN, C. und LEE, Y. C. (2016). A framework for fully integrated building information models in a federated environment. In: *Advanced Engineering Informatics*, 30(2):168–189.

SPATH, D. (2013). Produktionsarbeit der Zukunft - Industrie 4.0: [Studie]: Fraunhofer-Institut für Arbeitswirtschaft und Organisation. URL: https://pdfs.semanti-cscholar.org/7253/20d6e87641d8b737bd447b0d4fa3974a4fe0.pdf (Letzter Zugriff am: 15.02.2018).

STADLER, A. und KOLBE, T. (2007). Spatio-semantic Coherence in the Integration of 3D City Models. In: *Proceedings of the 5th International Symposium on Spatial Data Quality, Enschede*. URL: http://www.isprs.org/proceedings/xxxvi/2-c43/Session1/paper_Stadler.pdf.

STANDARD NORGE (2017). Regler for standardiseringsarbeid. URL: https://www.standard.no/en/standardisation/hvordan-lages-standarder/regler-for-standardiseringsarbeid/ (Letzter Zugriff am: 15.02.2018).

STANFORD UNIVERSITY (2016). Protégé: A free, open-source ontology editor and framework for building intelligent systems. URL: https://protege.stanford.edu/ (Letzter Zugriff am: 10.01.2018).

STATSBYGG (2006). *Experiences in development and use of a digital Building Information Model BIM according to IFC standards from the building project of Tromsø University College (HITOS) after completed Full Conceptual Design Phase: Tromsø University College (HITOS) for testing IFC: R&D project no. 11251 Pilot project*. Tromsø University College, Oslo, Norway.

STAUFER, M. J. (1987). *Piktogramme für Computer: Kognitive Verarbeitung, Methoden zur Produktion und Evaluation*, Band 2 von *Mensch Computer Kommunikation*. W. de Gruyter, Berlin and New York.

STEEL, J., DROGEMULLER, R. und TOTH, B. (2012). Model interoperability in building information modelling. In: *Software & Systems Modeling*, 11(1):99–109.

SUCCAR, B. (2010). Building Information Modelling Maturity Matrix. In: DIMA, I., UNDERWOOD, J. und ISIKDAG, U. (Herausgeber) *Handbook of Research on Building Information Modeling and Construction Informatics*, Advances in Civil and Industrial Engineering, 65–103. IGI Global.

SUTHERLAND, W. R. (1966). *The on-line graphical specification of computer procedures*. Dissertation, Massachusetts Institute of Technology. URL: http://dspace.mit.edu/bitstream/1721.1/13474/2/25697177-MIT.pdf (Letzter Zugriff am: 10.12.2018).

TAN, X., HAMMAD, A. und FAZIO, P. P. (2010). Automated Code Compliance Checking for Building Envelope Design. In: *Journal of Computing in Civil Engineering*, 24(2):203–211.

TAVARES, R. M. (2009). An analysis of the fire safety codes in Brazil: Is the performance-based approach the best practice? In: *Fire Safety Journal*, 44(5):749–755.

TIEN FOO SING QI ZHONG (2001). Construction and Real Estate NETwork (CORENET). In: *Facilities*, 19(11/12):419–428.

TRIMBLE (2018). Grasshopper-Tekla Live Link | Tekla User Assistance. URL: https://teklastructures.support.tekla.com/de/not-version-specific/en/ext_grasshopperteklalink (Letzter Zugriff am: 8/5/2018).

TRZECIAK, M. (2018). Towards Registration of Construction Drawings to Building Information Models. In: STEINER, M., THEILER, M. und MIRBOLAND, M. (Herausgeber) *Proc. of the 30th Forum Bauinformatik // 30. Forum Bauinformatik*. Bauhaus-Universität Weimar.

UHM, MI-YOUNG, PARK, YOUNG-HYUN, WON, JONG-SUNG, LEE und GHANG (2013). A Study on the Development Direction and Priority of the BIM-Based Hospital Design Validation Technology. In: *JOURNAL OF THE ARCHITECTURAL INSTITUTE OF KOREA Planning & Design*, 29(6):147–155.

UHM, M., LEE, G., PARK, Y., KIM, S., JUNG, J. und LEE, J.-K. (2015). Requirements for computational rule checking of requests for proposals (RFPs) for building designs in South Korea. In: *Advanced Engineering Informatics*, 29(3):602–615.

US GENERAL SERVICES ADMINISTRATION (2008). GSA Preliminary Concept Design BIM Guide. URL: https://www.gsa.gov/cdnstatic/GSA_BIM_Guide_Series_%281%29.pdf (Letzter Zugriff am: 10.02.2018).

VDI (2017). VDI 2552 Blatt 5:2017-10 - Entwurf. URL: https://www.beuth.de/de/technische-regel-entwurf/vdi-2552-blatt-5/277546257 (Letzter Zugriff am: 21.09.2017).

VDI (2017-02). VDI 1000: Richtlinienarbeit. URL: https://www.vdi.de/technik/richtlinien/vdi-1000/ (Letzter Zugriff am: 15.01.2018).

VISMANN, U. (Herausgeber) (2017). *Wendehorst Bautechnische Zahlentafeln*. Springer Fachmedien Wiesbaden, Wiesbaden, 36. Auflage. URL: http://dx.doi.org/10.1007/978-3-658-01689-0.

VITEK, C. und VITEK, T. (2017). *Baurecht*. Praxishandbuch. 2. Auflage.

VON BERTALANFFY, L. (1972). The History and Status of General Systems Theory. In: *Academy of Management Journal*, 15(4):407–426.

VRIES-BAAYENS, A. (1991). *CAD product data exchange: conversions for curves and surfaces*. Dissertation, TU Delft, Delft. URL: https://repository.tudelft.nl/islandora/object/uuid%3A3c671856-6016-4747-9b32-cb9713dc2267 (Letzter Zugriff am: 15.12.2018).

W3C (2004a). RDF Primer. URL: https://www.w3.org/TR/rdf-primer/ (Letzter Zugriff am: 10.04.2017).

W3C (2004b). SWRL. URL: https://www.w3.org/Submission/SWRL/ (Letzter Zugriff am: 10.01.2017).

W3C (2017). OWL. URL: https://www.w3.org/OWL/ (Letzter Zugriff am: 15.02.2018).

WEIDNER, I. (12.01.2017). Digitaler Zwilling. In: *Süddeutsche Zeitung*, 2017. URL: http://www.sueddeutsche.de/wirtschaft/digitales-bauen-digitaler-zwilling-1.3329982.

WELLER, W. (2008). *Automatisierungstechnik im Überblick: Was ist, was kann Automatisierungstechnik?* Beuth Wissen. Beuth Verlag GmbH, s.l., 1. Auflage.

WENDT, S. (1998). Formalismen und Anschauung. URL: http://www.fmc-modeling.org/download/publications/wendt_1998-formalismus_und_anschauung.pdf (Letzter Zugriff am: 09.01.2018).

WIENER, E. und CURRY, R. (1980). Flight-deck automation: Promises and problems. In: *Ergonomics*, 23(10):995–1011.

WIX, J. (2007). Information Delivery Manual: Guide to Components and Developement Methods. URL: http://iug.buildingsmart.org/idms/development/IDMC_004_1_2.pdf (Letzter Zugriff am: 15.02.2018).

YOUNG, N. W., JONES, S. A. und BERNSTEIN, H. M. (2009). *The Business Value of BIM: McGraw-Hill Construction*. SmartMarket Report.

YURCHYSHYNA, A., FARON-ZUCKER, C., LE THANH, N. und ZARLI, A. (2009). Towards an Ontology-enabled Approach for Modeling the Process of Conformity Checking in Construction. In: VAN ECK, P., GORDIJN, J. und WIERINGA, R. (Herausgeber) *Advanced information systems engineering*, Lecture notes in computer science. Springer, Berlin.

YURCHYSHYNA, A. und ZARLI, A. (2009). An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. In: *Automation in Construction*, 18(8):1084–1098.

ZHANG, J. und EL-GOHARY, N. (2012). Extraction of Construction Regulatory Requirements from Textual Documents Using Natural Language Processing Techniques. In: *Proceedings of the 2012 ASCE International Conference on Computing in Engineering*, 453–460.

ZHANG, J. und EL-GOHARY, N. (2015). Automated Information Transformation for Automated Regulatory Compliance Checking in Construction. In: *Journal of Computing in Civil Engineering*, 29(4):B4015001.

ZHANG, J. und EL-GOHARY, N. (2016a). A Prototype System for Fully Automated Code Checking. In: CHRISTODOULOU, S. (Herausgeber) *eWork and eBusiness in Architecture: ECPPM 2016 : Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016), Limassol, Cyprus, 7-9 September 2016*. CRC Press.

ZHANG, J. und EL-GOHARY, N. (2016b). Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking. In: CHRISTODOULOU, S. (Herausgeber) *eWork and eBusiness in Architecture: ECPPM 2016 : Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016), Limassol, Cyprus, 7-9 September 2016*. CRC Press.

ZHANG, J. und EL-GOHARY, N. (2016c). Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking. In: *Journal of Computing in Civil Engineering*, 04016037.

ZHANG, J., EL-GOHARY, N. und EL-GOHARY, N. M. (2017). Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. In: *Automation in Construction*, 73:45–57.

ZHONG, B. T., DING, L. Y., LUO, H. B., ZHOU, Y., HU, Y. Z. und HU, H. M. (2012). Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking. In: *Automation in Construction*, 28:58–70.

ZUBKE-VON THÜNEN, T. (1999). *Technische Normung in Europa: Mit einem Ausblick auf grundlegende Reformen der Legislative: Zugl.: Erlangen, Nürnberg, Univ., Diss., 1997*, Band 12 von *Beiträge zum europäischen Wirtschaftsrecht*. Duncker & Humblot, Berlin.