www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

KNOWLEDGE-BASED ENGINEERING FOR INFRASTRUCTURE FACILITIES: ASSISTED DESIGN OF RAILWAY TUNNELS BASED ON LOGIC MODELS AND ADVANCED PROCEDURAL GEOMETRY DEPENDENCIES

SUBMITTED: June 2015
REVISED: October 2015
PUBLISHED: October 2015 at http://www.itcon.org/2015/26
EDITOR: Amor R.

Javier Ramos Jubierre,
Chair of Computational Modeling and Simulation;
javier.jubierre@tum.de

André Borrmann,
Chair of Computational Modeling and Simulation;
andre.borrmann@tum.de

SUMMARY: *The current design of infrastructure facilities is mainly driven by the application of guidelines, codes and international standards. The complex interpretation and geometric representation of those rules impedes the dynamic search for alternative solutions. Parallel to this, in recent years there has been an increasing tendency toward the adoption of 3D modeling solutions. Of particular benefit in the design of infrastructure facilities is the use of parametric modeling systems since they allow the definition of flexible models that can easily be adapted to satisfy changes in the boundary conditions. By defining dependencies and constraints, engineers are able to capture the underlying engineering knowledge in the parametric model. However, the design knowledge that can be defined by those instruments is limited to a reduced number of constraints and simple algebraic relations among parameters that are clearly insufficient to describe the complexity of engineering rules. To close this technological gap, this paper presents a novel knowledge-based engineering (KBE) approach that captures the design knowledge engineers apply in the interpretation and generation of infrastructure models based on rules specific for infrastructure facilities. To achieve this, distinct knowledge units named logic models capture the generation knowledge needed to convert the abstract information engineers must deal with into a corresponding parametric geometry model. In addition, this paper presents a methodology to integrate logic models with parametric design systems. To this end, logic models are introduced as a new set of domain-specific features responsible for the generation and management of the geometry, connecting the results with other part of the model by advanced procedural geometry dependencies. Finally, the proposed methodology is verified in a real case study for a suburban railway tunnel that is in the planning stage in the city of Munich, Germany. The paper concludes with a detailed discussion of the proposed approach and an outlook on future developments.*

KEYWORDS: *logic models, parametric modelling, Knowledge-based Engineering, KBE*

REFERENCE: *Javier Ramos Jubierre, André Borrmann (2015). Knowledge-based engineering for infrastructure facilities: assisted design of railway tunnels based on logic models and advanced procedural geometry dependencies. Journal of Information Technology in Construction (ITcon), Vol. 20, pg. 421-441, http://www.itcon.org/2015/26*

1. INTRODUCTION

The globalization of world economies has driven an increasing population migration from rural areas to large and megacities. Close spatial proximity promotes collaboration between producers, reduces transportation costs and increases opportunities in labour markets (Henderson, 2000). As a consequence of urban agglomeration, authorities have been forced to seek new efficient public transportation solutions that reduce air pollution and respond to increasing environmental awareness among the population.

Parallel to this, technological advances in the construction of tunnels using Tunnel Boring Machines (TBM) has increased the number of tunnels in urban areas, allowing them to go deeper, become longer, and grow in diameter, and making subway transportation one of the most successful solutions (Guglielmetti et al., 2008). The design of a new subway infrastructure facility is, however, a challenging task where not only technical aspects such as soil conditions or tunnel diameters vary from project to project, but also the teams of engineers who work together (Dave and Koskela, 2009).

Despite these demanding requirements, the frequent exchange of information among specialized teams during the design phase is still mainly based on drawings. This technology is not capable of representing the relationships between the information elements they display (Liu et al., 2013). Thus, designers and engineers must be aware of changes in the project and manually update the geometric model, in a time-consuming and error-prone process.

In response to the limitations of drawing systems, one can observe a constantly increasing in the usage of parametric computer-aided design (CAD) systems in the last few years in the Architecture, Engineering and Construction (AEC) industry. Despite the model consistency and design acceleration that parametric CAD systems offer (CAMBASHI, 2009), the design process is still dominated by routine work. Engineers and experts invest 10% of their time on administrative duties, 70% on routine tasks and only 20% on effective work. This dilates the design process and impedes the creative searching of innovative solutions based on the *what-if* analysis (Cooper et al. 1999; Gardan and Gardan, 2003).

Much of the knowledge that engineers apply in the design of subway infrastructure facilities is based on engineering rules. Understanding and implementing their abstract knowledge in a parametric CAD system is not a simple or straightforward task. One example of such rules is found in the design of the alignment of an infrastructure facility. Instead of working with the 3D spatial-curve of the alignment, engineers prefer to work with two 2D curve representations, the horizontal and the vertical alignments, which enable the handling of the complexity and the various constraints involved in alignment design in a direct manner.

In short, in the design of an infrastructure facility, engineers invest up to 80% of their time in administrative and routine tasks. Much of this time is spent in the interpretation of engineering rules and their representation in a parametric 3D model.

To close this technological gap, this paper presents a novel knowledge-based engineering (KBE) methodology based on discrete knowledge units, that we called logic models, and their integration within parametric CAD systems. In particular, the presented approach addresses the automation of infrastructure's design by capturing engineering rules into a set of logic models that later are used to assist the engineer in the creation of the required parametric geometry. The time invested in repetitive work is reduced, allowing engineers and experts to concentrate on the conceptual design.

Additionally, this paper presents a new methodology for interconnecting the generation capabilities of several logic models. This integration defines advanced procedural dependencies that enable parametric systems to consistently update the geometry when a modification is performed.

The remainder of this paper is organized as follows: Section 2 discusses the state of the art regarding parametric CAD systems and generative KBE systems. Section 3 covers two points: Firstly, it presents a novel generative KBE approach that enables the encapsulation of the modeling process based on engineering *best practice* rules, and secondly, it discusses its integration into parametric CAD systems as a new set of construction operations. Section 4 provides a proof-of-concept of the described approach built on a real-world case study. Finally, Section 5 concludes the paper with a discussion of the contributions and limitations of the presented approach and proposes future developments.

2. BACKGROUND/RELATED WORK

2.1. Parametric modeling

The concepts underlying parametric CAD systems were developed in the mid 1980s (Roller 1991, Shah and Mäntylä 1995) and first implemented in a commercial product by Pro/ENGINEER in 1988 (PTC, 2014). Nowadays parametric CAD systems are the state-of-the-art in mechanical and aeronautical industries. Although the AEC industry is still mainly based on drawings, parametric systems have been successfully applied to model linear infrastructure facilities such as bridges and roadways (Obergriesser et al., 2011; Ji et al., 2013).

One of the reasons for this success is the flexibility of parametric models to adapt themselves to changing boundary conditions. On the one hand, parametric CAD systems maintain the construction steps needed to re-build the model instead of keeping only the final outcome geometry. On the other hand, parametric models define dependencies between these construction steps – also known as procedural or construction operations – that allow the user to modify a single operation while the system updates the complete geometry in a consistent manner.

Parametric sketches and construction history

Parametric models – also known as procedural or construction history models – implement a twofold approach based on parametric sketches and construction operations. In particular, 2D parametric sketches are flexibly defined by applying parametric dimensions and geometric constraints. Later on, based on these sketches, 3D volumes are created and modified through the consecutive application of construction operations such as extrusion, rotation or Boolean intersections of solids (Shah and Mäntylä, 1995; Monedero, 2000; Betting and Shah, 2001).

Parametric sketches are defined by three different types of objects: geometry elements (e.g. points, lines and circles), geometric constraints and dimensional constraints. From the two types of constraints, geometric constraints apply geometrical relations between pairs of geometry elements that specify their relative position (Sitharam et al., 2006). Figure 1 depicts some of the geometric constraints typically available in major parametric CAD systems.

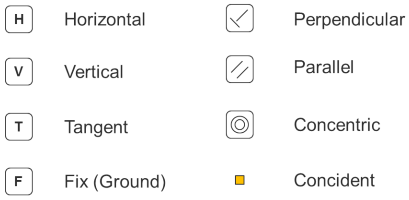

Figure 1: Geometric constraints typically provided by parametric CAD systems

Dimensional constraints, on the other hand, are used to restrict the size or the position of geometric elements. Furthermore, each dimension comprises a parameter that can be defined as a static value or as an algebraic equation where other parameters can be interrelated. Combined, these two types of constraints (geometric and dimensional) enable the generation of complex 2D designs that capture the design intent and provide a high degree of flexibility (Regli, et al., 2000; Chandrasegaran et al., 2013). Figure 2 and Figure 3 show two sketches where typical geometric and dimensional constraints have been applied.

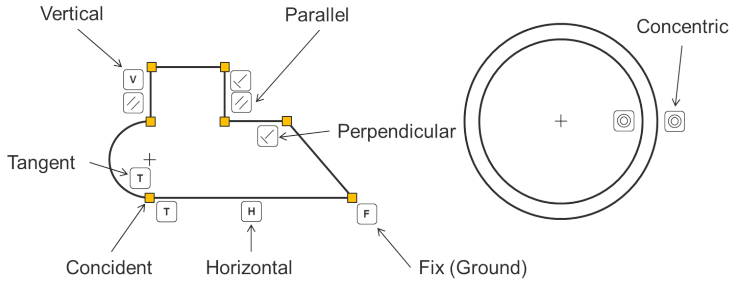

Figure 2: Example of geometric constraints applied to a 2D sketch

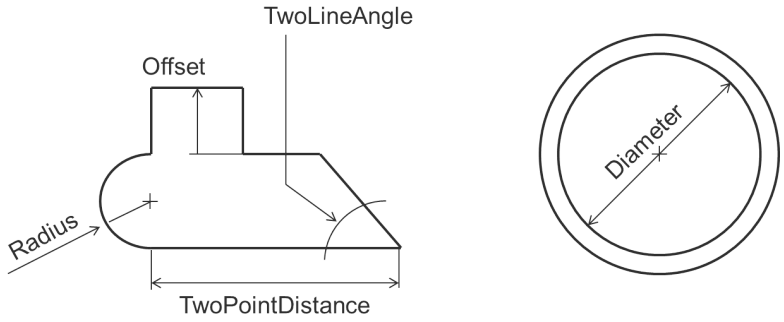

Figure 3: Example of dimensional constraints applied to a 2D sketch

Construction operations provided by parametric CAD systems can be classified in three groups:

- **Creation or sketch-based** operations need at least one sketch to create a volume. Typical operations are: extrusion, rotation and sweep operations. In Figure 4 a tunnel section is created by means of a parametric sketch and an extrusion operation.
- **Modification or non-sketch-based** operations are applied directly to a volume and do not need a sketch to be performed. Typical operations are: fillet, chamfer and Boolean operators.
- **Auxiliary geometry** operations create additional reference geometry needed to complete the construction task. Typical auxiliary elements are: work-planes, work-axes and work-points. An example is depicted in Figure 5 where a work-plane is defined perpendicular to a 3D spline at a given point. Later on, this work-plane can be defined as the reference surface for a parametric sketch.

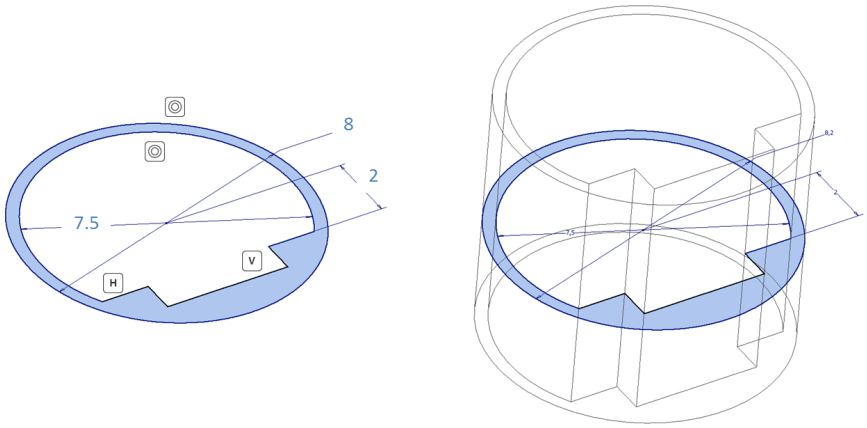

Figure 4: (left) parametric sketch of a tunnel cross section and (right) 3D extruded tunnel

Another important aspect that distinguishes parametric sketches from construction history in procedural models is the approach they use to solve the geometric problem. Parametric sketches analyze and solve the geometric constraint problem independently of the sequence its dimensions and constraints were introduced, in a methodology known as a variational approach. Differently, in the construction history the construction operations are sequentially retrieved, in a methodology known as procedural. (Anderl and Mendgen, 1998).

The variational approach handles the geometry and constraints defined in a parametric sketch as topological elements, whose size and placement must be calculated depending on the values of their parameters (Hoffmann and Joan-Arinyo 2005; Bettig and Hoffmann, 2011). One simple example is a triangle sketch defined by three lines, three dimensions and the necessary geometrical constraints. Thus, for a single set of topological elements, the variational approach will return different geometric solutions depending of the value defined on their parameters.

Parametric CAD systems use Geometric Constraint Solvers (GCS) to accomplish this task. Every time a topological modification in the sketch is introduced or one of the dimensions is updated, the GCS will start the

solving process. If the number of constraints (geometric and dimensional) equals the degree of freedom of the system, the GCS will return an updated size and placement for the different topological elements. However, if the system is defined with too many or insufficient constraints (over- or under-constraint) the GCS will not be able to find a solution and a warning message will be prompted.

On the contrary, the procedural approach employed on the construction history generates the final model by sequentially applying construction operations to the previously evaluated geometry. Unlike the variational approach, changes in the sequence of operations may end in a different outcome geometry.

Dependency relations in procedural models

As mentioned before, one important characteristic of procedural models is the available construction history defined in the modeling process. In addition, within the creation of the construction history, relations between operations are automatically generated by the parametric CAD system in a transparent manner to the end user. These relations can be understood as dependencies and allow the parametric CAD system to properly update the related geometry when an operation is modified or deleted.

A simple example of such dependencies can be found in the relation between an extrusion and a parametric sketch. Normally, the user begins with the definition of the sketch and after defining the height parameter, the CAD system will create the extruded volume and the dependency between both operations. This dependency has a clear direction – the extrusion is dependent on the parametric sketch, represented by a directed arc from the sketch to the extrusion (see Figure 5). This indicates that the two operations cannot be swapped in the construction history, as no extrusion can be created before its outlining geometry.

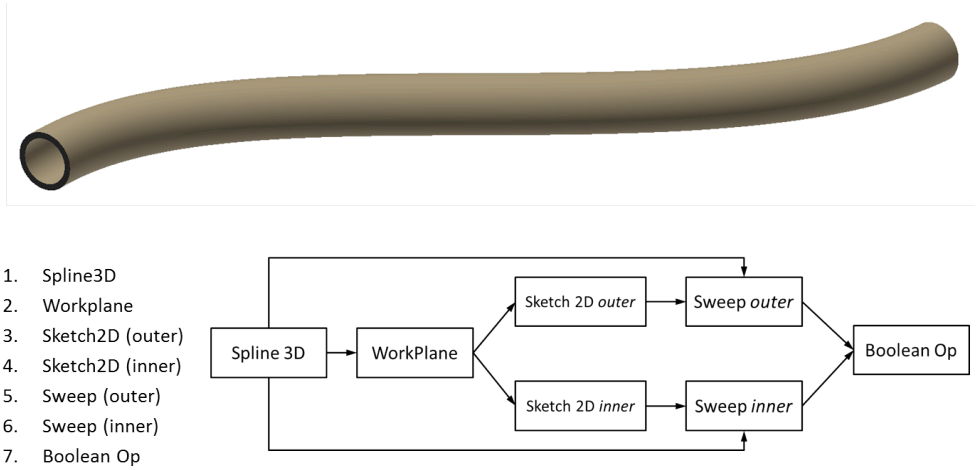

Figure 5: Parametric tunnel (top) modelled by seven construction operations (left) and its dependency diagram (right)

As a consequence, the geometric operations and their dependencies represent a directed acyclic graph (DAG) where the nodes are the construction operations and the edges the dependencies between them. Although the dependency diagram is unique for a given set of operations generating a desired geometric result, the sequence of executing operations is not necessarily fixed. Thus, the swapping of two operations is possible when they are consecutive and there is no dependency path between them. This property of the construction history can be seen in the example depicted in Figure 5, where there is no strict sequence of performing operations 5 and 6, and their execution can therefore be swapped without altering the resulting geometry.

Additionally, parametric CAD systems can manage procedural models that define several disconnected bodies. A disconnected body is defined as a set of construction operations that do not have dependencies on any other operation of the construction history. The corresponding dependency graph becomes a disconnected subgraph. This behavior plays a key role in the design of infrastructure models because this methodology enables the division of the complete project into a set of submodels that can be treated independently (Jubierre & Borrmann, 2013).

2.2. Knowledge-based Engineering

In the late 1980s, CAD systems became a standard tool in almost all industrial disciplines. At that time, CAD systems concentrated on simple geometry generation and manipulation rather than assisting the automatic generation of geometric models based on accumulated engineering knowledge (Regli et al. 2000, Chapman & Pinfold, 1999).

To answer this technological problem, in the late 1980s big corporations such Airbus and Jaguar began to develop KBE methodologies (Tech-Clarity, 2012). Defined as the crossroad of fundamental technologies such as Artificial Intelligence (AI), computer-aided design (CAD) and object-oriented programing (OOP), KBE approaches capture and reuse the knowledge engineers apply in the design process, reducing the routine work and consequently increasing the time available to develop more creative solutions (Chapman & Pinfold 2001, Skarka, 2007).

The basic architecture of KBE, as defined by La Rocca (2012), consist of three modules that analyze specific functional requirements such as size, cost, and performance, that process them using encapsulated knowledge contained in rules, codes, and design tables, and that return optimal engineering designs in the form of drawings or 3D models.

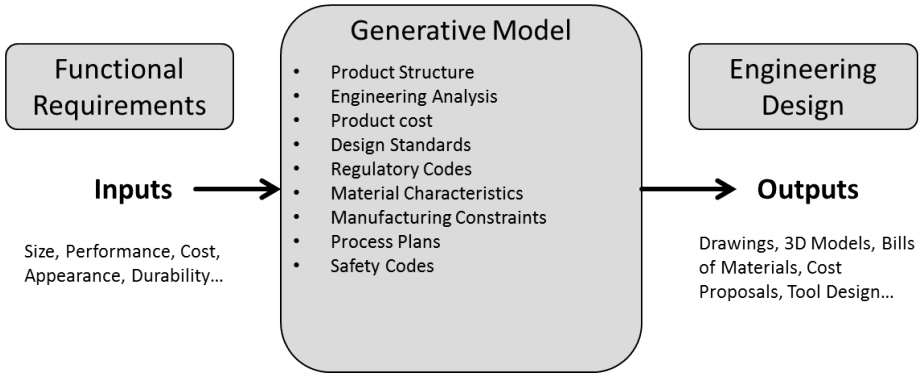

Figure 6: Basic KBE architecture (La Rocca, 2012)

Nowadays KBE systems have a well-established presence in mechanical and aeronautical industries. The interest in knowledge systems has increased so rapidly in the last two decades that it has been asserted that in the next decade KBE techniques will have the same importance that the CAD/CAM/CAE systems had in the 1990s (Howard, 1998).

Though we are seeing a slower adoption of modern modelling technologies in the AEC sector, we expect that knowledge-based systems will play a very important role in the near future. Chandrasegaran et al. (2013) have gone as far as forecasting a *knowledge revolution,* where traditional CAD systems would move from a geometry-based to a knowledge-based approach, coining the new term knowledge-aided design (KAD).

Generative KBE methods

Examining the different approaches for the implementation of KBE systems, Penoyer et al. (2000) categorized them by the solving method used and the type of user-interaction demanded. According to this classification, generative KBE approaches govern, without any other interaction, the creation or modification of the final geometry based on predefined constraints and user input information. This generative capability is of great importance in the design of subway infrastructures because we aim to separate the conceptual design from the tedious task of the geometry generation.

The literature includes several attempts at providing generative capabilities to KBE systems. Already in the late 1980s Adeli and Balasubramanyam (1988) described the first generative KBE system – known as design expert at that time – developed to calculate bridge trusses. Later, Gardan and Gardan (2003) reported an innovative approach, based on scripts, where the implementation and application tasks were split built on the expertise of the user – the *meta* description for the software engineer and the *knowledge* description for the CAD user. However, none of the previous examples linked KBE with parametric CAD systems.

One of the first approaches that linked a KBE system with a parametric CAD system was reported by Myung and Han (2001). Their approach presented a novel KBE methodology that was able to modify parametric assemblies based on the concepts of Design Unit, Functional Part and Functional Feature. While the concepts of design unit and functional part contained the knowledge that could manipulate assemblies and parts, the functional feature was the basic geometric unit proprietary to each CAD system. Although the functional features could be reused from one project to another, the main weakness of this approach remained the fact that each design unit or functional part was specific for a particular assembly or part and its knowledge could not be fully integrated until the model was finished.

Using a similar parametric approach Skarka (2007) presented an evolved methodology based on the MOKA approach and implemented it in Dassault CATIA. Instead of defining design units and functional parts, Skarka makes use of the two MOKA models, i.e. the informal model and the formal or generative model (Brimble & Sellini, 2000; Verhagen et al., 2012). For the informal model, Skarka developed basic conceptual charts based on the ICARE forms, while the generative task was left to the CATIA utilities. Like with the limitations of Myung and Han's method, the approach Skarka took needed a pre-modelled geometry to feed the generative model with geometry and parameters.

None of the two previous approaches is really able to generate geometry from scratch and even though both can update a pre-designed or template model, the description of the knowledge was always *a posteriori*, namely, after modeling the reference model.

In an attempt to advance the definition of the knowledge to the generation of the geometry, first van der Laan and van Tooren (2005) and later La Rocca (2011) developed a combined approach based on High Level Primitives (HLP) and Capability Modules (CM). Thus, the HLPs were defined as artefacts which contained the procedural knowledge specific for a single geometry and the CMs as artefacts that provide functionality to pre-defined types of HLPs. To clarify this approach with an example, van der Laan and van Tooren implemented this methodology for the design and structural calculation of airplane movables related to the topology of the airplane. They developed a set of different movable configurations that were encapsulated in a set of HLPs and CMs. When done this way, the KBE system could change the movable topology just by exchanging the HLP applied. Although the definition of the CMs was standard and therefore possible to define *a priori*, the definition of the HLPs was strongly coupled to the output geometry and is thus of limited usage in a creative process.

To increase the flexibility at the time that reducing the number of HLPs and CMs, Amadori et al. (2012) presented a derived approach based on the usage of High Level CAD templates (HLCt) that enable the instantiation and updating of a template model based on its boundary conditions. Even though this novel approach enables the modeling of a huge set of elements with a reduced amount of templates, the main shortcoming remains the definition of boundary conditions in a way that such templates can recognize and follow them.

3. LOGIC MODELS AND ADVANCED PROCEDURAL GEOMETRY DEPENDENCIES

3.1. Overview

The design of infrastructure facilities is strongly governed by guidelines, codes and national standards. Understanding and implementing the underlying abstract knowledge in a CAD system is not a simple task. Moreover, experts should interpret the demands of such engineering rules, match them to their specific requirements, and express their decisions in an appropriate product model. In conclusion, the design task is time consuming and hampers the creative searching of innovative solutions based on the *what-if* analysis.

One example of those engineering rules can be found in the description of the alignment of the infrastructure facility. As previously mentioned, the design of the alignment is a combination of 2D horizontal and vertical curves. For the horizontal alignment, which describes the curvature related to the XY-plane, engineers have to define parameters such as the minimal radius for a curve or the clothoid's parameter for the connection curves, which will establish the cant and the "smoothness" of the driving in a track. Similarly, for the vertical alignment, engineers must decide the values of the parameters that rule the length of crest and sag curves or the slope between two kilometric points. Finally, the information of both curves must be combined before being represented in one or more 3D spatial-curves.

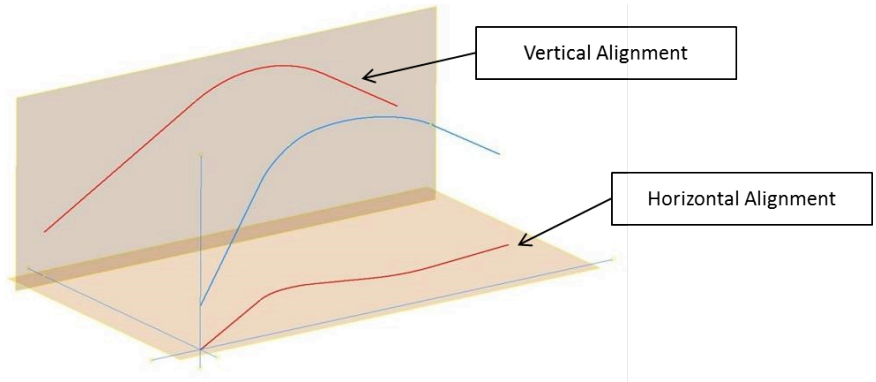

Figure 7: Alignment of an infrastructure as a combination of its horizontal and vertical alignment curves

One essential benefit of modeling these engineering rules in a parametric CAD system is that the outcome geometry produced after applying the rule is enclosed in a few construction operations that can easily be recognized and isolated in the construction history. Furthermore, the interpretation of such engineering rules can be easily captured in a KBE system in a way that the user only needs to provide the input information and then leave the system to return a new or updated procedural geometry.

Consequently, we propose a new generative KBE methodology based on logic models integrated in parametric CAD systems. In particular, each logic model encapsulates a different engineering rule and is able to interpret the input parameters and to generate a corresponding set of procedural operations. This approach also allows engineers and experts to modify complete sections of an infrastructure facility by the simple modification of one parameter, and to visualize the resulting geometry almost instantaneously.

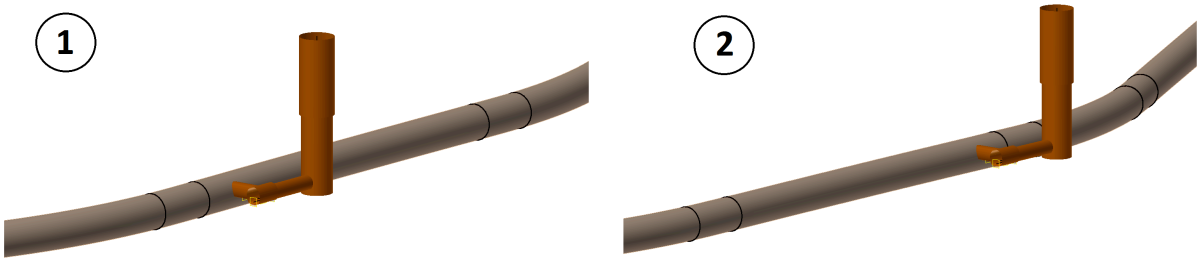

Figure 8: Rescue shaft attached to a subway tunnel in two different locations. The generation and modification of such geometries can easily be managed using logic models.

3.2. Logic model definition

As discussed in section 2.2, one limitation of the existing generative KBE systems is the fact that the knowledge is incorporated in the design process after a first geometric model is created. This limitation forces engineers to create large libraries of template models, which may answer specific design issues. Although this limitation is difficult to overcome, the design based on engineering rules give us the possibility to create KBE structures in advance.

Therefore, we define the concept of logic models as a component of the basic KBE architecture described by La Rocca (La Rocca, 2012), where the input is the abstract knowledge, the engineer must deal with, and the output the set of procedural operations needed by a parametric CAD system to model the outcome geometry. Figure 9 shows the architecture defined by our approach.

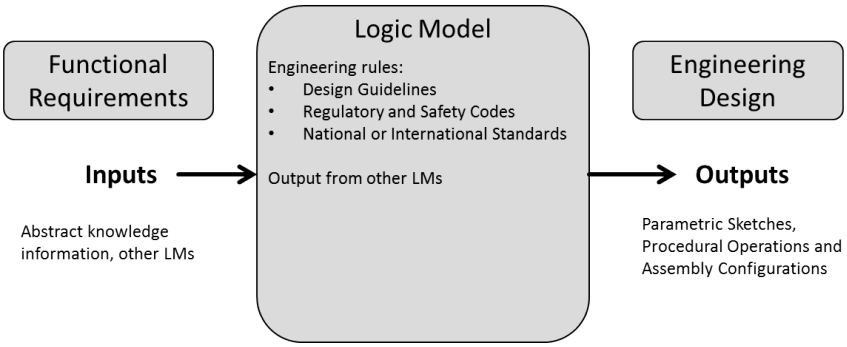

Figure 9: Logic Model architecture defined as a component of La Rocca's architecture.

To achieve this goal, we implemented the concept of logic models in a three-layer structure that allows the representation of the engineering knowledge and the geometry generation algorithms in an abstract way, making it independent of a specific CAD system. Hence, the top layer acts as an interface to the user and contains the logic unit. This unit encloses the knowledge the user must provide in order to fully define the selected engineering rule. The second layer, named logic interpreter, translates this abstract knowledge into a set of "neutral" (CAD system independent) procedural operations that parametric systems cannot directly employ. Finally, the geometry layer receives the neutral information from the logic interpreter and converts it into a set of proprietary procedural operations specific for one parametric system. In the following we provide a formal description of every layer.

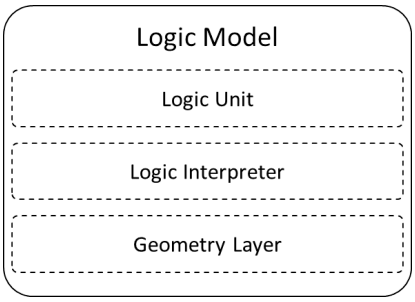

Figure 10: Logic model layer architecture.

Logic Unit (LU)

The logic unit embodies the abstract knowledge the engineer must provide to fulfil the requirements of the engineering rule. This unit remains independent of the CAD system and its information can be captured by a graphical user interface or defined by an external tool. Thus, in our approach, the information can be submitted in two ways: (1) based on the definition of the knowledge contained in a built model or (2) by the direct application of a neutral data format. One example of this duality can be found in the alignment model, where the content of the logic unit can be introduced directly in the designed model or read from an LandXML instance file.

The definition of our logic unit is close to the formal model described in the MOKA approach and which is usually modelled using the MOKA Modeling Language (MML) (Stokes 2001). Although MML was originally derived from the UML modeling language, we decided not to use it and to stick with the UML. On the one hand, this decision was based on the fact that we wanted to offer the possibility of developing logic units to any software expert without previous experience of system engineering process. On the other, the literature describes besides MML other process modeling languages – e.g. Alloy or SysML – that can be used instead (Bailey et al. 2004, He 2006).

For some widespread engineering rules, private companies and standardization organizations already developed neutral data models that enable the exchange of abstract knowledge, e.g. LandXML developed by Autodesk and OKSTRA developed by BASt, a German organization working for the Ministry of Transport (Rebolj et al. 2008, Schultze and Buhmann 2008). Although the majority of such models only exchange the outcome information, their content can be used as structure to capture and initialize the information of the logic model.

Logic Interpreter (LI)

The main task of the logic interpreter is to translate the abstract knowledge represented by the logic unit into a set of neutral procedural operations. In the scope of our research we developed a neutral data model that supports the exchange of procedural geometries between parametric CAD systems and which was successfully validated by enabling the exchange between the two parametric CAD systems, Autodesk Inventor and Siemens NX (Borrmann et al., 2014).

Although the standard input information for the interpreter is obtained from its logical unit, interpreters can also handle neutral procedural operations produced by other interpreters. This property enables the linking of several logic models and will be explained in the next section of this paper.

Similar to the definition of the logic unit, logic interpreters are defined independent of the selected parametric system and therefore can be also executed externally. Consequently, its development can be realized by software engineers in any available programing language.

Geometry Layer (GL)

Unlike the previously described layers, the geometry layer is embedded in the parametric system. Implementing this layer as a CAD module enables the geometry layer to accomplish two main tasks: firstly, to convert the neutral procedural operations into a set of proprietary construction operations that a specific CAD system can handle, and secondly, to lock the generated operations in front of manual modification by the user and thereby to avoid inconsistencies in the model.

The proprietary characteristic of the construction operations force logic models to implement different geometry layers depending on the parametric system selected. This need does not mean that a single logic model must embed all possible geometry layers. Usually engineers complete their design in a single parametric system and therefore only one geometry layer is needed. Only if a modification is required at the time to exchange the final model, the receiving CAD system will require a suitable geometry layer for its logic model.

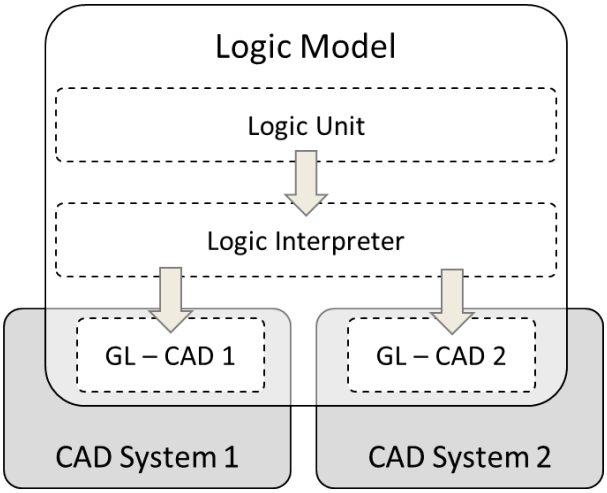

Figure 11: Integration of a Logic Model with two different CAD systems. The Logic Unit and the Logic Interpreter are independent of the CAD system selected while the Geometry Layer must be specifically defined for each system. Arrows shows the information exchange fluxes between layers. This figure does not imply the parallel or simultaneous use on two CAD systems.

Due to the fact that several logic models may need to convert the same type of neutral construction operations, e.g. parametric sketches, swept volumes, and Boolean operations, the strategy of using a common geometry layer can reduce the implementation effort. The common geometry layer acts as an interface between the different logic models and the parametric system. Having all geometry layers concentrated in a single module allows developers to enhance and update operation converters in a faster and consistent way.

3.3. Linking of logic models for more complex modeling

As the design of an infrastructure facility evolves, the knowledge needed to model it also becomes more complex. To avoid repetition in the knowledge's definition and the need of developing logic models that describes all possible scenarios, several logic models can be horizontally connected to achieve more complex modeling.

A simple example of two logic models that can be linked to achieve more complex modeling can be found in the alignment and axis curves of a ring tunnel. On the one hand, the logic model of the alignment contains the information needed to create a spatial-curve – described by the curve mid-way between the two rails – based on the horizontal and vertical alignments. On the other hand, the logic model responsible for the tunnel axis – used to guide the Tunnel Boring Machine (TBM) – creates a second spatial-curve that is shifted from the alignment curve by vertical and horizontal parameters. As the tunnel axis is based on the curve described by the alignment, both models can be linked to avoid redundancy and reduce the logic's complexity. Hence, a modification in the alignment model will update the geometry of the alignment and the axis curve – even when the tunnel axis model remains unchanged.

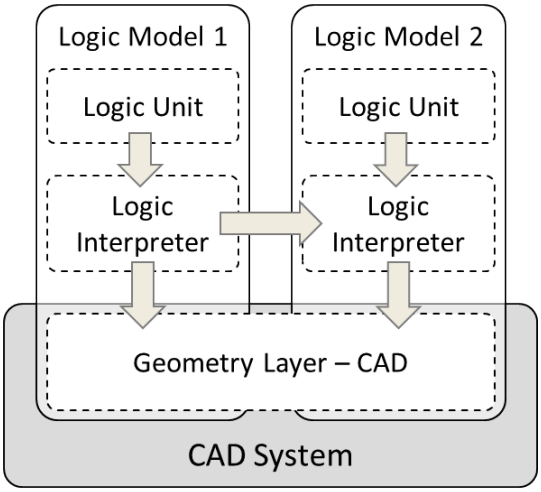

Figure 12: Integration of two Logic Models within one CAD system. The two logic models are linked through the logic interface layer to generate a more complex knowledge.

This behavior is achieved by horizontal connectors, which allow logic models to communicate among each other. Horizontal connectors implement a one-to-many communication that notifies the changes and the logic content from the related model to the linked models.

Horizontal connectors

Although the workflow in the logic model is mainly top-down, from the logic unit to the geometry layer, several logic models can be connected horizontally to create a more complex modeling scenario. Horizontal connections are defined as a one-to-many relationship and communication is only granted between connectors of the same type. In addition, horizontal connections provide two different types of communication between logic models: active and passive connections.

The active connection enables all linked models to be informed when an update arises in the related logic model. To implement this communication strategy we made use of the widely known observer-pattern (Gamma et al., 1994). This pattern defines two main elements, known as subject and observer objects. The subject object sends out update notifications without knowing the receivers, while the observer object subscribes to the notifications broadcast by a subject object. After receiving a notification, the linked model can start an update process to collect the changes. The passive connection enables linked logic models to access the information contained in the related logic model at any time. This communication mainly takes place during the model's initialization process when the logic interpreter is first executed and the neutral geometry created.

Although horizontal connectors are defined in an abstract way, the information they exchange is exclusively defined for a specific logic model. In particular, the related model offers to any linked logic model access to its

logic unit and logic interpreter independently of its purpose. Subsequently, it is a task of the linked model to establish structures that can handle the information provided.

To illustrate this behavior we have closer look on the connection between the alignment and the tunnel axis. In the creation stage, the logic model takes chance of the passive connection to request the content of the logic unit and the abstract construction operations created by the alignment logic unit. During the modeling workflow, each time the alignment is updated, the tunnel axis model receives a notification of this change and consequently starts an active connection to request only the modified information.

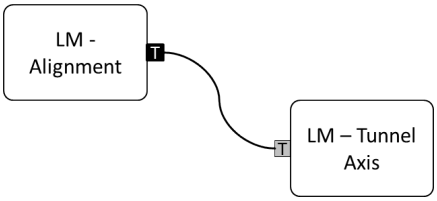

Figure 13: Example of a simple logic model diagram. The source-connector (black) of the alignment model is communicating its changes with the sink-connector (grey) placed in the tunnel axis model. 'T' is the connectors' name that distinguishes them from other connector types.

To represent the connections among logic models we make use of a logic model diagram where the nodes are the logic models and the edges the horizontal connections between models. To differentiate horizontal connections we place two differently colored connectors at the extremes of the edges depending on whether they are source or sink connectors. This nomenclature follows the concepts of subject and observer objects. Finally, every connector is represented by one letter that identifies the information it exchanges.

Due to the fact that each logic model pre-defines its own connections as soon as a new logic model is instantiated in the modeling process, the system will grant its communication. If the expected related model does not exist, the system will not allow the logic model to be initialized and an error will be generated.

3.4. Integration of logic models into parametric CAD systems

The literature and practical experience shows that CAD users prefer to work with feature-systems than with the direct manipulation of surface geometries (Regli et al., 2000; Bidarra, 1999). In parametric CAD systems, the non-sketch-based procedural operations described in section 2.1 usually follow this feature approach, where users only introduce the semantic information of the geometry they want to create or modify while the CAD system manipulates the geometry and establishes the necessary dependencies. Figure 14 shows an example of a graphical user interface for the hole feature in Autodesk Inventor. In this feature interface, the user can define not only the placement and size of the hole, but also semantic information such as type of hole or angle of the drill point.

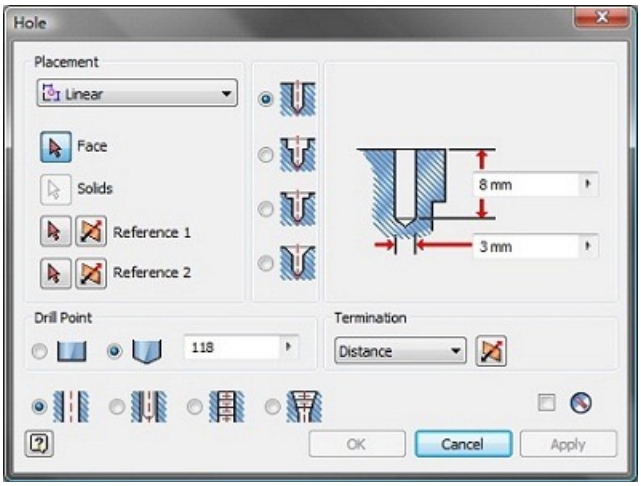

Figure 14: Hole-feature interface in Autodesk Inventor

Following this feature methodology model, we propose to integrate our logic model approach as a new set of domain-specific features in parametric CAD systems. This integration has two main tasks: (1) the generation of procedural geometry, and (2) the definition of advanced procedural geometry dependencies between construction operations that guarantees model consistency.

Geometry generation and consistency preservation through advanced procedural geometry dependencies

The successfully adoption of a KBE methodology is conditional on its integration in the engineer's workflow. This integration must additionally guarantee the consistency of the model enabling the coexistence of traditional and KBE geometry generation approaches.

In our approach each logic model is treated as a new parametric feature responsible for the interface with the user and for the geometry generation. In particular, our feature interface is a visual representation of the logic unit where the user introduces the abstract knowledge of the engineering rule. Next, as the user confirms the information, the logic interpreter will generate the set of neutral construction operations that will be provided to the geometry layer. Finally, in the geometry layer the neutral construction operations are translated and introduced at the bottom of the construction history in the parametric system. As the logic model's input is based either on abstract information or the result of other logic models, the dependency diagram of the existing model is not altered and its consistency guaranteed. In addition, as the construction steps generated by the logic model are native operations of the CAD system, every further modeling step based on those operations will be consistently updated by the dependencies of the parametric CAD system.

However, this behavior is not achieved when two or more logic models are linked. As can be seen in the dependency diagram in Figure 15 – where a small parametric tunnel was modelled using the two previously mentioned alignment and tunnel axis models – the parametric CAD system does not recognize any dependency between the splines produced by the two logic models, and therefore the geometry can become inconsistent if the user modifies the alignment.

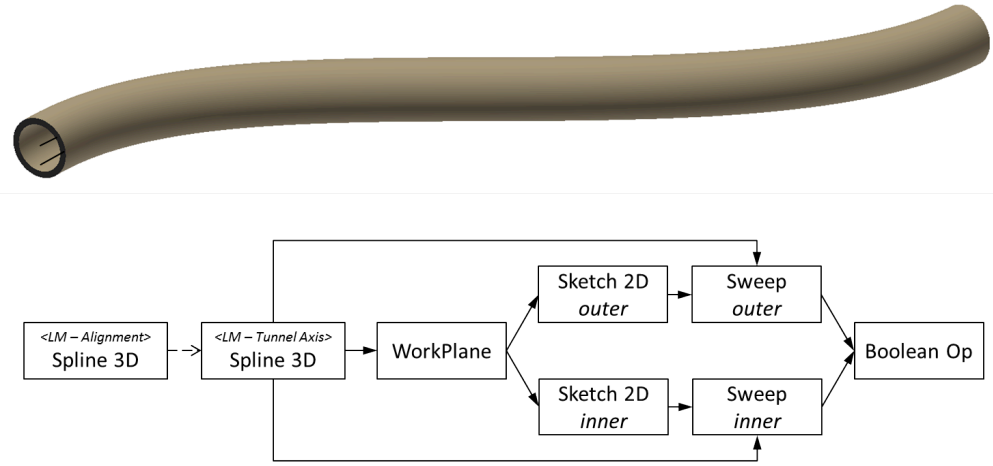

Figure 15: Railway tunnel modelled combining logic models and traditional parametric operations. Its dependency diagram shows the advanced procedural geometry dependency (dashed arrow) generated by the logic model and the parametric dependencies (solid arrows) generated by the CAD system.

To address this shortcoming we propose to extend the concept of horizontal connectors into a novel advanced procedural geometry dependency. Horizontal connectors, whose main purpose is to notify the update on a related logic model, can now be used as elements that trigger an update in the geometry. In this way, parametric dependencies guarantee consistency between procedural operations generated by CAD features, while the advanced procedural geometry dependencies keep consistency between geometries generated by logic models.

4. PROOF OF CONCEPT

To prove the feasibility of our approach, we applied it for modeling a subway tunnel that is soon to be constructed in the city of Munich, Germany. This infrastructure project, called the *Second main suburban track*,

has been designed to increase the limited capacity the actual main track shows and to provide a faster connection between the two ends of the city (Deutsche Bahn AG 2015).

The newly designed infrastructure facility is a nine-kilometer railway track with seven kilometers running underground in a twin-tunnel configuration. To avoid collisions with existing transportation networks the tunnels are planned 40 meters beneath the city's surface. This requirement, together with the length of tunnel sections – only three stops are planned within the city center – will make necessary to construct nine rescue shafts that allow passengers to safely leave the tunnel if required. Finally, an underground turnout has been designed for a future possible south branch. Altogether, this infrastructure facility is a highly challenging and dynamic project and the perfect scenario for testing our approach.

4.1. Logic models applied to the design of subway tunnels

For the case study we focused on the modeling of the tunnel section located between the stations Hauptbahnhof (city main station) and Marienhof that can be seen in Figure 16. Of the different engineering rules applied in the design of the proposed tunnel, we implemented four of them to test our approach. Although all four logic models are based on the same three-layer methodology already presented, each of them introduces slight differences to adapt them to their input and output requirements.

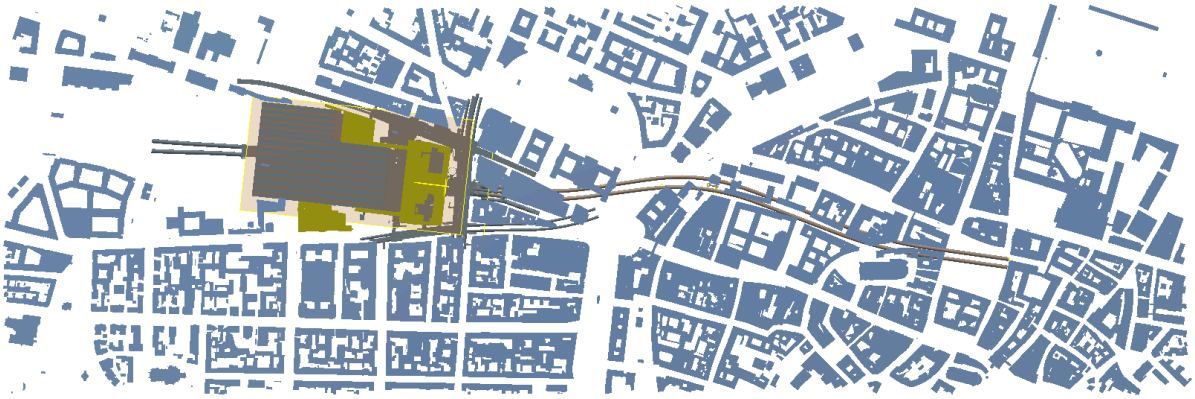

Figure 16: Partial model of the new second main suburban track planned for the city of Munich. In the model are represented the main station, one section of the tunnel and the inner city.

The first two models focus on the alignment and tunnel axis design, the third on the automatic generation of the ring used for the tunnel's lining and the fourth is responsible for the optimized combination of rings to minimize deviation of the tunnel within its axis. In the following we explain them in detail:

- The **Alignment Model (AM)** is responsible for converting the alignment information in a set of 3D splines in the parametric CAD system. The alignment of an infrastructure facility represents the curve mid-way between the two rails and is usually built on two 2D curve representations known as horizontal and vertical alignments. While the horizontal alignment is defined by straight segments, arcs and connection curves, the vertical alignment is only described by straight lines and their interpolation curves that generate the crest and sag curves. In our implementation, we divided the outcome alignment curve into a set of 3D spline segments that are equivalent in number to the divisions of the horizontal alignment. This division is not fixed and can be established differently for other infrastructure projects. Finally, for this logic model we developed the logic interpreter in such a way that the user can load the logic information directly from an exchange data format. More specifically, we use the LandXML data model as input for the logic model.

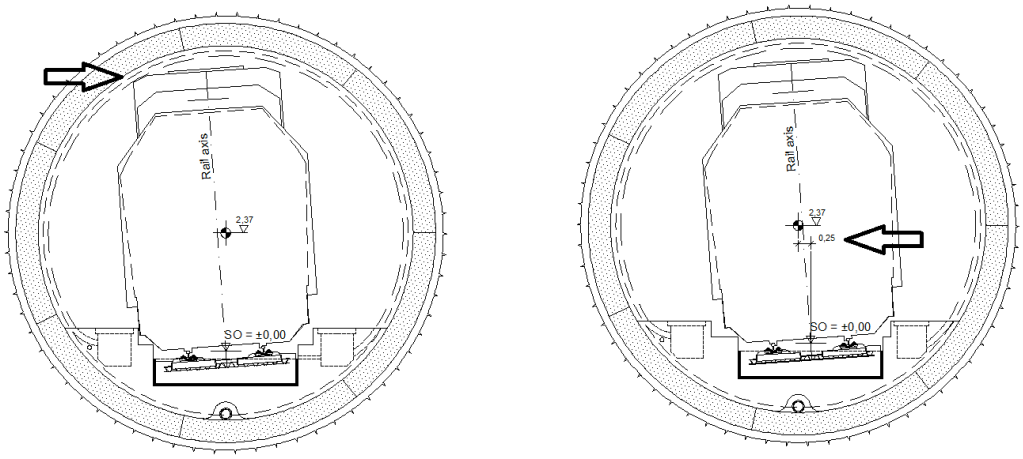

Figure 17: Example of how the clearance space of the train is affected by shifting the tunnel axis. On the left the axis is not shifted and the clearance of the train is overlapping the construction tolerances of the tunnel. On the right, the axis has been shifted and both spaces do not overlap.

- The **Tunnel Axis Model (TAM)** represents the alignment's shifted curve that guides the tunnel boring machine. By default the vertical shift parameter remains constant, while the horizontal parameter can differ for each curve. This shifting tries to minimize the tunnel's diameter caused by the train running on a canted curve. For the implementation of its logic unit we modelled the engineering rule as a table that allows tunnel experts to introduce a different horizontal shift parameter for each curve.

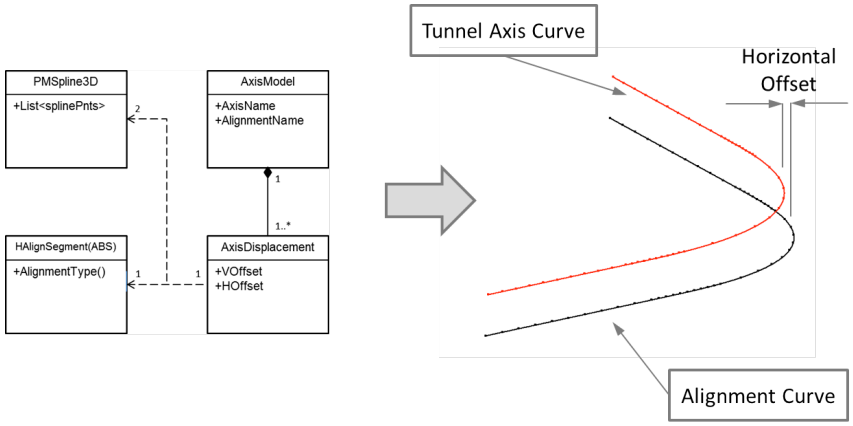

Figure 18: (left) UML diagram of the TAM's logic unit and its connection to the alignment model. (right) Geometrical representation of the tunnel axis curve in relation to the alignment curve

- The **Ring Design Model (RDM)** makes it possible to update a parametric ring model based on the input introduced in its logic unit. This logic model covers the inherent aspects of the ring, namely, the number of segments, its length, and its conicity. From one side rings have one or two tapered faces. The distance from its tapered surface and a theoretical parallel face is known as conicity and allow rings to follow a curved tunnel axis. The value of the conicity and the length of the ring define the minimum curvature radius a tunnel based on such a ring can follow. From the other side the number of segments in a ring is related to the diameter of the tunnel and the size of the key-segment, which is mounted last and closes the ring. Although the minimum value of conicity can be determined based on the curvature of the tunnel axis, the logic model is developed managing the design inputs as fixed values defined by the engineer, updating the model accordingly.

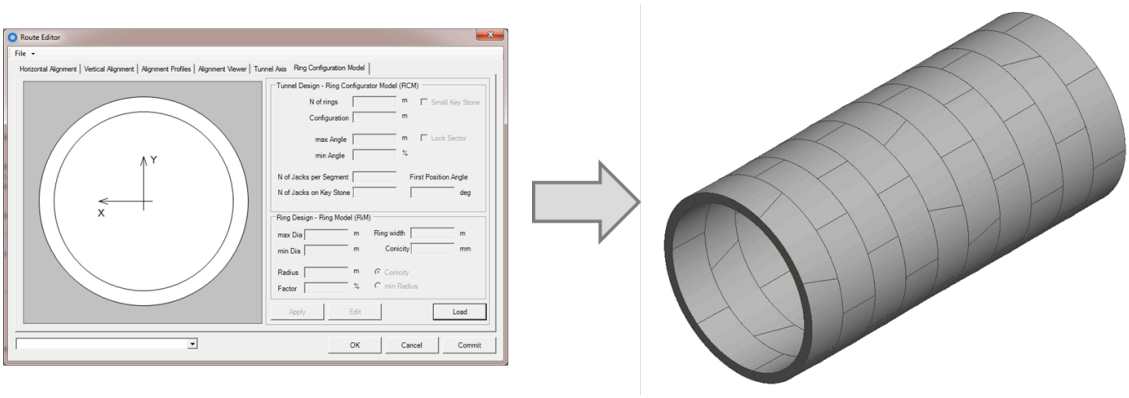

Figure 19: (left) Graphical user interface developed to introduce the logic information of the RDM and TCM. (right) Section of the outcome tunnel limited to ten rings

- The **Tunnel Configurator Model (TCM)** generates an assembly model based on the tunnel axis, the output ring defined by the RDM, and the limitations specified by the tunnel boring machine (TBM). Those limitations are usually restriction rules on the feasible combination of pairs of rings. From one side, rings are made out of several ring-segments that enable only a discrete number of assembly positions. From the other side some TBM delimits a lock sector where the key-segment cannot be mounted. Therefore an optimization process is needed in the logic interpreter that minimizes the deviation of the polygonal axis created by the selected combination of rings from the theoretical tunnel axis curve. For our prototypical implementation, we make use of the algorithm described by Socher (2013), which provides reliable results for short tunnels.

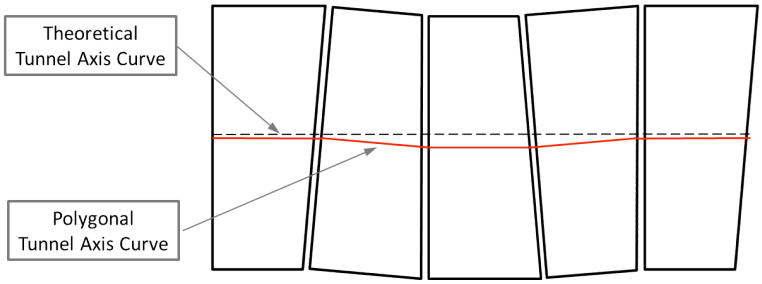

Figure 20: Tunnel section composed of five rings. The TCM creates an optimized combination of them to minimize the deviation between the theoretical and resulting polygonal tunnel axis curve

Besides the logic models, we additionally developed three horizontal connectors and its corresponding advanced geometrical dependencies that keep the parametric model updated in advance of changes. The horizontal connectors are defined by a single letter and are developed as follows:

- The connector **Track (T)** links in our case study the alignment and the tunnel axis models. In addition to the update events of the active connection, the track connection makes the horizontal alignment description and neutral procedural operations available to any logic model that implements the sink-plug.

- The connector **Axis (A)** links the tunnel axis and tunnel configurator models. An additional application of the axis connector – not covered in our case study – can link the tunnel axis with the ring design model. Thus, the ring's conicity parameter, defined as a fixed value in the ring design, could be dynamically updated with the alignment.

- The connector **Ring (R)** links the ring design and tunnel configurator models. This connector communicates the ring parameters defined in the logic unit.

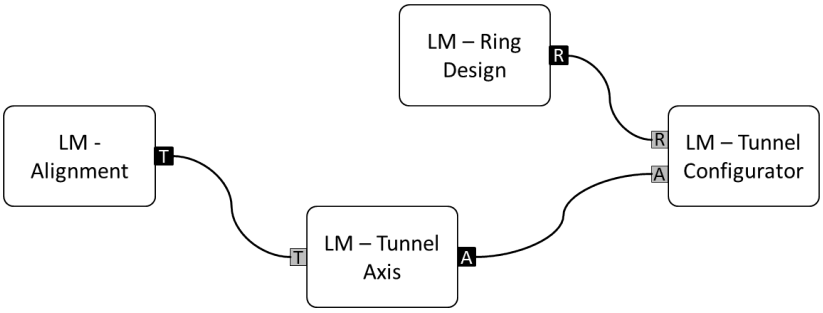

Figure 21: Instantiated logic model diagram and horizontal connections for the second main suburban tunnel in Munich

Finally, the partition of the horizontal alignment created by the alignment model makes it possible to represent the infrastructure facility based on several submodels. This modeling strategy enables the independent manipulation of one section of the tunnel. To avoid inconsistencies due to local modifications of the alignment, the logic model introduces extra geometrical dependencies among the submodels' splines that ensure the alignment's continuity on changes. This modeling methodology based on advanced procedural geometry dependencies guarantees not only the preservation of consistency of the model through the detailing process, but also through equivalent operations located in different submodels, making the logic model's approach more general and widely used.

4.2. Coupling logic models and parametric CAD systems on a real case study

For our prototypical implementation on the basis of a parametric CAD system we chose to use the commercial software Autodesk Inventor. The Autodesk product is one of the most widespread parametric systems and provides a convenient API development environment based on the .NET framework. Consequently, the combination of applications developed using the .NET programming language with *add-in* extensions of the parametric system achieved using the API environment, allowed us to follow a two-stage implementation strategy. First, we developed, using the .NET framework, the logic units and logic interfaces, which were generated separately from the parametric system selected. Second, we made use of the API environment to develop the geometry layer, which accesses the parametric CAD system and converts the neutral procedural operations into a set of construction operations proprietary to the parametric system.

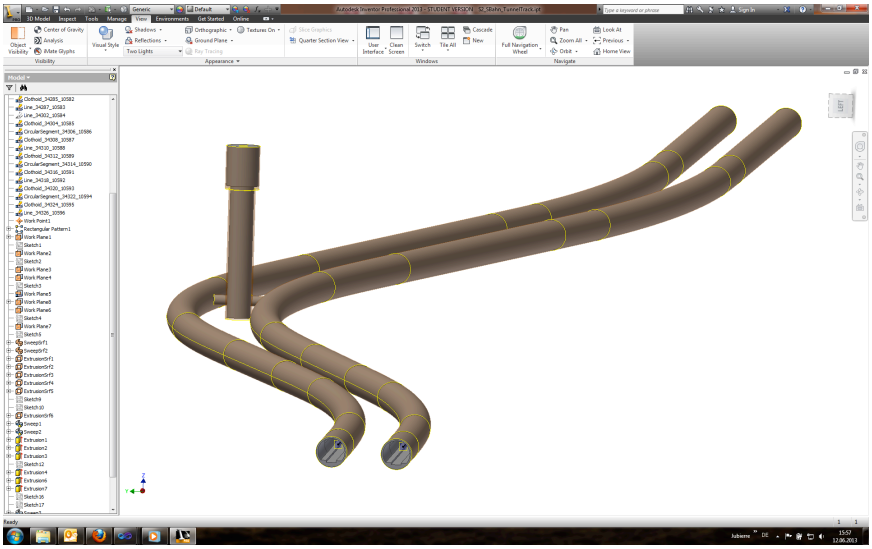

Figure 22: Tunnel and rescue shaft of the second main suburban track planned for the city of Munich

From a modeling point of view, the workflow begins with the definition of the alignment of the subway infrastructure. To perform this modeling step, the designer imports the alignment information directly from a

LandXML file. Once the alignment is converted into a set of 3D splines by the AM logic model, the definition of the tunnel axis can be introduced. To achieve this, the TAM provides a user interface where the value of the horizontal parameter can be defined separately for each curve. At this modeling point, the designer can carry on with the detailing process on the parametric system as usual, regardless of the logic model extension. Next, at the time of constructing the lining of the tunnel, the designer only needs to fill in the information requested by the user interface of the RDM and RCM, and the tunnel's lining will be automatically generated. Finally, to avoid inconsistencies in the generated geometry, logic models lock the construction operations to prevent manual manipulation. However, the interfaces defined by each logic models can be always reopened and its abstract information revised to complete a rapid and consistent modification of the final geometry.

4.3. Measuring efficiency gains

To find a metric that allows us to compare our approach with traditional modelling techniques is a difficult issue. First, there is a software implementation effort that due to the complexity in the development of a logic model an average engineer cannot alone undertake. We strongly believe that the development of new logic models must be the result of a collaboration between civil engineers and software developers.

Once this software implementation is done, the CAD users can integrate logic models into to their workflow as they would do it with any other feature of the parametric CAD system. For testing how much time is saved during the design stage and with every design's iteration we performed two experiments: Firstly, we selected from the presented case study a section of the tunnel containing 110 rings resulting in a total length of 300 meters. The time required by the TCM logic model to generate the model was about 25 seconds, while the manual assembly took an advanced Autodesk Inventor user almost two hours (1h 57min). Secondly, we modified the alignment of this tunnel section implementing a standard *what-if* analysis: Specifically, we changed the radius of a curve from 200 meters to 300 meters. This modification implied the update of the rotation parameter of 80 tunnel rings. Undertaking these changes manually took the advanced Inventor modeler almost 10 minutes (9 min 23 sec), while the logic model performed it almost instantaneously.

In conclusion, the employment of logic models and parametric CAD systems enables us to model our complex case study in a short time and with little effort. Moreover, the resulting parametric model is extremely flexible, i.e. the modification of one parameter, as the curvature radius of a horizontal alignment segment, will consistently update the complete tunnel model.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a new methodology that encapsulates the design knowledge engineers apply when constructing models of infrastructure facilities in order to fulfil technical demands, guidelines, codes or standards. The core concept of our approach is based on the definition of knowledge units denoted as logic models that encapsulate the generation knowledge needed to convert the abstract design rules engineers have to implement into a set of construction operations in a parametric CAD system.

We introduced logic models in a three-layer architecture that allow different experts to develop different layers depending on their expertise. On the top layer, the logic unit encapsulates the abstract knowledge experts and designers need to provide to fulfill the engineering rule. In our approach, logic units are integrated in parametric CAD systems as a new set of domain-specific feature interfaces. The second layer is the logic interpreter which is responsible for interpreting the abstract knowledge and translating it into a set of neutral construction operations. Although, these operations cannot be directly employed by the parametric system, they can be easily exchanged and used by other logic models. On the bottom layer, the geometry layer is responsible for converting the neutral construction operations provided by the logic interpreter into a set of proprietary operations specific to each parametric system.

In addition, several logic models can be interrelated to achieve more complex modeling results, reducing logic complexity and redundancy. This communication is achieved by the use of horizontal connectors, which inform the related logic models of any update arising on the referred model. This notification and update mechanism is introduced in the parametric CAD system as a novel advanced procedural geometry dependency. Consequently, construction operations created by two connected logic models, without any parametric dependencies among them, can still be updated, ensuring the consistency of the complete model.

The presented approach is general and applicable to a wide range of large infrastructure facilities such as roads, bridges or tunnels. Our Logic Model's architecture based on three layers enables the collaboration of different

experts in the development process. Later on, as our KBE methodology is totally integrated in the parametric CAD system, the end-user needs neither background knowledge in object-oriented programming nor advanced skills in parametric modeling, and gains flexibility in the modeling process for the *what-if* analysis.

With the presented case study we demonstrated that our approach is feasible for designing real-world infrastructure facilities. In particular, we developed four logic models that enable us to create and modify, at a highly detailed level, a parametric tunnel section of the second main suburban track, which will soon be constructed in the city of Munich, Germany. We showed that even in its most detailed stage the tunnel can be modified in a flexible and consistent way, from its more basic definition, i.e. its alignment.

After verifying our approach with the mentioned case study, we realize two main limitations. Firstly, the required software development task is highly demanding and requires the collaboration of civil engineers and software developers. Consequently, the implemented engineering rules must be carefully selected in order to balance the benefits gained with the effort created. Though the proposed common geometric layer clearly reduces this implementation effort, it may also create an additional burden when it has to be implementing in a multitude of CAD systems. Secondly, as soon as the engineering rule to be automated is getting closer to the finest level of abstraction, the logic interpreter needs to generate and manage larger sets of construction operations. This fact makes that complex geometry, such as the tunnel ring and the tunnel segments generated by the RDM, becomes rapidly too complex to be implemented in one single (monolithic) function. To solve this problem we created a simplified parametric model that the geometric layer can update with the results produced by the logic interface. This solution shows strong similarities with the High Level CAD templates (HLCt) proposed by Amadori et al. (2012). A future development of the logic model approach will thus be to explore the integration of HLCt in the logic interpreter layer for all those models based on complete geometries. This dual approach will reduce the implementation task and make the complete methodology more flexible.

Another suggested future development is the integration of logic models with *computational design software* (Autodesk, 2015). Such systems have their roots in visual programming languages and enable the easy and fast design of models with repetitive patterns. In particular, visual languages create geometry based on visual diagrams where nodes denote variables or geometry operations, and edges represent dependencies between these variables or operations. One of the most important characteristics here is the ability to generate loops over the generated geometry. This fact, together with the division of infrastructure models into parametric submodels as reported by Jubierre & Borrmann (2013), makes us believe that an integration of logic models as new type of nodes is possible and can increase the potential of such visual modeling tools.

6. ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the German Research Foundation (DFG) for funding the project under grant FOR 1546. We are also very thankful for the support of the engineering consultancy company *Obermeyer Planen und Beraten* for providing valuable insight into current planning practices and data for the presented real-world case study. We also thank *Deutsche Bahn* and *Landeshauptstadt München* for their support.

7. REFERENCES

Adeli, H. & Balasubramanyam, K. (1988). A Knowledge-Based System for Design of Bridge Trusses. Journal of Computing in Civil Engineering, 2(1), pp.1–20.

Amadori, K., Tarkian, M., Ölvander, J. & krus, P. (2012). Flexible and robust CAD models for design automation. Advanced Engineering Informatics, 26(2), pp.180–195.

Anderl, R. & Mendgen, R. (1998). Analyzing and optimizing constraint-structures in complex parametric CAD models, in: B. Bruderling, D. Roller (Eds.), Geometric Constraint Solving and Applications, Springer, Berlin, Germany, pp.58-81.

Autodesk (2015) Dynamo Studio. Computational BIM design software. Available: http://www.autodesk.com/products/dynamo-studio/overview, accessed May 31, 2015

Bailey, I., Dandashi, F., Ang, H., & Hardy, D. (2004). Using Systems Engineering Standards In an Architecture Framework. white paper eurostep company.

Bettig, B. & Shah, J.(2001). Derivation of a standard set of geometric constraints for parametric modeling and data exchange. Computer-Aided Design, 33 (1), pp.17–33.

Bettig, B., & Hoffmann, C. M. (2011). Geometric constraint solving in parametric computer-aided design. Journal of computing and information science in engineering, 11(2): 021001-021001-9. doi:10.1115/1.3593408

Bidarra, R. (1999). Validity maintenance in semantic feature modeling. PhD Thesis.

Brimble, R. & Sellini, F. (2000). The MOKA modelling language. In O. Dieng, Rose and Corby, Eds. Knowledge Management Methods, Models, Springer-Verlag Berlin Heidelberg, pp. 49–56.

Borrmann, A., Flurl, M., Jubierre, J.R., Mundani, R.-P. & Rank, E. (2014). Synchronous collaborative tunnel design based on consistency-preserving multi-scale models. Advanced Engineering Informatics, 28 (4), pp. 499-517. DOI: 10.1016/j.aei.2014.07.005

CAMBASHI LIMITED, 2009 AutoCAD 2010 Productivity Study. Available: http://www.cambashi.com/white-papers, accessed March 2013.

Chandrasegaran, S.K., Ramani, K., Sriram, R.D., Horváth, I., Bernard, A., Harik, R.F. & Gao, W. (2013). The evolution, challenges, and future of knowledge representation in product design systems. Computer-Aided Design, 45(2), pp.204–228.

Chapman, C.B. & Pinfold, M. (1999). Design engineering—a need to rethink the solution using knowledge based engineering. Knowledge-Based Systems, 12(5-6), pp.257–267.

Chapman, C.B. & Pinfold, M. (2001). The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. Advances in Engineering Software, 32(12), pp.903–912.

Cooper, S., Fan, I. & Li, G. (1999) A best practice guide – Achieving Competitive Advantage Through Knowledge-Based Engineering, Department of Trade and Industry (DTI), UK

Dave, B. & Koskela, L. (2009). Collaborative knowledge management—A construction case study. Automation in Construction, 18(7), pp.894–902.

Deutsche Bahn AG (2015) Die 2. Stammstrecke - Kernstück des Bahnknotens München. Available: http://www.2.stammstrecke-muenchen.de/startseite/, accessed June 1, 2015

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: elements of reusable object-oriented software. Adison-Wesley professional computing series. ISBN 0-201-63361-2

Gardan, N. & Gardan, Y. (2003). An application of knowledge based modeling using scripts. Expert Systems with Applications, 25(4), pp.555–568.

Guglielmetti, V, Grasso, P., Mahtab, A., Xu, S. (2008). Mechanized tunnelling in urban areas – design methodology and construction control. Taylor & Francis Group.

He, Y. (2006). Comparison of the Modeling Languages Alloy and UML. In Software Engineering Research and Practice, pp. 671–677.

Henderson, J.V. (2000). The effects of urban concentration on economic growth. In. NBER Working Paper No. 7503, Department of economics – Brown University, p.44.

Hoffmann, C. M., & Joan-Arinyo, R. (2005). A brief on constraint solving. Computer-Aided Design and Applications, 2(5), 655-663.

Howard, C.C. (1998). The virtual engineer, 21st century project development – Society of Manufacturing Engineers, ISBN:0-87263-491-4.

Ji Y., Borrmann A., Beetz J. & Obergrießer M. (2013): Exchange of Parametric Bridge Models using a Neutral Data Format. Journal of Computing in Civil Engineering, 27 (6), pp. 593-606, DOI: 10.1061/(ASCE)CP.1943-5487.0000286

Jubierre, J.R. & Borrmann, A. (2013). Cross-submodel consistency preservation in multi-scale engineering models, in: Proc. of the 14th International Conference on Civil, Structural and Environmental Engineering Computing (CSC2013), Sardinia, Italy.

La Rocca, G. (2011). Knowledge based engineering. Techniques to support aircraft design and optimization. PhD Thesis – TU Delft. ISBN/EAN 978-90-9026069-3

La Rocca, G. (2012). Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. Advanced Engineering Informatics, 26(2), pp.159–179.

Liu, F., Jallow, A.K., Anumba, C.J. & Wu, D. (2013). Building Knowledge Modeling: Integrating knowledge in BIM. In Proc of the 30th international Conference CIB W78. Beijing, China.

Monedero, J. (2000) Parametric design: a review and some experiences. Automation in Construction, 9, pp.369-377

Myung, S. & Han, S. (2001). Knowledge-based parametric design of mechanical products based on configuration design method. Expert Systems with Applications, 21, pp.99–107.

Obergriesser, M., Euringer, T., Borrmann, A. & Rank, E. (2011): Integration of geotechnical design and analysis processes using a parametric and 3D-model based approach. In: Proc. of the 2011 ASCE International Workshop on Computing in Civil Engineering. Miami, FL, USA.

Penoyer, J.A., Burnett, G., Fawcett, D.J. & Liou, S.-Y., (2000). Knowledge based product life cycle systems: principles of integration of KBE and C3P. Computer-Aided Design, 32(5-6), pp.311–320.

PTC (2014). PTC History and Acquisitions. Available: http://www.ptc.com/about/history, accessed November 2014.

Rebolj, D., Tibaut, A., Čuš-Babič, N., Magdič, A., & Podbreznik, P. (2008) Development and application of a road product model. In: Automation in Construction, Volume 17, pp. 719-728.

Regli, W.C., Hu, X., Atwood, M. & Sun, W. (2000). A survey of design rationale systems: approaches, representation, capture and retrieval. Engineering with computers, 16, pp.209–235.

Roller, D. (1991). An approach to computer-aided parametric design. Computer-Aided Design, 23(5), 385-391.

Schultze, C. & Buhmann, E. (2008) Developing the OKSTRA® Standard for the Needs of Landscape Planning in Context of Implementation for Mitigation and Landscape Envelope Planning of Road Projects. In Conference on information technologies in landscape architecture, pp. 310-320.

Shah, J.J. & Mäntylä, M. (1995). Parametric and Feature-based CAD/CAM: Concepts, Techniques, and Applications, Wiley Press Inc., New York.

Sitharam, M., Oung, J.J., Zhou, Y. and Arbree, A. (2006). Geometric constraints within feature hierarchies. Computer-Aided Design, 38, pp.22-38

Skarka, W. (2007). Application of MOKA methodology in generative model creation using CATIA. Engineering Applications of Artificial Intelligence, 20(5), pp.677–690.

Socher, A. (2013). Entwicklung eines Frameworks zur Generierung und Bewertung von Trassierungen im maschinellen Tunnelbau. Master Thesis at Ruhr-Universität Bochum - Lehrstuhl für Informatik im Bauwesen.

Stokes, M. (2001). Managing Engineering Knowledge – MOKA: Methodology for Knowledge Based Engineering Applications, Professional Engineering Publishing Limited, London

Tech-Clarity, (2012). Tech-Clarity Perspective: Best practices for developing industrial equipment. Tech-Clarity Inc.

Verhagen, W., Bermell-Garcia, P., van Dijk, R.E.C. & Curran, R. (2012). A critical review of Knowledge-Based Engineering: An identification of research challenges. Advanced Engineering Informatics, 26(1), pp.5–15.

van der Laan, A. & Van Tooren, M.J.L. (2005). Parametric modeling of movables for structural analysis. Journal of aircraft, 42(6).