TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Maschinenwesen
Lehrstuhl für Mikrotechnik und Medizingerätetechnik

Computergestützte Verifikation von Medizingerätenetzwerken

Max Emanuel Dingler

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende: Prof. Dr.-Ing. Birgit Vogel-Heuser

Prüfer der Dissertation:

1. Prof. Dr. rer. nat. Tim C. Lüth

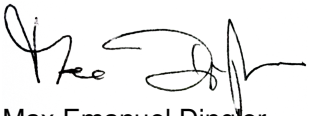
2. Prof. Dr.-Ing. habil. Alois Knoll

Die Dissertation wurde am 27.09.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 23.04.2018 angenommen.

Vorwort

Diese Arbeit entstand während meiner wissenschaftlichen Tätigkeit am Lehrstuhl für Mikrotechnik und Medizingerätetechnik unter der Leitung von Prof. Dr. rer. nat. Tim C. Lüth an der Technischen Universität München.
Im Rahmen des BMBF-Projekts *OR.Net* untersuchte der Lehrstuhl die dynamische, herstellerübergreifende Kommunikation von Medizingeräten mit Echtzeitanforderungen. Neben technischen Aspekten stellt die Konformitätsbewertung offener Netzwerke von Medizingeräten Hersteller, Betreiber und Anwender gleichermaßen vor neue Herausforderungen. Diese Arbeit beschreibt die Lösungen, die zur Bewältigung dieser Hürden entwickelt wurden.
Meinem Doktorvater Professor Tim Lüth danke ich sehr herzlich für die persönliche und fachliche Betreuung während meiner Promotion. Durch das mir entgegengebrachte Vertrauen durfte ich in unterschiedlichsten Leitungsfunktionen lehrreiche Erfahrungen machen, die mich weit über den wissenschaftlichen Rahmen hinaus begleiten werden.
Der Prüfungskommission danke ich sehr für die Begutachtung und Prüfung dieser Arbeit.
Meinem Mentor Dr.-Ing. Franz Irlinger verdanke ich sowohl wertvolle Anregungen bei der Konzeption dieser Arbeit, als auch zahlreiche persönliche Gespräche und Impulse, die mich über meine Zeit am Lehrstuhl sehr bereichert haben.
Unseren zahlreichen Projektpartnern aus Industrie und Forschung, insbesondere unseren TUM-Partnern *AIS*, *MITI* und *RES* danke ich für den fachlichen Austausch, die angenehme Zusammenarbeit und ihren Einsatz für den gemeinsamen erfolgreichen Projektabschluss.
Aufrichtiger Dank gilt meinen Kollegen, die mich über die vergangenen Jahre am Lehrstuhl durch alle Phasen dieses Projekts begleitet haben – für die fachlichen Anregungen und Diskussionen, die gegenseitige Motivation und den Rückhalt bei der gemeinsamen Projektarbeit, und die Nachtschichten vor dem Demonstrator Aufbau; allen voran meiner Gruppe *Navigation und Vernetzung* bestehend aus Christian Dietz, Franziska Klein, Tobias Lüddemann, Arne Menz und Jonas Pfeiffer.
Allen weiteren Kollegen, Studenten und Ehemaligen, deren namentliche Auflistung den Rahmen einer einzelnen Seite sprengen würde, danke ich von Herzen für die angenehme Zusammenarbeit und den freundschaftlichen Austausch über Fach- und Projektgrenzen hinweg.
Die Anfertigung dieser Dissertation und die vorausgehende intensive Projektarbeit wäre nicht möglich gewesen ohne den steten Rückhalt meiner Familie – besonderer Dank gilt daher abschließend meiner Frau Constanze, meinen Eltern Reinhard und Silvia, meinen Schwiegereltern Peter und Christine und meinen Geschwistern Flaminia, Nikolaus Alexander und Francesca Romana, deren fortwährende Unterstützung in allen Lebenslagen bedeutend zum Entstehen dieser Arbeit beigetragen hat.

München im September 2017



Max Emanuel Dingler

Inhaltsverzeichnis

1. Einleitung

Die vergangenen 15 Jahre waren in der operativen Medizin im stationären wie im niedergelassenen Bereich geprägt von der zunehmenden Bedeutung digitaler Daten und deren Integration. Maßgeblichen Anteil an dieser Entwicklung hat die durchgängige Digitalisierung von Geräten, sowie die fortschreitende Eröffnung klinischer Disziplinen für computer-assistierte Eingriffstechniken.
Dabei nimmt die digitale Bildgebung mit über 1 Mio. interventionellen Endoskopien im Jahr *(p.a.)* und knapp 3.5 Mio. diagnostischen Endoskopien p.a. in Deutschland allein im stationären Bereich (Statistisches Bundesamt, 2015) eine bedeutende Stellung ein. Diente die Endoskopie früher meist ausschließlich der Diagnostik, so ersetzt sie heute in weiten Teilen offen chirurgische und auch mikrochirurgische Prozeduren oder konkurriert zumindest mit diesen (Irion und Leonhard, 2017).
Bei weiteren digitalen Technologien verhält es sich ähnlich. Die intraoperative Navigation wurde beispielsweise in den 1990er-Jahren zunächst bei Eingriffen mit knöcherner Beteiligung – etwa in der Neurochirurgie, Orthopädie, Hals-Nasen-Ohren-Heilkunde oder Mund-Kiefer-Gesichtschirurgie – verwendet, kommt vereinzelt jedoch seit einigen Jahren auch in der Weichteilchirurgie zum Einsatz (Kenngott et al., 2015; Dickhaus und Metzner, 2017).
Bildgebende Systeme und Messsysteme sind jedoch bei Weitem nicht die einzigen Quellen digitaler Daten. Einen großen Anteil daran haben Peripheriegeräte, die zusammen mit diesen Systemen verwendet werden und intraoperativ laufend Biosignale, Messdaten, Steuerdaten und Konfigurationsparameter erzeugen und verarbeiten. Die typische endoskopische Ausrüstung besteht beispielsweise aus einen Geräteturm mit Monitor, Kamerasteuergerät, Video--Dokumentationsgerät, Kaltlichtquelle, Saug-/Spülpumpe und Insufflator (siehe Abb. 1). Je nach Eingriff kommen zusätzliche elektrisch gesteuerte Instrumente hinzu – etwa ein *Hochfrequenz-Gerät* (HF-Gerät), ein *Ultraschall-Dissektor* (US-Dissektor) oder ein Shaver (Liehn et al., 2011, 2014a; Aßmann et al., 2014).
Außerhalb der Endoskopie kommen weitere digitale Geräte (und damit geräteseitig erzeugte Daten) zum Einsatz. In der Neurochirurgie (kumuliert ca. 125.000 Eingriffe p.a. in Deutschland) gehören Operationsmikroskop, Laserschnittgerät, intraoperative Sonographie, Ultraschall-Zertrümmerer, Saug-/Spülpumpe, Bohrmaschine, Neuronavigation und *intraoperatives Neuromonitoring* (IONM) zu den typischen Arbeitsmitteln (Liehn et al., 2011; Sarnthein et al., 2012; Sadelfeld und Liehn, 2014).
Die Fülle an verfügbaren Daten könnte neuen Nutzen generieren, wenn diese über ein standardisiertes Protokoll ausgetauscht und von den Geräten eigenständig interpretiert würden – dies wird in den folgenden Abschnitten verdeutlicht.

Abbildung 1 Endoskopieturm der Firma Aesculap AG *(Aesculap)* mit Peripheriegeräten. Von oben nach unten: Monitore, Insufflator, Kaltlichtquelle, Kamerasteuergerät, Videodokumentationsgerät und Saug-/Spülpumpe.
Quelle: Aesculap AG.

1.1. Problemstellung – Vernetzung von Medizinprodukten

Aufbauend auf dem Voranschreiten der Digitalisierung in der Medizin ist seit einigen Jahren ein Trend zur *Integration* der Daten durch *Vernetzung* von Medizingeräten und Zubehör zu erkennen (vgl. Abb. 2). Vernetzung und Integration erstrecken sich über zahlreiche System- und Geräteklassen, die im Folgenden kurz skizziert werden:

1.1.1. Geräte der Intensivmedizin

Die Vernetzung in der Medizintechnik begann mit den Geräten der Intensivmedizin wie Spritzenpumpen, Beatmungsgeräten, oder Dialyse-Geräten für Nieren und Leber. Während diese zu Beginn der 80er Jahre isolierte Systeme mit proprietären (meist seriellen) Schnittstellen waren, besitzen sie heute eine Anbindung ins klinische IT-Netzwerk. Über diese können Behandlungsdaten ausgelesen, sowie Software- und Wartungsstände abgeprüft und ggf. Updates durchgeführt werden (Ahlbrandt et al., 2013). Weiter ist es nun möglich, Informationen über den Zustand des Patienten sowie Alarme über große Distanzen mittels der jeweils vorhandenen IT-Infrastruktur zu übertragen und anzuzeigen (Dickhaus und Metzner, 2017).

1.1.2. Klinische Informationssysteme

Klinische Informationssysteme *(KIS)* sollen den betrieblichen Ablauf in Krankenhäusern unterstützen, indem Patientendaten, Behandlungs- und Diagnosedaten sowie Daten aus Logistik und Verwaltung auf einer gemeinsamen Plattform abgelegt und verfügbar gemacht werden (Haas und Kuhn, 2017). Nahezu zeitgleich mit der Integration von Geräten der Intensivmedizin in den 80er Jahren begannen auch in diesem Bereich Standardisierungsbemühungen. Besondere Bedeutung haben dabei die Standards HL7 (Kommunikation von Untermodulen des KIS) und *DICOM* (Kommunikation speziell mit dem Radiologie-Informationssystem, insbesondere dem *Picture Archiving and Communication System* (PACS)) erreicht. Heute stehen Patientendaten aus KIS und PACS in vielen OP-Sälen an speziellen Terminals zur Verfügung. Einige Bildverarbeitungsoperationen lassen sich direkt auf modernen Bilderfassungssystemen wie der Computer-Tomographie *(CT)* oder Magnetresonanztomographie *(MRT)* größerer *Hersteller* durchführen und sodann wieder im PACS archivieren. Liegen die Ergebnisse dieser Verarbeitungsschritte wiederum in Standardformaten (meist DICOM) vor, können diese in das PACS zurückgespielt und beim Eingriff aufgerufen werden. Ist dies nicht möglich, so benötigt der Operateur einen zusätzlichen Rechner. In manchen Fällen werden KIS und PACS umgangen und eine direkte Verbindung mit Komponenten des OP-Saales bereitgestellt. Hierbei handelt es sich meist um Insellösungen (siehe Abschnitt 1.1.4), die nur mit dedizierten Komponenten weniger Hersteller funktionieren (Dickhaus und Metzner, 2017).

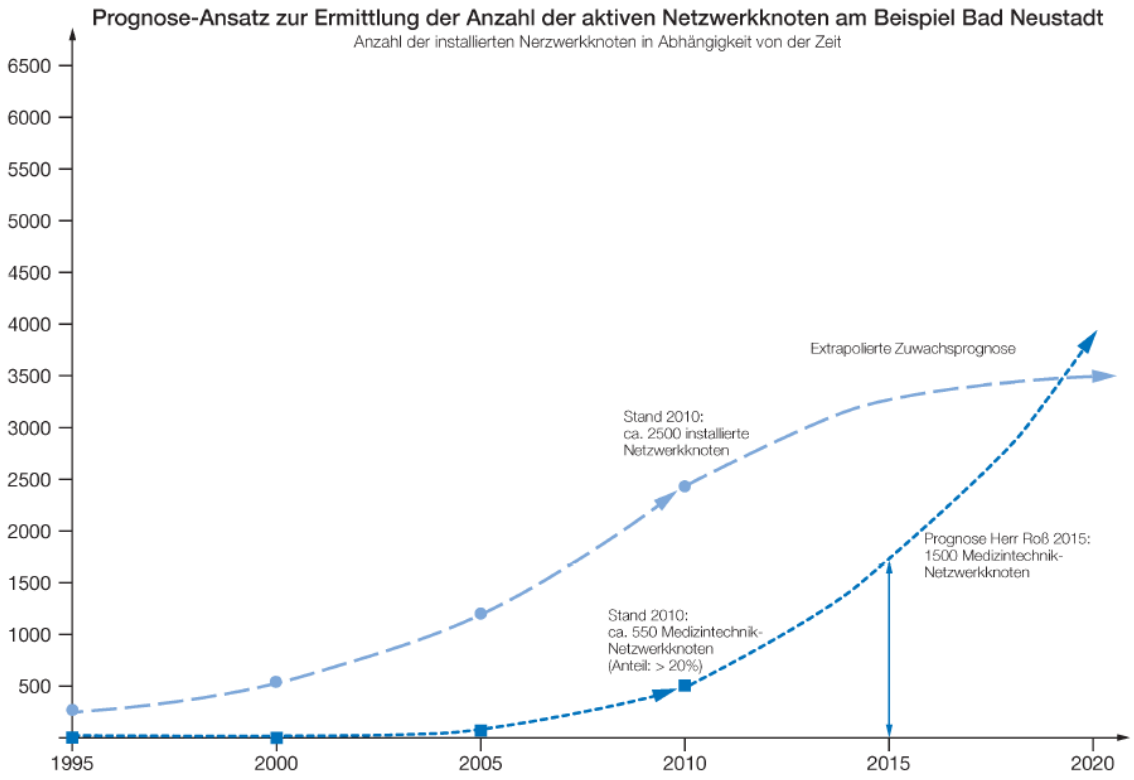


Abbildung 2 Extrapolierte Zuwachsprognose der Netzwerkknoten (hellblaue Linie), sowie des Anteils medizintechnischer Netzwerkknoten (dunkelblaue Linie) am Klinikum Bad Neustadt. Der Prognose zufolge ist davon auszugehen, dass der Anteil medizintechnischer Netzwerkknoten stetig wachsen und in den nächsten fünf Jahren fast 100% der Netzwerkknoten ausmachen wird. Unverändert entnommen aus (Ahlbrandt et al., 2013). Veröffentlicht unter der CC-Lizenz creativecommons.org/licenses/by-nc-nd/3.0/.

1.1.3. Integrierte Operationssäle

Eine spezielle Form der Datenintegration findet in sogenannten *integrierten Operationssälen* statt: Hierbei werden Medizingeräte und Zubehör *einzelner größerer* Hersteller über ein einheitliches, meist *proprietäres Kommunikationsprotokoll* (Dickhaus und Metzner, 2017) an eine zentrale Bedieneinheit angebunden (Abb. 3). Je nach Ausstattung werden so unterschiedliche Funktionen ermöglicht – etwa die zentrale Bedienung von Endoskopiegeräten und Peripheriegeräten (HF-Geräte, OP-Tische und -Leuchten, oder OP-Leuchten-Kameras), sowie der Dokumentations- und Telemedizineinheit durch einen zentralen Touchscreen im Sterilbereich oder Schwesternarbeitsplatz. Außerdem können wichtige Geräteparameter sowie Alarm- und Statusmeldungen einzelner Geräte auf dem zentralen Touchscreen angezeigt werden und Bilddaten auf unterschiedliche Monitore geroutet werden. Die Systeme verfügen über DICOM-Schnittstellen, um auf patienteneigene Bilddaten und Diagnosen zugreifen zu können. Über eine Schnittstelle zur Haustechnik können je nach Ausprägung Raumbeleuchtung, Klimaanlage etc. angesteuert werden. Integrierte Operationssäle kommen in allen chirurgischen Disziplinen, ob offen chirurgisch oder minimal-invasiv, zum Einsatz (Irion und Leonhard, 2017). Ziel *integrierter Operationssäle* ist es, Arbeitsabläufe zu optimieren, die Patientensicherheit zu erhöhen und die Kosteneffizienz zu verbessern (Wallwiener et al., 2011; Irion und Leonhard, 2017).

Abbildung 3 Integrierter Operationssaal „OR1“ der Firma *KARL STORZ GmbH & Co. KG* (Karl Storz). Die Geräte am Geräteturm (rechts im Bild) sind allesamt über ein firmeneigenes, proprietäres Bussystem miteinander verbunden. Geräteparameter können von der zentralen Bedieneinheit (Touchscreen links im Bild) aus dem sterilen Bereich heraus eingestellt werden. Alarm- und Statusmeldungen einzelner Geräte können auch zentral auf dem Touchscreen angezeigt werden. Außerdem können Videosignale anforderungsbasiert auf individuelle Bildschirme geroutet werden. Je nach Ausprägung können zudem Elemente der Haustechnik, wie etwa die Raumbeleuchtung, eingestellt werden.
Quelle: KARL STORZ GmbH & Co. KG.

1.1.4. Herstellergebundene Insellösungen

Als *Insellösungen* bezeichnet man herstellereigene, nicht-erweiterbare, lokale Netzwerke von Medizingeräten und Zubehör. Dies können einfache Punkt-zu-Punkt-Verbindungen, wie etwa ein US-Dissektor und der zugehörige Fußschalter sein (Abb. 4), oder aber komplexere Architekturen, die zumeist bei der intelligenten automatischen Steuerung von Geräten zum Einsatz kommen.

Ein Beispiel dafür ist die *Navigated Control Unit* (Lüth et al., 2001), mit der iatrogenen Verletzungen von empfindlichem Gewebe vorgebeugt werden kann. Dabei wird die Leistung angebundener chirurgischer Instrumente in Abhängigkeit ihrer Position durch Verknüpfung von Navigationsdaten, präoperativen CT-Daten und den Steuerdaten der Motoreinheit des Instruments geregelt (siehe Abb. 5). Sie wurde in der Wirbelsäulenchirurgie (Kneissler et al., 2003), der Dentalimplantologie (Koulechov und Lueth, 2004), der *funktionalen endoskopischen Sinus-Chirurgie* (FESS) (Strauss et al., 2005), sowie der HNO/Mastoidektomie (Luz et al., 2014) evaluiert. In letzterer Studie konnten durch die Verwendung von Navigated Con-

Abbildung 4 Punkt-zu-Punkt Verbindung von Medizingeräten und Zubehör am Beispiel eines US-Dissektors. Der Ultraschall-Generator wird von einem dedizierten Fußschalter über eine mechanisch und informationstechnisch proprietäre Schnittstelle angesteuert. Der Dissektor ist an den Generator ebenfalls über eine proprietäre Schnittstelle angebunden. Das Gerät besitzt keine weiteren Schnittstellen. Quelle: Söring GmbH.

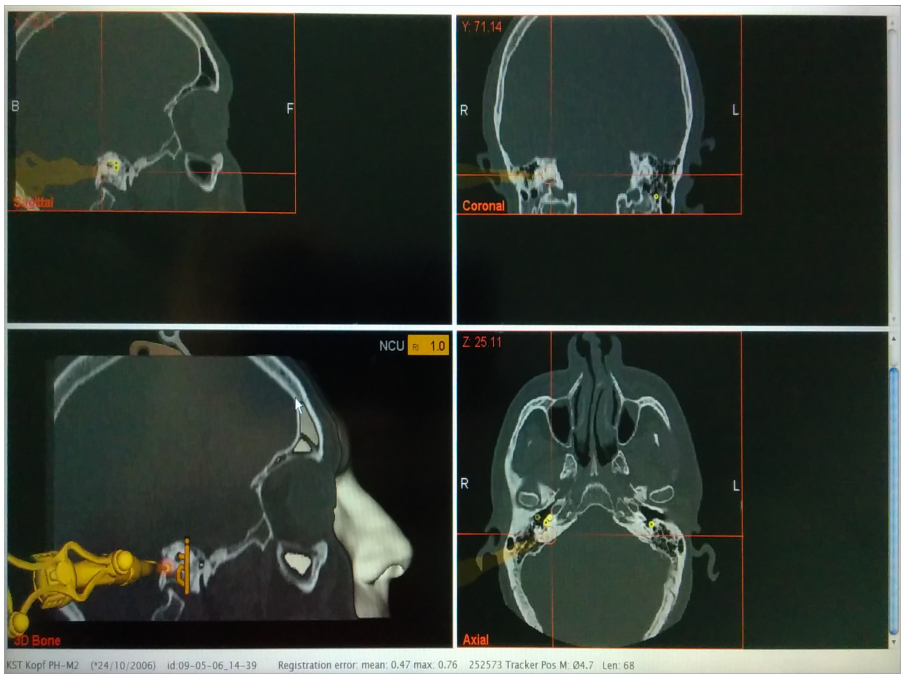


Abbildung 5 Screenshot der MiMed Navigationssoftware unter Verwendung der Navigated Control Unit. In den Bildern oben und rechts unten sind die 3 Ansichten von sagittal, coronal und axial zu sehen. Im Bild links unten ist das gelb eingeblendete Instrument zu erkennen, ebenso wie der gelb eingezeichnete Fazialis-Nerv. Wird der Abstand (rechts oben eingeblendet) geringer, so ändert sich die Instrumentenfarbe zu Rot und das Instrument wird abgeregelt – eine Funktionalität, die nur durch Verknüpfung der Navigation mit der Instrumentensteuerung zu erreichen ist.

trol bei einer simulierten Mastoidektomie ein besseres Operationsergebnis, geringerer physiologischer Aufwand und reduzierte Operationszeiten nachgewiesen werden.

Auch in anderen Anwendungsbeispielen finden Insellösungen Verwendung: Ein klassisches Problem der HF-Chirurgie ist die Freisetzung giftiger Rauchpartikel beim HF-Schneiden und -Koagulieren sowie der Anwendung von Laser. Diese setzen giftige Gase und Dämpfe (Kohlenmonoxid, Benzole) frei und transportieren ansteckende Krankheitserreger wie das Human Papillomvirus (Steege et al., 2016). Die Richtlinie TRGS 525, Kapitel 8.1 schreibt daher seit September 2014 vor, dass beim Einsatz derartiger Geräte entsprechende Absaugvorrichtungen eingesetzt werden müssen. Dabei kann es erforderlich sein, den Rauch direkt am Ort und im Moment der Entstehung abzusaugen (Bundesanstalt für Arbeitsschutz, 2014). Durch die Vernetzung mit entsprechenden eigenen Absaugkomponenten realisieren einzelne Hersteller von HF-Geräten eine intelligente, automatisierte Rauchgasabsaugung bei Betätigung des verursachenden Geräts.

Durch die Kombination integrierter Operationssäle mit neuartigen Steuerungs- und Regelungsfunktionen im *Surgical Deck* gelang es den Autoren in (Strauss et al., 2013), in der FESS deutliche Verbesserungen im Vergleich mit der Verwendung konventioneller Ausstattung festzustellen. Fast 90% der in dieser Studie befragten Anwender empfanden die Ergonomie als besser und über 70% erkannten einen Mehrwert des Surgical Deck für den Operationsablauf. Darüber hinaus fühlten sich 95% der Operateure von der Technik im Surgical Deck angemessen unterstützt.

1.1.5. Netzwerke aus Eigenherstellung

Im Gegensatz zu integrierten Operationssälen und Insellösungen, zeichnen sich Netzwerke aus *Eigenherstellung* dadurch aus, dass sie nicht als solche durch einen Hersteller in Verkehr gebracht, sondern vor Ort in der Gesundheitseinrichtung zusammengestellt werden (siehe auch Kapitel 2.1.2). So entstehende Netzwerke werden in der Fachliteratur als *medizinische IT-Netzwerke* (MIT-Netzwerke) bezeichnet. Sie können zu einer Vielzahl von Anwendungsfällen benutzt werden.

Die Autoren in (Strauss und Schmitz, 2016) beschreiben beispielsweise, wie man intraoperativ erzeugte Daten zu Dokumentationszwecken weiterverwenden kann. Demnach kann etwa laufend die Position des Instruments, die Leistung des Instruments (z.B. bipolare HF-Pinzette) und/oder die Drehzahl des Instruments (z.B. Shaver) bestimmt werden. Dadurch lassen sich Logfiles der benutzen Medizingeräte (etwa Leistung des HF-Geräts in der Nähe zur Dura der vorderen Schädelgrube, Umdrehungsfrequenz des Bohrers bei Passieren des Fazialiskanals im Felsenbein, Position des Saugers bei Eintreten in die Stirnhöhle) anfertigen und dauerhaft archivieren – für die Rekonstruktion der Operation, interne Auswertungen, Ausbildung oder auch als Beweis in haftungsrechtlichen Verfahren.

Die Autoren in (Stoecklein et al., 2015) beschreiben eine Studie in der Neurochirurgie, in der sie IONM-Daten (EMG-Kurven, somatosensorisch-evozierte Potenziale, motorisch-evozierte Potenziale und akustisch-evozierte Potenziale) in ein OP-Mikroskop eingeblendet haben. Dadurch sollten Kommunikationsfehler zwischen dem Chirurgen, der in das Mikroskop sieht, und dem Neurophysiologen, der die Kurven des IONM auswertet und kommuniziert, umgangen werden. Die Autoren konnten zeigen, dass die IONM-Daten ohne Beeinträchtigung des Sichtfeldes des Chirurgen in das Mikroskop eingeblendet werden konnten und dass das so entstehende System sicher und zuverlässig ist. Aus Sicht der Autoren liefert das neue System dem behandelnden Chirurgen wertvolle Zusatzinformationen und verbessert den Workflow.

Wie im Stand der Technik (Kapitel 2) deutlich werden wird, obliegt die Risiko- und Konformitätsbewertung derartiger Netzwerke dem *Betreiber* der Gesundheitseinrichtung. Dieser Aufgabe widmet sich diese Arbeit.

1.2. Risiken durch Vernetzung

Den Vorzügen der Vernetzung von Medizingeräten stehen jedoch neuartige Risiken gegenüber, die besonders im Bereich der Schnittstellen, sowie der Wechselwirkung von Mensch und Technik auftreten.

In den letzten zehn Jahren haben sich die Haftungsfälle im Bereich der Krankenhäuser mehr als verdoppelt und die Schadensummen deutlich erhöht. Auch die Anzahl der Vorkommnismeldungen ist in den letzten Jahren spürbar angestiegen (siehe Abb. 6). Von über 8000 Risikomeldungen im Jahr 2013 entfiel fast die Hälfte auf aktive Medizinprodukte, wobei von einer höheren Dunkelziffer auszugehen ist (Ahlbrandt et al., 2013; Langkafel, 2015; Kindler und Menge, 2017). 17 % der Ereignisse im Zeitraum von 2005 bis 2013 waren auf Design- oder Konstruktionsfehler zurückzuführen, wobei fehlerhafte Software und unzureichende Funktionalität zu den Hauptursachen gehörten (Schmola, 2016b; Kindler und Menge, 2017).

Nach Ansicht der Autoren in (Ahlbrandt et al., 2013) ist von einem relevanten Anteil an Problemen auszugehen, die durch Vernetzung von Medizinprodukten aufgetreten sind. Valide Zahlen, wie viele Fehler durch IT reduziert wurden, werden können oder hinzukommen, existieren aber nicht – die Hauptgründe dafür sind fehlender Datenaustausch, fehlender Fokus einschlägiger Studien, methodische Probleme und die allgemeine Fehlerkultur in Kliniken (Langkafel, 2015).
Der Autor in (Langkafel, 2015) nennt folgende häufige IT-Fehlerquellen:

- Mangelhafte Programmspezifikation. Die Software funktioniert zwar richtig, erfüllt aber nicht die an sie gestellten Erwartungen.
- Die Funktionalität ist nicht ausreichend und/oder entspricht nicht den Nutzererwartungen.
- Die von einem System bereitgestellten Informationen reichen nicht aus, um eine Handlung daraus abzuleiten.
- Fehlende oder unzureichende Systemintegration.
- Datenüberschuss: Die Menge an Daten blockiert eine Prozedur oder führt zu nicht akzeptablen Wartezeiten.

Diese und weitere Herausforderungen sollen in dieser Arbeit adressiert werden.

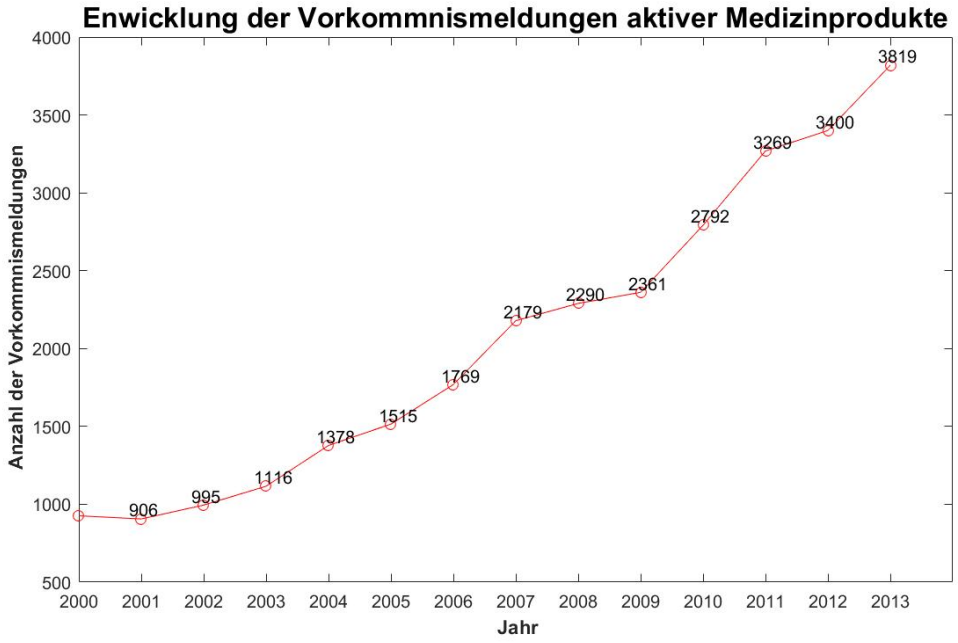


Abbildung 6 Entwicklung der Vorkommnismeldungen beim Bundesinstitut für Arzneimittel und Medizinprodukte *(BfArM)*. Daten übernommen aus (Kindler und Menge, 2017), da Originaldaten vom BfArM nicht mehr online verfügbar.

1.3. Zielsetzung der Arbeit

In dieser Arbeit sollen neue Methoden konzipiert und implementiert werden, die Betreiber von Gesundheitseinrichtungen bei der Konformitätsbewertung medizinischer IT-Netzwerke (siehe Abschnitt 1.1.5) unterstützen. Im Besonderen sollen die Methoden Betreibern ermöglichen, für ein gegebenes *Medizinisches IT-Netzwerk* (MIT-Netzwerk) zügig und korrekt bewerten zu können, ob Anforderungen beteiligter Hersteller, sowie eingriffsspezifische Anforderungen der *Anwender* hinsichtlich dessen Funktionsweise erfüllt werden können.
Dabei ist zu berücksichtigen, dass Betreiber nur eingeschränkte Kenntnis über die Funktionsweise einzelner *Module* (Medizingeräte und Zubehör) haben und Hersteller nicht vorhersehen können, mit welchen weiteren Modulen ihre Geräte verwendet werden. Der Betreiber soll dabei nur die *Verschaltung* der Module, d.h. die Verknüpfung von Datenquellen und -senken vorgeben. Darauf basierend soll automatisch ein Modell des MIT-Netzwerks erzeugt werden, welches das Verhalten desselben simuliert und gegebene Anforderungen automatisiert verifiziert.
Die Methode soll am Beispiel des OR.Net *Echtzeit-Bussystems* exemplarisch implementiert werden. Dieses Bussystem wurde im Rahmen des OR.Net Projekts an der TU München entwickelt und ermöglicht die herstellerübergreifende, echtzeitfähige Kommunikation von Medizingeräten und Zubehör (siehe Kapitel 4).
Bei der Implementierung soll auf die Besonderheiten des OR.Net Echtzeit-Bussystems eingegangen und im Besonderen ein skalierbares Modell für die Steuerung (*Managing Node*) des Netzwerks erarbeitet werden.

2. Stand der Technik

Die Voraussetzungen für den freien Warenverkehr von *Medizinprodukte*n in der EU werden auf europäischer Ebene durch die MDD (und ab 26. Mai 2020 verbindlich durch die MDR) festgelegt und im Zuge der Umsetzung europäischer Rechtsvorschriften in Länderrecht in das deutsche MPG übernommen.
Das MPG adressiert den Verkehr von Medizinprodukten und hat deren Sicherheit, Eignung und Leistung, sowie die Gesundheit und den Schutz von Patienten, Anwendern und Dritten zum Inhalt. Neben technischen und medizinischen Anforderungen enthält es auch Informationsanforderungen und Vorschriften für Betreiber und Anwender von Medizinprodukten. Die Erfüllung aller Sicherheitsanforderungen durch ein *Medizinprodukt* wird im Rahmen einer Konformitätsbewertung geprüft.
Da die MDR erst seit Mai 2017 in Kraft getreten ist, wird fortan die noch aktuelle MDD zitiert. Die dieser Arbeit zugrunde gelegten Anforderungen finden sich jedoch in ähnlichem Wortlaut auch in der MDR.

2.1. Konformitätsbewertung vernetzter Medizinprodukte

Bevor in Deutschland Medizinprodukte (insbesondere: miteinander verbundene Medizingeräte) am Menschen angewendet werden dürfen, müssen diese eines von vielen möglichen Konformitätsbewertungsverfahren durchlaufen. Ziel desselben ist stets, die Erfüllung der *Grundlegenden Anforderungen* an Medizinprodukte aus MDD Anhang I sicherzustellen. Diese richten sich vorrangig an die Sicherheit von Patienten und Anwendern, wie durch die einleitende Anforderung verdeutlicht wird – demnach müssen Medizinprodukte so ausgelegt und gefertigt sein, dass ihre Anwendung weder die Patientensicherheit, noch deren klinischen Zustand oder die Sicherheit und Gesundheit von Anwendern oder Dritten gefährdet (direkt oder indirekt), wenn sie unter den vorgesehenen Bedingungen für den vorgesehenen Verwendungszweck eingesetzt werden (MDD, Anhang I, Abs. 1).
Bei vernetzten Medizingeräten ergeben sich dabei je nach Ausprägung Besonderheiten: Man unterscheidet zwischen *monolithischen* Systemen und *offenen* Systemen.

2.1.1. Konformitätsbewertung monolithischer Systeme

Monolithische Systeme zeichnen sich dadurch aus, dass ihre Bestandteile vor dem *Inverkehrbringen* durch einen Hersteller vollständig spezifiziert sind. Dies können *Integrierte Operationssäle* größerer Hersteller (siehe Abschnitt 1.1.3) sein, oder aber auch herstellergebundene Insellösungen (siehe Abschnitt 1.1.4).
Die Abgrenzung ist aus einer rechtlichen Besonderheit erwachsen: Verlangt die Zweckbestimmung eines Medizinprodukts die Vernetzung mit weiteren Komponenten, so muss der Hersteller des Geräts nach MDD Anhang I Abs. 13.6 c) alle Merkmale der Komponenten benennen, die für eine sichere Kombination erforderlich sind. Meist begegnen Hersteller dieser Anforderung durch Angabe einer konkreten Liste kompatibler, oft eigener Komponenten in

der Zweckbestimmung des Medizingeräts. Für die Konformitätsbewertung eines monolithischen Systems ist sein Hersteller verantwortlich. Sie wird dadurch erleichtert oder gar erst ermöglicht, dass der Hersteller des Systems vor Inverkehrbringen vollständige Kenntnis über alle Komponenten des Systems hat.

2.1.2. Konformitätsbewertung offener Systeme

Anders verhält es sich, wenn Medizingeräte *über deren Zweckbestimmung hinaus* miteinander vernetzt werden. Systeme derart vernetzter Medizingeräte bezeichnet man als *offen* (auch: MIT-Netzwerk, siehe (DIN EN 80001-1)). Im Gegensatz zu monolithischen Systemen werden offene Systeme nicht als solche durch einen Hersteller in Verkehr gebracht; vielmehr entstehen sie gewissermaßen *vor Ort* in der Gesundheitseinrichtung (siehe auch Abschnitt 1.1.5). Installation und Betrieb offener Systeme sind wiederum streng reguliert: Die MPBetreibV erlaubt *Betreibern* von Gesundheitseinrichtungen in §2 Abs. 1 und 3 grundsätzlich nur die Anwendung von Medizinprodukten im Rahmen ihrer Zweckbestimmung (MPBetreibV).

Nutzt ein Betreiber Medizingeräte und weitere Komponenten *außerhalb* deren Zweckbestimmung (z.B. durch unzulässige Vernetzung von Geräten und Zubehör), so ist er aus Sicht des §3 Abs. 21 MPG *Eigenhersteller* des so entstandenen Systems und daher nach §10 Abs. 2 MPG sowie §12 Abs. 1 MPG selbst zuständig für die Konformitätsbewertung (MDD; MPV; MPG). Der Verstoß kann mit einer Geldbuße von 25.000 € geahndet werden (§42 Abs. 2 Nr. 16 MPG), wobei Ansprüche auf Schadensersatz und/oder Schmerzensgeld davon unberührt sind. Die Haftung fällt erst dann wieder auf den Hersteller einer einzelnen Komponente zurück, wenn nachgewiesen werden kann, dass der den Schaden verursachende Fehler ausschließlich auf diese Komponente zurückgeht und keinen Zusammenhang zum kombinierten Produkt hat (Weimer, 2010).

Im Gegensatz zu monolithischen Systemen liegt also die Verantwortung für die Konformitätsbewertung offener Systeme beim *Betreiber* und nicht beim *Hersteller* einzelner Teilsysteme.

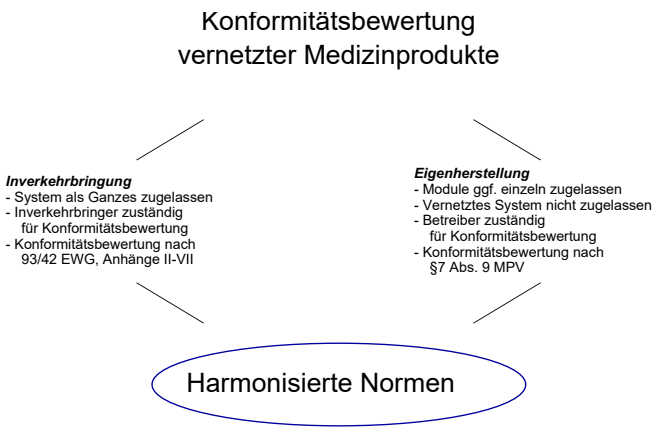


Abbildung 7 Verschiedene Konformitätsbewertungsverfahren für Inverkehrbringer und Eigenhersteller.

2.2. Verfahren zur konformen Produktrealisierung

In Abschnitt 2.1 wurde deutlich, dass bei monolithischen Systemen der Hersteller, bei MIT-Netzwerken aber der Betreiber für die Konformitätsbewertung zuständig ist. Im Folgenden wird gezeigt, wie dieser Pflicht in den einzelnen Entwicklungsphasen nachgekommen wird.
Um die Konformität mit den Grundlegenden Anforderungen sicherzustellen, gibt es eine Reihe an unterstützenden *Harmonisierten Normen*, welche konkrete Anforderungen an alle Phasen der Produktrealisierung und des begleitenden *Risikomanagements* beschreiben.
So schreibt die *DIN EN ISO 13485* in Kapitel 7 speziell vor, dass der Hersteller von Beginn der Entwicklung an die Qualitätsziele und Anforderungen an das Produkt festlegen und die Einhaltung dieser Anforderungen in geeigneten Abständen verifizieren muss. Die Anforderungen können die folgende Natur haben (siehe DIN EN ISO 13485 7.2.1, 7.3.2 und Abb. 8):

- Kundenanforderungen
- Anforderungen der Herstellerorganisation
- Sicherheitsanforderungen
- Weitere normative Anforderungen
- Gesetzliche und regulatorische Anforderungen
- Anforderungen aus Erfahrungswissen
- Andere für Funktion, Design und Entwicklung wesentliche Anforderungen
- Ergebnisse aus dem Risikomanagement

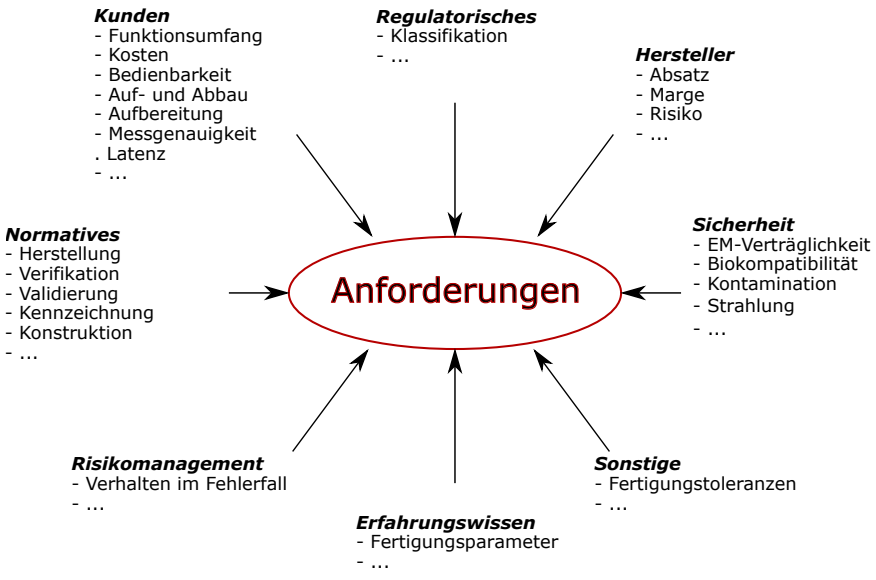


Abbildung 8 Beispielhafte Anforderungen und ihre Quellen.

Keineswegs wird verlangt, dass die Anforderungen an das Produkt von Beginn der Entwicklung an *vollständig* spezifiziert sind. Vielmehr werden im Laufe der Entwicklung ständig neue Erkenntnisse gewonnen, die als Eingabe in das *Risikomanagement* fließen und Entwicklungsänderungen bewirken.
Die einzelnen Phasen der Produktrealisierung können nach (Nedetzky und Müllner, 2013) verkürzt wie folgt beschrieben werden:

Projektstart In dieser Vorklärungsphase sollen die oft nur grob definierten Zielsetzungen und Anforderungen detailliert und die technische Machbarkeit erstmals bewertet werden. Wichtig ist in dieser Phase, Annahmekriterien für das Produkt, sowie Verifizierungs- und Validierungstätigkeiten zu definieren, die als Nachweis der Erfüllung ebendieser Kriterien verwendet werden können (DIN EN ISO 13485, 7.1).

Design Input In dieser Phase muss bereits genau spezifiziert sein, was zu entwickeln ist, wie das Produkt realisiert wird und (im Falle eines *Inverkehrbringers*) wann das Produkt für die Vermarktung bereitsteht. Die Annahmekriterien des Produkts sind durch messbare Parameter und Akzeptanzkriterien beschrieben und beinhalten sämtliche Anforderungen, die zum Projektstart für gültig befunden wurden. Es dürfen weiterhin Änderungen an Anforderungen, Design und Entwicklung, sowie Verifizierungs- und Validierungsplänen vorgenommen werden, allerdings müssen diese *gelenkt* sein, d.h. sie müssen bewertet, genehmigt, verifiziert und validiert werden ((DIN EN ISO 13485) 7.3.6,7.3.7).

Design Output In dieser Phase werden die vom tatsächlich implementierten Produkt erreichten Ist-Spezifikationen mit den im Design Input vorgegebenen Soll-Spezifikationen abgeglichen. Die hierbei zu erstellenden Dokumente sind vorrangig Protokolle über die durchgeführten Prüfungen des Produkts über die Annahmekriterien. Abweichungen werden bewertet und gegebenenfalls im Rahmen einer Entwicklungsänderung ausgeräumt.

Produktfreigabe Ist die Verifikation in der Design Output Phase abgeschlossen, d.h. die Soll-Spezifikationen sind in ausreichendem Maße erfüllt, so erfolgt die Validierung des Produkts. Ziel derselben ist es, das Produkt auf seine Eignung unter realen Einsatzbedingungen zu überprüfen. Bei erfolgreicher Validierung kann das Produkt für die Vermarktung freigegeben werden.

2.3. Entwicklungsbegleitende Verifikation und Risikomanagement

Wie in Abschnitt 2.2 beschrieben, ist die Verifikation der Anforderungen im Stand der Technik keineswegs ein *einmaliger* Vorgang. Vielmehr muss laufend in geeigneten Phasen der Entwicklung die Eignung der Entwicklungsergebnisse für die an sie gestellten Anforderungen verifiziert werden. Die U.S. Food and Drug Administration *(FDA)* beschreibt dies in ihrem *Wasserfallmodell* (siehe Abb. 9). Demnach sind für jede Entwicklungsphase geeignete Prüf- und Messmethoden festzulegen, mit deren Hilfe die Einhaltung der Anforderungen verifiziert werden kann.
Eng verknüpft mit der Konformitätsbewertung ist das Risikomanagement, wie aus dem zweiten Satz der Grundlegenden Anforderungen deutlich wird: Etwaige Risiken, die aus der Anwendung von Medizingeräten erwachsen, müssen verglichen mit deren nützlicher Wirkung für den Patienten vertretbar und in hohem Maße mit dem Schutz der Gesundheit und Sicherheit vereinbar sein (MDD, Anhang I Kap. 1 Abs. 1). In der MDR ist das Risikomanagement selbst als Grundlegende Anforderung aufgenommen (MDR, Anhang I, Kap. 1, Abs. 3).
Das Risikomanagement ist also ein verpflichtender Bestandteil der Konformitätsbewertung und fließt über die DIN EN ISO 13485 direkt in die Produktrealisierung ein:

> *„Die Organisation muss dokumentierte Anforderungen für das Risikomanagement während der gesamten Produktrealisierung erarbeiten. Es müssen Aufzeichnungen geführt werden, die sich aus dem Risikomanagement ergeben [...]“ (DIN EN ISO 13485, 7.1).*

Das Risikomanagement zieht sich so durch alle Phasen der Produktentwicklung, wobei einige Risiken direkt durch normative Anforderungen abgedeckt werden. Wann immer einschlä-

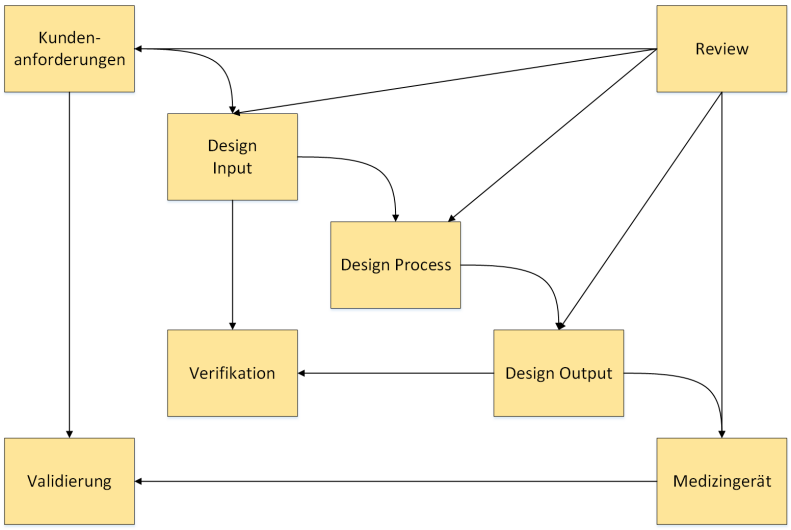


Abbildung 9 Wasserfall-Modell der FDA. Jede Entwicklungsphase von Anforderungsanalyse bis Produktfreigabe muss einem Review unterzogen werden, in dem die Ergebnisse bewertet und ggf. mit vorhandenen Anforderungen abgeglichen werden. Angelehnt an (Nedetzky und Müllner, 2013).

gige Normen verwendet werden, um Kriterien für die Auslegung des Medizinprodukts zu finden, wird dies von der *DIN EN ISO 14971* als Aktivität zur Risikobeherrschung angesehen (DIN EN ISO 14971, 5, Anm. 2).
Bevor im Folgenden das Risikomanagement von Medizinprodukten näher beschrieben wird, sei der Zusammenhang zum Anforderungskatalog und dessen Verifikation angeführt:
Sämtliche Risiken in Bezug auf ein Medizinprodukt sind durch den Hersteller zu erfassen und als Anforderungen in die Design- und Entwicklungsvorgaben aufzunehmen (DIN EN ISO 13485, 7.3.2. e)). Sind weiter zur Minderung von Risiken Maßnahmen erforderlich, so müssen diese implementiert und deren Wirksamkeit verifiziert werden (DIN EN ISO 14971, 3.5).
Auf diese Weise entstehen durch das Risikomanagement laufend neue Anforderungen, deren Erfüllung durch das Produkt verifiziert werden muss.
Zusammenfassend wird deutlich, dass der Entwicklungsprozess bei Medizinprodukten stark auf dokumentierten Anforderungen basiert. Diese werden vor Beginn des Produktdesigns angelegt, im Zuge der Entwicklung vervollständigt und laufend durch das Risikomanagement angereichert. Die hierzu gängigen Methoden werden im folgenden Abschnitt erläutert.

2.4. Risikomanagement für Hersteller – Verfahren und Standards

Der Begriff Risikomanagement beschreibt laut DIN EN ISO 14971, Kapitel 2, Abs. 2.22 die „systematische Anwendung von Managementstrategien, Verfahren und Praktiken auf die Aufgaben der Analyse, Bewertung, Beherrschung und Überwachung von Risiken“.

2.4.1. Risikoanalyse

Für die Analyse von Risiken stehen vielfältige Methoden zur Verfügung. Qualitative Methoden der Risikoanalyse zielen in der Regel darauf ab, Abläufe, die zu einem Unfall führen können, zu identifizieren. Ausgehend vom Gesamtsystem wird versucht, Gefahren und ihre Ursachen zu erkennen, um diese zu reduzieren oder auszuschließen. Quantitative Methoden hingegen analysieren mittels statistischer Untersuchungen Häufigkeit und Auswirkungen bestimmter Ereignisse (Büchel et al., 2014).
Im Folgenden werden die gängigsten Risikoanalysetechniken nach DIN EN ISO 14971 vorgestellt. Die Beschreibungen sind sinngemäß aus (Ericson, 2016) übernommen.

Preliminary Hazard Analysis Die *Preliminary Hazard Analysis* (PHA) ist eine Analysetechnik, die in den frühen Designphasen für Systeme angewendet wird, über die noch wenige Details bekannt sind. Die PHA leitet aus einer gegebenen Liste an Gefährdungen neue Gefährdungssituationen und deren Ursachen sowie Gegenmaßnahmen ab.

Fault Tree Analysis Die *Fault Tree Analysis* (FTA) ist eine Analysetechnik, die auf die Auffindung von Ursachen und die Bewertung der Eintrittswahrscheinlichkeit unerwünschter Ereignisse abzielt. Sie wird in jeder Phase der Entwicklung und des Lebenszyklus von Systemen verwendet. Bei der FTA wird eine gegebene Gefährdung betrachtet und dann im sogenannten *fault tree* graphisch sämtliche Pfade aufgezeichnet, die zum gegebenen Ereignis führen können. Die FTA setzt Expertenkenntnisse voraus.

Failure Mode and Effects Analysis Die *Failure Mode and Effects Analysis* (FMEA) ist eine Analysetechnik, welche die Auswirkungen von Fehlern einzelner Systembestandteile untersucht. Sie kann während der gesamten Designphase eingesetzt werden und erfordert hohe Expertenkenntnis.

2.4.2. Risikobewertung

Die Risikobewertung hat zum Ziel, die Eintrittswahrscheinlichkeit eines Fehlers und die zu erwartenden Folgen zu quantifizieren. Die DIN EN ISO 14971 macht hierzu in Anhang D konkrete Vorschläge: Die Eintrittswahrscheinlichkeit kann demnach in die Kategorien *häufig* ($\geq 10^{-1}$% der Fälle), *wahrscheinlich* ($\geq 10^{-2}$% der Fälle), *gelegentlich* ($\geq 10^{-3}$% der Fälle),

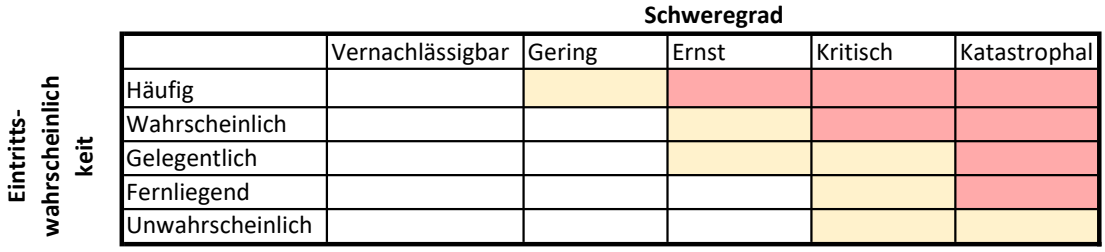


Abbildung 10 Bewertung von Risiken nach DIN EN ISO 14971. Die Spalten der Tabelle stehen für die zu erwartenden Folgen, die Zeilen für die Eintrittswahrscheinlichkeit eines Risikos. Beispielhaft eingefärbt sind inakzeptable Risiken in Rot, akzeptable Risiken in Weiß und Risiken, die nach Möglichkeit reduziert werden sollen, in Gelb.
Angelehnt an DIN EN ISO 14971, Anhang D.

fernliegend ($\geq 10^{-4}\%$ der Fälle) und *Unwahrscheinlich* (weniger als $10^{-4}\%$ der Fälle) eingeteilt werden. Die zu erwartenden Folgen hingegen werden eingeteilt in die Klassen *katastrophal* (Tod des Patienten), *kritisch* (Dauerhafte Behinderung oder lebensbedrohliche Schädigung), *ernst* (Schädigung oder Behinderung, die medizinischen Eingriff erfordert), *gering* (zeitweilige Schädigung oder Behinderung, die keinen medizinischen Eingriff erfordert) und *vernachlässigbar* (Unannehmlichkeiten oder zeitweilige Beschwerden). Pro Risiko wird unter Verwendung obiger Begriffe eine Bewertung durchgeführt und in einer Tabelle wie in Abb. 10 dokumentiert. Die eigentliche Risikobewertung, d.h. die Entscheidung, welche Risiken als vertretbar oder unvertretbar bewertet werden, obliegt dem Hersteller und ist nicht normiert. Eine Möglichkeit besteht darin, obige Tabelle (Abb. 10) in vertretbare Bereiche (links-unten) und unvertretbare Bereiche (rechts-oben) einzuteilen. In letzterem Falle sind Methoden zur Risikominderung erforderlich.

2.4.3. Risikobeherrschung

Die Möglichkeiten der Risikobeherrschung fallen nach DIN EN ISO 14971 in die Kategorien

- Integrierte Sicherheit durch Design,
- Schutzmaßnahmen im Medizinprodukt oder im Herstellungsprozess,
- Informationen zur Sicherheit,

wobei der Standard verlangt, dass Hersteller diese Optionen in der vorgegebenen Reihenfolge prüfen. Ziel der genannten Maßnahmen ist es, Schweregrad oder Eintrittswahrscheinlichkeit eines Risikos (oder beides) zu reduzieren. Die Umsetzung der Maßnahmen, sowie deren Wirksamkeit, muss vom Hersteller verifiziert werden. Nach Durchführung aller risikomindernden Maßnahmen wird das Restrisiko durch den Hersteller erneut bewertet und zudem untersucht, ob durch die Einführung der Maßnahmen neuartige Risiken entstanden sind.

2.4.4. Risikoüberwachung

Nach Abschluss aller risikomindernden Maßnahmen prüft der Hersteller die Akzeptanz des Gesamt-Restrisikos. Keineswegs ist jedoch dadurch der Prozess des Risikomanagements beendet – vielmehr beginnt hier die Phase der *Überwachung*, in der Daten aus der Herstellung und nachgelagerten Phasen ausgewertet und auf deren Sicherheitsrelevanz bewertet werden. Hierzu gehören nach DIN EN ISO 14971, Anhang F.7 und DIN EN ISO 13485, Kapitel 7 insbesondere

- neue oder überarbeitete Normen und gesetzliche Auflagen,
- Kundenrückmeldungen und Beschwerden zum Medizinprodukt oder vergleichbaren Produkten mit Sicherheitsbezug,
- Zwischenfälle mit dem Medizinprodukt oder vergleichbaren Produkten und
- klinische Studien.

Der Risikomanagement-Prozess bei *Herstellern* erstreckt sich so über den gesamten Produktlebenszyklus. Der Risikomanagement-Prozess ist in Abb. 11 abschließend dargestellt.

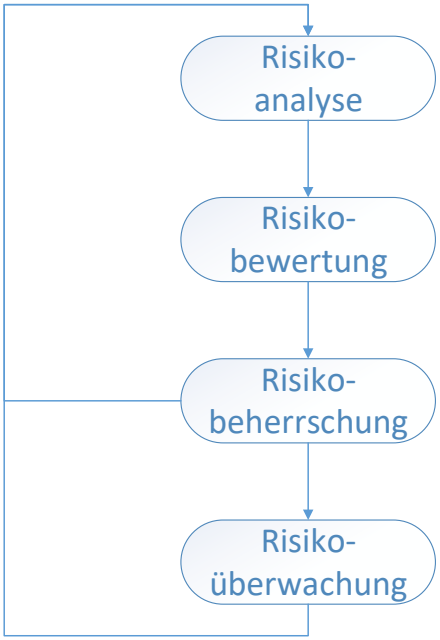


Abbildung 11 Risikomanagement-Prozess mit den Phasen Analyse - Bewertung - Beherrschung - Überwachung. Angelehnt an DIN EN ISO 14971.

2.5. Risikomanagement in Gesundheitseinrichtungen – Verfahren und Standards

Übergeordnetes Ziel des Risikomanagements in Gesundheitseinrichtungen ist es, Schaden von Patienten und Anwendern abzuwenden und eine sichere Versorgung zu gewährleisten. Dabei müssen insbesondere medizinisch-technische Geräte auf potenzielle Risiken geprüft werden (Schmola, 2016a).

2.5.1. Risikomanagement für kommerzielle Systeme

Für Systeme mit bestehender Zulassung beschränkt sich die Risikomanagement-Aktivität der Gesundheitseinrichtung auf die Erfüllung ihrer wesentlichen Pflichten aus der MPBetreibV. Diese sind (Kruber, 2015a)

- die Organisation der Instandhaltung aktiver Medizinprodukte und der Betrieb einer zentralen Reparaturannahme für Medizinprodukte (§4 MPBetreibV)
- die Pflege eines Medizinproduktebuchs (§7 MPBetreibV) und eines Bestandsverzeichnisses aller aktiven Medizinprodukte (§8 MPBetreibV)
- die Terminierung und Organisation der verpflichtenden Sicherheitstechnischen Kontrollen (§6 MPBetreibV) und messtechnischen Kontrollen (§11 MPBetreibV)
- die Überwachung und Dokumentation der Funktionsprüfung am Einsatzort bei Erstinbetriebnahme durch den Hersteller (oder eine befugte Person (§5 MPBetreibV, Abs. 3)) von Medizinprodukten aus MPBetreibV Anlage 1 (siehe §5 Abs. 1 MPBetreibV)
- die Sicherstellung, dass nur Medizinprodukte mit CE-Kennzeichnung in Betrieb genommen werden (ausgenommen Sonderanfertigungen, Medizinprodukte aus Eigenherstellung, Medizinprodukte mit Ausnahmeregelung durch das BfArM (§11 Abs. 1 MPG) sowie solche zur klinischen Prüfung).

Anwender haben gemäß §2 Abs. 5 MPBetreibV vor der Anwendung eines *Medizinprodukts* dessen Funktionsfähigkeit und ordnungsgemäßen Zustand zu prüfen. Untersuchungsgegenstände sind hierbei Sichtprüfungen, Prüfung auf ordnungsgemäßen Zusammenbau, Sauberkeit, Freiheit von äußeren Schäden und auch ordnungsgemäße Geräte-Selbsttests. Darüber hinaus hat der Anwender die Gebrauchsanweisung zu beachten. Diese Pflichten gelten auch für mit den Medizingeräten verbundenes Zubehör (Jäkel, 2016).

2.5.2. Risikomanagement für medizinische IT-Netzwerke

Wiederum anders verhält sich die Situation im Umfeld des Risikomanagements von MIT-Netzwerken – hierbei nimmt der Standard *DIN EN 80001-1*, der die Verantwortlichkeiten und Rollen für den Risikomanagement-Prozess von MIT-Netzwerken definiert, eine zentrale Rolle ein. Der Standard ist jedoch nicht harmonisiert und kann daher nicht für den Konformitätsnachweis mit der MDD herangezogen werden.

Ziel standardkonformer Risikomanagement-Aktivitäten ist der Betrieb sicherer und effektiver MIT-Netzwerke in Kliniken.
Der Risikomanagement-Prozess erstreckt sich laut Standard über alle Phasen von Planung und Dokumentation der Risikomanagement-Aktivitäten, über Analyse, Bewertung und Beherrschung von Risiken bis hin zum Management von Änderungen und Konfigurationen, Überwachung im Betrieb und Meldung von unerwünschten Ereignissen.
Diese Aktivitäten betreffen dabei sowohl Planung und Design des Netzwerks, als auch Inbetriebnahme, Einbindung von Geräten, Änderungen an Geräten oder Netzwerk, bis hin zu Betrieb und Außerbetriebnahme einzelner Geräte oder des Netzwerks.
Der Projektplan eines Vernetzungs-Projekts enthält unter anderem eine exakte Spezifikation des Netzwerks und seiner Konfiguration, welche Anforderungen an das Netzwerk existieren, wie diese erfüllt werden sollen und wie die Erfüllung verifiziert werden soll.
Für die Risikoanalyse sind insbesondere Hersteller-Angaben erforderlich, die Eigenschaften und Anforderungen des Medizinprodukts beschreiben (DIN EN 80001-1).
Die Verantwortlichkeiten und deren Wechselwirkungen innerhalb der Gesundheitseinrichtung sind in Abb. 12 dargestellt. Eine zentrale Rolle nimmt demnach der sogenannte *Risikomanager* ein, der für das Risikomanagement verantwortlich ist. Er wird von der Leitung benannt und steht in ständigem Austausch mit den klinischen Abteilungen, der IT-Abteilung, der Medizintechnik und den Herstellern (DIN EN 80001-1; Ahlbrandt et al., 2013).

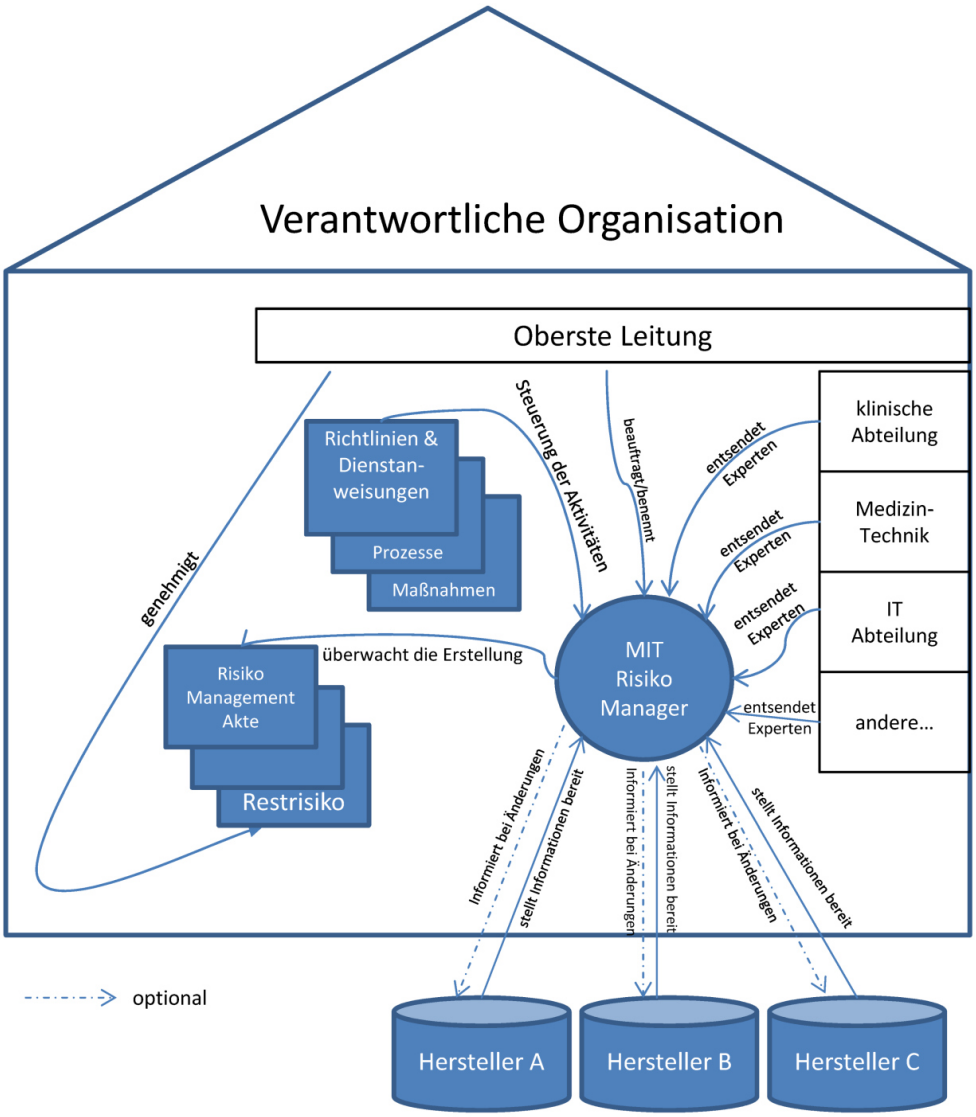


Abbildung 12 Rollenverteilung für das Risikomanagement nach IEC 80001-1. Eine zentrale Rolle nimmt der MIT-Risikomanager ein, der von der obersten Leitung bestellt wird. Von Seiten der Betreiberorganisation nehmen sowohl Anwender (klinische Abteilung) und Medizintechniker, als auch IT-Spezialisten am Risikomanagement-Prozess teil. Der Risikomanager erhält die technischen Informationen, die für das Risikomanagement erforderlich sind, vom Hersteller. Unverändert entnommen aus (Ahlbrandt et al., 2013), dort angelehnt an DIN EN 80001-1. Veröffentlicht unter der CC-Lizenz creativecommons.org/licenses/by-nc-nd/3.0/.

2.6. Kommerzielle Produkte zur Unterstützung der Konformitätsbewertung

Um Hersteller bei der Erfüllung der einschlägigen Standards zum Risikomanagement zu unterstützen, gibt es eine Reihe kommerzieller Tools und Dienstleistungen.

2.6.1. Software-Tools zur Unterstützung des Risikomanagements für Hersteller von Medizinprodukten

Neben Standard-Tools wie *Microsoft Excel®* gibt es spezialisierte Software für die Erfüllung einschlägiger Standards zum Risikomanagement. Während der genaue Funktionsumfang in der wissenschaftlichen Literatur nicht beschrieben ist, sind in der nachfolgenden Tabelle 1 Angaben der Anbieter aufgeführt. Auffällig ist, dass alle bekannten Tools an die DIN EN ISO 14971 angelehnt sind und die Umsetzung der darin vorgeschlagenen Analysetechniken (FMEA, FTA, PHA) unterstützen. Alle Tools richten sich primär an Inverkehrbringer von Medizingeräten.

2.6.2. Dienstleistungen für die Konformitätsbewertung von medizinischen IT-Netzwerken für Betreiber

Der Markt für die Konformitätsbewertung von MIT-Netzwerken besteht überwiegend aus Beratungsleistungen (vgl. (Lüth et al., 2016)). Um bei der Erfüllung regulatorischer Anforderungen bei Eigenherstellung von MIT-Netzwerken zu unterstützen, bietet etwa der TÜV einen sogenannten *SystemCheck* für Medizinprodukte aus Eigenherstellung an, dessen genauer Leistungsumfang in der wissenschaftlichen Literatur jedoch nicht beschrieben ist. Nach Angaben des TÜV analysiert der Prüfer die technischen, betrieblichen und organisatorischen Risiken einer freien Gerätekombination, um dem Betreiber zu ermöglichen, eine Konformitätserklärung in Anlehnung an Anhang VIII der MDD auszustellen.
Die Telekom Healthcare Solutions bietet Beratungsdienstleistungen an, in denen die Implementierung der DIN EN 80001-1 für eine Gesundheitseinrichtung diskutiert wird und mögliche Nutzen/Risiken identifiziert werden.
Ähnliche Leistungen bietet die Firma medfacilities: Sie übernimmt die Beratung zur Implementierung der DIN EN 80001-1 im Krankenhaus. Für die Uniklinik Köln hat die Firma die Prozesse für das Risikomanagement nach DIN EN 80001-1 exemplarisch implementiert (Schömig und Heinen, 2014).

Anbieter	Produkt	Adressierte Standards	Funktionen
AQA Company	IMSXpress	ISO 14971	FMEA Risikomanagement-Bericht Maßnahmen-Verwaltung
Bayoomed	Qware® Riskmanager	ISO 14971 ISO 62366 ISO 60601-1-6 ISO 60601-3 DIN EN 80001-1	PHA FMEA Maßnahmen-Verwaltung Wissensverwaltung Bug-Tracking
CAQ	FMEA.Net	ISO 14971 ISO/TS16949	Wissensverwaltung
greenlight.guru	RM Software	ISO 14971	FMEA Risikomanagement-Bericht Maßnahmen-Verwaltung
Orcanos	QPack FMEA Risk Management Tool	ISO 14971	FMEA Bug Tracking Maßnahmen-Verwaltung Risikomanagement-Bericht
SurgiTAIX	Carad	ISO 14971	FMEA FTA

Tabelle 1 Übersicht kommerzieller Tools für das Risikomanagement (alphabetisch sortiert, da Marktanteile unbekannt). Diese Tools unterstützen Hersteller dabei, das vorhandene Wissen über das eigene System entlang jeder Phase des Risikomanagements normkonform zu dokumentieren. Dazu werden unter anderem die in der DIN EN ISO 14971 vorgeschlagenen Analysetechniken implementiert.
Angelehnt an (Lüth et al., 2016).

2.7. Stand der Forschung

Ziel aktueller Forschungsarbeiten im Bereich der Interoperabilität von Medizingeräten ist die Bereitstellung von Rahmenbedingungen, in denen Anwender Medizingeräte und Zubehör frei miteinander vernetzen und anwenden können. Verglichen mit integrierten Operationssälen besteht die Besonderheit darin, dass Hersteller einzelner Komponenten vor deren Inbetriebnahme keine Kenntnis haben über die Zusammensetzung des Netzwerks, in das sie eingebunden werden. Die Forschung wird durch zwei große Initiativen maßgeblich vertreten: Dem deutschen BMBF-Projekt OR.Net und dem amerikanischen Medical Device Plug and Play (MDPnP) Projekt (Leitung: Julian Goldman, Harvard Medical School).

2.7.1. Beschreibungsmethoden zum Zwecke der Interoperabilität

Verbindender Gedanke beider Initiativen ist die Realisierung Service Orientierter Architekturen in der Medizintechnik. Dabei beschreiben Geräte und Systeme ihre Funktionalität in maschinenlesbarer Form (als *Services*), um beliebigen weiteren Systemen im Netzwerk das Auffinden und Ausüben der Funktionalität zu ermöglichen (Kasparick et al., 2015b).
Besonders weite Verbreitung haben dabei die Beschreibungsstandards der IEEE 11073-Familie (ursprünglich entwickelt für die Integration von Geräten der Intensivmedizin) in Verbindung mit *Devices Profile for Web Services* (DPWS) und *Data Distribution Service* (DDS) gefunden (Gregorczyk et al., 2014; Plourde et al., 2014; Kasparick et al., 2015a).
Die Autoren in (Kasparick et al., 2015a) stellen die Erweiterungsstandards IEEE 11073-20702 (MDPWS), IEEE 11073-20701 (Binding) und IEEE 11073-10207 (semantische Interoperabilität) für die IEEE 11073 vor. Dadurch sollen diejenigen Elemente service-orientierter Kommunikation von Medizingeräten (auch außerhalb des Intensivbereichs) beschrieben werden können, die durch die bestehenden IEEE 11073-Standards noch nicht abgedeckt sind. Eine neuartige Entwicklung besteht in der Hinzunahme von *Zuständen* in die Gerätespezifikation. Über diese kann nach (Kasparick et al., 2015b) zur Laufzeit beschrieben werden, welchen aktuellen Wert eine Variable des Gerätes hat, ob dieser Wert auch *gültig* ist und zu welchem Zeitpunkt dieser Wert beobachtet wurde.
Eine Besonderheit des OR.Net-Projekts ist die Hinzunahme eines *Ethernet-Powerlink* (EPL)-basierten **Echtzeit-Bussystems** (Pfeiffer et al., 2015; Lüth et al., 2016), mit dessen Hilfe auch zeitkritische Sensordatenströme sowie Steuer- und Regelsignale innerhalb der offenen Architektur übertragen werden können. Zusätzlich zu den oben genannten Beschreibungsmerkmalen der IEEE 11073-Standards, verlangt die Einbindung in das Echtzeitnetzwerk die Beschreibung weiterer technischer Anforderungen, die für den zuverlässigen Betrieb des Netzwerks vonnöten sind (Dietz et al., 2016).
Wie in den vorigen Kapiteln dargestellt, genügt jedoch die rein technische Möglichkeit des Datenaustauschs nicht, um frei vernetzte Systeme von Medizingeräten am Patienten einsetzen zu dürfen. Die Forschung nähert sich daher aktuell der Frage, wie man die Beschreibung einzelner Geräte ergänzen kann, um dem Betreiber die Konformitätsbewertung eines frei zusammengefügten Systems derart beschriebener Geräte zu erleichtern.

2.7.2. Beschreibungsmethoden zum Zwecke der Konformitätsbewertung

Die Gruppe um J. Goldman hat den Standard ASTM F2761-09 mitentwickelt, welcher die abstrakte Architektur eines Integrated Clinical Environments und Anforderungen an konforme Implementierungen beschreibt. In ihrem Artikel *Design Pillars for Medical Cyber-Physical System Middleware* (Arney et al., 2014) formulieren die Autoren einen Anforderungskatalog an *Middleware* zur Realisierung von MIT-Netzwerken, die sie aus 10-jähriger Validierungstätigkeit mit konformen Prototypen im klinischen Einsatz extrahieren. Darunter finden sich die Folgenden:

- Jede Komponente sollte eine einheitliche Schnittstellenbeschreibung aufweisen.
- Das Netzwerk muss über die Frequenz, Genauigkeit und Verlässlichkeit von Daten informiert sein.
- Vorhandene Standards sollten zu einem maximalen Ausmaß weiterverwendet werden.
- Das System muss den Anwender informieren, wenn Anforderungen nicht eingehalten werden können.
- Die Konnektivität einzelner Komponenten, sowie kritische System- oder Komponentenzustände müssen stets erkenntlich sein.
- Geräte können zu jedem Zeitpunkt hinzugenommen oder abgezogen werden.
- Es müssen einfache Pfade zur Erfüllung regulatorischer Anforderungen definiert werden.

Um den in (Arney et al., 2014) genannten Anforderungen zu begegnen, haben die Autoren in (King et al., 2013) einen neuartigen Beschreibungsansatz für Medizingeräte unter Verwendung modaler Ein-/Ausgabe-Automaten gewählt. Mithilfe dieser formalen Beschreibungsmethode möchten die Autoren Eigenschaften eines MIT-Netzwerks maschinengestützt verifizieren (*Safety*-Eigenschaften, *Liveness*-Eigenschaften). Die Autoren evaluieren ihren Ansatz anhand einer PCA-Infusionspumpe: Diese wird verwendet, um Traumapatienten die eigenständige Dosierung von Schmerzmitteln zu ermöglichen. Dabei kann es vorkommen, dass sich der Patient eine Überdosis verabreicht, die zu respiratorischen Komplikationen führt, was wiederum von Sensoren detektiert werden kann. In der Evaluation erstellen die Autoren händisch unterschiedliche Modelle der PCA-Infusionspumpe und zeigen, wie maschinengestützt geprüft werden kann, ob die Auslegung der Pumpe die genannten Komplikationen verhindern kann. Sie verwenden dazu das MIO Workbench Tool und den PRISM *Model Checker* (Kwiatkowska et al., 2002). Die Autoren schließen ihre Arbeiten jedoch mit der Bemerkung ab, dass sich die gewählte Beschreibungsmethode nicht für die Beschreibung von Echtzeit Systemen eignet und die Definition unterschiedlicher Datentypen vermissen lässt.
Die Autoren in (Kühn und Leucker, 2014) formulieren Anforderungen an die Spezifikation von Komponenten von MIT-Netzwerken. Diese soll für Verifikationsmaßnahmen herangezogen werden, um Risiken zu entdecken, die durch den Verbundbetrieb von Geräten entstehen, die zuvor nicht im Verbund getestet werden konnten. Die Autoren erwähnen einige Beschreibungsmethoden (etwa UML Komponentendiagramme, Interface Automaten, modale

Automaten), mit deren Hilfe Eigenschaften des Geräteverbunds zur Laufzeit verifiziert werden können, ohne jedoch eine davon zu implementieren. Sie merken im selben Artikel an, dass die genannten Beschreibungsmethoden modifiziert werden müssen, um gewisse Interaktionsphänomene zu beschreiben (z.B. Austausch von Daten, Rückgabewerte von Funktionen). Gleichsam soll nach Ansicht der Autoren die Verifikation gewünschter Eigenschaften berechenbar bleiben. Sie formulieren folgende Thesen:

- Geräte müssen derart beschrieben sein, dass ein maschinengestützter Integrationstest vom Betreiber durchgeführt werden kann.
- Die Schnittstellen müssen derart beschrieben sein, dass die Frage nach Kompatibilität der Geräte berechenbar ist.

Auch die Autoren in (Mildner et al., 2015) schlagen vor, Eigenschaften und Anforderungen eines Medizinprodukts maschinenlesbar zu beschreiben. Nach Ansicht der Autoren sollen fünf Gesichtspunkte durch die Beschreibung abgedeckt werden:

1. Die Zweckbestimmung des Medizingeräts
2. Die technische Schnittstellenbeschreibung nach IEEE 11073
3. Die Bedienschnittstelle des Geräts
4. Risiken durch Austausch von Komponenten, technische Risiken durch Kommunikation
5. Netzwerkanforderungen (Bandbreite, Jitter etc.)

Eine Sprache, in der obige Elemente beschrieben werden können, stellen die Autoren jedoch nicht vor. Der Betreiber ist nach Ansicht der Autoren vor Inbetriebnahme eines Verbunds derart beschriebener Geräte in der Pflicht, die Erfüllung der Anforderungen durch das eigene Netzwerk und die Kompatibilität von Gerätebeschreibungen sicherzustellen. Die Autoren nehmen in Anspruch, eine vollständige Beschreibung angegeben zu haben, mit deren Hilfe eine sichere dynamische Vernetzung von Medizingeräten realisiert werden kann. Aus den Beschreibungen der Autoren ist nicht unmittelbar ersichtlich, ob und mit welchen Tools Betreiber beim Test der Kompatibilität unterstützt werden sollen.

2.7.3. Prozesse zur Konformitätsbewertung

Aufbauend auf dem Artikel (King et al., 2013) entwickeln die Autoren in (King et al., 2015) ein Modell einer „Ökosphäre“, in welcher sie ein Konzept für die beteiligten Akteure und Abläufe entlang des Lebenszyklus künftiger, integrierbarer Medizinprodukte skizzieren. Dabei schlagen sie unter anderem modellbasierte Methoden und *Model Checking* (siehe Kapitel 4) für den Nachweis der Sicherheit von Medizinprodukte-Kombinationen vor. Die Anforderungen an das MIT-Netzwerk sind dabei nach Ansicht der Autoren vollständig in der Beschreibungsdatei des Medizinprodukts hinterlegt und müssen durch einen Akteur im Netzwerk verifiziert werden. Insgesamt sind folgende Akteure benannt:

- Das Standardisierungskonsortium: Dieser Akteur ist zuständig für die Erstellung von Regelwerken für Teilnehmer an der Ökosphäre. Dazu gehören die Standardisierung der Kommunikationsprotokolle, der Schnittstellenbeschreibungen und die Beschreibung von Regeln, die entscheiden, ob eine Gerätekombination zulässig ist oder nicht.
- Der Hersteller ist für das Inverkehrbringen von Medizinprodukten verantwortlich. Dies schließt die Konformität mit den Regelwerken des Standardisierungskonsortiums ein.
- Der Systemintegrator: Aufgabe des Systemintegrators ist es, die Funktion des Medizinprodukte-Netzwerks zu definieren und die Sicherheit des Gesamtsystems zu beurteilen.
- Der Verkäufer der Plattform ist zuständig für den Nachweis deren korrekter Funktionalität. Dies schließt insbesondere ein, dass die Prüfung der Kompatibilität und der Nachweis der Systemsicherheit durch die Plattform korrekt durchgeführt werden.
- Der Zertifizierer ist nach Ansicht der Autoren zuständig dafür, die Konformität eines Medizinprodukts mit den Regeln der Ökosphäre zu prüfen und durch ein digitales Zertifikat zu bestätigen. Die Plattform kann dann anhand des Zertifikats eigenständig ermitteln, ob die Konformität gegeben ist.
- Der Betreiber hat nach Ansicht der Autoren keine Konformitätsbewertungspflichten.

2.7.4. Lösungsansätze anderer Disziplinen

Außerhalb der Medizintechnik wurden formale Methoden zur Sicherstellung der Fehlerfreiheit von Systemen bereits vielfach umgesetzt.
Die Autorin in (Wardana, 2009) wendet sie etwa in der Automatisierungstechnik an und übersetzt dort Continuous Function Charts *(CFC)* – eine grafische Programmiersprache für verteilte Systeme – in Timed Automata, um die Fehlerfreiheit so programmierter Automatisierungsanlagen (z.B. Versorgungstanks, Rührkesselreaktoren) nachzuweisen. Die Autoren in (Hackenberg et al., 2014) erarbeiten und implementieren eine formale Methode, mit der sie einen Designfehler in einer Pick-and-Place-Einheit aufdecken konnten.
Ähnliche Ansätze finden sich in der industriellen Robotik: Die Autoren in (Weißmann et al., 2011) übersetzen Programme industrieller Roboter (z.B. für das Karosserie Schweißen in der Automobilindustrie) in Protocol Meta Language *(PROMELA)*, um deren Freiheit von Deadlocks und Kollisionen nachzuweisen.
Stochastische Methoden werden beispielsweise für autonome Fahrzeuge in (Althoff, 2010) verwendet, um deren Sicherheit zu bewerten.
Da sich die genannten Methoden jedoch zumeist auf statische Systeme beziehen, die in einer einheitlichen Sprache programmiert werden, sind die Methoden nicht ohne Modifikation auf MIT-Netzwerke anzuwenden – hier besteht die Herausforderung darin, dass *unterschiedliche Hersteller* verschieden programmierte, teils komplexe Geräte in ein Netzwerk einbringen, das für jeden Eingriff neu konfiguriert wird.

2.8. Defizite am Stand der Technik

Die intensive Forschungsarbeit an MIT-Netzwerken unterstreicht, was im Stand der Technik deutlich geworden ist: Die Konformitätsbewertung solcher Systeme stellt Hersteller und Betreiber gleichermaßen vor neue Herausforderungen. Im Einzelnen gelten folgende Defizite:

2.8.1. Risikomanagement ist ressourcenintensiv

Risikomanagement erfordert Expertenwissen, das auf Betreiber-Seite meist nicht vorhanden ist (Woskowski, 2014; Kühn und Leucker, 2014). Gleichwohl ist die Eignung des Personals zur Durchführung des Risikomanagements normative Anforderung nach DIN EN ISO 14971, Kapitel 3.3. Aufgrund der häufig dürftigen Interaktion zwischen Medizintechnik und IT gibt es im Bereich Vernetzung häufig keine Risikomanagement-Aktivitäten (Gärtner, 2017). Die Umsetzung der DIN EN 80001-1 zur Sicherstellung des sicheren Betriebes eines medizinischen IT-Netzwerks erfordert zusätzliches Personal und Dokumentation sowie Kenntnisse der DIN EN ISO 14971 (Ahlbrandt et al., 2013; Kruber, 2015b).

2.8.2. Bestehende Standards und Werkzeuge bilden Vernetzung nicht ab

Bestehende Normen und Standards bilden die Anforderungen an sichere Medizingerätekommunikation nicht ab (Hatcliff et al., 2012). Kommerzielle Tools zur Erfüllung normativer Anforderungen (z.B. DIN EN ISO 14971) sind vorwiegend auf die konsistente *Dokumentation* von Risiken und risikomindernder Maßnahmen ausgelegt. Es gibt jedoch derzeit keine Methoden, mit denen Betreiber die Einhaltung von Anwender- und Hersteller-Anforderungen schnell, bedarfsabhängig und ohne externe Hilfe verifizieren können. Weder in Europa (siehe Abschnitt 2.1.2, 2.6.2) noch in den USA (Hatcliff et al., 2012) gibt es derzeit klare Prozesse zur Erfüllung regulatorischer Anforderungen bei der Zulassung von MIT-Netzwerken.

2.8.3. Keine geeigneten Beschreibungsmethoden für Medizingeräte

Methoden aus dem Stand der Forschung (z.B. nach IEEE 11073-x), nach denen Geräte allein anhand deren Ein- und Ausgabedaten beschrieben werden, erlauben keinen Rückschluss auf das Geräte- und Systemverhalten (King et al., 2013). Diesbezügliche Anforderungen beteiligter Hersteller und Anwender können daher nicht maschinenlesbar beschrieben, oder gar verifiziert werden. In der medizintechnischen Forschung werden formale Ansätze zwar diskutiert, jedoch keine Methoden zur Umsetzung bereitgestellt. Lösungen anderer Fachdisziplinen sind nicht ohne Modifikation auf MIT-Netzwerke anzuwenden, da sie anstelle dynamisch veränderlicher MIT-Netzwerke statische Anlagen betrachten, die in einer einheitlichen Sprache programmiert werden.

Zusammenfassend ist das Risikomanagement offen vernetzter Medizinprodukte ressourcenintensiv und derzeit nicht in einem praxistauglichen Prozess abzubilden. Da es **weder eine Beschreibungsmethodik für MIT-Netzwerke** gibt, die hinreichend aussagekräftig ist, **noch eine Methode für die Formulierung von Anforderungen**, ist auch deren **maschinengestützte Verifikation nicht möglich**.

3. Computergestützte Verifikation von Medizingerätenetzwerken

3.1. Eigener Ansatz

Ziel der vorliegenden Arbeit ist der Entwurf und die Implementierung einer *Verification Toolbox* für die computergestützte Verifikation dynamisch veränderlicher MIT-Netzwerke am Beispiel des OR.Net Echtzeit-Bussystems. Diese baut auf den folgenden zu erzielenden Teilergebnissen auf:

- **Formale Beschreibung einzelner Module:** Es sollen Methoden erarbeitet werden, mit denen technische Eigenschaften und Anforderungen eines *Moduls* (angebundenes Medizingerät oder Zubehör) in Textform standardisiert beschrieben werden können. Die Beschreibungen schließen insbesondere das *Verhalten* einzelner Module, d.h. die zeitliche Abfolge von deren Ereignissen, sowie Echtzeitanforderungen (max. Latenz, Frequenz, Rechenzeit) mit ein.
- **Erzeugung digitaler Gerätemodelle:** Auf Basis der Formalisierung soll eine Software implementiert werden, welche die Module digital abbildet und dem Nutzer ermöglicht, die Verknüpfung von Datenquellen und -senken (Verschaltung) zu editieren.
- **Formale Beschreibung der Interaktion der Module:** Sind Modulverhalten und -verschaltung beschrieben, muss die Interaktion der Module unter Berücksichtigung des verwendeten Datenaustauschmechanismus beschrieben werden. Im Falle des Echtzeit-Bussystems müssen Eigenheiten des EPL Protokolls modelliert werden.
- **Erzeugung eines digitalen Netzwerkmodells:** Es soll eine Software entstehen, mit der das formal spezifizierte Netzwerk digital repräsentiert werden kann. Mit dem digitalen Modell des Netzwerks können erste Plausibilitätsprüfungen vorgenommen werden. Typische Fragen, die hierbei untersucht werden können, sind:
  - Ist das Netzwerk funktionsfähig, oder ist der Ablauf an irgendeiner Stelle blockiert?
  - Verhält sich das Netzwerk erwartungsgemäß?

  Die formale Analyse des Netzwerks erfordert jedoch noch eine präzise Formulierung der gewünschten Eigenschaften wie folgt:
- **Automatisierte Verifikation von Anforderungen:** Zuletzt soll eine Software entwickelt werden, welche die Erfüllung formaler Anforderungen durch das beschriebene System automatisiert verifiziert. Dazu muss eine geeignete Sprache identifiziert werden, mit der Anforderungen formuliert werden können. Bei der Wahl derselben ist auf die effiziente Berechenbarkeit, sowie die Anwendbarkeit im klinischen Umfeld zu achten.

3.2. Erwartete Vorteile

Die beschriebene Toolbox ist die weltweit erste, welche die zuvor genannten Eigenschaften erfüllt. Zugleich erfüllt sie folgenden Zweck:

- Sie unterstützt Betreiber bei der Verifikation von Anforderungen durch ein MIT-Netzwerk (weniger Fehler).
- Sie verkürzt die Dauer der Verifikation von Anforderungen für ein gegebenes MIT-Netzwerk (kürzere Zeit).
- Sie ermöglicht vor Inbetriebnahme die Simulation des MIT-Netzwerks durch den Betreiber zum Zwecke der Prüfung des Systemverhaltens.

3.3. Abgrenzung

- **Echtzeit-Bussystem:** Das *Echtzeit-Bussystem* ist im Rahmen des OR.Net-Projekts an der TU München entstanden. Der Aufbau der *Konnektoren* und *Funktionsmodule* sowie die Implementierung des *Managing Node* (MN) sind nicht Bestandteil dieser Arbeit. Auch der EPL-Standard ist nicht in dieser Arbeit entstanden.
- **Algorithmen zur Verifikation:** Die Algorithmen zur Verifikation sowie das in Kapitel 5 eingeführte *UPPAAL*-Tool bestanden schon vor dieser Arbeit und wurden übernommen. Das UPPAAL-Tool liest zum Programmstart eine *Extensible Markup Language* (XML)-Datei auf, welche die Systemspezifikation enthält. Diese Datei wird durch die Verification Toolbox dieser Arbeit automatisiert erzeugt.
- **Beschreibungsmethoden:** Auch wenn die bestehenden Beschreibungsmethoden modifiziert und auf die konkrete Problemstellung adaptiert wurden, kann eine Urheberschaft nicht in Anspruch genommen werden. Wo in dieser Arbeit ein neuer Beitrag geleistet wurde, ist dies deutlich hervorgehoben. Die Teilsprache von *Timed Computation Tree Logic* (TCTL) für die Formulierung der Anforderungen wurde unverändert übernommen.
- **Einschränkungen:** Diese Arbeit entstand parallel zu laufenden Arbeiten am Echtzeit--Bussystem. Es werden daher Annahmen über Fähigkeiten des Echtzeit-Bussystems getroffen, deren Implementierung zum Zeitpunkt der Einreichung noch nicht abgeschlossen, aber durch das Entwicklungsteam des *MiMed* und des MN-Herstellers als machbar bewertet wurde. Dazu gehören insbesondere:

  1. Die dynamische Zuteilung von Adressen an Controlled Nodes (CNs).
  2. Die automatisierte Adaptation des MN auf die angeschlossenen Netzteilnehmer.

4. Konzept und Systementwurf

Die Kernidee der vorliegenden Arbeit ist die formale Spezifikation aller Netzwerk-Komponenten zum Zwecke der maschinengestützten Verifikation formal beschriebener Anforderungen. In der wissenschaftlichen Literatur ist dieser Ansatz umfassend beschrieben (z.B. in (Baier et al., 2008d)) und wird als Model Checking bezeichnet.
Die ersten Arbeiten zum Model Checking stammen parallel von (Clarke und Emerson, 1982) und (Queille und Sifakis, 1982).
Der folgende Abschnitt dient als kurze Einführung, um Begriffe zu klären, die in dieser Arbeit benötigt werden. Er folgt im Wesentlichen den Ausführungen in (Baier et al., 2008e).

Einführung - Model Checking

Die Disziplin des Model Checking befasst sich mit der Frage, wie man Eigenschaften eines Systems mit mathematischen Methoden nachweisen kann. Dabei sind zwei wesentliche Komponenten vonnöten: Zum einen ein hinreichend deskriptives, mathematisch eindeutiges **Modell des Systems** und zum anderen eine präzise **Formulierung von Anforderungen**, die durch das System erfüllt werden sollen (siehe Abb. 13). Typische Anforderungen, die durch Model Checking verifiziert werden sollen, sind:

- Ist ein Berechnungs-Ergebnis in Ordnung? (*Correctness*)
- Kann das System in einen Zustand gelangen, aus dem es sich nicht herausbewegen kann? (*Deadlock*)
- Werden Wunsch-Zustände erreicht und kritische Zustände gemieden? (Liveness und Safety)
- Reagiert das System innerhalb einer definierten Zeitschranke auf eine Eingabe? (*Realtime Property*)

Gegeben ein formales – d.h. mathematisches – Modell des Systems und der Anforderungen, traversiert ein Model Checker (ein Programm, das Methoden des Model Checking implementiert) systematisch sukzessive sämtliche Systemzustände und prüft, in welchen Zuständen eine Anforderung gültig ist oder nicht. Model Checking hat in zahlreichen Disziplinen Einzug gehalten. Prominente Beispiele sind die Überarbeitung des ISDN-Standards nach der Analyse durch (Holzmann, 1994), die Überarbeitung des IEEE Futurebus+ Cache Coherence Protokolls nach der Aufdeckung formaler Fehler durch (Clarke et al., 1995), sowie die Enthüllung vierer schwerwiegender Fehler in der Steuerung eines Raumschiffs vor der Deep Space 1 Mission durch (Havelund et al., 2001). Dabei sind je nach Disziplin unterschiedliche Systemeigenschaften von Interesse. Mit wachsendem Beschreibungsumfang der gewählten Spezifikationssprache kann der Nachweis von Eigenschaften des Systems schwieriger oder gar unmöglich werden. In hinreichend mächtigen Beschreibungssprachen sind nie alle Eigenschaften des Systems beweisbar (Gödel, 1931).

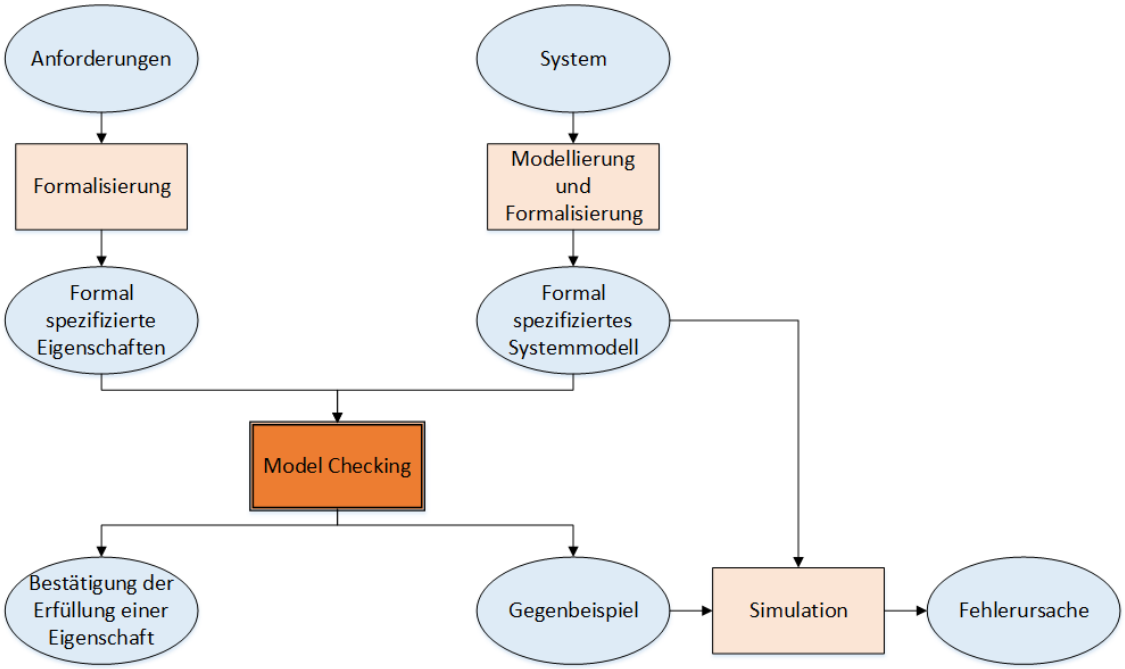


Abbildung 13 Das Prinzip des Model Checking. Anforderungen werden formalisiert, um eine Verifikation gegenüber einer formalen Spezifikation zu ermöglichen. Angelehnt an (Baier et al., 2008d).

Eine Herausforderung ist daher stets, geeignete Beschreibungsmethoden für das jeweilige System und die zu verifizierenden Anforderungen zu finden. Vor der Einführung der in dieser Arbeit gewählten Spezifikationsmethode folgt eine kurze Beschreibung des Echtzeit-Bussystems, um den erforderlichen Beschreibungsumfang abzustecken und Begrifflichkeiten zu klären. Eine detaillierte Beschreibung des Echtzeit-Bussystems findet sich in (Lüth et al., 2016).

Einführung - das Echtzeit-Bussystem

Das Echtzeit-Bussystem ist ein an der TU München entwickeltes echtzeitfähiges Bussystem für die herstellerübergreifende Kommunikation von Medizingeräten und Zubehör. Es ist *dynamisch* insofern, als die Verknüpfung der Datenquellen und -senken zur Laufzeit geändert werden kann. Die Basis-Technologie hinter dem Echtzeit-Bussystem ist der *Ethernet-Powerlink* (EPL)-Standard (detaillierte Beschreibung in Kapitel 4.2). Dieser sieht einen designierten *Managing Node* (MN) vor, der die gesamte Kommunikation im Netzwerk steuert. Medizingeräte und Zubehör werden über sogenannte *Konnektoren* an den Bus angebunden. Diese fungieren als Übersetzer zwischen dem standardisierten Protokoll im Echtzeit-Bussystem und den geräteeigenen Schnittstellen (siehe Abb. 14). Zudem können sogenannte Funktionsmodule eingebunden werden, welche auf Basis der standardisierten Netzwerk-Daten dedizierte Funktionen wahrnehmen (automatische Leistungssteuerung, Verrechnung von Navigationsdaten etc.). Komponenten, die über das Echtzeit-Bussystem miteinander kommunizieren, werden in dieser Arbeit Module genannt. Dies können Medizingeräte und Zubehör einschließlich ihrer Konnektoren, Funktionsmodule oder der MN sein.

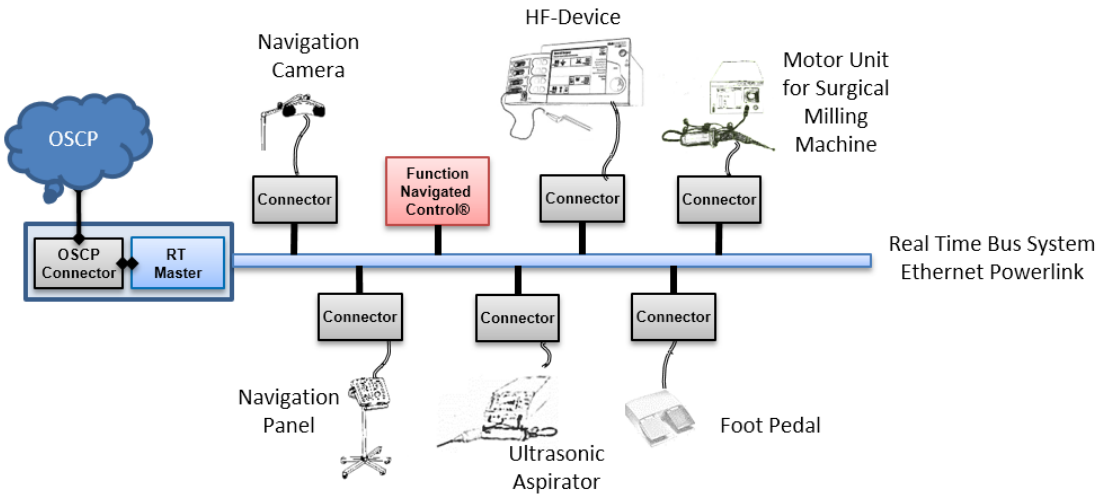


Abbildung 14 Das OR.Net Echtzeit-Bussystem. Der RT-Master (links, blau) überwacht und taktet das gesamte Netzwerk. Medizingeräte und Zubehör werden über Konnektoren (grau) an das Bussystem angebunden. Funktionsmodule (rot) nehmen neuartige Funktionen (Leistungssteuerung, Verrechnung von Navigationsdaten, ...) auf Basis der standardisierten Daten im Netzwerk war. Das Echtzeit-Bussystem hat ein Gateway zur *Open Surgical Communication Protocol* (OSCP) (blaue Wolke) um nicht-zeitkritische Daten in das Netzwerk einzuspeisen.

Formale Beschreibung des Echtzeit-Bussystems

In dieser Arbeit wird das Echtzeit-Bussystem einschließlich seiner Komponenten durch ein modulares System nebenläufiger, kommunizierender Automaten modelliert. Dabei beschreibt jede Komponente ihre *eigenen* Eigenschaften und Anforderungen in einer einheitlichen Syntax. Abhängig von der logischen Verschaltung der Komponenten (wer spricht mit wem worüber) entsteht durch die Komposition der so beschriebenen Komponenten ein neuer Automat, welcher die Eigenschaften des komponierten Systems beschreibt.

Einführung - Umfang der Beschreibungsmethode

Vor Beginn der mathematischen Modellierung lohnt es zu untersuchen, welche Elemente der Geräte zu beschreiben sind. Als Beispiel betrachte man hierzu (vgl. (Dingler et al., 2017)) die chirurgische Saug-/Spülpumpe *Endomat LC* der Firma *KARL STORZ GmbH & Co. KG* (Karl Storz) aus Abb. 15. Diese Rollenpumpe kann zum Spülen oder Absaugen von Flüssigkeit bei chirurgischen Eingriffen verwendet werden. Sie verfügt über einen Knopf zum Ein- und Ausschalten. Vor Inbetriebnahme muss die Pumpe über einen zusätzlichen Knopf initialisiert werden. Sie hat zwei Betriebsmodi (High/Normal) bei denen die Pumpe jeweils voreingestellte Förderraten (in ml/min) realisiert. Innerhalb der beiden Modi kann der Anwender die Förderraten einstellen – diese wird dann für die Dauer der Anwendung von der Pumpe gespeichert. Zwischen beiden Modi schaltet ein diskreter, zweistufiger Einpedal-Fußschalter um. Wird der Fußschalter nicht betätigt, löst die Pumpe nicht aus. Drückt der Anwender das Fußpedal halb durch, so saugt bzw. spült die Pumpe mit *normaler* Intensität, bei vollem Durchdrücken schaltet die Pumpe in den *High* Zustand. Die aktuelle Saug-/Spülleistung wird über ein Display auf der Vorderseite der Pumpe angezeigt. Um die zuverlässige Ansteuerung zu ermöglichen, sollten diese Steuersignale maximal 50 ms alt sein.

Aus dieser Beschreibung wird deutlich, dass die Beschreibungsmethode zumindest die Modellierung folgender Elemente ermöglichen muss: **Zustände** (z.B. Ein/Aus), **Speicherung**

Abbildung 15 Eingrenzung des Beschreibungsumfangs der Methode anhand der Saug-/Spülpumpe *Endomat LC* der Firma Karl Storz. Angelehnt an (Dingler et al., 2017).

von Variablen (etwa für die eingestellte Förderrate), **Ein-/Ausgabesignale** (für die Auslösung oder auch das Display) sowie einen **Zeitbegriff** zur Analyse von Risiken aus Überschreitung der Echtzeitanforderungen.

Notation

In dieser Arbeit werden spezielle Automaten – *TEFAs* – zur Beschreibung der Netzwerkkomponenten verwendet. Vorab einige Konventionen zur Notation (siehe Symbolverzeichnis):
Seien A und B beliebige Mengen. In dieser Arbeit wird $|A|$ für die Kardinalität von A (d.h. die Anzahl der Elemente in A), $B \backslash A := \{b \in B : b \notin A\}$ für das Komplement von A in B, $B^A := \{f : A \to B : f \text{ Funktion}\}$ für die Menge der Abbildungen von A nach B und speziell $2^A := \{0,1\}^A$ für die Potenzmenge von A verwendet.
Gegeben eine Funktion $f : A \to B$ (also $f \in B^A$) und eine Teilmenge $C \subseteq A$, so beschreibt $f|_C : C \to B$ die Einschränkung von f auf die Teilmenge C (also $f|_C(x) := f(x)$ für $x \in C$).
Für jedes $k \in \mathbb{N}$ sei $[k] := \{1, \dots, k\}$ die Menge der ersten k natürlichen Zahlen.
Für einen beliebigen Vektor $x = (x_1, \dots, x_n)$ ist $x\left[\tilde{x}_j / x_j\right] := (x_1, x_2, \dots, \tilde{x}_j \dots, x_n)$ der Vektor, der an allen Einträgen – bis auf den j-ten – mit x übereinstimmt und dessen j-ter Eintrag $\tilde{x}_j$ ist.
Wie üblich beschreibt das Symbol $\wedge$ die logische Konjunktion („UND“) und $\vee$ die logische Disjunktion („ODER“). Ist eine Aussage φ hinreichend für eine Aussage ψ, so wird dies durch die Zeichensequenz $\varphi \Rightarrow \psi$ beschrieben. In diesem Fall heißt ψ *notwendig* für φ. Ist φ notwendig und hinreichend für ψ zugleich, so wird wahlweise $\varphi \Leftrightarrow \psi$, oder φ ***genau dann, wenn*** **(gdw)** ψ geschrieben.

4.1. Methoden zur Beschreibung des Echtzeit-Bussystems

In dieser Arbeit werden zur Modellierung des Echtzeit-Bussystems *Timed Extended Finite Automata* verwendet. Diese mathematischen Objekte erweitern die Theorie klassischer endlicher Automaten (*Finite Automata*) um einen Zeitbegriff (*Timed*) und Variablen (*Extended*).
Endliche Automaten werden seit vielen Jahrzehnten für die Modellierung und Verifikation verwendet. Die unterschiedlichen Grundformen von Automaten (etwa *endliche Automaten*, *Mealy-Automaten* (Mealy, 1955), *Moore-Automaten* (Moore, 1956), *Büchi-Automaten*, *Petrinetze* (Petri, 1962), *Statecharts* (Harel, 1987) uvm.) basieren alle auf derselben Semantik endlicher Transitionssysteme (siehe Bemerkung 5), die eine vollautomatische Verifikation ermöglicht (Beyer, 2002). All diesen Ansätzen ist jedoch auch gemein, dass der Begriff der *Zeit* nicht explizit modelliert werden kann; sie sind daher für die Modellierung von Echtzeitsystemen nicht geeignet, da deren Korrektheit davon abhängt, *wann* ein Ereignis auftritt.
Die wesentlichen theoretischen Erkenntnisse über Automaten mit Zeitbegriff gehen auf die Pionierarbeit von R. Alur und D. Dill in (Alur und Dill, 1994) zurück.
Die hier gewählte Definition des *Timed Extended Finite Automaton* (TEFA) führt die Anforderungen an *Timed Automata* aus (Alur und Dill, 1994) und (Bengtsson und Yi, 2004) zusammen und erweitert sie um Variablen. Da die Literatur vor allem in Hinblick auf die Variablen vorwiegend intuitive Definitionen anhand von Beispielen bereitstellt (z.B. (Baier et al., 2008c), (Pajic et al., 2014)), werden diese Erweiterungen speziell für diese Arbeit mathematisch präzisiert (insbesondere Definitionen 4.1.6 – 4.1.9).

4.1.1. Grundlagen

Im folgenden Abschnitt werden die elementaren Begriffe zu Alphabeten, Grammatiken, Prädikaten und Transitionssystemen eingeführt. Die folgenden beiden Definitionen sind in ähnlicher Form in Standard-Lehrbüchern zu finden (vgl. (Hofmann und Lange, 2011), (Ernst et al., 2015a)).

Definition 4.1.1 (Alphabet, String, Kleene-Abschluss und Konkatenation)
Ein Alphabet A *ist eine nicht-leere, endliche Menge. Elemente der Form*

$$x_1 x_2 \dots x_k := (x_1, x_2, \dots x_k) \in A^k \ (k \in \mathbb{N})$$

heißen Strings *der Länge* k*. Der leere* String *der Länge* 0 *wird stets mit* ϵ *bezeichnet. Die Menge* $A^* := \bigcup_{k=0}^{\infty} A^k$ *mit* $A^0 := \{\epsilon\}$ *heißt der* Kleene-Abschluss *von* A*. Sei nun* B *ein weiteres Alphabet. Die* Konkatenation *ist eine Abbildung*

$$\cdot : A^* \times B^* \to (A \cup B)^*, (a, b) \mapsto ab,$$

welche zwei Strings aus A^* *und* B^* *aneinanderreiht.*

Definition 4.1.2 (Grammatik und erzeugte Sprache)
Eine Grammatik G *ist ein Tupel* $G = (N, \Sigma, P, S)$ *mit einer endlichen Menge* N *(genannt:* Nichtterminale*), einem Alphabet* Σ*, einer endlichen Menge* $P \subseteq ((N \cup \Sigma)^* | \Sigma^*) \times (N \cup \Sigma)^*$

(genannt: Ableitungsregeln*) und einem Startsymbol $S \in N$. Seien nun $u \in (N \cup \Sigma)^* \backslash \Sigma^*$ und $(p_1, p_2) \in P$ so, dass $u = ap_1b$ für geeignete $a, b \in (N \cup \Sigma)^*$. Man schreibt dann $u \rightsquigarrow_G ap_2b$ und sagt, dass ap_2b aus u* abgeleitet *wurde.*
Ist $w_1, \dots w_n \in (N \cup \Sigma)^*$ eine Folge von Strings so, dass $w_i \rightsquigarrow_G w_{i+1}$ für alle $i \in 1, \dots, n-1$, so schreibt man auch $w_1 \rightsquigarrow_G^n w_n$.
Ist $u \in (N \cup \Sigma)^* \backslash \Sigma^*$, so bezeichnet $Abl_n^{(G)}(u) := \{w \in \Sigma^* : u \rightsquigarrow_G^n w\}$ die Menge der $n-$Schritt-Ableitungen von u.
Schließlich heißt $L^{(G)}(u) := \bigcup_{n=0}^{\infty} Abl_n^{(G)}(u)$ die aus u durch G erzeugte Sprache und speziell $L(G) := L^{(G)}(S) \subseteq \Sigma^*$ die durch G erzeugte Sprache.

Bemerkung 1 (Algorithmus zur Berechnung der erzeugten Sprache)
Obige Definition der Ableitung zeigt, dass man für die Ermittlung der durch eine Grammatik *erzeugten Sprache folgendermaßen vorgeht:*

1. *Wähle eine Ableitungsregel $p = (p_1, p_2)$ und suche im Startsymbol S den String p_1.*
2. *Ersetze in S beliebig viele Instanzen des Strings p_1 durch p_2.*
3. *Wiederhole Schritt 1 und 2 mit weiteren Ableitungsregeln (auch p selbst) so lange, bis der resultierende String nur noch aus Elementen des Alphabets Σ besteht (also ganz in Σ^* enthalten ist).*

Die Sprache $L(G)$ besteht exakt aus den Strings, die auf diese Weise gewonnen werden.

Beispiel 4.1.1 (Berechnung der erzeugten Sprache)
Sei $G = (N, \Sigma, P, S)$ mit $N := \{STR\}$, $\Sigma := \{a, \wedge\}$, $P := \{(STR, a), (STR, STR \wedge STR)\}$ und $S := STR$. Ziel ist es, $L(G)$ zu berechnen.

1. *Man betrachte die erste Ableitungsregel $p = (STR, a) \in P$.*
   Klarerweise ist $p_1 = STR$ in $S = STR$ enthalten (Schritt 1) und daher darf $p_1 = STR$ durch $p_2 = a$ ersetzt werden (Schritt 2). Es folgt $STR \rightsquigarrow a$. Da $a \in \Sigma \subseteq \Sigma^*$, folgt $a \in L(G)$ (Schritt 3).
2. *Nun betrachte man die zweite Ableitungsregel $q = (STR, STR \wedge STR)$. Da wieder $q_1 = STR$ in $S = STR$ enthalten ist, folgt $STR \rightsquigarrow STR \wedge STR$ (Schritte 1 und 2). Da nun aber $STR \wedge STR \notin \Sigma^* \supseteq L(G)$, ist $STR \wedge STR \notin L(G)$. Man wiederholt daher (Schritt 3) Schritte 1 und 2. Dabei gibt es unterschiedliche Möglichkeiten:*
   a. *Ersetze jedes STR in $STR \wedge STR$ durch a (Ableitungsregel 1). Dann folgt $STR \wedge STR \rightsquigarrow a \wedge a \in \Sigma^*$ und es folgt $a \wedge a \in L(G)$.*
   b. *Ersetze nur ein STR in $STR \wedge STR$ durch a (Ableitungsregel 1), also $STR \wedge STR \rightsquigarrow STR \wedge a \notin L(G)$.*
   c. *Ersetze ein STR in $STR \wedge STR$ durch a (Ableitungsregel 1) und ein STR durch $STR \wedge STR$ (Ableitungsregel 2), also $STR \wedge STR \rightsquigarrow a \wedge STR \wedge STR \notin L(G)$ oder $STR \wedge STR \rightsquigarrow STR \wedge STR \wedge a \notin L(G)$.*

Da in den Schritten b und c jeweils noch das Nichtterminal-Symbol STR auftaucht, sind die resultierenden Strings nicht Teil der erzeugten Sprache, daher kehrt man zurück zu Schritt 1. Wendet man nun Ableitungsregel 1 etwa auf alle STR im String $STR \wedge STR \wedge a$ aus Schritt c an, so erkennt man $a \wedge a \wedge a \in L(G)$.

Induktiv macht man sich auf diese Weise leicht klar, dass $L(G) = \left\{ \bigwedge_{k=1}^{n} a \,:\, n \in \mathbb{N} \right\}$.

Bemerkung 2 (Backus-Naur Form, angelehnt an (Ernst et al., 2015b))
In dieser Arbeit wird zur Verkürzung der Notation bei Grammatiken *folgendermaßen vorgegangen: Statt die Ableitungsregeln in der Form*

$$P = \{(p,\, q)\,,\, (p,\, r)\,, (p,\, s)\,,\, (q,\, rr)\,,\, (q,\, ss)\}$$

zu schreiben, werden sie in der Form

$$\begin{aligned} p &::= q|r|s \text{ mit} \\ q &::= rr|ss \end{aligned}$$

zusammengefasst. Der Trennstrich „|“ steht also für ein logisches „oder“. Das Startsymbol S steht Ausdrücken obiger Form immer auf der linken Seite der ersten Zeile (in diesem Beispiel $S = p$). In der Regel wird nur das Alphabet Σ angegeben (in diesem Fall: $\Sigma = \{r,\, s\}$), die Nicht-Terminalsymbole ergeben sich dann aus dem Kontext (in diesem Beispiel: $N = \{p,\, q\}$). In obigem Beispiel 4.1.1 würde also verkürzt stehen:
$\mathbf{L(G)}$ *ist die Sprache, die durch die Grammatik*

$$STR ::= \mathbf{a}|STR \wedge STR$$

auf dem Alphabet $\mathbf{\Sigma} = \{\mathbf{a},\, \wedge\}$ erzeugt wird.

Definition 4.1.3 (Sprache der Zwangsbedingungen, vgl. (Alur und Dill, 1994))
Sei A ein Alphabet, $\epsilon \in A^*$ der leere String und $M \subseteq \mathbb{Z}$ eine endliche Teilmenge der ganzen Zahlen. Die Sprache der Zwangsbedingungen *$\Phi_M(A)$ ist definiert als diejenige Sprache, die auf dem Alphabet $A \dot\cup \{\leq,\, \geq\} \dot\cup M$ durch die Grammatik*

$$Bedingung ::= \epsilon|EXPR \text{ mit}$$

$$EXPR ::= (a \leq m)\,|\,(a \geq m)\,|EXPR \wedge EXPR,\ (a \in A,\, m \in M)$$

erzeugt wird. Elemente aus $\Phi_M(A)$ werden in dieser Arbeit Bedingungen *genannt. Dabei schreibt man häufig etwa $a = 4$ anstelle von $(a \leq 4) \wedge (a \geq 4)$, um die Notation schlank zu halten. Wenn aus dem Kontext ersichtlich ist, um welche endliche Teilmenge $M \subseteq \mathbb{Z}$ es sich handelt, oder wenn dies unerheblich ist, wird M in der Notation weggelassen und schlicht $\Phi(A)$ geschrieben.*

Bemerkung 3
Elemente aus $\Phi(A)$ sind zunächst rein syntaktische Konstrukte – schließlich wurde keine Anforderung an die Menge A gestellt, welche etwa dem Ausdruck $a_1 \leq 4$ Bedeutung geben

könnte (z.B. $A \subseteq \mathbb{R}$). Die nächste Definition zeigt aber, wie das Konstrukt im intuitiven Sinne benutzt werden kann. Im Zusammenhang mit Automaten kann man sich $\Phi(A)$ vorstellen als die Menge aller Forderungen an obere und untere Schranken von Elementen aus A.

Definition 4.1.4 (Erfüllung einer Bedingung, angelehnt an (Baier et al., 2008f))
Sei D ein Alphabet, $\epsilon \in D^*$ der leere String, $\varphi \in \Phi(D)$ eine Bedingung über D und $f \in \mathbb{R}^D$ eine reellwertige Funktion. Die Erfüllung der Bedingung φ durch die Funktion f, in Symbolen $f \vDash \varphi$, wird induktiv wie folgt definiert:

- *Ist $\varphi = \epsilon$, so gilt $f \vDash \varphi$.*
- *Ist $\varphi = (x \leq m)$ (bzw. $\varphi = (x \geq m)$) für ein $x \in D$ und $m \in \mathbb{N}_0$, so gilt*

$$f \vDash \varphi \text{ genau dann, wenn } \textit{(gdw)}\ f(x) \leq m \ (\textit{bzw.}\ f(x) \geq m).$$

- *Ist $\varphi = \bigwedge_{k=1}^{m} \varphi_k$ für ein $m \in \mathbb{N}_0$ mit $\varphi_k \in \left\{ \left(x_k \leq m_k\right), \left(x_k \geq m_k\right) : x_k \in D, m_k \in \mathbb{N}_0 \right\}$ für alle $k \in [m]$, so gilt*

$$f \vDash \varphi \textit{ gdw } f \vDash \bigwedge_{k=1}^{m-1} \varphi_k \textit{ und } f \vDash \varphi_k,$$

  wobei $\bigwedge_{k=1}^{0} \varphi_k := \epsilon$.

Die nun folgende Definition eines Transitionssystems ist Grundlage für zahlreiche aufbauende Beschreibungsmethodiken; dazu gehören insbesondere die in dieser Arbeit verwendeten TEFAs.

Definition 4.1.5 (Transitionssystem, vgl. (Baier et al., 2008c))
Ein Transitionssystem $TS = \left(T, t_0, A, \rightarrow, P, L\right)$ ist ein Tupel mit folgenden Elementen:

- *Einer endlichen Menge T an Zuständen mit Anfangszustand $t_0 \in T$,*
- *einem Alphabet A, genannt:* actions,
- *einer Transitionsrelation $\rightarrow \subseteq T \times A \times T$,*
- *einer endlichen Menge P an Prädikaten und*
- *einer Zuordnung $L : S \rightarrow 2^P$.*

Für Elemente der Relation $\rightarrow$ schreibt man häufig $A \xrightarrow{a} B$, anstelle von $(A, a, B) \in \rightarrow$.

Bemerkung 4 (Verkürzte Notation)
In manchen Zusammenhängen werden Transitionssysteme nur dazu verwendet, die Semantik des Zustandswechsels einer darauf aufbauenden Beschreibungsmethode zu definieren. Daher wird häufig P und L weggelassen und schlicht von einem Transitionssystem $T = \left(T, t_0, A, \rightarrow\right)$ gesprochen.

Bemerkung 5 (Semantik eines Transitionssystems, vgl. (Baier et al., 2008c))
Das Verhalten *eines Transitionssystems kann man sich wie folgt vorstellen: Das System startet im Zustand t_0. Aus der Menge der* ausgehenden Transitionen *von t_0, d.h. aller Elemente*

der Form $t_0 \xrightarrow{*} x$ ($*\in A$, $x \in T$), kann nun eine Beliebige gewählt und wahrgenommen werden. Die Auswahl findet nicht-deterministisch statt. Ist etwa $t_0 \xrightarrow{a} t_1$ die gewählte Transition, so führt das Transitionssystem die action a *aus und* wechselt *in den Zustand t_1. Dieses Verfahren wird solange fortgeführt, bis ein Zustand ohne ausgehende Transitionen erreicht ist (möglicherweise unendlich lange). Die Zuordnung L ordnet einem Zustand $t \in T$ einen Satz an Prädikaten zu, die in diesem Zustand erfüllt sind (siehe auch Definition 4.1.4).*

Beispiel 4.1.2 (Transitionssystem Trinkautomat, vgl. (Baier et al., 2008c))
Man betrachte die Grafik in Abbildung 16. Darin wird ein Transitionssystem $TS = (T, t_0, A, \rightarrow, P, L)$ beschrieben, mit folgenden Elementen:

- $T := \{PAY, SODA, CHOOSE, BEER\}, t_0 := PAY,$
- $A := \{insert\_coin, \tau, get\_soda, get\_beer\},$
- *Die Transitionsrelation $\rightarrow$ ermöglicht die Transitionen*
  - $PAY \xrightarrow{insert\_coin} CHOOSE$
  - $CHOOSE \xrightarrow{\tau} SODA$
  - $CHOOSE \xrightarrow{\tau} BEER$
  - $SODA \xrightarrow{get\_soda} PAY$
  - $BEER \xrightarrow{get\_beer} PAY$

Die Menge P, sowie die Zuordnung L können frei gewählt werden. Beispielsweise könnte man setzen $P := \{coin\_inserted \wedge no\_beverage\}$, um auszudrücken, dass eine Münze eingeworfen, aber noch kein Getränk ausgegeben wurde. In diesem Fall wäre

$$L(PAY) = \emptyset,\ L(CHOOSE) = L(SODA) = L(BEER) = \{coin\_inserted \wedge no\_beverage\}.$$

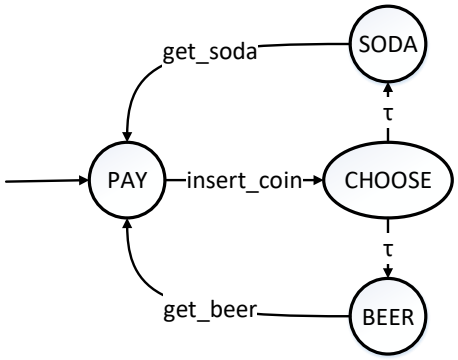


Abbildung 16 Einfaches Transitionssystem, das einen Getränkeautomaten beschreibt. Das System kennt die Zustände PAY, $SODA$, $CHOOSE$ und $BEER$ und die Actions τ, get_soda, get_beer und insert_coin. Es startet im Zustand PAY und wechselt bei Eingabe einer Münze in den Zustand $CHOOSE$. Daraufhin entscheidet es nicht-deterministisch, ob es in den Zustand $BEER$ oder $SODA$ übergeht und gibt bei der darauffolgenden Transition Bier (get_beer) oder Soda (get_soda) aus. Danach startet die Prozedur von Anfang an. Angelehnt an (Baier et al., 2008c).

4.1.2. Neue Erweiterungen

Die nun folgenden beiden Definitionen ermöglichen die mathematische Modellierung von Wertzuweisungen zu Variablen und Uhren bei Automaten. Genauer soll das Zurücksetzen von Uhren, also Ausdrücke der Form $clock = 0$ modelliert werden. Genauso sollen affine Zuordnungen der Form $y = a \cdot x + b$ auf Transitionen formalisiert werden, wobei x, y Variablen des Automaten und a, $b \in \mathbb{Z}$ ganze Zahlen sind. Während das Zurücksetzen von Uhren in der Literatur bereits formal behandelt wurde (z.B. (Alur und Dill, 1994)), wurde letzterer Aspekt bislang nur informell beschrieben.

Definition 4.1.6 (Affine Zuordnungen und Resets)
Sei wieder A ein beliebiges Alphabet, $\epsilon \in A^*$ der leere String und $M \subseteq \mathbb{Z}$ eine endliche Teilmenge der ganzen Zahlen. Die Sprache $\mathcal{A}(A)$ der affinen Zuordnungen *von A ist definiert als die von der Grammatik*

$$\textit{Zuordnung} ::= \mathit{EXPR} | \epsilon \textit{ mit}$$

$$\mathit{EXPR} ::= (y \to a \cdot x + b) | \mathit{EXPR} \wedge \mathit{EXPR} \quad (x, y \in A,\ a, b \in M \subset \mathbb{Z})$$

auf dem Alphabet $\Sigma = A \dot\cup \{\to, \cdot, +\} \dot\cup M$ erzeugten Sprache. Der Ausdruck $y \to a \cdot x$ steht kurz für die Zuordnung $y \to a \cdot x + 0$ und $y \to x + b$ für die Zuordnung $y \to 1 \cdot x + b$. Weiter wird die Sprache der Resets $\mathcal{A}_0(A) \subseteq \mathcal{A}(A)$ über die Grammatik

$$\textit{Reset} ::= \mathit{EXPR} | \epsilon \textit{ mit}$$

$$\mathit{EXPR} ::= (x \to 0 \cdot y) | \mathit{EXPR} \wedge \mathit{EXPR} \quad (x, y \in A)$$

definiert.

Bemerkung 6
Man beachte (wie in Bemerkung 3), dass auch die Menge $\mathcal{A}(A)$ zunächst nur ein syntaktisches Konstrukt ist, da das Alphabet A frei gewählt werden kann.

Definition 4.1.7 (Induzierte Abbildung)
Sei wieder A ein beliebiges Alphabet und $\epsilon \in A^*$ der leere String. Jedes $\varphi \in \mathcal{A}(A)$ induziert eine Abbildung

$$[\varphi] :\ \mathbb{Z}^A \to \mathbb{Z}^A,\ \mu \mapsto [\varphi]\mu,$$

wobei $[\varphi]\mu$ induktiv folgendermaßen definiert wird:

- *Ist $\varphi = \epsilon$ der leere String, so ist $[\varphi]\mu = \mu$.*
- *Ist $\varphi = (y \to a \cdot x + b)$ für feste a, $b \in \mathbb{Z}$ und x, $y \in A$, so ist für $z \in A$*

$$[\varphi]\mu(z) = \mathbf{1}_{A \setminus \{y\}}(z) \cdot \mu(z) + \mathbf{1}_{\{y\}}(z) \cdot (a \cdot \mu(x) + b) = \begin{cases} \mu(z) & \textit{wenn } z \neq y \\ a \cdot \mu(x) + b & \textit{sonst} \end{cases}.$$

- *Ist schließlich $\psi \in \mathcal{A}(A)$ beliebig, so existieren nach Definition 4.1.6 endlich viele Zuordnungen der Form $\psi_i = \left(y_i \to a_i \cdot x_i + b_i\right)$ $(i = 1, \dots n)$ derart, dass $\psi = \bigwedge_{i=1}^{n} \psi_i$.*

In diesem Fall wird für $z \in A$

$$[\psi]\mu(z) := \mathbf{1}_{A \setminus \bigcup_{i=1}^{n} \{y_i\}}(z)\mu(z) + \sum_{k=1}^{n} \mathbf{1}_{\{y_k\}}(z) \left(a_k \cdot \left[\bigwedge_{m=1}^{k-1} \psi_m \right] \mu\left(x_k\right) + b_k \right)$$

definiert.

Die nachfolgende Bemerkung zeigt, wie ein Reset schnell berechnet werden kann.

Bemerkung 7 (Berechnung eines Resets)
Sei A ein Alphabet. Nach Definition 4.1.7 hat ein Reset stets die Form $\psi = \bigwedge_{k=1}^{m}(y_k \to 0 \cdot x_k)$ mit $x_k,\ y_k \in A$ für alle $k \in [m]$. Seien nun $\mu \in \mathbb{Z}^A$ und $z \in A$ beliebig. Man sieht

$$\begin{aligned}
[\psi]\,\mu(z) &= \mathbf{1}_{A \setminus \bigcup_{k=1}^{m} \{y_k\}}(z)\mu(z) + \sum_{k=1}^{m} \mathbf{1}_{\{y_k\}}(z) \cdot \left(0 \cdot \left[\bigwedge_{i=1}^{k-1} \left(y_i \to 0 \cdot x_i\right) \right] \mu\left(x_k\right) \right) \\
&= \mathbf{1}_{A \setminus \bigcup_{k=1}^{m} \{y_k\}}(z)\mu(z) = \begin{cases} \mu(z) & \text{wenn } z \notin \{y_k \,:\, k = 1, \dots m\} \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

Bemerkung 8
Bemerkung 7 zeigt, dass die induzierte Abbildung von $\psi := (x \to 0 \cdot y)$ unabhängig ist von y. Daher schreibt man kurz $\psi := (x \to 0)$. Außerdem zeigt die Bemerkung, dass die induzierte Abbildung unabhängig ist von der Reihenfolge, in der ψ definiert wurde ($\psi := (x \to 0) \wedge (y \to 0)$ vs. $\tilde{\psi} := (y \to 0) \wedge (x \to 0)$). Um die Notation weiter zu verkürzen schreibt man daher für $X = \left\{x_1, \dots x_n\right\} \subseteq A$ auch $\psi := (X \to 0)$ anstelle von $\psi := \left(x_1 \to 0\right) \wedge \dots \wedge \left(x_n \to 0\right)$. Unter Beachtung von Bemerkung 7 ist also

$$[X \to 0]\mu(z) = \begin{cases} \mu(z) & \text{wenn } x \notin X \\ 0 & \text{sonst} \end{cases}.$$

Insbesondere stimmt dieses allgemeinere Konstrukt aus Definition 4.1.7 mit dem speziellen Reset-Konstrukt aus (Alur und Dill, 1994) überein.

Lemma 4.1.1 (Vereinfachte Berechnung der induzierten Abbildung)
Sei A ein Alphabet, $\mu \in \mathbb{Z}^A$ eine Abbildung und $\psi = \bigwedge_{k=1}^{n} \left(y_k \to a_k x_k + b_k\right) \in \mathcal{A}(A)$ eine Zuordnung. Ist $z \in A \setminus \bigcup_{k=1}^{n} \left\{y_k\right\}$, so ist $[\psi]\,\mu(z) = \mu(z)$.

`Beweis`
Ist $z \in A \setminus \bigcup_{k=1}^{n} \left\{y_k\right\}$, so ist

$$\begin{aligned}
[\psi]\mu(z) &= \underbrace{\mathbf{1}_{A \setminus \bigcup_{i=1}^{n} \{y_i\}}(z)}_{=1}\,\mu(z) + \sum_{k=1}^{n} \underbrace{\mathbf{1}_{\{y_k\}}(z)}_{=0} \left(a_k \cdot \left[\bigwedge_{m=1}^{k-1} \psi_m \right] \mu\left(x_k\right) + b_k \right) \\
&= \mu(z),
\end{aligned}$$

womit die Behauptung gezeigt ist.

Beispiel 4.1.3 (Berechnung einer induzierten Abbildung)
Sei $A = \{x, y, z\}$ und $\mu \in \mathbb{Z}^A$ mit $\mu(x) = 1$, $\mu(y) = 2$, $\mu(z) = 3$. In diesem Beispiel wird die affine Zuordnung $\psi := (x \to 3 \cdot y) \wedge (y \to 2 \cdot x + 1)$ betrachtet und beispielhaft $[\psi]\mu$ berechnet.
Wegen Lemma 4.1.1 ist

$$[\psi]\mu(z) = \mu(z) = 3,$$

da z in keinem Ausdruck von ψ vorkommt. Weiter ist

$$\begin{aligned}
[\psi]\mu(x) &= \underbrace{\mathbf{1}_{A\setminus\{x,y\}}(x)}_{=0}\,\mu(x) + \underbrace{\mathbf{1}_{\{x\}}(x)}_{=1} \cdot \left(3 \cdot \underbrace{[\epsilon]\mu(y)}_{=\mu(y)} \right) + \underbrace{\mathbf{1}_{\{y\}}(x)}_{=0} \cdot (2 \cdot [x \to 3 \cdot y]\mu(x) + 1) \\
&= 3 \cdot \mu(y) \\
&= 6.
\end{aligned}$$

Zuletzt ist

$$\begin{aligned}
[\psi]\mu(y) &= \underbrace{\mathbf{1}_{A\setminus\{x,y\}}(y)}_{=0}\,\mu(y) + \underbrace{\mathbf{1}_{\{x\}}(y)}_{=0} \cdot (3 \cdot [\epsilon]\mu(y)) + \underbrace{\mathbf{1}_{\{y\}}(y)}_{=1} \cdot (2 \cdot [x \to 3 \cdot y]\mu(x) + 1) \\
&= 2 \cdot \underbrace{[x \to 3 \cdot y]\mu(x)}_{=3\cdot\mu(y)} + 1 \\
&= 2 \cdot (3 \cdot \mu(y)) + 1) \\
&= 2 \cdot (3 \cdot 2) + 1 = 13,
\end{aligned}$$

womit die Abbildung $[\psi]\mu$ vollständig spezifiziert ist.

Bemerkung 9
Aus obigem Beispiel wird deutlich, dass die induzierte Abbildung exakt die sequentielle Zuordnung von Variablenwerten formalisiert. In Matlab-Code entspricht obige Prozedur der Anweisung

$$x = 1;\ y = 2;\ z = 3;\ x = 3 * y;\ y = 2 * x + 1;$$

Insbesondere wird dadurch deutlich, dass die Reihenfolge der Zuordnung durchaus Einfluss auf die induzierte Abbildung hat.

Im Bereich der TEFAs wird später der Begriff der *Broadcast Synchronisation* eingeführt, in dem mehrere Automaten gleichzeitig Transitionen vornehmen und dabei Daten manipulieren, die möglicherweise Einfluss auf das Verhalten einzelner TEFAs haben. Bemerkung 9 zeigt, dass die Reihenfolge der Variablenbelegung Einfluss auf das Ergebnis haben kann. Im Folgenden wird daher ein Kriterium, welches das deterministische Verhalten gekoppelter Automaten bei einer Broadcast Synchronisation sicherstellt. Dazu benötigt man zunächst den Begriff der Permutation, der in mathematischen Standard-Lehrbüchern nachzulesen ist.

Definition 4.1.8 (Permutation)
Sei A ein Alphabet. Eine Permutation *auf A ist eine bijektive Abbildung $\rho : A \to A$. Die Menge aller Permutationen auf A bezeichnet man mit A_σ.*

Lemma 4.1.2
Sei D ein Alphabet, $\mu \in \mathbb{Z}^D$ eine Abbildung und $\psi = \bigwedge_{k=1}^{n} \left(y_k \to a_k x_k + b_k\right) \in \mathcal{A}(D)$ so, dass $x_k \neq y_l$ für alle $k, l \in [n]$. Dann ist für jede Permutation $\rho \in \{1, \ldots k\}_\sigma$

$$[\psi]\mu = \left[\bigwedge_{k=1}^{n} \left(y_{\rho(k)} \to a_{\rho(k)} x_{\rho(k)} + b_{\rho(k)}\right)\right] \mu$$

und damit $[\psi]\mu$ unabhängig von der Reihenfolge der Zuordnungen $\left(y_k \to a_k \cdot x_k + b_k\right)$.

`Beweis`
Sei $z \in D$, $k \in \mathbb{N}$, $\rho \in [k]_\sigma$ und $\psi_\rho := \bigwedge_{k=1}^{n} \left(y_{\rho(k)} \to a_{\rho(k)} x_{\rho(k)} + b_{\rho(k)}\right)$. Zu zeigen ist, dass $\left[\psi_\rho\right] \mu = [\psi]\, \mu$. Ist $z \in D \backslash \bigcup_{k=1}^{m} \left\{y_{\rho(k)}\right\} = D \backslash \bigcup_{k=1}^{m} \left\{y_k\right\}$, so ist nach Lemma 4.1.1

$$\left[\psi_\rho\right] \mu(z) = \mathbf{1}_{D \backslash \bigcup_{k=1}^{m} \{y_{\rho(k)}\}}(z) \mu(z) = \mu(z) = [\psi]\, \mu(z).$$

Ist hingegen $z \in \bigcup_{k=1}^{m} \left\{y_{\rho(k)}\right\}$, so ist

$$\left[\psi_\rho\right] \mu(z) = \sum_{k=1}^{m} \mathbf{1}_{\left\{y_{\rho(k)}\right\}}(z) \cdot \left(a_{\rho(k)} \cdot \left[\bigwedge_{i=1}^{k-1} \left(y_{\rho(i)} \to a_{\rho(i)} \cdot x_{\rho(i)}\right)\right] \mu\left(x_{\rho(k)}\right)\right).$$

Da nun für alle $k \in [m]$ nach Voraussetzung $x_{\rho(k)} \notin \bigcup_{i=1}^{k-1} \left\{y_{\rho(k)}\right\}$, ist wieder nach Lemma 4.1.1

$$\left[\bigwedge_{i=1}^{k-1} \left(y_{\rho(i)} \to a_{\rho(i)} \cdot x_{\rho(i)}\right)\right] \mu\left(x_{\rho(k)}\right) = \mu\left(x_{\rho(k)}\right),$$

und daher

$$\left[\psi_\rho\right] \mu(z) = \sum_{k=1}^{m} \mathbf{1}_{\left\{y_{\rho(k)}\right\}}(z) \cdot \left(a_{\rho(k)} \cdot \mu\left(x_{\rho(k)}\right)\right) = \sum_{k=1}^{m} \mathbf{1}_{\left\{y_k\right\}}(z) \cdot \left(a_k \cdot \mu\left(x_k\right)\right) = [\psi]\, \mu(z),$$

wobei sich die vorletzte Gleichung aus der Kommutativität der Addition ganzer Zahlen ableitet und die letzte Gleichung wie zuvor mit Lemma 4.1.1 gezeigt werden kann.

`q.e.d.`

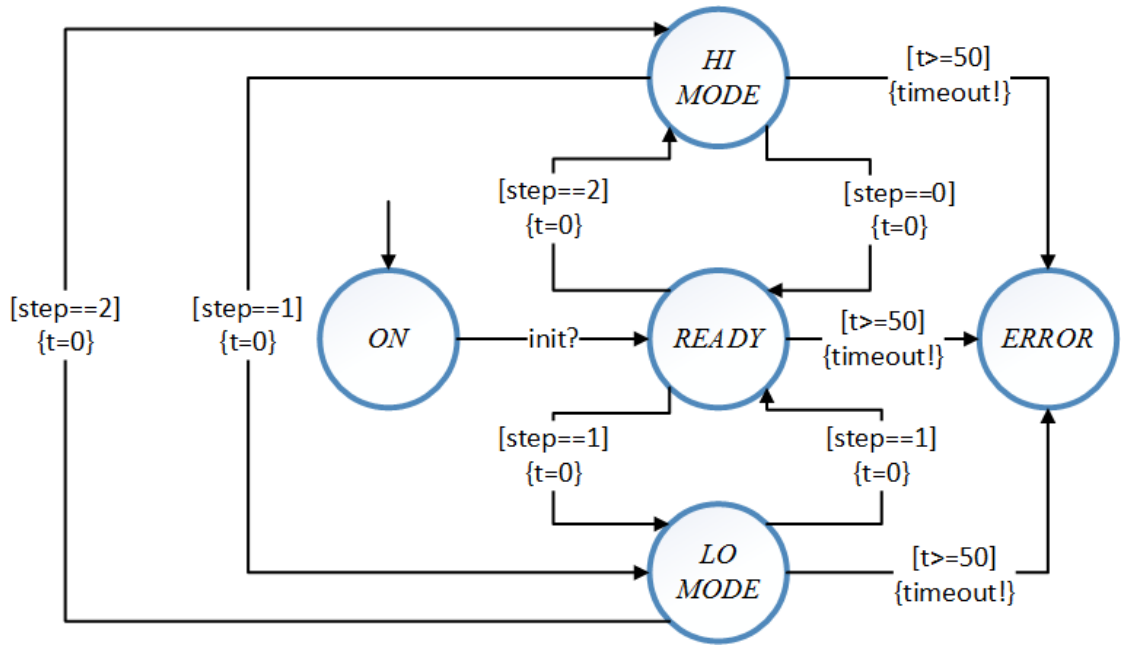


Abbildung 17 Modell einer Saug-/Spülpumpe als TEFA. Die grauen Kästen definieren die Locations. *ON* ist die initiale Location. Das Alphabet besteht aus den Signalen *init* und *timeout*. Die Variable *step* nimmt die Werte 0,1 oder 2 an. Je nach Wert wechselt der TEFA in eine der Locations *HI MODE*, *READY* oder *LO MODE*. Eine Uhr *t* wird bei jeder Zustandstransition zurückgesetzt. Wird der Wert länger als 50 ms nicht aktualisiert, geht der TEFA in die Location *ERROR* über.

4.1.3. Timed Extended Finite Automata

Kombiniert man vorige Konstrukte mit den Definitionen von *Timed Automata* aus (Alur und Dill, 1994) und (Bengtsson und Yi, 2004), so ist es nun möglich, die Klasse der Automaten zu definieren, die in dieser Arbeit zur Modellierung von *Modulen* verwendet werden.

Definition 4.1.9 (Timed Extended Finite Automaton (TEFA))
Ein TEFA $\mathcal{T}$ *ist ein Tupel* $\mathcal{T} = (S, s_0, C, V, A, \tau, v_0, I, \Delta)$*. Dabei ist* S *eine nichtleere, endliche Menge (genannt:* `Locations`*),* $s_0 \in S$ *die* `Initial Location`*,* C *die endliche Menge der* `Uhren`*,* V *die endliche Menge der* `Variablen` *mit* $C \cap V = \emptyset$*,* A *ein* `Alphabet`*,* $\tau \in A$ *die* `silent action` *und* $v_0 \in \mathbb{Z}^V$ *die Funktion der* `Initialwerte`*. Die Funktion* $I : S \to \Phi(C)$ *ordnet Locations Bedingungen (genannt:* `Invarianten`*) an Uhren zu. Die Relation*

$$\Delta \subseteq S \times \Phi(C \cup V) \times A \times \mathcal{A}_0(C) \times \mathcal{A}(V) \times S$$

ist eine Menge an Kanten zwischen Locations mit einer Bedingung an die Uhren und Variablen aus $\Phi(C \cup V)$ *(genannt:* `guard`*), einem auslösenden* `Signal` *aus* A*, eines Uhren-Resets, sowie einer Zuordnung von Variablenwerten aus* V*.*

Ein Beispiel-*TEFA* findet sich in Abb. 17.

Bemerkung 10 (CSP-Notation, vgl. (Brookes et al., 1984))
TEFAs werden häufig im Zusammenhang mit Ein-/Ausgabesystemen verwendet. Wie in Abbildung 17 zu sehen, teilt man dazu das Alphabet $A = A! \dot\cup A?$ *disjunkt in* Ausgabesignale *aus* $A!$ *(actions) und* Eingabesignale *aus* $A?$ (co-actions*). Wo nötig, wird CSP-Notation genutzt, um anzudeuten, ob ein Signal* a *als action (**a!**) oder co-action (**a?**) zu interpretieren ist.*

Die nun folgende Definition (aufbauend auf (Baier et al., 2008f)) überführt die abstrakte Definition von Uhren und Variablen als Elemente beliebiger Mengen zu reellen *Zeitpunkten* und *Werten*. Sie ist nötig, um das *Verstreichen von Zeit* zu modellieren und ist von essentieller Bedeutung für das Verständnis der Semantik von TEFAs (siehe Definition 4.1.11).

Definition 4.1.10 (Zeitpunkt, Wertzuweisung, Realisierung)
Sei $\mathcal{T}$ ein TEFA mit den Uhren $C = \{c_1, \dots c_n\}$ und Variablen $V = \{v_1, \dots v_n\}$. Ein `Zeitpunkt` *$u \in [0, \infty)^C$ ist eine Abbildung, die jeder Uhr $c \in C$ eine nicht-negative reelle Zahl zuordnet. Eine* `Wertzuweisung` *$w \in \mathbb{Z}^D$ ist eine Abbildung, die jeder Variable eine ganze Zahl zuweist. Das Tupel **MAP** $= (u, w) : C \times V \to [0, \infty) \times \mathbb{Z}$ heißt* Realisierung.
Für einen Zeitpunkt $u \in [0, \infty)^C$ und ein $d \in [0, \infty)$ wird der Zeitpunkt $u + d$ punktweise durch

$$[u + d](c) = u(c) + d$$

definiert. Dies modelliert das Verstreichen von d Zeiteinheiten. In der gesamten Arbeit wird $u_0 \equiv 0$ als der Zeitpunkt 0 verwendet.

Definition 4.1.11 (Semantik des TEFA, aufbauend auf (Baier et al., 2008f))
Sei $\mathcal{T} = (S, s_0, C, V, A, \tau, v_0, I, \Delta)$ ein TEFA. Die Semantik entspricht der eines Transitionssystems $\langle T, t_0, ([0, \infty) \cup A), \to \rangle$ (siehe Definition 4.1.5), wobei die Zustände $T = S \times [0, \infty)^C \times \mathbb{Z}^V$ Tupel aus Locations, Zeitpunkten und Wertzuweisungen sind und $t_0 = (s_0, u_0, v_0)$ den Anfangszustand beschreibt. Die Transitionsrelation $\to \subseteq T \times ([0, \infty) \cup A) \times T$ hat folgende Eigenschaften:

- *Für $d \in [0, \infty)$ gilt $(l, u, w) \xrightarrow{d} (l, u + d, w) \Leftrightarrow \forall \tilde{d} \in [0, d] : u + \tilde{d} \vDash I(l)$.*
- *Für $a \in A$ gilt $(l, u, w) \xrightarrow{a} (\tilde{l}, \tilde{u}, \tilde{w}) \Leftrightarrow \exists \delta := (l, G, a, \varphi, \psi, \tilde{l}) \in \Delta$ mit **MAP** $\vDash G$, $\tilde{u} = [\varphi]u$, $\tilde{w} = [\psi]w$ und $\tilde{u} \vDash I(\tilde{l})$,*

wobei bei letzterem Punkt auch $a = \tau$ erlaubt ist.

Definition 4.1.12 (Pfad, Erreichbarkeit, aufbauend auf (Baier et al., 2008b))
Man schreibt $(l, u, w) \xrightarrow{*} (\tilde{l}, \tilde{u}, \tilde{w})$, wenn es ein $\rho \in \to$ gibt mit $\rho_1 = (l, u, w)$ und $\rho_3 = (\tilde{l}, \tilde{u}, \tilde{w})$. Dies entspricht der Vorstellung, dass der Zustand $(\tilde{l}, \tilde{u}, \tilde{w})$ vom Zustand (l, u, w) aus in einem Schritt erreichbar ist. Weiter bezeichnet für $n \in \mathbb{N}_0 \cup \{\infty\}$ die Menge

$$[[\mathcal{T}]]_n := \left\{ t \in T^n : \text{Für alle } k \in \{1, \dots, n-1\} : t_k \xrightarrow{*} t_{k+1} \right\}$$ *die* Pfade der Länge n

und

$$[[\mathcal{T}]] := \bigcup_{k \in \mathbb{N}_0} [[\mathcal{T}_k]]$$ *die Pfade endlicher Länge.*

Zuletzt bezeichnet für einen Pfad $t \in [[\mathcal{T}]]$ der Länge $n_0 \in \mathbb{N}_0$ das Konstrukt

$$S_t := \{ t_k : k \in \{1, \dots, n_0\} \}$$

die Menge der in ihm enthaltenen Zustände.

4.1.4. Kommunizierende Automaten

Automaten können formal miteinander Informationen austauschen und simultane Zustandsübergänge realisieren. In den nun folgenden beiden Definitionen wird zwischen zwei Arten der Synchronisation unterschieden: *Binäre Synchronisation*, in der je zwei Netzteilnehmer sich synchronisieren und Broadcast Synchronisation, in der ein Netzteilnehmer ein synchronisierendes Signal schickt und *alle* Netzteilnehmer, die dieses verarbeiten können sich darauf synchronisieren. Die folgende Definition ist angelehnt an die in (Milner, 1980) eingeführte *parallele Komposition* und wird in der gängigen Literatur als *Interleaving* bezeichnet. Sie modelliert alle möglichen Reihenfolgen, in denen nebenläufige, kommunizierende Prozesse ablaufen können.

Definition 4.1.13 (Interleaving, aufbauend auf (Baier et al., 2008f))
Sei $\left(\mathcal{T}_i = (S_i, s_0^{(i)}, C_i, V_i, A_i, \tau, v_0^{(i)}, I_i, \Delta_i)\right)_{i=1}^{n}$ *eine endliche Folge von TEFAs. Sei weiter* $S := \prod_{i=1}^{n} S_i$, $\vec{s}_0 := \left(s_0^{(1)}, \ldots, s_0^{(n)}\right)$, $C := \bigcup_{i=1}^{n} C_i$, $V := \bigcup_{i=1}^{n} V_i$, $A := \bigcup_{i=1}^{n} A_i$, $v_0 := \sum_{j=1}^{n} \mathbb{1}_{V_j} \cdot v_0^{(j)} \cdot \left(\sum_{k=1}^{n} \mathbb{1}_{V_k}\right)^{-1}$ *und* $I := \bigwedge_{i=1}^{n} I_i$.
Der Interleaving Automat $\mathcal{T}_1 \parallel \mathcal{T}_2 \parallel \ldots \parallel \mathcal{T}_n$ *der* $\left(\mathcal{T}_i\right)_{i=1,\ldots n}$ *ist definiert als ein Transitionssystem* $\langle T, t_0, \rightarrow \rangle$, *wobei* $T = S \times [0,\infty)^C \times \mathbb{Z}^V$ *die Menge der Zustände und* $t_0 = \left(\vec{s}_0, u_0, v_0\right)$ *den Anfangszustand beschreibt. Die Relation* $\rightarrow \subseteq T \times ([0,\infty) \cup A) \times T$ *erfüllt:*

1. *Für* $d \in [0,\infty)$ *gilt* $(l,u,w) \xrightarrow{d} (l, u+d, w) \Leftrightarrow \forall \tilde{d} \in [0,d] : u + \tilde{d} \vDash I(l)$.

2. *Für* $k \in [n]$ *und* $a \in A$ *gilt* $(l, u, w) \xrightarrow{a} \left(l\left[\tilde{l}_k / l_k\right], \tilde{u}, \tilde{w}\right) \Leftrightarrow \exists \delta := \left(l_k, G, a, \varphi, \psi, \tilde{l_k}\right) \in \Delta_k$ *mit*
   $\boldsymbol{MAP}|_{C_k \times V_k} \vDash G$, $\tilde{u} = [\varphi]u$, $\tilde{u} \vDash I(\tilde{l}_k)$ *und* $\tilde{w} = [\psi]w$.

3. *Es gilt* $(l,u,w) \xrightarrow{\tau} \left(l\left[\tilde{l}_i / l_i\right]\left[\tilde{l}_j / l_j\right], \tilde{u}, \tilde{w}\right)$ *für alle* $i, j \in [n]$, $i \neq j$ *mit* $\left(l_i, G_i, a!, \varphi_i, \psi_i, \tilde{l}_i\right) \in \Delta_i$, $\left(l_j, G_j, a?, \varphi_j, \psi_j, \tilde{l}_j\right) \in \Delta_j$ *so, dass*
   $\boldsymbol{MAP}|_{C_k \times V_k} \vDash G_k$ *für* $k \in \{i,j\}$, $\tilde{u} = \left[\varphi_i \wedge \varphi_j\right] u$, $\tilde{u} \vDash I\left(l\left[\tilde{l}_i / l_i\right]\left[\tilde{l}_j / l_j\right]\right)$ *und* $\tilde{w} = \left[\psi_i \wedge \psi_j\right] w$.

Der erste Unterpunkt obiger Definition modelliert das *Verstreichen von* d *Zeiteinheiten*. Der zweite Unterpunkt zeigt, dass Interleaving sämtliche Zustandsübergänge der Teilautomaten $\mathcal{T}_i$ erlaubt und *zusätzlich* gepaarte Zustandsübergänge je *zweier* Teilautomaten (siehe dritter Unterpunkt). Die gepaarten Zustandsübergänge bezeichnet man auch als *binäre Synchronisation* im Gegensatz zu der in Definition 4.1.14 eingeführten *Broadcast Synchronisation* (Behrmann et al., 2004). Man beachte beim dritten Unterpunkt die subtile Definition von $\tilde{w}$: Obschon der Zustandsübergang zweier Teilautomaten *synchron* ist, findet die Belegung der Variablen sequentiell statt. Derjenige Teilautomat, der eine action auslöst, definiert den Wert der Variable. Derjenige Teilautomat, der eine *co-action* ausführt, übernimmt den so definierten Wert und belegt davon abhängig weitere Variablen (siehe hierzu Bemerkung 9).
Da die Funktion v_0 zur Ermittlung der Anfangswerte offenbar über alle Anfangswerte mittelt, ist darauf zu achten, dass die Variablenmengen disjunkt oder die Zuordnungen identisch sind.

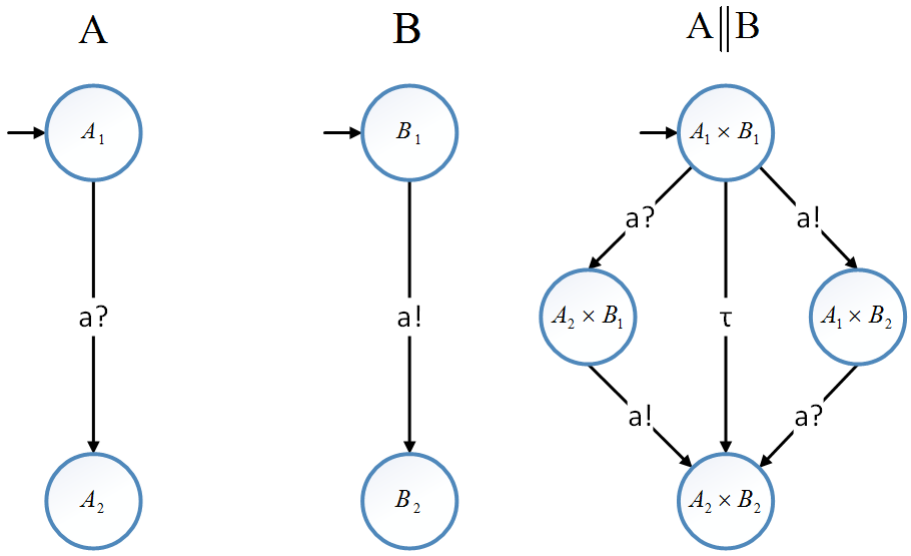


Abbildung 18 Zwei TEFAs A und B und deren durch Interleaving erzeugter Automat $A \parallel B$. Aus Gründen der Übersichtlichkeit wurden keine Uhren und Daten eingeführt. Die Locations in $A \parallel B$ ergeben sich aus dem kartesischen Produkt der jeweiligen Locations von A und B. Alle Transitionen der TEFAs A und B bleiben erhalten. Das Signal a erzeugt zudem eine *binäre Synchronisation* von A und B. Dabei wird eine *silent Transition* von $A_1 \times B_1$ zu $A_2 \times B_2$ genommen, die mit τ markiert ist.

Definition 4.1.14 (Broadcast Synchronisation, aufbauend auf (Behrmann et al., 2004))
Seien wieder $\mathfrak{T}_i = (S_i, s_0^{(i)}, C_i, A_i, \tau, V_i, I_i, \Delta_i)\ (i = 1, \ldots n)$ *TEFAs und* $S, V, C, A, \vec{s}_0, v_0$ *und* I *wie in Definition 4.1.13. Sei* $\mathbb{B} \subseteq A$ *eine Menge an* `Broadcast Signalen`. *Bei der Broadcast Synchronisation* $\left(\mathcal{T}_1 \parallel \mathcal{T}_2 \parallel \ldots \parallel \mathcal{T}_n\right)_{\mathbb{B}}$ *erfüllt die Transitionsrelation* Δ *Folgendes:*

1. *Für* $d \in [0, \infty)$ *gilt* $(l, u, w) \xrightarrow{d} (l, u + d, w) \Leftrightarrow \forall \tilde{d} \in [0, d] : u + \tilde{d} \vDash I(l)$.
2. *Für* $k \in [n]$ *und* $a \in A$ *gilt* $(l, u, w) \xrightarrow{a} \left(l\left[\tilde{l}_k / l_k\right], \tilde{u}, \tilde{w}\right) \Leftrightarrow \exists \delta := \left(l_k, G, a, \varphi, \psi, \tilde{l}_k\right) \in \Delta_k$ *mit*
   $\boldsymbol{MAP}|_{C_k \times V_k} \vDash G$, $\tilde{u} = [\varphi]u$, $\tilde{u} \vDash I\left(\tilde{l}_k\right)$ *und* $\tilde{w} = [\psi]w$.
3. *Für* $a \in A \backslash \mathbb{B}$ *gilt* $(l, u, w) \xrightarrow{\tau} \left(l\left[\tilde{l}_i / l_i\right]\left[\tilde{l}_j / l_j\right], \tilde{u}, \tilde{w}\right)$ *für alle* $i, j \in [n]$, $i \neq j$ *mit* $\left(l_i, G_i, a!, \varphi_i, \psi_i, \tilde{l}_i\right) \in \Delta_i, \left(l_j, G_j, a?, \varphi_j, \psi_j, \tilde{l}_j\right) \in \Delta_j$ *so, dass*
   $\boldsymbol{MAP}|_{C_k \times V_k} \vDash G_k$ *für* $k \in \{i, j\}$, $\tilde{u} = \left[\varphi_i \wedge \varphi_j\right] u$, $\tilde{u} \vDash I\left(l\left[\tilde{l}_i / l_i\right]\left[\tilde{l}_j / l_j\right]\right)$ *und* $\tilde{w} = \left[\psi_i \wedge \psi_j\right] w$.
4. *Es gilt* $(l, u, w) \xrightarrow{\tau} \left(\hat{l}, \hat{u}, \hat{w}\right)$ *gdw*
   a. $\exists i \in [n], b \in \mathbb{B} : \left(l_i, G_i, b!, \varphi_i, \psi_i, \hat{l}_i\right) \in \Delta_i$ *mit* $u|_{C_i \times V_i} \vDash G_i$ *und*
   b. $\forall j \in \Gamma := \left\{k \in [n] \backslash \{i\} \text{ mit } \left(l_k, u|_{C_k}, w|_{V_k}\right) \neq \left(\hat{l}_k, \hat{u}|_{C_k}, \hat{w}|_{V_k}\right)\right\}$ *gilt:*
      $$\exists \delta := \left(l_j, G_j, b?, \varphi_j, \psi_j, \hat{l}_j\right) \in \Delta_j \text{ mit } \boldsymbol{MAP}|_{C_j \times V_j} \vDash G_j$$
   c. $\forall j \notin \Gamma$ *gilt:* $\nexists \delta := \left(l_j, G_j, b?, \varphi_j, \psi_j, \hat{l}_j\right) \in \Delta_j$ *mit* $\boldsymbol{MAP}|_{C_j \times V_j} \vDash G_j$ *und*
   d. $\hat{u} = \left[\varphi_i \wedge \bigwedge_{k \in \Gamma} \varphi_k\right] u$, $\hat{u} \vDash I\left(\hat{l}\right)$ *und* $\hat{w} = \left[\psi_i \wedge \bigwedge_{k \in \Gamma} \psi_k\right] w$.

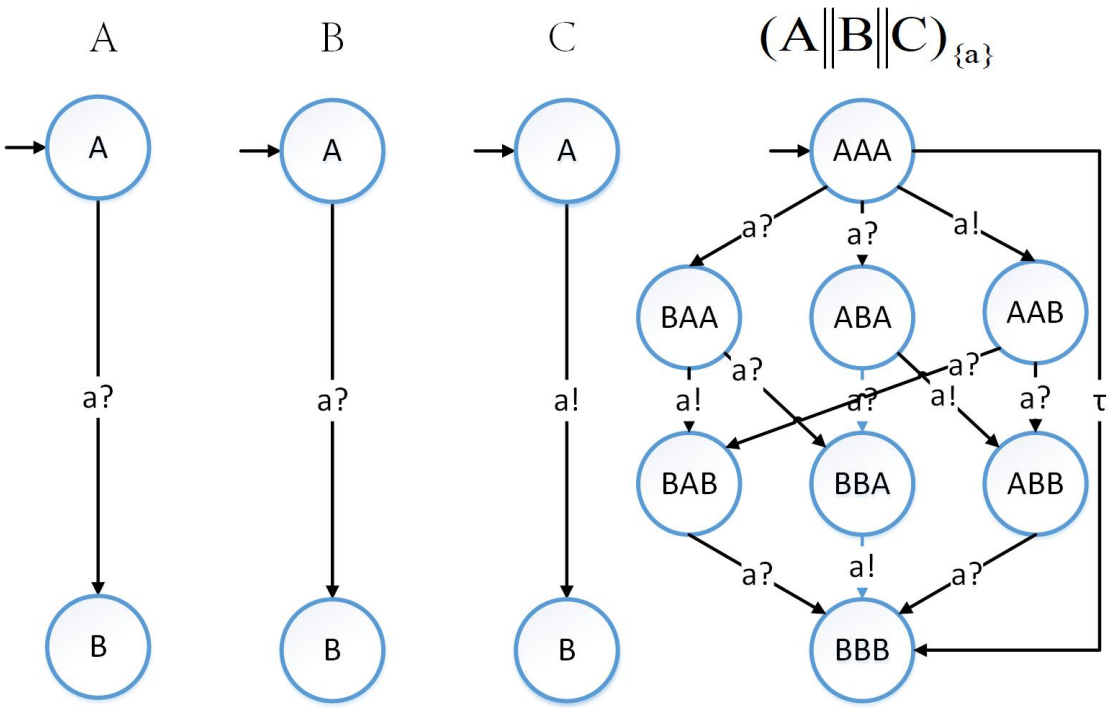


Abbildung 19 Drei TEFAs A, B und C und deren durch Broadcast Synchronisation erzeugter Automat $(A \parallel B \parallel C)_{\{a\}}$. Das *Broadcast Signal* ist a. Wieder ergeben sich die Locations durch das kartesische Produkt der ursprünglichen Locations. Über die Transitionen von A, B und $A \parallel B \parallel C$ hinaus erzeugt das *Broadcast Signal* eine Broadcast Synchronisation von A, B und C. Dabei wird eine *silent Transition* von $A_1 \times B_1 \times C_1$ zu $A_2 \times B_2 \times C_2$ genommen, die mit τ markiert ist.

Wieder modelliert der erste Unterpunkt das Verstreichen von d Zeiteinheiten und der zweite Transitionen der Teilautomaten. Der dritte Unterpunkt schließt Transitionen durch Interleaving (siehe Definition 4.1.13) für Signale ein, die nicht den Broadcast Synchronisation-Signalen zuzuordnen sind. Der letzte Unterpunkt beschreibt Folgendes:

a. Es gibt einen Auslöser der Synchronisation.

b. Eine Zustandsänderung eines Teilautomaten kann nur durch die ausgelöste Synchronisation hervorgerufen werden.

c. Jeder Teilautomat, der eine Zustandstransition durch die ausgelöste Synchronisation wahrnehmen kann, tut dies auch.

d. Uhren und Variablen werden entsprechend der Vorgaben der Teilautomaten aktualisiert. Man beachte, dass $\hat{u}$ wegen Bemerkung 7 von der Reihenfolge der $(\varphi_k)_{k\in\Gamma}$ unabhängig und deshalb wohldefiniert ist. Hingegen kann $\hat{w}$ je nach Belegung der Reihenfolge der $(\psi_k)_{k\in\Gamma}$ ganz unterschiedliche Werte zuordnen. Wie im Laufe der Arbeit aber deutlich werden wird, wird dieses Problem wegen Lemma 4.1.2 im Falle des Echtzeit-Bussystems nicht auftauchen.

Zur Systemspezifikation fehlt nun nur noch ein letzter Schritt. Möchte man modellieren, dass eine Transition *ausschließlich* durch binäre Synchronisation oder Broadcast Synchronisation genommen werden kann, so muss man aus vorigen Automaten sämtliche Pfade entfernen,

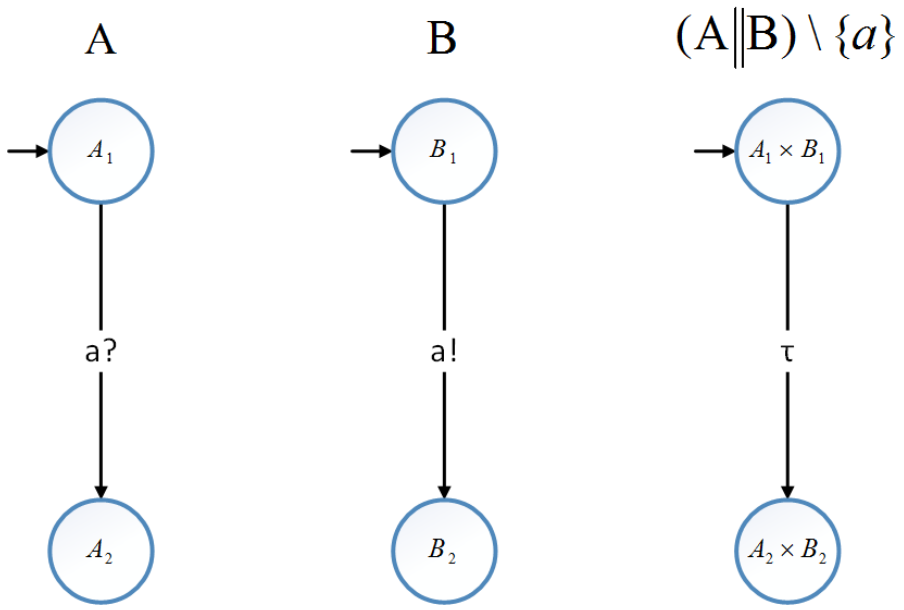


Abbildung 20 Zwei TEFAs A und B und der durch Interleaving und Hiding erzeugte Automat $(A \parallel B)\backslash\{a\}$. Diejenigen Transitionen der Automaten A und B, die das versteckte Signal a beinhalten, werden im Hiding-Automat gestrichen (vgl. Abb. 18). Übrig bleibt allein die binäre Synchronisation τ.

welche einen unabhängigen Zustandsübergang eines Teilautomaten auf ein synchronisierendes Signal hin ermöglichen. Dies geschieht in der nun folgenden Definition.

Definition 4.1.15 (Hiding, angelehnt an (Milner, 1980))
Sei $\mathcal{T} = (S, s_0, C, V, A, \tau, v_0, I, \Delta)$ *ein TEFA und* $\mathbb{H} \subseteq A$ *eine beliebige Teilmenge des Alphabets. Die Transitionsrelation* $\Delta^{(\mathbb{H})}$ *des TEFA* $\mathcal{T}\backslash\mathbb{H} := \left(S, s_0, C, A\backslash\mathbb{H}, \tau, \vec{d}_0, I, \Delta^{(\mathbb{H})}\right)$ *setzt sich wie folgt zusammen:*

$$\Delta^{(\mathbb{H})} := \left\{\delta \in \Delta : \delta_4 \notin \mathbb{H}\right\}.$$

Es werden also exakt diejenigen Transitionen aus Δ *gestrichen, welche ein Signal aus* $\mathbb{H}$ *beinhalten (siehe Abb. 20).*

Nun sind alle Voraussetzungen geschaffen, um die Module des Echtzeit-Bussystems formal zu beschreiben. Man beginnt hierzu mit der Beschreibung des Übertragungsprotokolls.

4.2. Beschreibung des Übertragungsprotokolls

Wie in der Einführung zu Kapitel 4 beschrieben, baut das Echtzeit-Bussystem auf dem IEEE 802.3 Ethernet-konformen *Ethernet-Powerlink* (EPL)-Standard auf. EPL bedient sich des *Time Division Multiple Access* (TDMA)-Prinzips, das Kollisionen in der Datenübertragung verhindert, indem jeder Netzteilnehmer nur zu dedizierten Zeitpunkten Daten übertragen darf. Dabei fordert ein einziger *Managing Node* (MN) – meist ein Industrie-PC – sukzessive die Daten jedes einzelnen Netzteilnehmers (sogenannte *Controlled Nodes* (CNs)) mithilfe von *Poll Request* (preq)-Nachrichten an (*Polling*). Die preq-Nachrichten enthalten darüber hinaus die für den jeweiligen CN vorgesehenen Daten (*Payload*). Jeder CN übermittelt wiederum seine Daten mit der *Poll Response* (pres)-Nachricht an den MN.
Die Kommunikation im Echtzeit-Bussystem folgt gleichbleibenden *Zyklen*: In der ersten ***Start of Cycle*-Phase** synchronisiert der MN in gleichbleibendem Takt die CNs mit einem *Start of Cycle* (SoC)-Signal; darauf folgt die ***isochrone* Phase**, in der der MN die CNs in gleichbleibender Reihenfolge zu gleichbleibenden Zeitpunkten *pollt* und jeder CN in pres-Nachrichten seine Daten an den MN übermittelt. Die letzte Phase ist die *asynchrone* Phase, in der nicht-zeitkritische Daten übermittelt werden (Ethernet POWERLINK Standardisation Group, 2013). Der Zyklus ist in Abb. 21 dargestellt. **Die asynchrone Phase ist nicht Teil der Modellierung dieser Arbeit.** Bereits jetzt lässt sich das System aus n EPL-Knoten formal beschreiben: Ist $\mathbb{PQ}_n := \{preq_i : i \in [n]\}$, $\mathbb{PS}_n := \{pres_j : j \in [n]\}$ und $\mathbb{P}_n := \mathbb{PQ}_n \cup \mathbb{PS}_n$, so gilt

$$System = \left(MN \parallel CN_1 \parallel CN_2 \parallel \ldots \parallel CN_n\right)_{\{soc\}} \setminus \left(\mathbb{P}_n \cup \{soc\}\right).$$

Es folgt nun die detaillierte Beschreibung der Elemente des EPL-Netzwerks. In Abb. 22 findet sich eine Übersicht der zu modellierenden Prozesse und deren Zusammenhang.

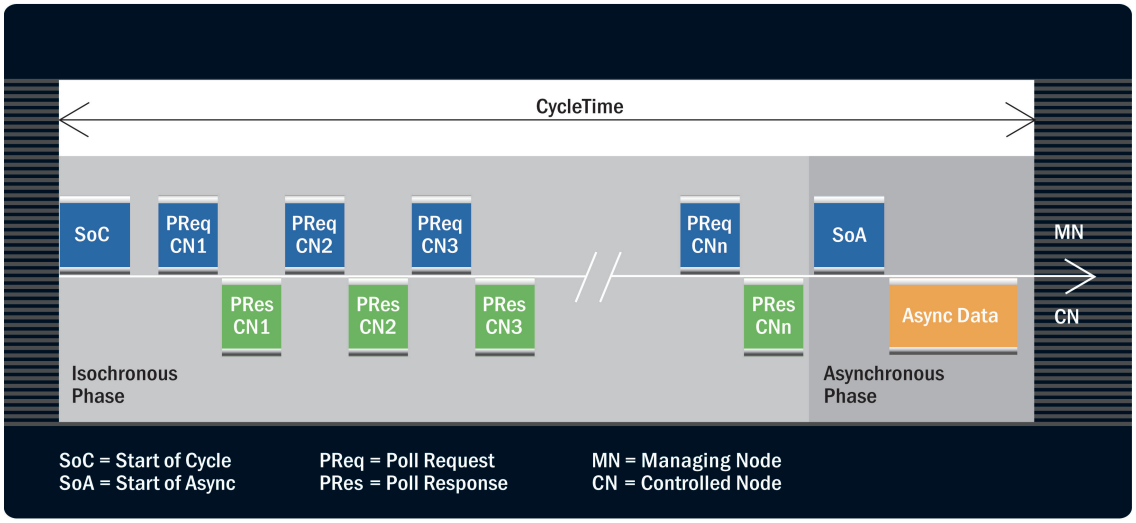


Abbildung 21 Ein EPL Zyklus beginnt mit der SoC-Phase, in der alle CNs synchronisiert werden. Danach wird über preq-Nachrichten sukzessive jeder CN gepollt. CNs übermitteln ihre Daten mit pres-Antworten an den MN. Zum Schluss des Zyklus folgt die asynchrone Phase, in der nicht-zeitkritische Daten ausgetauscht werden.
Quelle: Ethernet Powerlink Standardisation Group.

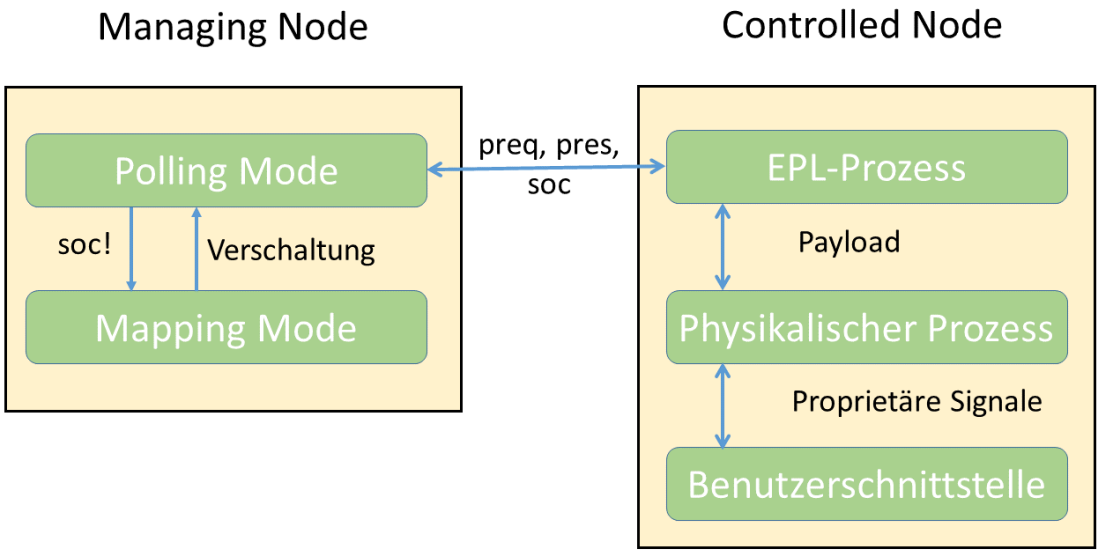


Abbildung 22 Übersicht über die formal zu beschreibenden Teilprozesse und ihre Interaktionen. Sowohl MN als auch alle CNs werden je durch ein System nebenläufiger Automaten beschrieben. MN und CNs interagieren über die preq-, pres- und soc-Signale.

4.2.1. Der Managing Node

Der *Managing Node* (MN) wird modelliert als ein System zweier nebenläufiger kommunizierender TEFAs: *Polling Mode* und *Mapping Mode*. Im Polling Mode wird die isochrone Phase modelliert. Der Mapping Mode modelliert hingegen den Prozess, der für die dynamische Zuordnung von Datenquellen und -senken im Echtzeitnetzwerk zuständig ist.

Modellierung des Polling Mode Der Polling Mode ist der Kernprozess des MN in der *isochronen* Phase. Um das Echtzeit-Bussystem dynamisch zu gestalten, ist der MN anders als in vielen Automatisierungsszenarien sehr einfach gehalten – seine Aufgabe besteht nur darin, sukzessive jeden *Controlled Node* (CN) mit *Poll Request* (preq)- Nachrichten zu pollen und dabei den jeweiligen Payload zu übertragen. Steuerungsaufgaben wie z.B. in (Witsch und Vogel-Heuser, 2011) nimmt der MN im Echtzeit-Bussystem nicht wahr. Besteht das Netzwerk aus dem MN und n CNs (z.B. $CN_1, \ldots, CN_n$), so wird der Polling Mode modelliert als ein TEFA mit $2n+1$ Locations (zwei Locations pro CN und eine *Start of Cycle* (SoC) Location).

Formale Spezifikation Jedem CN_i $(i = 1, \ldots n)$ werden im Polling Mode zwei Locations zugeordnet: Eine $Polled_i$-*Location* und eine $Respond_i$-Location. Der Prozess startet in der SoC-Location und pollt CN_1 mit der preq1-Nachricht (binäre Synchronisation). Kommt die entsprechende *Poll Response* (pres)-Nachricht zusammen mit dem Payload innerhalb eines vordefinierten Zeitslots (der Invarianten von $Polled_1$) beim MN an, schreitet der MN mit CN_2 voran und so fort. Am Ende des Zyklus synchronisiert der MN über die soc-Nachricht alle CNs (Broadcast Synchronisation). Bei der Datenübertragung im Echtzeit-Bussystem liegt

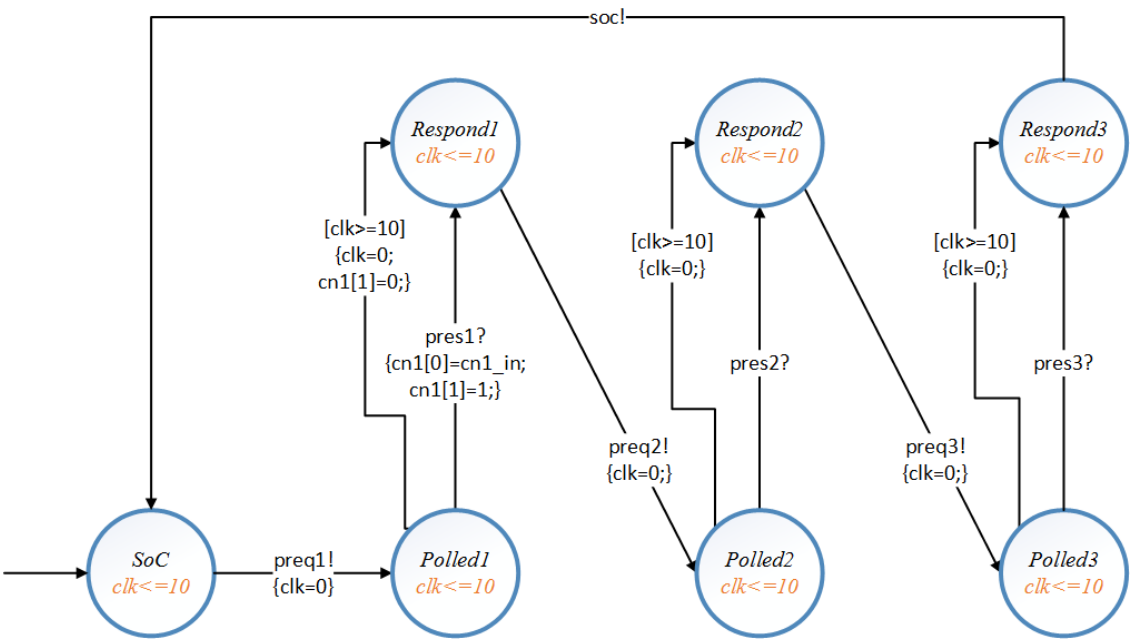


Abbildung 23 Beispielhafter Polling Mode des MN. Der Prozess beginnt in der Location SoC. Dort verharrt er maximal so lange, bis die Uhr *clk* auf 10 gezählt hat. Mit der preq1!-Nachricht wird der erste CN abgefragt und die Uhr zurückgesetzt. Der CN hat nun wiederum 10 Zeiteinheiten, um zu antworten. Geht die pres1?-Antwort bis dahin ein, so aktualisiert der MN den Payload von CN1 und setzt das Gültigkeits-Flag auf *true* ($cn1[1] = 1$). Andernfalls ($clk \geq 10$) wird der Payload nicht aktualisiert und das Gültigkeits-Flag auf *false* gesetzt. Der Prozess geht nun in die Location *Respond1* über und wiederholt das Vorgehen mit CN2. Zum Schluss des Zyklus informiert das Broadcast soc!-Signal alle CNs, dass ein neuer Zyklus beginnt.

eine Besonderheit vor: Neben jedem Datensatz schickt der MN stets ein dem Datensatz zugeordnetes *Gültigkeits-Flag* mit, das dem jeweiligen CN Aufschluss über die Gültigkeit des Datensatzes gibt. Üblicherweise verarbeitet der CN den Payload genau dann, wenn das Flag *true (1)* ist. Der Prozess ist in Abb. 23 dargestellt und in Tabelle 2 formalisiert. Der Mapping Mode wird im Abschnitt 4.4 (Beschreibung der Verschaltung) spezifiziert.

S	$\{SoC, Polled_1, Respond_1, \ldots Polled_n, Respond_n\}$
$\mathbf{s_0}$	SoC
C	$\{clk\}$
V	$\bigcup_{j=1}^{n} V_{CN_j}$
A	$\{soc!, preq1!, pres1!, \ldots preqn!, presn!\}$
$\mathbf{v_0}$	$\sum_{j=1}^{n} \mathbb{1}_{V_j} \cdot v_0^{(j)} \cdot \left(\sum_{k=1}^{n} \mathbb{1}_{V_k}\right)^{-1}$
I	$I : S \to \Phi(\{clk\}),\ x \mapsto (clk \leq 10)$

Tabelle 2 Tabellen-Notation des Polling Mode-Prozesses als TEFA. Die Anfangswertzuweisung v_0 mittelt bei jeder Variable über alle Anfangswerte, die der Variablen durch die beteiligten CNs zugeordnet wird. In der Praxis ist dies häufig nur ein CN, da die Variablenmengen überschneidungsfrei sind.

4.3. Beschreibung von Medizingeräten und Zubehör

Jedes Medizingerät (mitsamt Konnektor) wird als System nebenläufiger TEFAs modelliert. Dabei sind jeweils der *Physikalische Prozess* und der Ethernet-Powerlink *(EPL)*-Prozess zu modellieren.

4.3.1. EPL-Prozess

Der EPL-Prozess modelliert den Prozess des Datenaustauschs über EPL auf dem CN. Der Prozess unterscheidet sich von CN zu CN nur anhand des EPL-Index. Er folgt stets demselben Muster, das in Abb. 24 zu sehen ist.

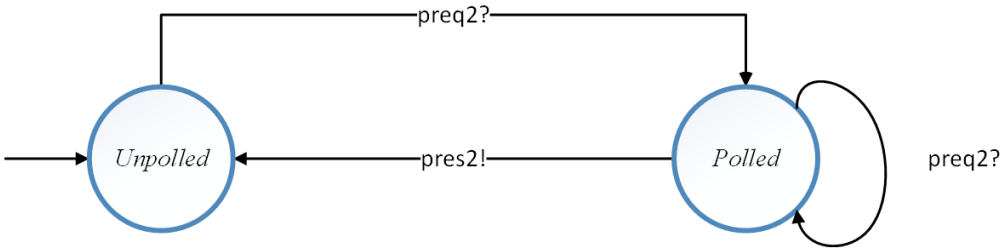


Abbildung 24 Der EPL-Prozess von CN Nummer 2 beginnt in der `Unpolled`-Location und wartet auf die preq-Anfrage. Trifft diese ein, so wechselt der EPL-Prozess in die Location `Polled`. Trifft eine erneute `preq`-Anfrage ein, wiederholt sich das Vorgehen. Andernfalls antwortet der Prozess in Form der `pres2!`-Antwort und beginnt von vorne.

Formale Spezifikation Der Prozess besitzt die beiden Locations *Unpolled* und *Polled* und startet in Ersterer. Auf die preq-Anfrage des MN (binäre Synchronisation) hin wechselt der Prozess in die Polled-Location und antwortet sodann mit der pres-Antwort. Sollte zwischen den beiden Schritten eine erneute preq-Anfrage des MN eintreffen, wiederholt der Prozess das Vorgehen aus dem ersten Schritt (siehe z.B. Abb. 24 und Tab. 3). In der Realität wird zusammen mit den pres- und preq-Signalen auch der Payload übermittelt. Da die am MiMed entwickelten Konnektoren jedoch den Payload erst mit dem soc!-Signal parsen (Synchronität), wird die Übermittlung erst in diesem Schritt modelliert (siehe z.B. Abb. 25).

S	{Polled, Unpolled}
s_0	Unpolled
C	$\emptyset$
V	$\emptyset$
A	$\{preq2?, pres2!\}$

Tabelle 3 Tabellenspezifikation des EPL-Prozesses am Beispiel des CN Nummer 2.

4.3.2. Physikalischer Prozess

Der physikalische Prozess modelliert das Verhalten des Medizingerätes, das durch den Nutzer erkennbar ist. Der Prozess wird vom Hersteller des Moduls modelliert und dokumentiert.

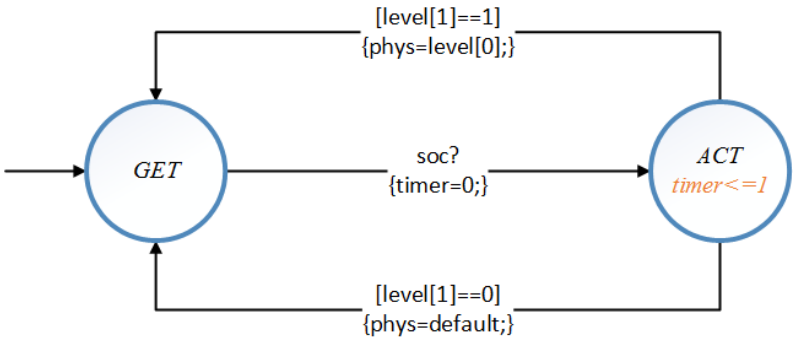


Abbildung 25 Der (stark vereinfachte) physikalische Prozess der Saug-/Spülpumpe beginnt in der Location `GET`. Sobald das `SoC`-Signal (Broadcast Synchronisation) eintrifft, wechselt der physikalische Prozess in die Location `ACT` und setzt die interne Uhr zurück (`timer=0`). Die Variable `level` wird zwischenzeitlich von einem anderen Prozess, z.B. dem Fußschalter, manipuliert. Der physikalische Prozess verweilt in den Location `ACT` höchstens eine Zeiteinheit (`timer<=1`) und untersucht, ob der eingetroffene Payload `level` *gültig* ist (`level[1]==1`) oder *ungültig* (`level[1]==0`). Abhängig davon realisiert die Pumpe die Soll-Intensität (`phys=level[0]`) oder setzt sie auf einen internen *default*-Wert wechselt zu `GET`.

Modellierung des physikalischen Prozesses Der Prozess wird als System nebenläufiger TEFAs modelliert. Er kommuniziert über einen *Shared Memory* mit dem EPL-Prozess. Der physikalische Prozess unterscheidet sich von Gerät zu Gerät und ist der Gestaltung des *Herstellers* überlassen. Ein Beispiel-Prozess findet sich in Abb. 25. Der Prozess wird als System nebenläufiger TEFAs modelliert. Er kommuniziert über einen Shared Memory mit dem EPL-Prozess. Der physikalische Prozess unterscheidet sich von Gerät zu Gerät und ist der Gestaltung des Herstellers überlassen. Ein Beispiel-Prozess findet sich in Abb. 25.

S	{GET, ACT}
$\mathbf{s_0}$	GET
C	$\{timer\}$
V	{level, phys, default}
A	$\{soc?\}$
$\mathbf{v_0}$	level, phys, default $\mapsto 0$
I	ACT $\mapsto (timer \leq 1)$

Tabelle 4 Tabellenspezifikation des Prozesses aus Abb. 25.

4.4. Beschreibung der Verschaltung

In den vorigen Kapiteln wurden Methoden vorgestellt, um einzelne Systembestandteile und deren synchrone Kommunikation formal zu modellieren. Bislang unbeachtet bleibt der Aspekt der Verschaltung, d.h. *welcher Netzteilnehmer welche Daten* manipuliert (siehe Kapitel 4.1.4). Dies ist Aufgabe des Mapping Mode, der Teil des MN ist und nebenläufig zum Polling Mode ausgeführt wird.

Definition 4.4.1 (Verschaltung)
Seien Q und S beliebige endliche Mengen. Eine Verschaltung $V = (Q, S, \Delta)$ *ist ein Tupel mit* $\Delta \subseteq Q \times S$*. Elemente der Menge* Q *werden als* Quellen *bezeichnet, Elemente der Menge* S *als* Senken.

Bei einer Verschaltung werden also Quellen ihren Senken zugeordnet. In dieser Arbeit wurde für die Zuordnung eine Benutzerschnittstelle implementiert, mit deren Hilfe Daten von TEFAs einander einfach zugeordnet werden können.
Die Verschaltung könnte auch anhand standardisierter Variablenbezeichner (z.B. nach den neuen semantischen Standards der IEEE 11073-Familie, siehe (Andersen et al., 2016; Kasparick et al., 2015b)) durch die Software automatisiert vorgeschlagen werden. Da im Allgemeinen jedoch mehrere Bezeichner in einem Netzwerk übereinstimmen werden (z.B. im Falle zweier Fußschalter zur Steuerung unterschiedlicher Geräte), ist eine vollautomatische Generierung der Verschaltung nicht sinnvoll. Bevor die formale Spezifikation vorgenommen wird, ist vorab in Abb. 26 ein beispielhafter Mapping Mode-Prozess aufgezeichnet.

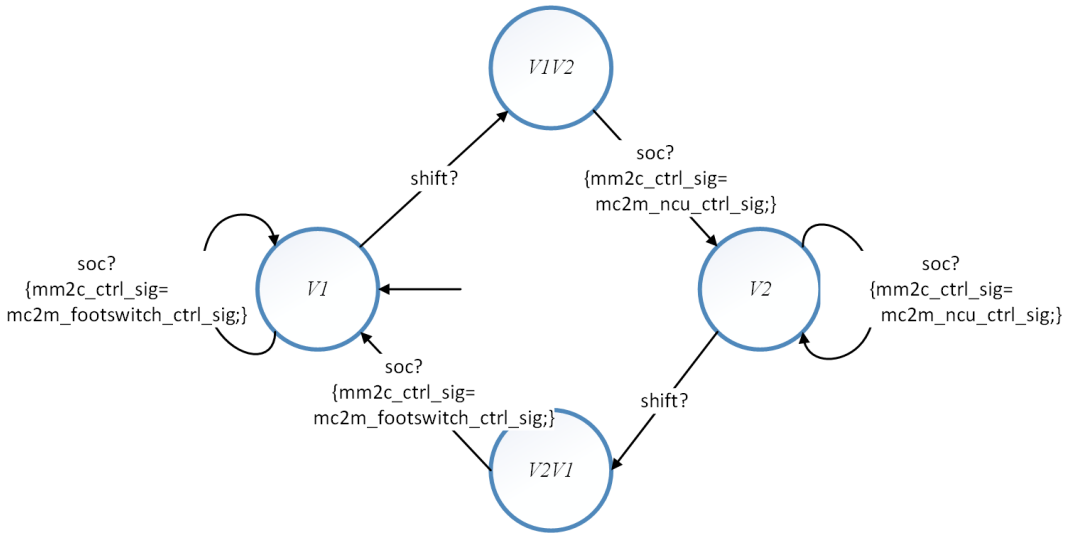


Abbildung 26 Der Mapping Mode des MN startet in der Location V1. Diese entspricht der Verschaltung, in der das Steuersignal $mm2c\_ctrl\_sig$ vom MN die Werte als Eingabe erhält, die als Ausgabe vom Fußschalter kamen ($mc2m\_footswitch\_ctrl\_sig$). Die *shift*-Nachrichten dienen dazu, von einer Verschaltung in die nächste zu wechseln. Trifft innerhalb eines Zyklus keine *shift*-Nachricht ein, so verharrt der Prozess in der ersten Verschaltung und aktualisiert mit jedem SoC-Signal den Payload. Trifft hingegen eine *shift*-Nachricht ein, so wechselt der Prozess in eine Zwischen-Location (V1V2), wartet auf das nächste soc-Signal und realisiert ab dann die neue Verschaltung $V2$, in der das Steuersignal dem Ausgabesignal der NCU entspricht. Zur Nomenklatur der Variablen siehe Abschnitt 5.2.4.

Formale Spezifikation Aus Modellierungsperspektive ist der Mapping Mode ein einfacher *Extended Finite Automaton* ohne Zeitbegriff, der nebenläufig zum Polling Mode ausgeführt wird. Insbesondere sind daher I und C nicht zu spezifizieren. Er startet in einer vordefinierten Default-Verschaltung. Über dedizierte Nachrichten (binäre Synchronisation) kann der MN die Verschaltung wechseln. Dabei geht der Mapping Mode auf das *shift*-Signal hin zunächst in einen Zwischenzustand über, wartet auf das SoC-Signal (Broadcast Synchronisation) des Polling Mode und realisiert daraufhin die neue Verschaltung. Der MN kann eine beliebige endliche Anzahl an *Verschaltungen* realisieren. Sind m unterschiedliche Verschaltungen $(W_1, \dots W_m)$ vordefiniert, so hat der Mapping Mode m^2 verschiedene Locations. Diese setzen sich wie folgt zusammen:

Jeder Verschaltung $W_1, \dots W_m$ wird eine Location zugeordnet. Zudem gibt es für jeden Übergang von Verschaltung p zu Verschaltung q $(p, q \in [m])$ einen Zwischenzustand. Da es bekanntermaßen $\binom{m}{2}$ ungeordnete Paare (p, q) gibt, gibt es doppelt so viele geordnete Paare, also $2 \cdot \binom{m}{2} = m(m-1)$ und daher insgesamt $m + m(m-1) = m^2$ Locations. Der Beispielprozess aus Abb. 26 ist in nachfolgender Tabelle 5 tabellarisch formalisiert.

S	$\{W_1, \dots W_n\} \cup \{W_i W_j \mid i, j \in [n], i \neq j\}$
$\mathbf{s_0}$	W_1
C	$\emptyset$
V	$\bigcup_{j=1}^{n} V_{CN_j}$
A	$\{soc?, shift?\}$
$\mathbf{v_0}$	$\sum_{j=1}^{n} \mathbb{1}_{V_j} \cdot v_0^{(j)} \cdot \left(\sum_{k=1}^{n} \mathbb{1}_{V_k}\right)^{-1}$

Tabelle 5 Tabellen-Notation des Mapping Mode-Prozesses als Timed Extended Finite Automaton.

Da nun die Spezifikation des Systems abgeschlossen ist, befasst sich das folgende Kapitel mit der formalen Beschreibung der Anforderungen, die an dieses System gestellt werden.

4.5. Beschreibung der Anforderungen durch temporale Logik

Mithilfe der oben eingeführten Formalismen kann das Echtzeit-Bussystem inklusive der angeschlossenen Medizingeräte unter Vorgabe der Verschaltung (siehe Abschnitt 4.4) durch ein System kommunizierender TEFAs formal beschrieben werden. Ein System mit n CNs wird durch die Gleichung

$$System = \left(MN \parallel CN_1 \parallel CN_2 \parallel \ldots \parallel CN_n\right)_{\{soc\}} \setminus \left(\left\{preq_i : i \in [n]\right\} \cup \left\{pres_j : j \in [n]\right\}\right)$$

beschrieben. Die Analyse solcher formalen Modelle erfolgt anhand von *Anforderungen*, welche an das System gestellt werden und durch das System entweder erfüllt sind, oder nicht (Verifikation). Um die rechnergestützte Verifikation der Anforderungen zu ermöglichen, müssen diese in geeigneter Weise formalisiert werden (siehe Abb. 13). In dieser Arbeit wird dazu eine Sprache verwendet, die der Klasse der *temporalen Logiken* zugehörig ist.
Temporale Logiken wurden ursprünglich entwickelt, um Formalität in natürliche Sprachen zu bringen (Kristoffersen, 1998). Später wurde der Ansatz zunächst von Amir Pnueli (Pnueli, 1986) in die Informatik transferiert und schließlich in unterschiedlichsten Varianten fortgeführt. In der Praxis haben sich besonders die Logiken *Computation Tree Logic* (CTL) (Clarke et al., 1986), *Linear Temporal Logic* (LTL) (Pnueli und Manna, 1992) und *Branching Time Logic* (CTL*) durchgesetzt. Weitere Sprachen sind μ-Calculus (Kozen, 1983) und Hennessy Milner Logic *(HML)* (Hennessy und Milner, 1985). Einige dieser Logiken wurden um einen Zeitbegriff erweitert, so etwa getimter μ-Calculus T_μ (Henzinger et al., 1994) und *Timed Computation Tree Logic* (TCTL) (Alur et al., 1990, 1993).

4.5.1. Die Spezifikationssprache

In dieser Arbeit wird die Teilsprache von TCTL aus (Behrmann et al., 2004) verwendet. Die Sprache TCTL unterscheidet *lokale Aussagen* und *Pfadaussagen*, wobei Pfade (siehe Definition 4.1.12) mögliche Abfolgen von Zuständen sind. Lokale Aussagen beziehen sich immer auf einzelne Zustände, Pfadaussagen hingegen auf ganze Pfade innerhalb eines TEFA.

Definition 4.5.1 (Lokale Aussagen, Spezifikationssprache, vgl. (Behrmann et al., 2004)) *Sei* $\mathcal{T} = (S, s_0, C, V, A, \tau, v_0, I, \Delta)$ *ein* Timed Extended Finite Automaton *(TEFA). Die Sprache* lokaler Aussagen $\mathfrak{Loc}$ *baut auf dem Alphabet*

$$\mathcal{E} := \{\wedge, \neg\} \cup \{\mathcal{T}.s \mid s \in S\} \cup \Phi(C \cup V)$$

auf und wird durch die Grammatik

$$REQ ::= \epsilon \mid EXPR \text{ mit}$$

$$EXPR ::= a \mid EXPR \wedge EXPR \mid \neg EXPR \qquad (a \in \{\mathcal{T}.s \mid s \in S\} \cup \Phi(C \cup V))$$

definiert. Die Spezifikationssprache wiederum ist

$$\mathfrak{Spec} := \{d \cdot a : d \in \mathfrak{Quant}, a \in \mathfrak{Loc}\}, \text{ wobei } \mathfrak{Quant} := \left\{\forall\Box, \forall\Diamond, \exists\Box, \exists\Diamond\right\}.$$

Bemerkung 11 (Semantik der Pfadaussagen)
*Aussagen aus $\mathfrak{Spec}$ setzen sich nach Definition 4.5.1 nach dem Muster **X Y** φ zusammen, wobei **X** $\in \{\exists, \forall\}$, **Y** $\in \{\Diamond, \Box\}$ und $\varphi \in \mathfrak{Loc}$ eine lokale Aussage ist. Ersetzt man obige Symbole nach folgendem Muster*

$$\exists \rightarrow \textit{Für manche Pfade gilt}, \forall \rightarrow \textit{Für alle Pfade gilt}, \Diamond \rightarrow \textit{manchmal}, \Box \rightarrow \textit{immer},$$

kann man umgangssprachliche Aussagen formulieren (siehe Beispiel 4.5.1 und Abb. 27).

Beispiel 4.5.1 (Übersetzung von Aussagen in Umgangssprache)
Im folgenden Beispiel werden unterschiedliche Aussagen aus $\mathfrak{Spec}$ über ein System mit der lokalen Aussage "Drehzahl<1000" vorgestellt:
$\forall\Box(Drehzahl < 1000) \leftrightarrow$ ***Für alle Pfade gilt*** | ***immer*** | $Drehzahl < 1000$
(Die Drehzahl ist immer kleiner als 1000).
$\exists\Box(Drehzahl < 1000) \leftrightarrow$***Für manche Pfade gilt*** | ***immer*** | $Drehzahl < 1000$
(Die Drehzahl kann dauerhaft unter 1000 bleiben).
$\forall\Diamond(Drehzahl < 1000) \leftrightarrow$***Für alle Pfade gilt*** | ***manchmal*** | $Drehzahl < 1000$
(Zu irgendeinem Zeitpunkt ist die Drehzahl sicher kleiner als 1000).
$\exists\Diamond(Drehzahl < 1000) \leftrightarrow$***Für manche Pfade gilt*** | ***manchmal*** | $Drehzahl < 1000$
(Die Drehzahl kann kleiner als 1000 werden).

Die Semantik der so geformten Aussagen ist in Abb. 27 visualisiert.

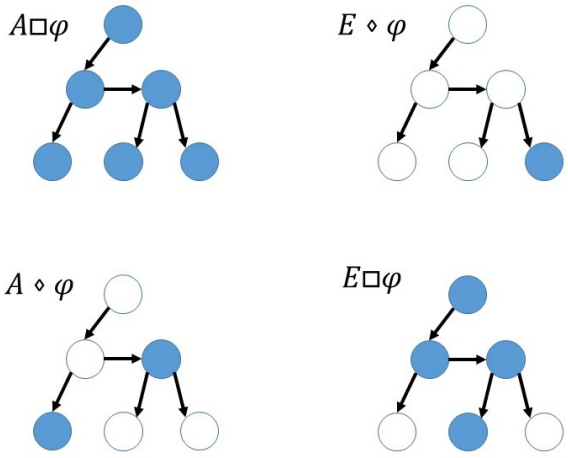


Abbildung 27 Unterschiedliche Sätze der Logik. Die Knoten der jeweils als Graph beschriebenen Systeme sind so eingefärbt, dass die Anforderung des jeweils nebenstehenden Satzes erfüllt sind. Angelehnt an (Behrmann et al., 2004).

Definition 4.5.2 (Gültigkeit lokaler Aussagen, angelehnt an (Baier et al., 2008f))
Sei $\mathcal{T} = (S, s_0, C, V, A, \tau, v_0, I, \Delta)$ *ein TEFA. Die Gültigkeit einer lokalen Aussage* $\rho \in \mathfrak{Loc}$ *in einem Zustand* $Z = (l, u, v) \in S \times [0,\infty)^C \times \mathbb{Z}^V$*, in Symbolen* $Z \vDash \rho$*, wird induktiv definiert (siehe auch Definition 4.1.4). Dabei gilt für* $s \in S$ *und* $G \in \Phi(C \cup V)$

$$\begin{aligned} Z \vDash G \quad &\text{genau dann, wenn } (gdw) \quad (u, v) \vDash G \text{ (siehe Definition 4.1.4),} \\ Z \vDash \mathcal{T}.s \quad &gdw \quad z = s \text{ und} \\ Z \vDash \epsilon \quad &\text{immer.} \end{aligned}$$

Für allgemeine $\varphi, \psi \in \mathfrak{Loc}$ *gilt nun*

$$\begin{aligned} Z \vDash \varphi \wedge \psi \quad &gdw \quad Z \vDash \varphi \text{ und } Z \vDash \psi \text{ und} \\ Z \vDash \neg\varphi \quad &gdw \quad Z \nvDash \varphi. \end{aligned}$$

Bemerkung 12 (Syntaktische Konstrukte)
Verkürzend werden im Folgenden die üblichen Abkürzungen genutzt:

$$\begin{aligned} a \vee b \quad &:= \quad \neg(\neg a \wedge \neg b) \\ a \Rightarrow b \quad &:= \quad (\neg a \vee b) \end{aligned}$$

für $a, b \in \{\mathcal{T}.s \mid s \in S\} \cup \Phi(C \cup V)$.

Nachdem in Definition 4.5.2 beschrieben wurde, was die Aussage „Zustand Z erfüllt ρ“ (für $\rho \in \mathfrak{Loc}$) bedeutet, wird in der folgenden abschließenden Definition formalisiert, was „Automat $\mathcal{T}$ erfüllt ψ“ für $\psi \in \mathfrak{Spec}$ bedeutet.

Definition 4.5.3 (Gültigkeit von Pfadaussagen, angelehnt an (Baier et al., 2008f))
Sei $\mathcal{T} = (S, s_0, C, V, A, \tau, v_0, I, \Delta)$ *ein TEFA und* $\varphi \in \mathfrak{Loc}$*. Man schreibt*

$$\begin{aligned} \mathcal{T} \vDash \forall\Box\varphi \quad &gdw \quad \forall tr \in [[\mathcal{T}]]\, \forall s \in S_{tr} : s \vDash \varphi \\ \mathcal{T} \vDash \forall \diamond \varphi \quad &gdw \quad \forall tr \in [[\mathcal{T}]]\, \exists s \in S_{tr} : s \vDash \varphi \\ \mathcal{T} \vDash \exists\Box\varphi \quad &gdw \quad \exists tr \in [[\mathcal{T}]]\, \forall s \in S_{tr} : s \vDash \varphi \\ \mathcal{T} \vDash \exists \diamond \varphi \quad &gdw \quad \exists tr \in [[\mathcal{T}]]\, \exists s \in S_{tr} : s \vDash \varphi \end{aligned}$$

und im Fließtext „$\mathcal{T}$ *erfüllt die Formel“.*

Die Autoren aus (Clarke und Emerson, 1981) konnten zeigen, dass die Entscheidbarkeit (true/false) derartiger Aussagen mit in der Länge des Ausdrucks polynomiell wachsendem Platzaufwand (PSPACE) berechenbar ist (Baier et al., 2008d).

4.6. Generierung des Systemmodells und des Anforderungskatalogs

Sind die physikalischen Prozesse aller Module (wie in Tabelle 4) durch die Hersteller, und deren Verschaltungen durch den Betreiber spezifiziert, wird nach den Vorgaben aus Tabellen 2 und 5 automatisiert das Modell für den MN zusammen mit allen EPL-Prozessen der Module (Tabelle 3) generiert. Im Kernstück dieser Arbeit, der Verification Toolbox, wird das so definierte Gesamtsystem aufgebaut, maschinenlesbar beschrieben und in ein bestehendes Framework transportiert, das dessen Simulation erlaubt und formale Anforderungen aus Spec automatisiert verifiziert.

Neben Betreibern und Anwendern können auch Hersteller Anforderungen ihres Geräts an das System formulieren, indem sie diese in der jeweiligen Beschreibungsdatei des Geräts hinterlegen. Durch die in dieser Arbeit entwickelte Verification Toolbox werden die Anforderungen der Betreiber und Anwender, die in einer separaten Textdatei editiert werden können, und die Anforderungen der Hersteller, die in den Beschreibungsdateien kodiert sind, eingelesen und zu einem Anforderungskatalog zusammengefasst. Beispielhaft wurde in dieser Arbeit folgende Funktion implementiert: Hersteller können für jede Location angeben, ob diese gemieden oder besucht werden muss. Wenn eine Location besucht werden muss, kann zudem angegeben werden, ob sie *dauerhaft* oder *einmal* besucht werden muss. Die Struktur der Verification Toolbox ist in Abb. 28 aufgezeichnet.

Eine Besonderheit des in dieser Arbeit beschriebenen Ansatzes ist die automatisierte Erzeugung der Systemspezifikation auf Basis der formalen Beschreibung der einzelnen Systemkomponenten und der Spezifikation des *Verschaltungs*-Zustands. Während die Spezifikation der Systembestandteile vom Hersteller der jeweiligen Komponente vorgenommen wird, editiert der Betreiber die unterschiedlichen Verschaltungen.

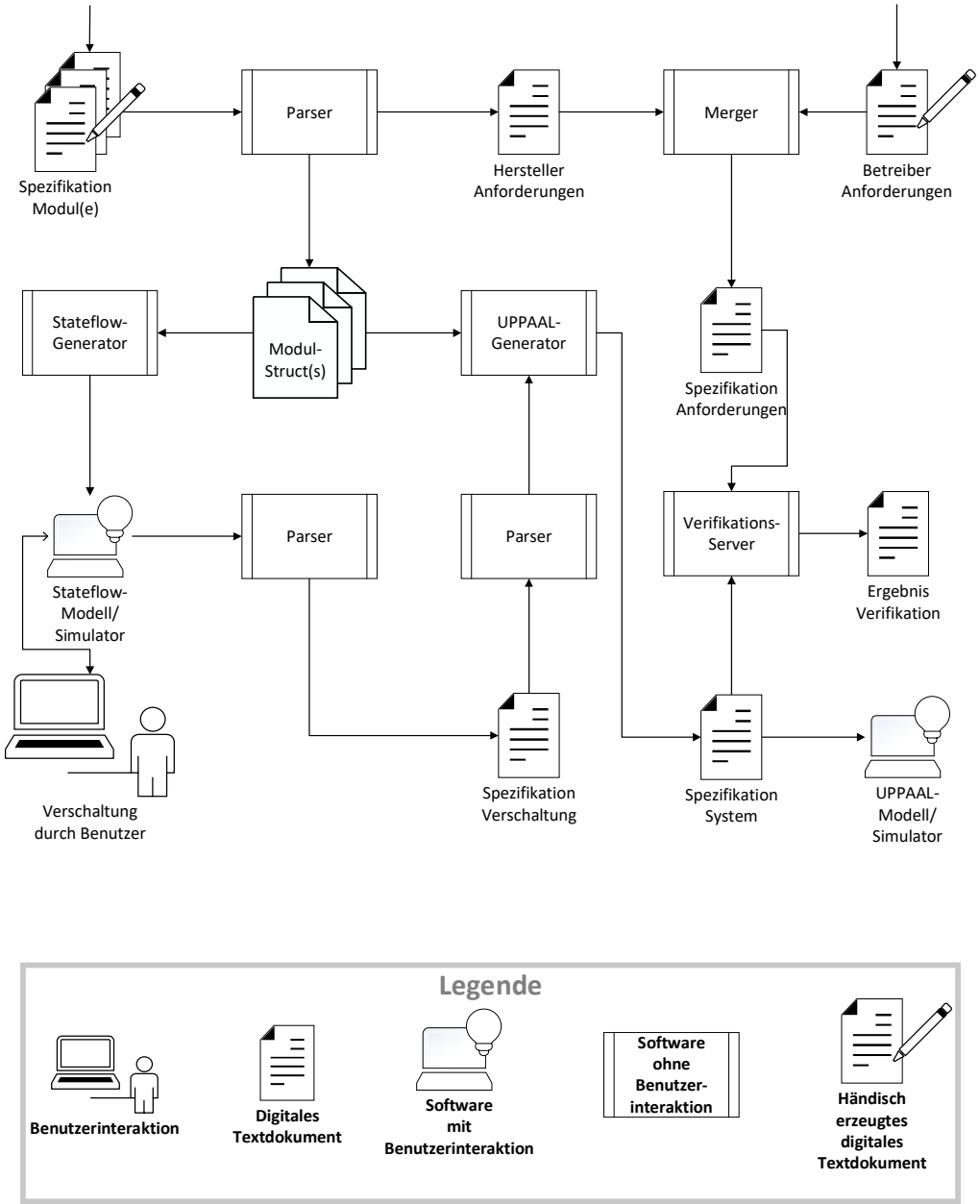


Abbildung 28 Übersicht über die Struktur der Verification Toolbox. Die Symbole sind im grauen Kasten erklärt. Händisch zu spezifizieren sind demnach die einzelnen beteiligten Module, sowie Anforderungen in Bezug auf den Anwendungsfall. Technische Anforderungen leiten sich auch automatisch aus den Beschreibungsdateien ab. Sind alle Elemente spezifiziert, so werden aus den Modulen Structs erzeugt, die für die automatisierte Generierung von Stateflow-Charts und UPPAAL-Modellen herangezogen werden können. Sind die Stateflow-Charts erzeugt, können diese durch den Anwender verschaltet und die verschiedenen Verschaltungen in das UPPAAL-Modell aufgenommen werden. Das Modell kann sodann entweder zu Plausibilitätszwecken simuliert, oder sofort gegen die Anforderungen verifiziert werden. Das Ergebnis der Verifikation wird automatisch dokumentiert. Aufbauend auf (Dingler et al., 2017).

4.7. Automatisierte Verifikation der Anforderungen

Sind System und Anforderungen modelliert, kann das System verifiziert, d.h. die Anforderungen auf Gültigkeit geprüft werden. Wie in Kapitel 5 deutlich wird, gibt es eine Fülle an Algorithmen und Tools, die diese Herausforderung adressieren. Die Implementierung von Model Checking-Algorithmen ist nicht Teil dieser Arbeit – hier wurde auf das etablierte UPPAAL-Tool zurückgegriffen. In diesem kurzen Absatz sollen nur die wichtigsten Grundzüge genannt werden, die für die Verifikation von TEFAs wichtig sind. Übersichten der vom UPPAAL-Tool genutzten Algorithmen finden sich in (Behrmann et al., 2002) und (Bengtsson und Yi, 2004). Eine umfassende Übersicht über Model Checking-Algorithmen findet sich in (Baier et al., 2008d).

4.7.1. Reduktion des Zustandsraums

Die zentrale Idee für jeden Verifikationsalgorithmus auf TEFAs ist es, den zunächst überabzählbar großen Zustandsraum auf einen endlichen Zustandsraum einzuschränken, ohne dabei von Eigenschaften des Systems zu abstrahieren, die das Ergebnis beeinflussen könnten. Der zentrale Begriff dabei ist die *Clock Equivalence* – eine Äquivalenzrelation, die die überabzählbar vielen Werte, welche jede einzelne Uhr annehmen kann, auf endlich viele reduziert. Für eine reelle Zahl $x \in \mathbb{R}$ bezeichne $\lfloor x \rfloor := max\{n \in \mathbb{Z} : n \leq x\}$ die größte ganze Zahl, die kleiner oder gleich x ist und $frac(x) := x - \lfloor x \rfloor$ ihren reellen Rest.

Definition 4.7.1 (Clock Region, vgl. (Alur und Dill, 1994))
Sei $\mathcal{T} = (S, s_0, C, V, A, \tau, v_0, I, \Delta)$ein TEFA und φ eine TCTL-Formel. Für jede Uhr $c \in C$ sei $k_c \in \mathbb{N}$ die größte Zahl, mit der c in irgendeiner Transition von $\mathcal{T}$ oder in φ verglichen wird. Zwei Zeitpunkte $\eta_1, \eta_2 \in [0, \infty)^C$ heißen äquivalent, geschrieben $\eta_1 \cong \eta_2$, wenn

- *für alle $c \in C$ gilt $\eta_1(c) > k_c$ und $\eta_2(c) > k_c$, oder*
- *für alle $c_1, c_2 \in C$ mit $\eta_1(c_1), \eta_2(c_1) \leq k_{c_1}$ und $\eta_1(c_2), \eta_2(c_2) \leq k_{c_2}$ gilt*
  - $\lfloor \eta_1(c_1) \rfloor = \lfloor \eta_2(c_1) \rfloor$ *und* $frac\left(\eta_1(c_1)\right) = 0 \Leftrightarrow frac\left(\eta_2(c_1)\right) = 0$,
  - $frac\left(\eta_1(c_1)\right) \leq frac\left(\eta_1(c_2)\right) \Leftrightarrow frac\left(\eta_2(c_1)\right) \leq frac\left(\eta_2(c_2)\right)$.

Die Äquivalenzklasse eines Zeitpunkts $\eta \in [0, \infty)^C$ heißt Clock Region *und wird mit $[\eta] := \{\tilde{\eta} : \eta \cong \tilde{\eta}\}$ beschrieben.*

Die Menge der so entstehenden Äquivalenzklassen ist endlich, da für eine feste Uhr c alle Zeitpunkte u in eine der Klassen

$$u(c) = 0,\ 0 < u(c) < 1,\ u(c) = 1,\ 1 < u(c) < 2,\ \ldots, u(c) = k_c,\ u(c) > k_c$$

fallen (siehe auch Abb. 29). Auf diese Weise ist es möglich, die zunächst überabzählbar vielen Zeitpunkte unter Erhaltung der für die Fragestellung wesentlichen Eigenschaften auf

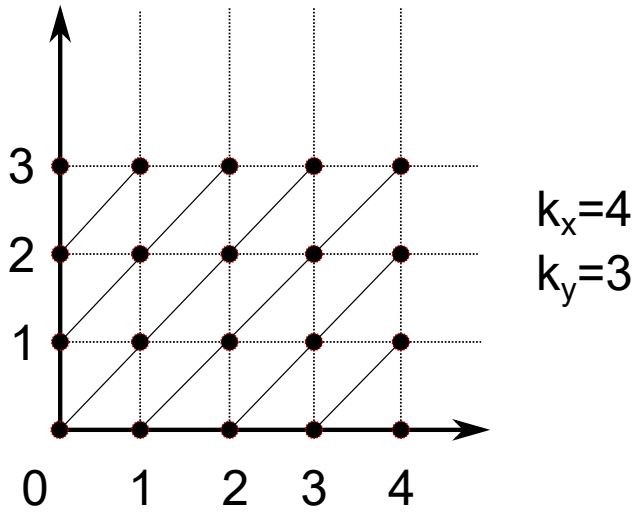


Abbildung 29 Äquivalenzklassen von Zeitpunkten bei zwei Uhren und Grenzen $k_x = 4$, $k_y = 3$. Für die Auswertung der Aussage $x \leq 2$ ist es unerheblich, ob $u(x) = \sqrt{\pi}$ oder $u(x) = 1.7$. Die Aussage stimmt allgemein genau dann, wenn $\lfloor u(x) \rfloor < 2$ oder $u(x) = 2$. In diesem konkreten Beispiel zerfallen überabzählbar viele Werte in nur $(2 * k_x + 2) \cdot (2 * k_y + 2) = 80$ Äquivalenzklassen. Auf diese Weise wird der unendlich große Zustandsraum auf einen endlichen Zustandsraum „abstrahiert". Angelehnt an (Alur und Dill, 1994).

eine endliche Menge äquivalenter Zeitpunkte zu reduzieren. Diese Methode lässt sich offensichtlich auf Variablen übertragen. Von Zustandsäquivalenz $(l, u, w) \cong \left(\tilde{l}, \tilde{u}, \tilde{w}\right)$ spricht man dann, wenn $l = \tilde{l}$, $u \cong \tilde{u}$ und $w \cong \tilde{w}$.

Auf diese Weise wird der Zustandsraum insgesamt – je nach Eigenschaft φ – auf einen endlichen Zustandsraum beschränkt, ohne dabei Eigenschaften zu verletzen, die für die Fragestellung φ von Bedeutung wären. Auf eine genaue Definition des so entstehenden Regionen-Automaten (siehe z.B. Definition 9.58. in (Baier et al., 2008f)) wird hier verzichtet – für das Verständnis ist ein solcher in Abbildung 30 aufgezeichnet. Durch die Überführung des unendlichen Zustandsraums in einen endlichen, kann das System effektiv algorithmisch analysiert werden. Der Nachweis, dass die Verifikation des Regionen-Automaten mit der des ursprünglichen TEFA übereinstimmt, wird beispielsweise in (Baier et al., 2008f) geführt.

4.7.2. Verifikationstechniken (angelehnt an (Kristoffersen, 1998))

Die einfachste Verifikationsmethode besteht darin, den gesamten Zustandsraum eines TEFAs zu ermitteln und zu prüfen, in welchen Zuständen eine Aussage Gültigkeit hat. Das Kernproblem dabei ist das *State Explosion Problem*: Der Zustandsraum des entsprechenden Regionen-Automaten wächst exponentiell mit der Anzahl der verwendeten Variablen, Uhren und auch Schranken k_c. Für komplexere Systeme erfordert daher dieser Ansatz mehr Speicherplatz, als in der Praxis zur Verfügung steht.

Die Kunst ist daher, je nach Verifikationsproblem nur denjenigen Teil des Zustandsraumes zu untersuchen, der für die jeweilige Verifikationsaufgabe relevant ist. Diese generische Tech-

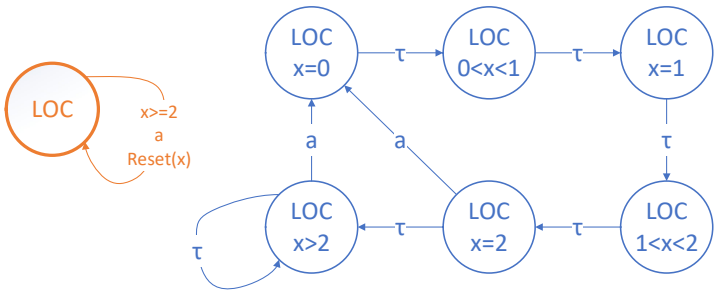


Abbildung 30 Überführung eines einfachen TEFA ohne Variablen in ein Transitionssystem mit endlich vielen Zuständen. Während der vermeintlich einfachere TEFA im oberen Bildabschnitt überabzählbar viele Zeitpunkte der Uhr x zulässt und somit überabzählbar großen Zustandsraum hat, wird in der unteren Version von den konkreten Zeitpunkten von x in geeigneten Klassen abstrahiert und so ein gleich aussagekräftiger TEFA mit endlichem Zustandsraum erzeugt. Angelehnt an (Baier et al., 2008f).

nik wird als Bounded Model Checking (auch: *On-the-fly Model Checking*) bezeichnet. Sie ist immer dann von Vorteil, wenn die Verifikationsaufgabe nicht die Betrachtung des gesamten Zustandsraums erfordert – etwa in der Erreichbarkeitsanalyse (siehe nächster Abschnitt).
Für die Verifikation von Aussagen der Form $\forall\Box\varphi$ benötigt man hingegen weiterführende Techniken (vgl. Abschnitt 4.7.2). Die wichtigsten darunter sind Reduced Ordered Binary Decision Diagrams *(ROBDD)* und Partial Order Reduction *(POR)*. Diese Techniken werden in den folgenden Abschnitten kurz vorgestellt.

Erreichbarkeitsanalyse

Die Erreichbarkeitsanalyse betrachtet die Fragestellung, ob ein Zustand $z = (l, u, v)$ des Regionen-Automaten eines TEFA vom Startzustand $z_0 = (l_0, u_0, v_0)$ aus erreichbar ist. Hierzu muss beispielsweise (siehe Einführung) keineswegs der gesamte Zustandsraum erforscht werden – vielmehr genügt es, beginnend beim Startzustand z_0 zu prüfen, ob dieser bereits der gesuchte Zustand ist ($z = z_0$). Ist dies nicht der Fall, so geht man zu einem der (endlich vielen) Folgezustände $(z_i)_{i=1,\ldots n}$, etwa z_1, über und wiederholt die Prozedur.
Ist $z \neq z_1$, so ist *Tiefensuche* (DFS) oder *Breitensuche* (BFS) möglich – erstere würde die Prozedur mit dem Folgezustand $z_{1,1}$ fortsetzen, die BFS hingegen mit dem Zustand z_2. Auf diese Weise wird weiter verfahren, bis ein Zustand z^* mit z übereinstimmt, also $z = z^*$. Die Suche kann auch *rückwärts* erfolgen – man startet also nicht beim Startzustand z_0 (und untersucht Folgezustände), sondern beim Zielzustand z (und untersucht Vorgänger-Zustände). Alle Varianten sind im UPPAAL-Tool implementiert. In welchen Situationen welche der beiden Methoden vorteilhaft ist, scheint noch nicht hinreichend erforscht zu sein. Ebenso wenig erforscht scheint die Frage, wann DFS und wann BFS vorzuziehen ist. BFS hat im Falle einer verifizierten Anforderung den Vorteil, inhärent den kürzesten Pfad zum Zielzustand zu ermitteln. DFS ist jedoch häufig die schnellere Verifikationsmethode (vgl. Kapitel 7, Messungen und Experimente).

Reduced Order Binary Decision Diagrams

ROBDD (Bryant, 1986) sind Graphen, die zur kompakten Darstellung binärer Funktionen der Form $f : \{0,1\}^n \rightarrow \{0,1\}$ dienen. Durch geeignete Modifikation des Graphen (diese kann zum Zeitpunkt der Erzeugung ohne vorige Kenntnis des vollständigen Graphen vorgenommen werden) kann dieser eine sehr kompakte Darstellungsform bieten (vgl. Abb. 31).
Im Rahmen des *Symbolic Model Checking* wird diese Technik für die kompakte Darstellung komplexer Transitionssysteme verwendet, indem Zustände durch Tupel binärer Variablen repräsentiert werden. Durch diese Darstellungsform gelang es den Autoren in (Lind-Nielsen et al., 2001) erstmals, ein Transitionssystem mit 10^{476} Zuständen verifizieren.

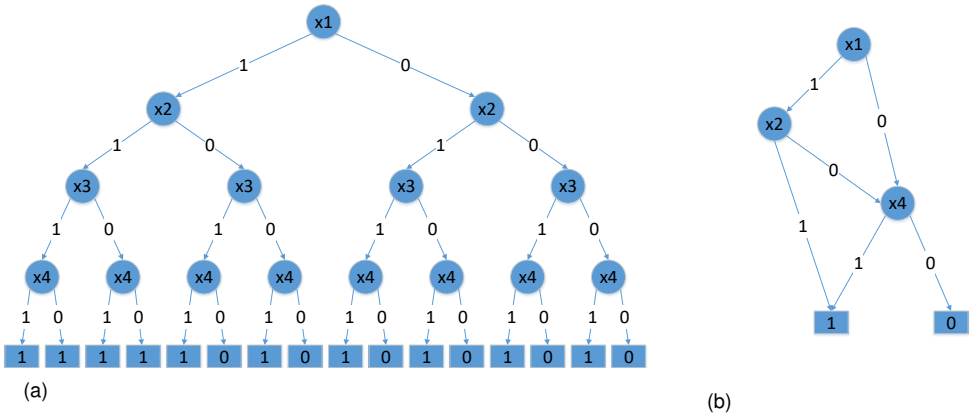


Abbildung 31 Entscheidungsdiagramme für den Ausdruck $x_1 \cdot x_2 + x_4$. Vollständiges Entscheidungsdiagramm links (Abb. 31a) und kompakte Darstellungsform mit demselben Informationsgehalt rechts (Abb. 31b). Angelehnt an (Bryant, 1986).

Partial Order Reduction

Partial Order Reduction (POR) (Katz und Peled, 1989; Valmari, 1991) ist eine Methode zur Reduktion des Zustandsraums nebenläufiger Prozesse. Sie gründet auf der Beobachtung, dass ein und derselbe Zustand durch unterschiedliche Ausführungsreihenfolgen (siehe Interleaving, Definition 4.1.13) von Transitionen erreicht werden kann. Alle Pfade, die in diesem Sinne zu demselben Ergebnis führen, können zu einem einzigen Vertreter zusammengefasst werden. Dabei ist zu beachten, dass entlang der Pfade kein Zustand auftritt, der Einfluss auf das Ergebnis der Verifikation hat.
Das auf diese Weise reduzierte Transitionssystem kann *on-the-fly* generiert werden, benötigt also nicht zuerst die Modellierung des vollständigen Zustandsraums (Clarke et al., 1999). Durch POR ist es möglich, die Größe des Zustandsraumes bedeutend zu reduzieren. Die Autoren in (Clarke et al., 1999) konnten durch Anwendung entsprechender Methoden bei unterschiedlichen Algorithmen bis zu 98% der besuchten Zustände, 99.5% der Transitionen und 89.5% des Speicherbedarfs und 95% der benötigten Zeit einsparen.

4.8. Der neue Ansatz in der Gesamtschau

Betrachtet man die in der Problemstellung genannten Herausforderungen der Konformitätsbewertung und die in dieser Arbeit erarbeiteten Lösungen, so ergibt sich in der Gesamtschau der Ablauf aus Abb. 32. Dieser wird durch folgende Akteure bestimmt:

1. **Das Standardisierungsgremium** standardisiert zwei Bereiche:
   - Die Kommunikation zwischen den Geräten. Dies schließt neben den technischen Mechanismen des Datenaustauschs auch die Standardisierung der auszutauschenden Daten und deren Bezeichner ein (siehe auch (Kasparick et al., 2015b; Andersen et al., 2016)).
   - Die Minimalanforderungen an eine Spezifikationsdatei.
2. **Der Hersteller einzelner Module** ist zuständig für die Entwicklung des jeweiligen Moduls und der Erstellung der zugehörigen technischen Dokumentation (losgelöst vom Aspekt der Vernetzung). Zudem ist der Hersteller wie bisher zuständig für die Ausstellung der Konformitätsbescheinigung mit den Grundlegenden Anforderungen. Zusätzlich zu den bisherigen Tätigkeiten, ist es Aufgabe des Herstellers, die maschinenlesbare Spezifikation seines Moduls und dessen technischer Anforderungen bereitzustellen. Der Hersteller ist verantwortlich für die Konformität der Module mit ihrer jeweiligen Spezifikation und die Erstellung einer entsprechenden Konformitätserklärung.
3. **Der Hersteller des Echtzeit-Bussystems** muss sicherstellen, dass das Echtzeit-Bussystem konform ist mit den Vorgaben des Standardisierungskonsortiums und eine entsprechende Konformitätserklärung ausstellen.
4. **Der Betreiber/Risikomanager** ist vor Inbetriebnahme des Gesamtsystems in der Verantwortung, das *vereinfachte Konformitätsbewertungsverfahren* durchzuführen. Wie im folgenden Abschnitt beschrieben, schließt dies die folgenden Tätigkeiten ein:
   - Virtuelle Komposition des Systems
   - (Wiederholte) Formulierung von Anforderungen
   - (Wiederholte) Verifikation der Anforderungen
   - Dokumentation der Verifikation

   Hinzu kommen die weiteren Tätigkeiten im Zusammenhang mit der Konformitätsbewertung aus §7 Abs. 9 MPV, die jedoch nicht Gegenstand dieser Arbeit sind.
5. **Der Anwender** ist weiterhin nur zuständig für die Anwendung des Systems. Wie bisher, muss vor der Anwendung sichergestellt sein, dass das System erfolgreich ein Konformitätsbewertungsverfahren durchlaufen hat. Vernetzungsprojekte werden durch den Anwender angestoßen. Der Anwender wirkt bei der Formulierung klinischer Anforderungen mit.

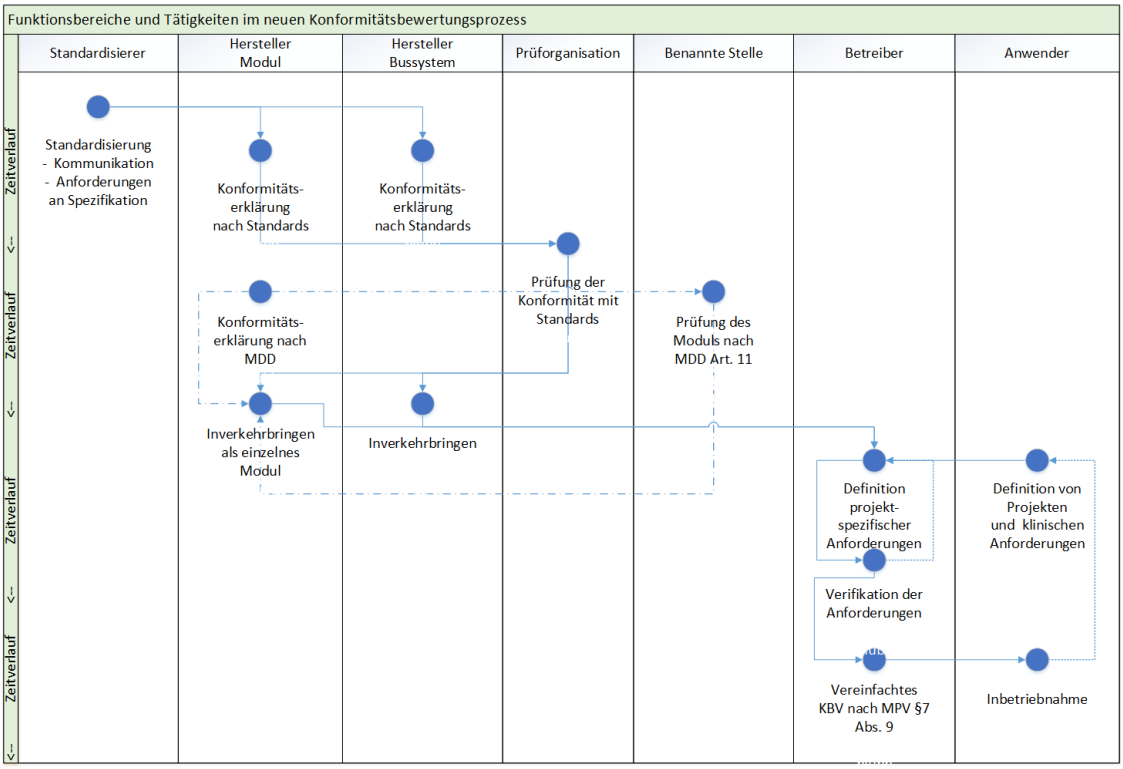


Abbildung 32 Entwurf eines neuen Konformitätsbewertungsverfahrens. Für jedes *Modul* wird die bestehende Konformitätsbewertung (unterbrochene Linie) künftig gesondert nach MDD und nach Standards vorgenommen, die sich sowohl auf die Beschreibung der Funktionalität, als auch der Anforderungen beziehen. Anforderungen stammen von Hersteller sowie Betreiber und Anwender. Sind alle Anforderungen verifiziert, kann das vereinfachte Konformitätsbewertungsverfahren nach §7 Abs. 9 MPV durchgeführt und das System in Betrieb genommen werden.

6. **Die Prüforganisation des Standardisierungsgremiums** ist zuständig für die Prüfung und Bescheinigung der Konformität der Modulspezifikation mit den Anforderungen des Standardisierungsgremiums. Zudem bescheinigt die Prüforganisation die Konformität des implementierten Kommunikationsprotokolls mit den Anforderungen des Standardisierungsgremiums.

7. **Die benannte Stelle** ist wie bisher zuständig für die Bewertung der Konformität einzelner Module mit den Grundlegenden Anforderungen. Diese ist nicht zu verwechseln mit der Konformität eines Moduls mit seiner Spezifikation.

Der gesamte Prozess ist abschließend in Abb. 32 festgehalten.

5. Realisierung

5.1. Software Bestandteile

Die Bestandteile der Verification Toolbox sind in Abb. 33 dargestellt. Da der Ziel-Anwender der Verification Toolbox ein Krankenhausbetreiber ohne technischen Hintergrund ist (siehe Kapitel 3.2), wurde bei der Auslegung darauf geachtet, dass dieser bei nur ein einziges Software-Tool bedienen muss. In dieser Arbeit wurde Matlab r2016b für die Realisierung gewählt.
Die Gerätespezifikation wird durch den Hersteller des Geräts vorgenommen und ist eine ASCII-Datei, die der Syntax aus Abschnitt 5.2 folgt. In Anlehnung an die IEEE 11073, werden die Gerätespezifikationen als *MDIB* bezeichnet. In einem Verzeichnis sind sie anhand des Präfix **mdib_** im Dateinamen zu erkennen. Wichtig dabei ist, dass der Hersteller lediglich den physikalischen Prozess des Geräts beschreiben muss. Der EPL-Prozess wird durch die Verification Toolbox automatisiert erzeugt.
Wird der physikalische Prozess durch den Hersteller als eine Menge nebenläufiger Automaten definiert, so ist ein Hauptprozess zu identifizieren, der als MDIB im Verzeichnis abgelegt wird. Die Beschreibungen der Nebenprozesse sind durch das Präfix **add_** zu erkennen.
Neben den von den beteiligten Herstellern an das System gestellten Anforderungen kann auch der Betreiber eingriffsspezifische Anforderungen an das System definieren und ebenfalls in eine einfache ASCII-Datei schreiben. Diese ist im selben Verzeichnis wie die Gerätespezifikation abzulegen und mit dem Präfix **req_** im Dateinamen zu versehen. Die Syntax der Anforderungsspezifikation ist in Abschnitt 5.4 spezifiziert.
Sind die Spezifikationen abgeschlossen, sucht ein Matlab-Programm im vorgegebenen Verzeichnis alle Dateien mit dem MDIB-Präfix, öffnet eine Simulink-Umgebung und erzeugt in derselben aus jeder MDIB einen *Stateflow*-Chart. Der Anwender der Verification Toolbox kann nun die einzelnen Stateflow-Charts verknüpfen, um Datenquellen und -senken virtuell zu verschalten (siehe Abb. 36). Ist die Verschaltung abgeschlossen, generiert ein Matlab-Skript die Systemspezifikation für den verschalteten Automaten (siehe Abschnitt 4.6).
Der Anwender der Verification Toolbox kann daraufhin den *Verification Call* in Matlab aufrufen, welcher die Verifikation der Anforderungen durch die Systemspezifikation veranlasst. Die Verifikation selbst wird im UPPAAL-Tool durchgeführt (Abschnitt 5.6.1). Das Ergebnis wird in Matlab ausgegeben und in einer ASCII-Textdatei automatisch niedergeschrieben.

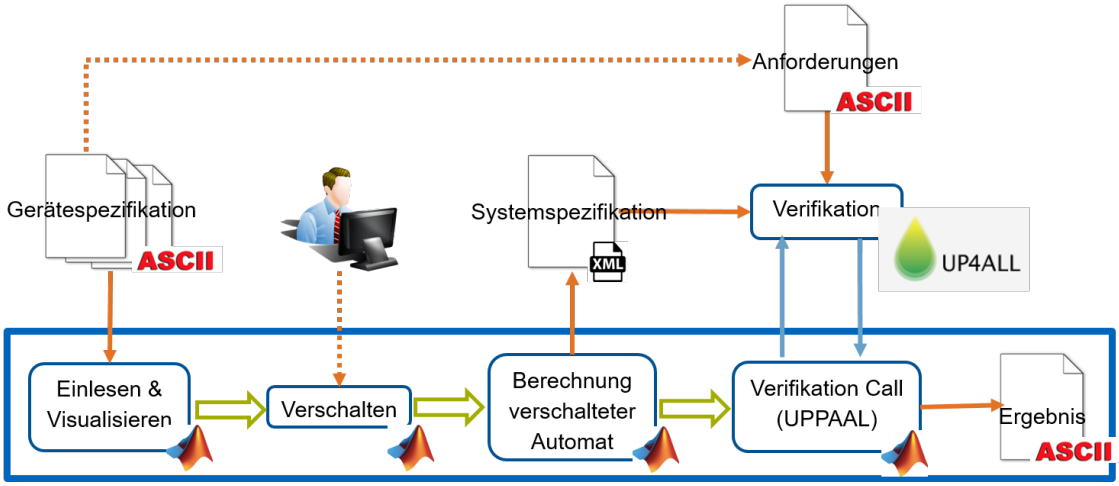


Abbildung 33 Jede Gerätespezifikation wird als ASCII-Datei aufgeschrieben, die der Syntax aus Abschnitt 5.2 folgt. Die Verschaltung wird ebenfalls als Textdatei in der Syntax aus Abschnitt 5.3 aufgeschrieben, oder aus einer Stateflow-GUI, die der Anwender per Drag-and-Drop bedienen kann, automatisiert erzeugt. Aus diesen beiden Bestandteilen wird die Systemspezifikation im XML-Format erzeugt. Der Nutzer formuliert seine Anforderungen in einer Textdatei. Der Anwender des Systems nutzt durchgängig Matlab. Im Hintergrund ist die Verifikation mit UPPAAL v4.1.19 realisiert. Matlab- und UPPAAL-Logo von Herstellerseiten entnommen.

5.2. Realisierung der Gerätespezifikation

Die Gerätespezifikation findet in einer einfachen ASCII-Textdatei mit Hilfe von *Key-Value-Pairs* statt. Das bedeutet, dass Eigenschaften des Geräts stets gepaart mit einem Bezeichner in loser Reihenfolge aufgeführt werden können (Beispiel: (Gerät:Pumpe//Seriennummer:1) bzw. (Seriennummer:1//Gerät:Pumpe)) und nicht in fester Reihenfolge mit vordefinierter Semantik (Beispiel: (Pumpe//1)). Dies beugt Fehlern in der Spezifikation vor, erleichtert die Lesbarkeit und den Ausbau der Verification Toolbox. Als Trennzeichen zwischen *Key* und *Value* wird wie oben der Doppelpunkt verwendet. Key-Value-Pairs werden wie oben durch doppelte Schrägstriche getrennt.

5.2.1. Spezifikation der Module

Für die vollständige Spezifikation eines Moduls müssen nach Definition 4.1.9 in der Beschreibungsdatei folgende Elemente spezifiziert werden:

- Locations, Initial Location und Invarianten
- Uhren
- Variablen mit Anfangswerten
- Ein- und Ausgabesignale
- Transitionen

All diese Elemente werden wie oben einzeln, durch Klammerung voneinander unterscheidbar aufgelistet. Um welches der obigen Elemente es sich handelt, wird durch den Key `Class`

beschrieben (z.B. `Class:Location`). Abhängig von der `Class` werden unterschiedliche Attribute geparst. Im Folgenden werden die zu einer `Class` gehörigen Attribute aufgeführt. In jeder der folgenden Tabellen bezeichnet die *Value-Domain* den Definitionsbereich des *Keys*. Die Attribute **Optional** bzw. **Required** spezifizieren, ob der Key belegt werden **kann** (Optional) oder **muss** (Required).

5.2.2. Spezifikation der Locations

Die folgende Tabelle listet alle Keys auf, die für das Element Location geparst werden. Neben rein beschreibenden *Eigenschaften*, wie zum Beispiel eindeutigen Location-Bezeichnern (Handle), können hier durch den Hersteller erste *Anforderungen* definiert werden. Im Key *Reachability* kann der Hersteller deklarieren, ob und wie oft eine Location erreicht oder gemieden werden kann oder muss.

Key-Value-Tabelle für Locations eines TEFA

Key	**Value-Domain**	**Optional (O) Required (R)**	**Beschreibung**
Class	Location	R	Die beschriebene Klasse ist eine Location
Handle	char	R	Eindeutiger Bezeichner der Location
InitLocation	Boolean	O	TRUE, wenn es sich um die Initial Location handelt
Invariant	$\Phi(C)$	O	*Invariante*
Reachability	Never Possible Mandatory Recurrent Permanent	O	Location... ...darf nicht... ...kann einmal... ...muss einmal... ...kann dauerhaft... ...muss dauerhaft ... ...erreicht werden
Urgent	Boolean	O	TRUE, wenn in dieser Location keine Zeit verstreichen darf

Tabelle 6 Keys für das Element **Location**. Locations werden durch einen eindeutigen Bezeichner HANDLE, ihre Invariante INVARIANT und eine optionale Erreichbarkeitsanforderung REACHABILITY beschrieben.
Durch das INITLOCATION-Flag kann beschrieben werden, ob es sich um die Initial Location handelt. Ein auf true gesetztes URGENT-Flag ist eine kompakte Schreibweise, die semantisch äquivalent ist zu einer Uhr t, die bei allen eingehenden Transitionen auf null gesetzt wird, zusammen mit der Invarianten $t <= 0$, die zu der Location gehört.

5.2.3. Spezifikation der Uhren

Uhren werden ausschließlich durch ihren Bezeichner beschrieben.

Key-Value-Tabelle für Uhren eines TEFA

Key	Value-Domain	Optional (O) Required (R)	Beschreibung
Class	Clock	R	Die beschriebene Klasse ist eine Uhr
Handle	char	R	Eindeutiger Bezeichner der Uhr

Tabelle 7 Uhren werden nur über ihren eindeutigen Bezeichner HANDLE spezifiziert.

5.2.4. Spezifikation der Variablen

Bei jedem Modul wird zwischen *lokalen Variablen* und *globalen Variablen* sowie zwischen *Eingabevariablen* und *Ausgabevariablen* unterschieden (Abb. 34). *Lokale Variablen* bezeichnen in diesem Zusammenhang Variablen eines Moduls, welche nicht über das Netzwerk propagiert werden, sondern nur der Komponente selbst bekannt sind. Im Gegensatz dazu bezeichnen *Globale Variablen* diejenigen Variablen, welche über die gemeinsame Netzwerkinfrastruktur gesendet werden und potentiell von allen Modulen mitgehört werden können. Eingabevariablen bezeichnen Variablen einer Netzwerkkomponente, die von außen (Anwender, weitere Komponenten) manipuliert werden. Ausgabevariablen bezeichnen hingegen Variablen einer Netzwerkkomponente, welche die Komponente selbst manipuliert (Anzeige, Wirkung, weitere Komponenten).

Arten von Eingabevariablen

Eingabevariablen können entweder durch den Anwender über die Bedienschnittstelle des Geräts manipuliert werden (*Lokale Variable*), oder durch ein anderes Modul über die Netzwerkschnittstelle (*Globale Variable*). Lokale Eingabevariablen werden also durch An-/Aus-Knöpfe, Schieberegler, Pedalstellungen und allgemein sämtliche Bedienelemente eines Moduls bestimmt. In der Beschreibungsdatei wird lokalen Eingabevariablen das Attribut **„manual_get“** zugewiesen.

Globale Eingabevariablen sind sämtliche Variablen, die ein Modul über die Netzwerkschnittstelle empfangen kann. Dazu gehören beispielsweise Positionsdaten eines Navigationsrechners oder Steuerparameter einer Saug-/Spülpumpe. In der Beschreibungsdatei wird globalen Eingabevariablen das Attribut **„epl_get“** zugewiesen.

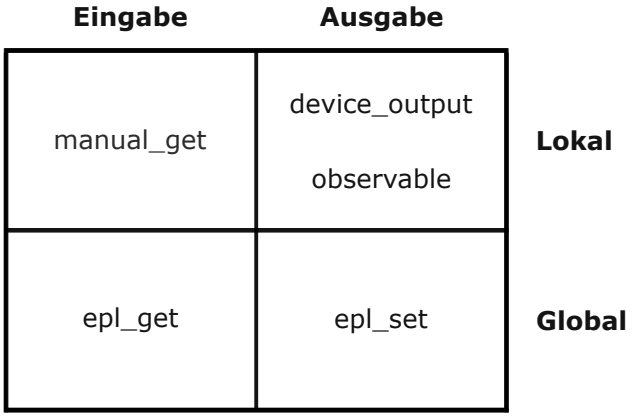


Abbildung 34 Klassen von Ein- und Ausgabe-Variablen der Module. Diese können je global oder lokal sein. Globale Variablen werden im Netzwerk propagiert, lokale Variablen sind nur dem Modul selbst bekannt.

Arten von Ausgabevariablen

Die Ausgabevariablen eines Moduls können lokal oder global sein. Lokale Ausgabevariablen sind entweder Geräte-Informationen wie Display-Ausgaben oder Alarme, oder beobachtbare Wirkungen des Geräts, wie etwa die aktuelle Drehzahl einer chirurgischen Fräse. Geräte-Informationen wird in der Beschreibungsdatei das Attribut **„device_output“** zugewiesen, während beobachtbaren Wirkungen das Attribut **„observable“** zugewiesen wird. Globalen Ausgabevariablen, d.h. Ausgabevariablen welche über das Netzwerk gehen, wird das Attribut **„epl_set“** zugewiesen.

Interne Variablen

In vielen Modellierungen kann es nötig oder hilfreich sein, den physikalischen Prozess eines Medizingeräts selbst als eine Menge nebenläufiger Automaten darzustellen. Dies ist zum Beispiel dann der Fall, wenn parallel zur Hauptfunktion über die Bedienoberfläche des Medizingeräts Einstellungen vorgenommen werden können (z.B. Leistung des US-Dissektors). In diesem konkreten Fall wird so über die Bedienoberfläche der Wert der Variable *power* eingestellt und an den Hauptprozess übergeben, damit dieser seine Funktionalität danach ausrichtet. Während diese Variablen im Nebenprozess als manual_get deklariert werden, werden sie Im Hauptprozess durch das Attribut **„internal_pass“** (siehe Anhang – Beispiel *Ultraschall-Dissektor* (US-Dissektor)) beschrieben. Damit wird deutlich, dass es einen Nebenprozess gibt, für den die Variable eine Ein- oder Ausgabe darstellt, die dem Hauptprozess intern kommuniziert wird. In der Modellgenerierung (siehe Abschnitt 5.6.4) wird so vermieden, dass zwei Kopien derselben Variable angelegt werden.
Ein weiterer Sonderfall ergibt sich, wenn der Wert einer Variable durch ein eingehendes Signal (z.B. „UP“) manipuliert wird. In diesem Fall wird die Variable als **„internal_get“** deklariert (siehe Anhang – Beispiel Saug-/Spülpumpe).

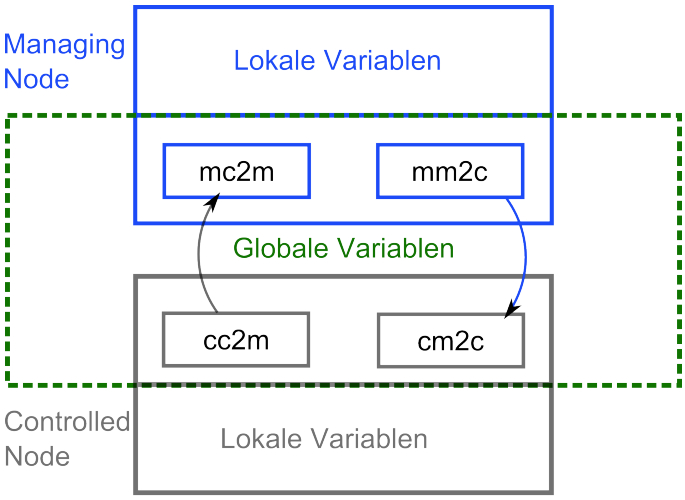


Abbildung 35 Nomenklatur der Variablen in dieser Arbeit. Aufgrund der Besonderheit des Echtzeit-Bussystem existieren von einem Datum immer zwei Versionen – eine auf dem CN, der das Datum empfängt oder erzeugt und eine auf dem MN. Welche Version gemeint ist, lässt sich am ersten Buchstaben des Präfixes erkennen (m=MN, c=CN). Welche Quelle und welche Senke das Datum hat, lässt sich an der angehängten Zeichenfolge erkennen (Muster: QUELLE2SENKE).

Nomenklatur der Variablen

Wie in Kapitel 4.2 beschrieben, ist der MN die zentrale Schaltstelle für die gesamte Kommunikation im Echtzeit-Bussystem – insbesondere werden alle Daten, welche von CN zu CN übertragen werden sollen, zunächst vom MN zwischengespeichert. Für globale Ein- und Ausgabevariablen gibt es also stets zwei Versionen – eine, die auf dem jeweiligen CN hinterlegt ist und eine, die auf dem MN hinterlegt ist. Um das Verständnis zu erleichtern, wird in dieser Arbeit vor jedes Datum ein Präfix geschrieben, welches ausdrückt **wo** das Datum hinterlegt ist (**m** für MN und **c** für CN), sowie Quelle und Senke des Datums (**c2m** für Daten, welche vom CN zum MN gesendet werden sollen oder wurden und **m2c** für Daten, welche vom MN zu einem CN gesendet werden sollen oder wurden). Die Nomenklatur ist in Abb. 35 dargestellt.

Variablen werden durch Elemente beschrieben, deren Nomenklatur in weiten Teilen an die neuen Standards der IEEE 11073-Familie angelehnt ist (etwa Resolution, MetricCategory, LowerRange, UpperRange). Wie bei Locations kann außer der Spezifikation der *Eigenschaften* der Variablen bei deren Beschreibung auch *Anforderungen* durch den Hersteller dokumentiert werden. So kann beispielsweise im Falle einer Eingabevariable durch das Feld LatencyAcceptance angegeben werden, wie hoch die Ende-zu-Ende-Latenz zwischen Quelle und Senke maximal sein darf, um die einwandfreie Gerätefunktion zu gewährleisten.

Key-Value-Tabelle für Variablen eines TEFA

Key	Value-Domain	Optional (O) Required (R)	Beschreibung
Class	Variable	R	Die beschriebene Klasse ist eine Variable
DefVal	uint8	O	Defaultwert, auf den die Variable im Fehlerfall zurückgesetzt wird
Frequency	uint8	O	Abfragefrequenz (in Hz)
Handle	char	R	Eindeutiger Bezeichner der Variable
InitVal	uint8	R	Anfangswert der Variable
LatencyAcceptance	uint8	O	Tolerierte Latenz zwischen Quelle und Senke (in ms)
LowerRange	uint8	O	Untere Schranke des Wertebereichs
MetricCategory	manual_get device_output observable epl_get epl_set internal_get internal_pass	R	Lokale Eingabe (siehe 5.2.4) Geräte-Information Gerätewirkung (siehe 5.2.4) Globale Eingabe Globale Ausgabe Benutzereingabe (siehe 5.2.4) Interne Variable (siehe 5.2.4)
Resolution	uint8/100	R	Auflösung des Wertebereichs
UpperRange	uint8	O	Obere Schranke des Wertebereichs

Tabelle 8 Keys für die Beschreibung von Variablen in Anlehnung an die neue IEEE 11073-Familie. Es können sowohl Eigenschaften (z.B. Resolution), als auch Anforderungen (z.B. LatencyAcceptance) durch den Hersteller spezifiziert werden.

5.2.5. Spezifikation der Signale

Neben den EPL-Signalen, die jedes Modul empfangen, verarbeiten und senden kann, verarbeiten manche Module proprietäre Signale – wie etwa das Drücken eines ON/OFF-Schalters an der Bedienoberfläche oder Signale, die der internen Prozesssynchronisation dienen (siehe Beispiel Navigationskamera). Alle Keys zur Beschreibung eines Signals sind in Tabelle 9 aufgelistet.

Key-Value-Tabelle für Signale eines TEFA

Key	**Value-Domain**	**Optional (O) Required (R)**	**Beschreibung**
Class	Signal	R	Die beschriebene Klasse ist ein Signal
Frequency	int	O	Frequenz, mit der das Signal erzeugt werden kann (Default: unbegrenzt schnell)
Handle	char	R	Eindeutiger Bezeichner des Signals
SignalCategory	GET LOCAL SET	R	Indikator, ob ein Eingabesignal (z.B. Druckknopf), ein lokales Signal (z.B. zur Prozesssynchronisation) oder ein Ausgabesignal (z.B. ein Alarm) definiert wird

Tabelle 9 Moduleigene Signale werden über einen Bezeichner und die maximale Frequenz, mit der sie ein- oder ausgehen, beschrieben.

5.2.6. Spezifikation der Transitionen

Die Beschreibung der Transitionen eines Moduls ist in Abschnitt 4.1.3 erklärt. In der Implementierung wurden die Begriffe ähnlich gewählt. Sie sind in Tabelle 10 zusammenfassend aufgeführt.

Key-Value-Tabelle für Transitionen eines TEFA

Key	**Value-Domain**	**Optional (O) Required (R)**	**Beschreibung**
Class	Transition	R	Die beschriebene Klasse ist eine Transition
Destination	Menge L der Locations	R	Ziel-Location einer Transition
Guard	$\Phi(C \cup V)$	O	Bedingungen an Uhren & Daten i.S.v. zulässigen Intervallen (siehe Definition 4.1.3)
Handle	char	O	Eindeutiger Bezeichner der Transition
Source	Menge L der Locations	R	Quell-Location einer Transition
Sync	preq pres SoC A	O	Ein- und Ausgabesignale
Update	$\mathcal{A}(V)$	O	Zuordnung von Variablenwerten (siehe Definition 4.1.6)

Tabelle 10 Transitionen werden durch einen eindeutigen Bezeichner HANDLE, sowie Quell- und Ziel-Location (SOURCE bzw. DESTINATION) beschrieben. Eine *Transition* kann nur genommen werden, wenn die Bedingung GUARD erfüllt ist. Sie löst ein UPDATE aus, durch das die Werte von Variablen und Uhren verändert werden können. Das (optionale) Signal, das die Transition auslöst, wird in SYNC spezifiziert.

5.2.7. Spezifikation der Geräte-Attribute

Zusätzlich zu obigen, formal erforderlichen Elementen, benötigt die Verification Toolbox für jedes Gerät eine allgemeine Beschreibung, die etwa den Gerätenamen, Hersteller etc. beinhaltet.

Key-Value-Tabelle für Geräteattribute eines TEFA

Key	**Value-Domain**	**Optional (O) Required (R)**	**Beschreibung**
Class	Device	R	Die beschriebene Klasse ist ein Gerät
Manufacturer	char	R	Hersteller
ModelName	char	R	Gerätename
ModelNumber	char	O	Bestellnummer des Geräts
ProcessingTime	uint8	O	siehe (Dietz et al., 2016)
ResponseTime	uint8	R	Antwort-Dauer auf eine preq-Anfrage
SerialNumber	char	O	Seriennummer des Geräts
udi	char	R	Unique Device Identifier (weltweit eindeutige Adresse des Geräts, vom Hersteller zu vergeben)

Tabelle 11 Zusatzinformationen über das Modul. Hier werden Elemente beschrieben, die abseits von den formalen Aspekten aus Abschnitt 4.1.3 über jedes Modul bekannt sein müssen – insbesondere der Name des Herstellers und des Geräts.

5.3. Realisierung der Verschaltungsspezifikation

Die Verschaltung des Systems wird in einer einfachen ASCII-Datei niedergeschrieben (händisch, oder mit dem Stateflow-GUI aus Abb. 33). Die Grammatik baut auf dem Alphabet V aller Gerätevariablen auf und folgt der Syntax

$$\begin{aligned} MAP &::= (Source//Sink)|MAP \cdot MAP|MAP < \texttt{NEW} > MAP \text{ mit} \\ Source &::= SRC : V \text{ und} \\ Sink &::= SNK : V. \end{aligned}$$

Verschaltungen werden durch den Ausdruck < NEW > voneinander getrennt. Jede Verschaltung besteht aus einem oder mehreren sukzessiven geklammerten Tupeln von Zuordnungen von Quellen (Source) zu Senken (Sink). Eine einfache Verschaltungsdatei könnte daher wie folgt aussehen:

```
(SRC:mc2m_FOOT_pedal_val//SNK:mm2c_EndomatLC_pedal_val)
<NEW>
(SRC:mc2m_FOOT_pedal_val//SNK:mm2c_Sonoca300_pedal_val)
```

Sie repräsentiert zwei Verschaltungen. In der ersten wird das Fußpedal-Signal `mc2m_FOOT_pedal_val` auf den entsprechenden Eingang `mm2c_EndomatLC_pedal_val` der Pumpe gelegt, bei der zweiten hingegen auf den des USDs (`mm2c_Sonoca300_pedal_val`). Der daraus erzeugte Mapping Mode wird später in Abschnitt 5.6.6 erklärt.

5.4. Aufbau der Anforderungsspezifikation

Die Anforderungsspezifikation wird in einer einfachen ASCII-Datei vorgenommen. In dieser können die TCTL-Formeln aus Abschnitt 4.5.1 sowie Kommentare editiert werden. Kommentare werden wie in vielen Programmiersprachen durch die Zeichensequenz „/*“ eingeleitet und durch die Zeichensequenz „*/“ abgeschlossen. Aus Sicht der Semantik bezieht sich innerhalb der Spezifikationsdatei ein Kommentar immer auf die *nachfolgende* Formel. Eine beispielhafte Spezifikationsdatei findet sich im Anhang.

5.5. Erzeugung der Gerätemodelle in Stateflow

Bevor das vollständige Systemmodell in UPPAAL erzeugt werden kann, muss die Verschaltung des Systems definiert werden (siehe Abschnitt 4.4). In dieser Arbeit werden hierzu Gerätemodelle in Stateflow realisiert, die der Anwender der Verification Toolbox händisch verschalten kann (siehe Abb. 36). Aus dieser Verschaltung wird eine Textdatei erzeugt, die der Syntax aus Abschnitt 5.3 folgt. In Vorveröffentlichungen (Dingler et al., 2015, 2016; Lüth et al., 2016) wurden innerhalb von Matlab/Stateflow zusätzliche Prüfmechanismen eingebaut, die etwa Auskunft darüber geben, ob der Definitionsbereich von einander zugeordneten Daten übereinstimmt.

5.6. Erzeugung des Systemmodells in UPPAAL

In dieser Arbeit wird der UPPAAL Model Checker für die Verifikation des Echtzeit-Bussystems verwendet. Es folgt eine kurze Beschreibung von UPPAAL, bevor mit den Einzelheiten der Modellerzeugung begonnen wird.

5.6.1. Der UPPAAL Model Checker

UPPAAL ist eine Toolbox für die Verifikation von Systemen, die durch eine spezielle Klasse kommunizierender Automaten beschrieben werden können - insbesondere TEFAs. UPPAAL verwendet eine Untersprache von TCTL, die eine effiziente Verifikation erlaubt (Kristoffersen, 1998). Die Toolbox wurde von den Universitäten Uppsala (Schweden) und Aalborg (Dänemark) entwickelt. Sie stellt eine grafische Benutzeroberfläche bereit, mit der ein System edi-

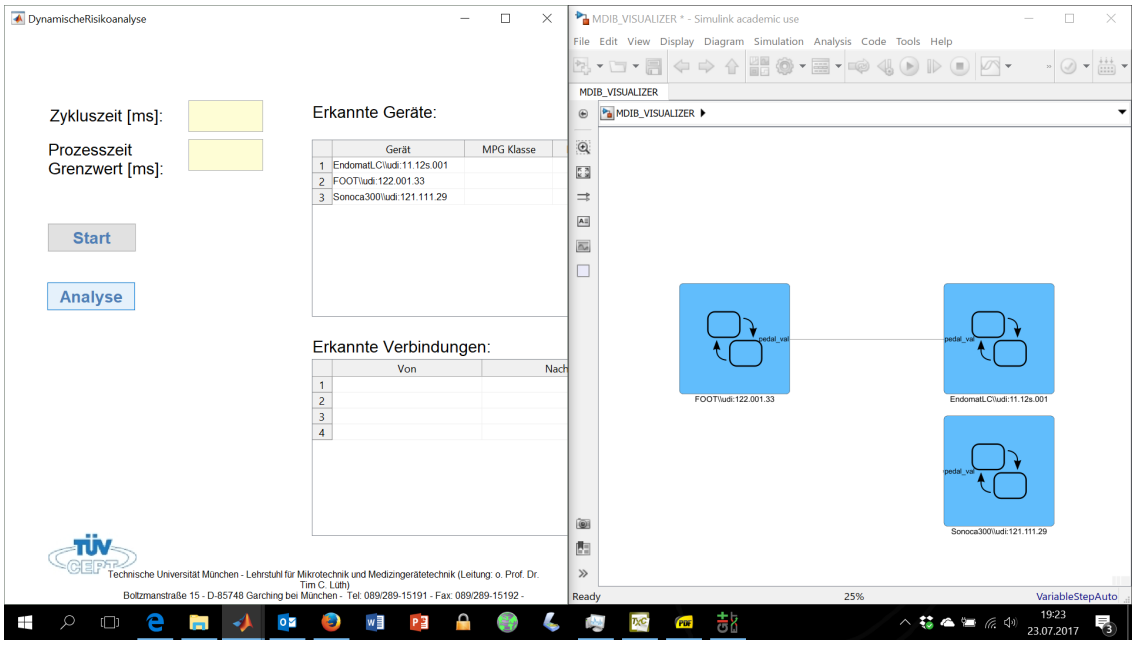


Abbildung 36 Matlab-GUI mit Verschaltungseditor. Links im Bild sind die Steuerknöpfe der GUI, rechts können die Gerätemodelle (blaue Kästen) verschaltet werden. Mit jedem Klick auf den Knopf „Analyse“ wird die Verschaltung abgelegt.

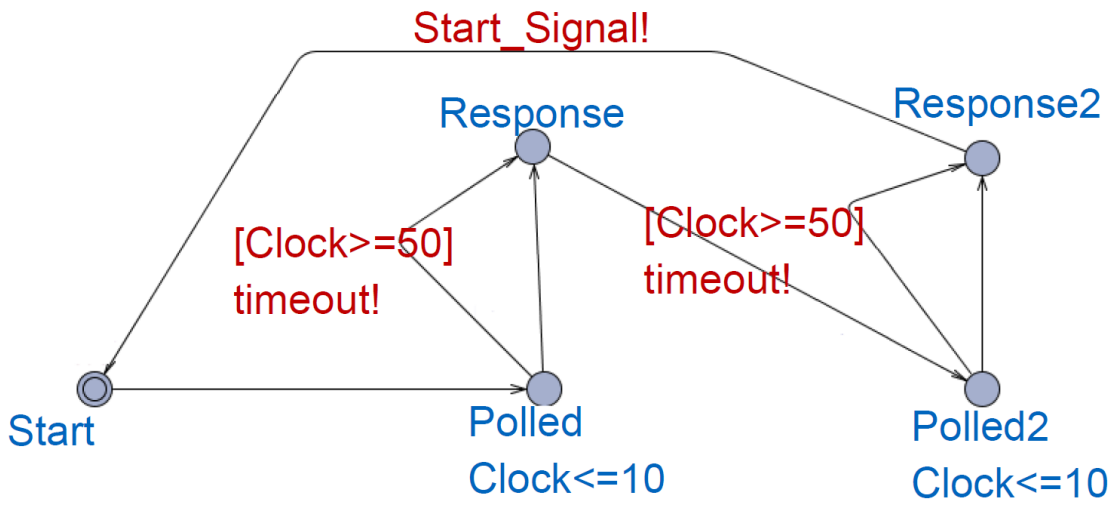


Abbildung 37 TEFA in UPPAAL. Locations werden durch graue Punkte repräsentiert, Transitionen zwischen Locations durch Pfeile. Aus Gründen der Lesbarkeit wurde die Schrift außerhalb von UPPAAL vergrößert.

tiert werden kann. Eine zusammenfassende Übersicht über das Tool ist in (Behrmann et al., 2004) beschrieben. Demnach wurde die erste Version von UPPAAL im Jahr 1995 vorgestellt (Larsen et al., 1997). Diese wird in Abständen mehrerer Jahre laufend weiterentwickelt – die aktuelle Release 4.1.19 wurde im Juli 2014 veröffentlicht. UPPAAL baut auf den Dissertationen (Kristoffersen, 1998; Pettersson, 1999; Jensen, 1999; Nielsen, 2000; Bengtsson, 2002; David, 2003; Fersman, 2003; Behrmann et al., 2003; Fehnker, 2002) auf.

5.6.2. Elemente des UPPAAL Model Checkers

UPPAAL hat eine **grafische Benutzeroberfläche**, mit deren Hilfe per Drag-and-Drop Automaten editiert werden können. Locations werden durch graue Punkte repräsentiert und können mit einem Namen, sowie der zugehörigen Invarianten beschrieben werden. *Transitionen* zwischen Locations werden, wie üblich, durch Pfeile repräsentiert (siehe Abb. 37).
Mithilfe der grafischen Benutzeroberfläche wird jeder TEFA des Netzwerks in einem eigenen ***Template*** editiert (siehe Abb. 38).
Zudem hat jedes Template eine eigene ***Template Declaration Section***, in der lokale Variablen, lokale Uhren und lokale Signale eines TEFAs eingetragen werden können.
Globale Variablen i.S.v. Abschnitt 4.6 und globale Signale, wie etwa preq, pres und SoC, werden hingegen in der **Project Declaration Section** des Gesamtsystems deklariert (siehe Abb. 39).
Bevor das System simuliert und verifiziert werden kann, müssen alle Templates instanziiert werden. Dies wird im Abschnitt ***System Declarations*** realisiert (siehe Abb. 40).
Das Abbild des so grafisch editierten Systems wird in einer unverschlüsselten *Extensible Markup Language* (XML)-Datei gespeichert, die von UPPAAL eingelesen wird, wenn das System erneut geöffnet wird. Dies wiederum ermöglicht eine automatisierte Modellgenerierung durch Generierung einer entsprechenden XML-Datei. In den folgenden Abschnitten wird dieses Vorgehen erklärt.

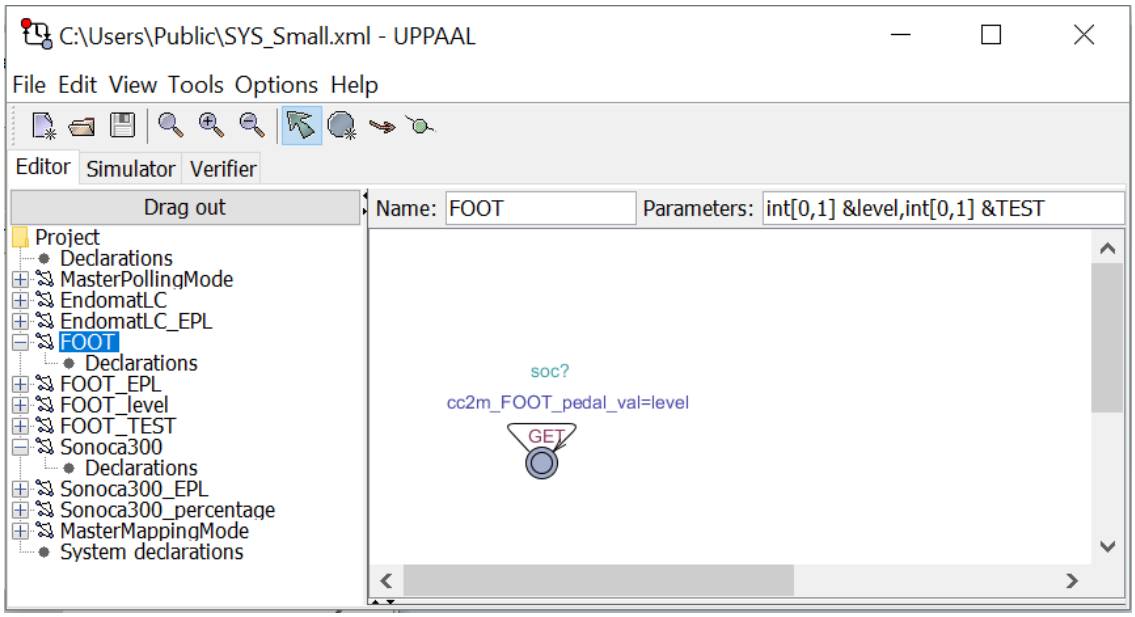


Abbildung 38 UPPAAL-Template für einen Fußschalter. Links in der Menüleiste sind alle TEFAs des Netzwerks aufgelistet. Rechts im Bild kann der markierte TEFA händisch editiert werden. TEFAs können *parametrisiert* werden, d.h. ihr Verhalten je nach übergebenen Parametern ändern. Diese Parameter können rechts neben dem Namen des Templates übergeben werden.

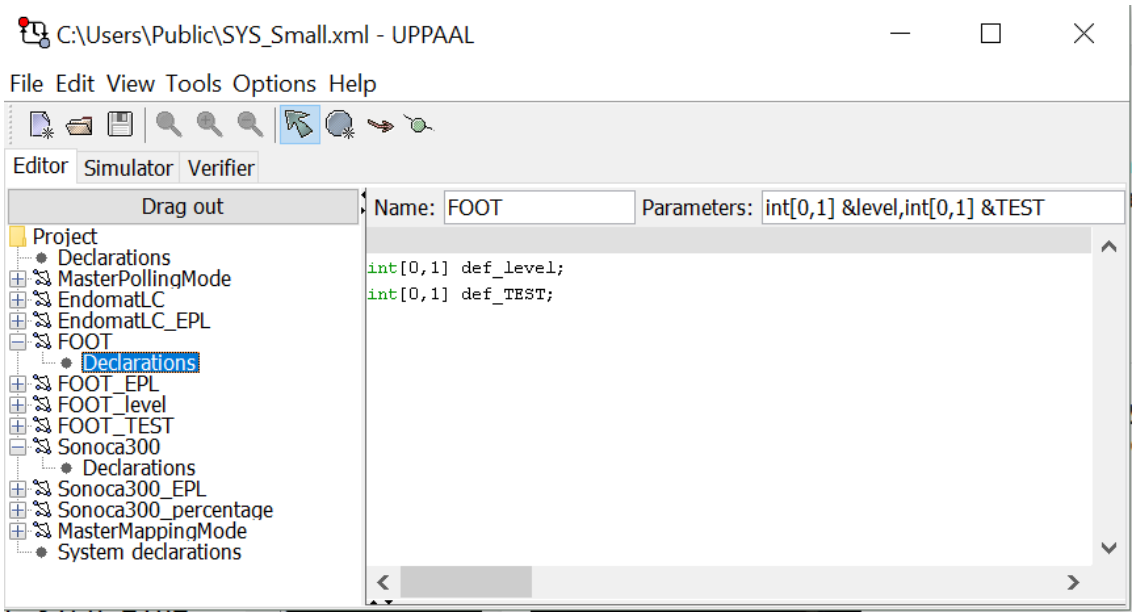


Abbildung 39 Lokale Declaration Section eines UPPAAL-Templates für einen Fußschalter. Links in der Menüleiste sind alle TEFAs des Netzwerks aufgelistet. Unterhalb jedes TEFA in der linken Spalte befindet sich eine *Declaration*-Section. Hier können lokale Variablen des TEFA deklariert werden. Globale Variablen werden in der Declaration-Section des Gesamtsystems *Project* deklariert.

5.6.3. Weitere Model Checker für temporale Logiken

Neben UPPAAL gibt es zahlreiche weitere Model Checker. Eine umfangreiche Übersicht findet sich in (Fokkink, 2007). Die wichtigsten Tools werden nachfolgend kurz beschrieben:

- Stateflow ist eine Umgebung für die Modellierung, Simulation und Verifikation kombinato-

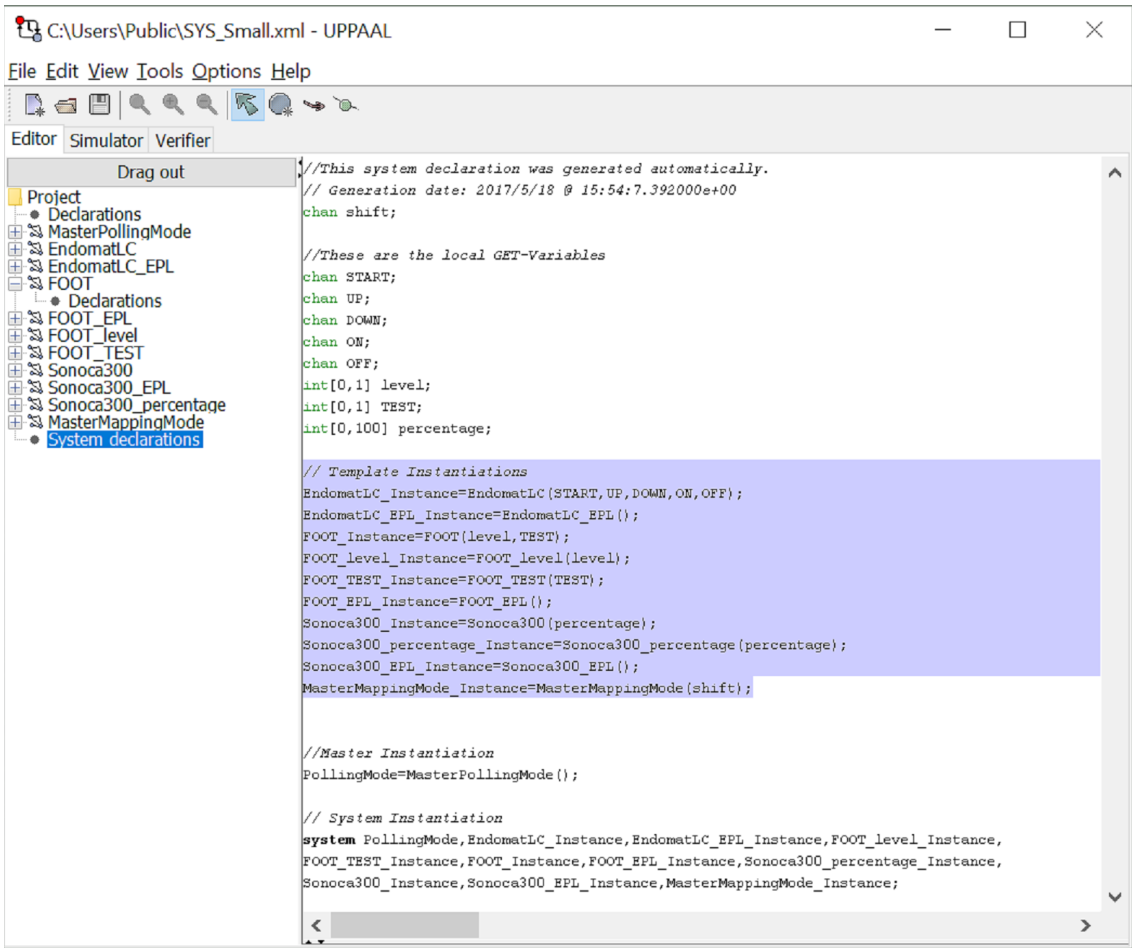


Abbildung 40 System Declaration Section eines UPPAAL-Systems. Im blau markierten Bereich werden die einzelnen Systeme ggf. parametrisiert und dann instanziiert. Im letzten Abschnitt wird das System, bestehend aus den instanziierten Templates, deklariert.

rischer und sequenzieller Entscheidungslogik auf Grundlage von Zustandsautomaten und Flussdiagrammen. Stateflow ist in Simulink eingebettet. In Stateflow können unterschiedliche Klassen von Automaten (Mealy-Automaten, Moore-Automaten, Statecharts) und weitere Entscheidungslogik (Wahrheitstabellen, Flussdiagramme) editiert, simuliert und analysiert werden. Als Verifikationstool hat sich Simulink/Stateflow jedoch nicht durchgesetzt; da Stateflow zudem eine eigene Semantik zugrunde legt, welche nur die Analyse deterministischer Automaten erlaubt, können nicht-deterministische Systeme nicht analysiert werden (Pajic et al., 2014; Dingler et al., 2016).

- SPIN (Holzmann, 1990) ist ein Tool für die Analyse von Kommunikationsprotokollen. Als Eingabe nimmt SPIN Kompositionen endlicher Automaten, die in *Protocol Meta Language* (PROMELA) beschrieben sind, und LTL-Formeln. SPIN prüft generische Eigenschaften wie Deadlocks und Erreichbarkeit. Zur Verifikation nutzt SPIN die Methode der POR. Während SPIN ursprünglich nur für endliche Automaten konzipiert war, folgte schon bald eine Erweiterung RT-SPIN für die Verifikation von Echtzeitsystemen unter Verwendung der neuen Beschreibungssprache RT-Promela (Kristoffersen, 1998; Tripakis und Courcoubetis, 1996; Fokkink, 2007).

- Symbolic Model Verifier *(SMV)* (McMillan, 1993) ist ein symbolischer Model Checker, der ROBDD verwendet um CTL-Formeln zu verifizieren. SMV wurde in den USA entwickelt und war das erste Tool, das ROBDD für Verifikationszwecke verwendet hat, wodurch Automaten mit mehr als 10^{20} Zuständen analysiert werden konnten. Bei Verletzungen formaler Anforderungen kann SMV wie UPPAAL Gegenbeispiele produzieren und ausgeben (Fokkink, 2007; Kristoffersen, 1998).

- KRONOS (Daws et al., 1996) ist ein Model Checker für die Verifikation nebenläufiger, getimter Automaten mithilfe der Temporallogik TCTL. Unter KRONOS ist das Ergebnis der Berechnung eine Bedingung, unter der die Formel erfüllt ist. Das Tool wurde im Forschungslabor VERIMAG entwickelt. Es erlaubt keine explizite Modellierung von Variablen, was jedoch durch die Modellierung entsprechender Zustände entlang des gewünschten Definitionsbereichs der Variablen umgangen werden kann. Die in Abb. 13 beschriebenen Gegenbeispiele sind in KRONOS nur für einfache Safety-Eigenschaften vorgesehen (Beyer, 2002; Kristoffersen, 1998). KRONOS wird in der Literatur vielfältig beschrieben, die letzte Version ist jedoch im Jahr 2002 ausgegeben worden.

Da Kronos unter den genannten Tools das einzige ist, das TCTL unterstützt, die Entwicklung desselben aber nicht erkennbar vorangetrieben wird, wurde in dieser Arbeit das UPPAAL-Tool genutzt.
Die Erzeugung der Gerätemodelle für UPPAAL wurde in Matlab r2016b realisiert. Für jedes Gerät wird ein UPPAAL-Template erzeugt.

5.6.4. Zuordnung von Variablen zu Templates

Der folgende Abschnitt zeigt, wie die abstrakt in Abschnitt 4.6 definierten Variablenklassen den UPPAAL-Templates aus Abschnitt 5.6.2 zugeordnet werden.

Besonderheiten bei lokalen Eingabevariablen

Lokale Eingabevariablen sollen in UPPAAL simuliert und miteinbezogen werden können. Da diese in der Realität gewissermaßen „zufällig“ erzeugt werden, muss in der Verification Toolbox ein Zufalls-Generator für alle lokalen Eingabesignale und -variablen erzeugt werden. Das UPPAAL-Tool hat hierfür das „select“-Feature vorgesehen, das einer Variable einen zufälligen Wert innerhalb eines gewissen Intervalls zuschreibt. Der Ausdruck *select='level: int[0,1]'* auf einer gegebenen Transition bedeutet etwa, dass bei Auslösung der Transition ein zufälliger Integer-Wert zwischen 0 und 1 gewählt wird, und die Variable $level$ mit diesem Wert versehen wird. In dieser Arbeit wird das „select“-Feature folgendermaßen genutzt: Ist in der Beschreibungsdatei eine lokale *Eingabevariable* deklariert, so wird ein weiteres Template mit einer einzigen Location und einer einzigen Transition erzeugt, die allein mit dem „select“-Feature beschrieben ist (siehe Abb. 41).

Zuordnung von TEFA-Bestandteilen zu UPPAAL-Sections

Template Declarations (EPL)	[LEER]
Template Declarations (Physikalischer Prozess)	Lokale Ausgabevariablen (observable, device_output, siehe Abschnitt 5.2.4). Lokale Uhren.
Template Declarations (Polling Mode)	ResponseTimes der CNs. Lokaler master_timer
Template Declarations (Mapping Mode)	[LEER]
Project Declarations	SoC-, pres- und preq-Signale. Alle globalen Variablen sämtlicher CNs. Wie in Unterabschnitt 5.2.4 beschrieben, werden dazu für Globale Variablen jeweils eine MN-Kopie und eine CN-Kopie mit den entsprechenden Präfixen generiert.
System Declarations	Initialisierung lokaler Eingabe-Variablen und lokaler Signale. Instanziierung aller Templates. Systemdeklaration.
Parameter	Lokale Eingabevariablen und Pass-Variablen. Lokale Signale und Pass-Signale.
Template	Pro CN mindestens 2 Stück: EPL-Prozess und Physikalischer Prozess. Für jede Eingabevariable und für jedes Eingabesignal des TEFAs wird ein zusätzliches Template erzeugt. Für den MN zwei separate Templates: Polling Mode und Mapping Mode.

Tabelle 12 Zuordnung von Elementen der TEFAs zu den UPPAAL-Sections.

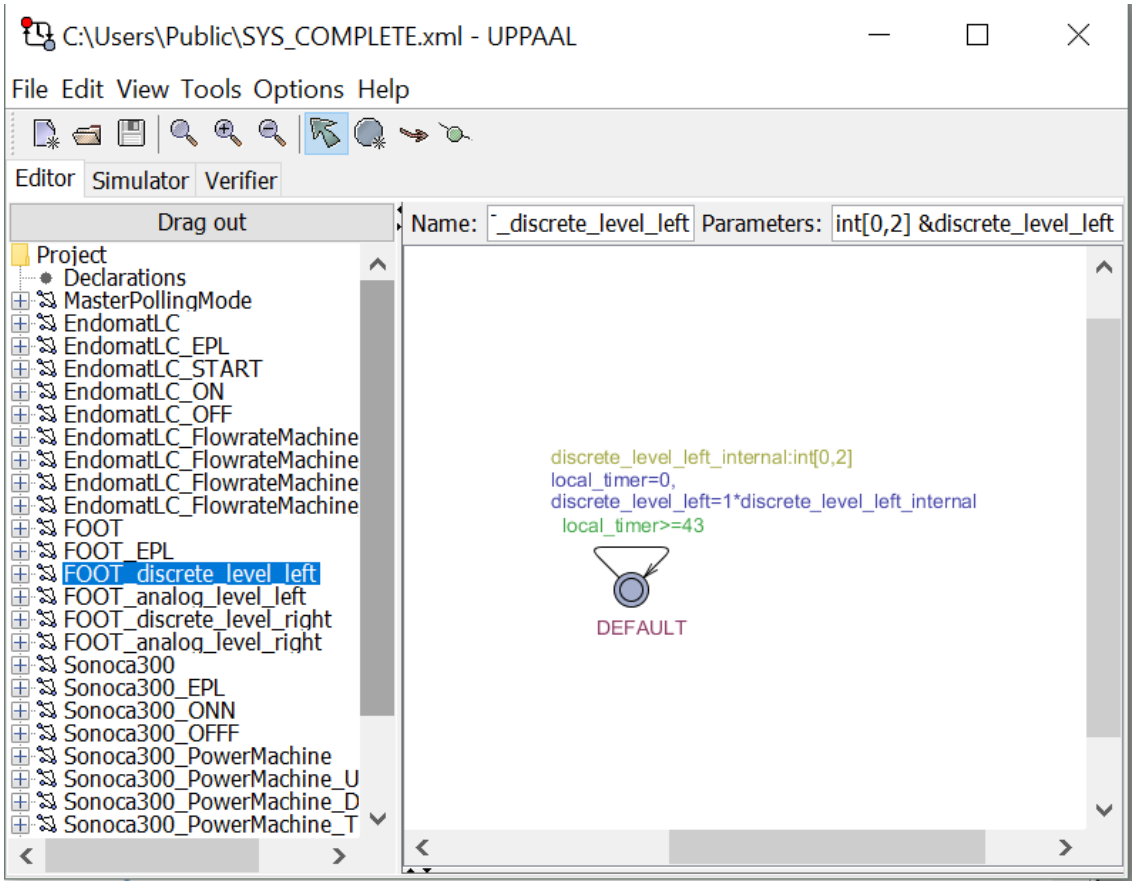


Abbildung 41 Automatisch erzeugter Generator für lokale Eingabevariablen. Durch Angabe der maximalen Frequenz in der Beschreibungsdatei, kann zusätzlich eine Uhr erzeugt werden, die den Generator an der zu schnellen Ausführung hindert.

5.6.5. Übersicht über die wichtigsten Funktionen der Modellgenerierung

Im Folgenden werden die wichtigsten Funktionen der Verification Toolbox kurz dargestellt. Zugunsten der Lesbarkeit sind einige Funktionsnamen abgeändert.

Die Funktion `parse_mdib`

Die Funktion liest die Eingabe-Beschreibungsdatei ein und generiert daraus den in Tabelle 13 beschriebenen Geräte-Struct. Zur Syntax der Beschreibungsdatei siehe Abschnitt 5.2.

Beschreibung der Funktion parse_mdib

Eingabe	Filename der Beschreibungsdatei eines Geräts
Ausgabe	Struct mit folgenden Feldern: • Template (Geräteinformationen) • Signals • Locations • Transitions • Clocks • Metrics (Gerätevariablen)

Tabelle 13 Beschreibung der Funktion parse_mdib.

Das Skript `generate_stateflow_blocks`

Dieses Skript durchsucht ein gegebenes Verzeichnis nach allen Gerätebeschreibungsdateien. Diese sind durch das MDIB-Präfix im Dateinamen zu identifizieren. Ist die Suche abgeschlossen, werden durch wiederholten Aufruf der Funktion parse_mdib Geräte-Structs generiert. Aus jedem resultierenden Struct wird ein Stateflow-Chart erzeugt – dabei werden nur diejenigen Datenquellen und -senken in das Stateflow-Modell übernommen, die über das Echtzeit-Bussystem ausgetauscht werden.

Die Funktion `edit_mapping`

Die Funktion edit_mapping öffnet das Matlab-*Graphical User Interface* (GUI) aus Abb. 36, in dem sämtliche Gerätebeschreibungen des Quell-Verzeichnisses aufgeführt werden. Ein Klick auf *Start* öffnet die Simulink-Umgebung, in der die Gerätebeschreibungen mithilfe der

Funktion generate_stateflow_blocks visualisiert werden. Sobald die Geräte per Drag-and-Drop verschaltet sind, wird diese durch einen Klick auf den Button *Analyse* in einer Textdatei niedergeschrieben. Auf diese Weise können wiederholt Verschaltungen generiert werden. Nach Schließen des GUI werden durch die Prozedur `merge_all_maps` alle so generierten Textdateien in der Syntax aus Abschnitt 5.3 zusammengeführt. Dabei ist zu beachten, dass die erste editierte Verschaltung als die Default-Verschaltung übernommen wird.

Die Funktion `parse_mapping`

Die Funktion liest die Beschreibung der Verschaltung ein und generiert daraus eine Zelle an Structs. Jeder Struct repräsentiert eine Zuordnung von Datenquelle zu Datensenke. Jede Verschaltung wird so durch eine Folge an Structs beschrieben, die zeilenweise aneinandergereiht werden. Eine Verschaltung entspricht also eine Spalte der Ausgabezelle. Siehe Abschnitt 5.3 zur Syntax der Verschaltungsspezifikation. Jede Verschaltung V besteht aus m_V Tupeln der Form „Senke=Quelle“. In Tabelle 14 ist n die Anzahl an definierten Verschaltungen und $m := max\left\{m_V : \text{V Verschaltung}\right\}$.

Beschreibung der Funktion parse_mapping

Eingabe	Filename der Spezifikation der Verschaltung
Ausgabe	$m \times n$ Zelle aus Structs mit Feldern: • Source (Datenquelle) • Sink (Datensenke)

Tabelle 14 Beschreibung der Funktion parse_mapping.

Die Funktion `generate_mappingmode`

Diese Funktion generiert einen TEFA aus der Zelle, die durch die Funktion parse_mapping generiert wurde (siehe Tabelle 14). Die Verschaltung aus der ersten Spalte der Zelle gilt dabei als die Start-Verschaltung. Die Vorgehensweise zur Erzeugung des TEFA ist in Abschnitt 4.4 detailliert beschrieben. Der TEFA wird in derselben Syntax wie die Gerätespezifikationen in eine Textdatei geschrieben, um ihn der Funktion parse_mdib zugänglich zu machen. Die Textdatei, welche die Verschaltungen beschreibt, kann auch aus Stateflow heraus in Verbindung mit den Funktionen `map2file` und `merge_all_maps` automatisiert generiert werden.

Die Funktion `uppaalgen`

Aufgabe der Funktion uppaalgen ist es, die Information der Gerätebeschreibungen aus Tabelle 13 in die UPPAAL-Sektionen aus Tabelle 12 einzupflegen. Durch diese Funktion werden

Beschreibung der Funktion generate_mappingmode

Eingabe	Filename der Spezifikation der Verschaltung
Ausgabe	Textdatei, die den Mappingmode (siehe Abschnitt 4.4) als TEFA beinhaltet.

Tabelle 15 Beschreibung der Funktion generate_mappingmode.

für ein Gerät der EPL-Prozess und der physikalische Prozess generiert. Zudem werden gegebenenfalls Eingabe-Generatoren (siehe Abschnitt 5.6.4) erzeugt und System- sowie Projektdeklaration mit den Geräte-Informationen versehen.

Das Skript `create_model_for_dir`

Das Skript generiert ein UPPAAL-Modell als XML-Datei auf Basis aller Beschreibungsdateien, die im spezifizierten Verzeichnis (siehe Eingabe) liegen. Diese werden am Präfix 'mdib_' automatisch erkannt. Zudem wird der Mapping Mode generiert und in das System aufgenommen (siehe Funktion generate_mappingmode). Anschließend wird mit der Funktion generate_masterfile der Polling Mode generiert und in das System aufgenommen. Das Skript startet automatisch das UPPAAL-Tool und ermöglicht erstmals die Begutachtung, Simulation und Verifikation des Gesamtsystems.

Beschreibung des Skripts create_model_for_dir

Eingabe	Name des Verzeichnisses, in dem sich alle Beschreibungsdateien befinden.
Ausgabe	XML-Datei mit der vollständigen Systemspezifikation.

Tabelle 16 Beschreibung des Skripts create_model_for_dir.

Die Funktion `launch_uppaal`

Möchte der Anwender der Verification Toolbox das System vor der formalen Verifikation simulieren wollen, kann durch die Funktion launch_uppaal die UPPAAL-Applikation gestartet werden. Diese lädt die Systemspezifikation und ermöglicht so neben deren Simulation auch die Prüfung, ob alle gewünschten Module in das System aufgenommen wurden, sowie die Modifikation des Systems. Ein modifiziertes System kann unter anderem die Berechnungsdauer signifikant reduzieren und für erste Plausibilitätsüberlegungen herangezogen werden (siehe Kapitel 7.1).

Die Funktion `verify_properties`

Diese Funktion wird nach der vollständigen Systemspezifikation aufgerufen. Die Funktion fertigt ein Protokoll an, in dem die Testergebnisse für jede angeführte Anforderung niedergeschrieben werden. Als Eingabeargument kann spezifiziert werden, ob im Falle der Nicht-Erfüllung von Anforderungen ein Gegenbeispiel aufgeschrieben werden soll und in welcher Reihenfolge die Verifikation durchgeführt werden soll (*Depth-First-Search* vs. *Breadth-First-Search*). UPPAAL ermöglicht auch die Einstellung weiterer Prüfspezifika, die jedoch in der Verification Toolbox nicht implementiert sind.

5.6.6. Erzeugung des Managing Node

Der MN wird als ein Paar der nebenläufigen TEFAs Polling Mode und Mapping Mode modelliert (siehe Abschnitt 4.2.1).

Der Pollingmode

Der Polling Mode ist derjenige TEFA, der für das sukzessive Abfragen aller CNs zuständig ist. Der Algorithmus zur Erzeugung des Polling Mode ist in Abschnitt 4.2.1 erklärt. Der Pollingmode für das bekannte Setup aus Fußschalter, Saug-/Spülpumpe und US-Dissektor ist in Abb. 42 aufgezeichnet. In der *Declaration-Section* des Polling Mode werden automatisch die Antwortzeiten (*ResponseTimes*) der CNs eingetragen (siehe Tabelle 11) und der *master_timer* initiiert. Der master_timer startet mit jedem Zyklus neu. Nachdem die Rückgabewerte eines CNs abgefragt wurden, wartet der MN höchstens bis zum Ablauf der vom CN abhängigen ResponseTime. Antwortet der CN innerhalb dieser Zeitschranke, geht der MN zum nächsten CN über. Andernfalls werden alle Rückgabewerte des (verspäteten) CN auf FALSE gesetzt (wie in Abb. 42 beim zweiten CN.). Jeder CN, der auf die entsprechenden Werte angewiesen ist bekommt auf diese Weise mitgeteilt, dass die Werte ungültig sind.

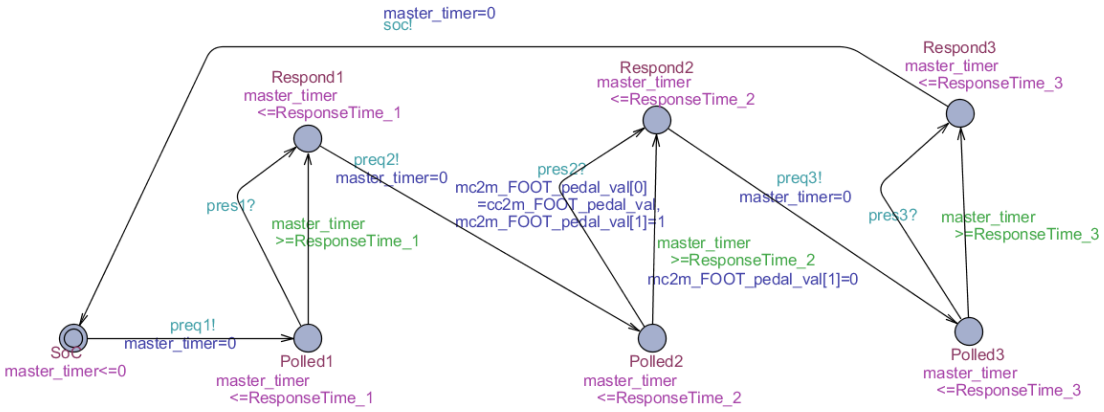


Abbildung 42 Automatisch erzeugter Pollingmode. Neben pres- und preq-Signalen werden automatisch Timeouts generiert, um festzustellen wann ein CN ausgefallen ist. Die Timeouts werden aus der Beschreibungsdatei des jeweiligen CN ausgelesen und in die Declaration Section des Polling Mode geschrieben. Die Elemente im Bild sind zugunsten der Lesbarkeit händisch neu arrangiert.

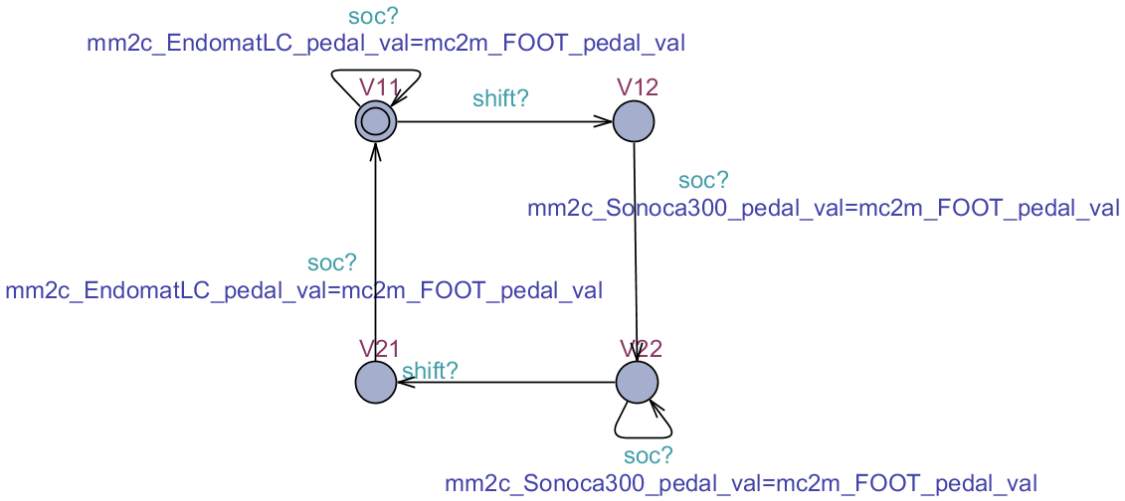


Abbildung 43 Mapping Mode für zwei Verschaltungen. Der TEFA startet in der Location V11. Mit jedem neuen Zyklus (soc?) werden die Daten in der Senke aktualisiert – in diesem Fall wird dem EndomatLC am Fußpedaleingang der Fußpedalwert übergeben. Bei einem shift-Signal, das z.B. per Touch ausgelöst werden kann, wechselt der TEFA in einen Zwischenzustand und vollzieht die neue Datenzuordnung beim Start des nächsten Zyklus. In diesem Fall wird der Fußschalter-Wert auf den Eingang im Sonoca300 gelegt.

Der Mapping Mode

Der Mapping Mode ist derjenige TEFA, der für die Zuordnung von Datenquellen zu -senken zuständig ist. Er wird aus der Textdatei aus Abschnitt 5.3 erzeugt, die wiederum mithilfe des Stateflow-Generators aus Abschnitt 5.5 generiert werden kann. Der TEFA in Abb. 43 wurde beispielsweise aus der Verschaltungsspezifikation in Abschnitt 5.3 automatisch generiert. Ist das System vollständig modelliert, kann ein gegebener Anforderungskatalog in Bezug auf dieses System verifiziert werden. In Abb. 44 ist das UPPAAL-GUI mit dem Anforderungskatalog abgebildet. Darüber können Anforderungen geprüft, editiert, entfernt oder in Umgangssprache angezeigt werden. In dieser Arbeit wurde eine Matlab-Schnittstelle realisiert, die diese Funktionen implementiert.

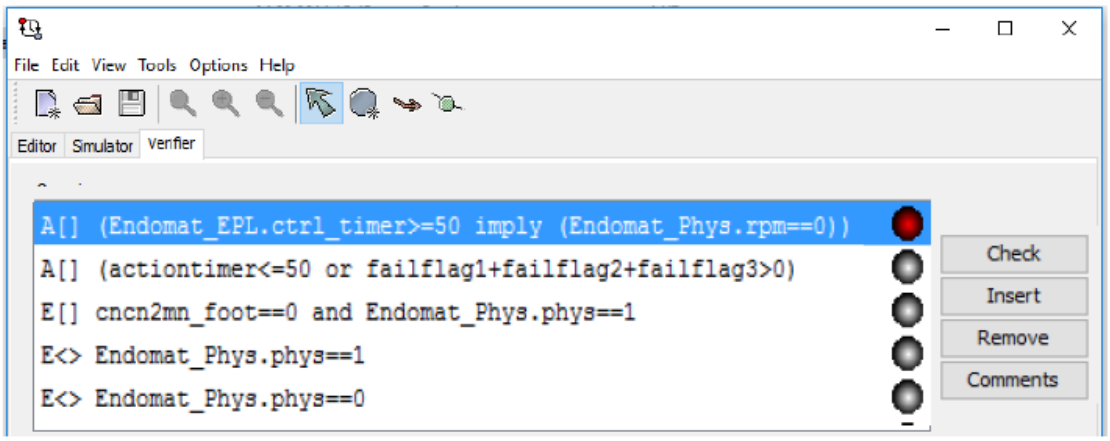


Abbildung 44 Verifikation der Anforderungen auf Knopfdruck im UPPAAL Tool. Die Anforderungen sind zeilenweise aufgelistet. Ein rotes Lämpchen bedeutet die Widerlegbarkeit der Anforderung, ein grünes Lämpchen die Gültigkeit. Graue Lämpchen bedeuten, dass die Anforderung noch nicht geprüft wurde.

5.7. Modellierung des OR.Net Echtzeitbussystems

Im folgenden Kapitel werden mithilfe der in dieser Arbeit entwickelten Methode beispielhafte Risiken des Echtzeit-Bussystem untersucht. Bevor damit begonnen werden kann, bedarf es einer lückenlosen Spezifikation verschiedener Module des Echtzeit-Bussystem- dies soll in diesem Abschnitt vorbereitend geleistet werden. An dieser Stelle sei angemerkt, dass die Modellierung der Module eigenständig durch den Autor auf Basis empirisch wahrgenommenen Geräteverhaltens erfolgt ist. Die Modellierung dient dazu, die Ausdruckskraft der TEFAs in Verbindung mit der entwickelten Verification Toolbox zu offenbaren und hat nicht den Anspruch, die volle Funktionalität jedes Hersteller-Moduls akkurat abzubilden. Der Echtzeit--Bussystem besteht aus

- einem MN (vollautomatisch erzeugt, siehe Abschnitte 4.2.1, 4.4),
- einem Fußschalter des Herstellers steute Schaltgeräte GmbH & Co. KG *(steute)* (Abschnitt 5.7.1)
- einem US-Dissektor „Sonoca 300“ des Herstellers Söring GmbH *(Söring)* (Abschnitt 5.7.2),
- einem *Hochfrequenz-Gerät* (HF-Gerät)-Gerät „Autocon“ des Herstellers *KARL STORZ GmbH & Co. KG* (Karl Storz),
- einem HF-Gerät-Gerät „Maxium“ des Herstellers Gebrüder Martin GmbH & Co. KG *(KLS Martin)*,
- einem HF-Gerät-Gerät „ME-MB3“ des Herstellers KLS Martin,
- einer Motoreinheit „Unidrive Neuro“ für aktive Instrumente (Fräser, Shaver, Knochensäge) des Herstellers Karl Storz,
- einer Saug-/Spülpumpe „Endomat LC“ des Herstellers Karl Storz (Abschnitt 5.7.3),
- einem Navigationsrechner „Navigation Panel Unit“ des Herstellers Ergosurg GmbH *(Ergosurg)*,
- einer Navigationskamera „Vicra“ des Herstellers Northern Digital Inc. *(NDI)*,
- einem IONM „C2“ des Herstellers inomed Medizintechnik GmbH *(inomed)*,
- einem *Funktionsmodul* „Navigated Control“ (Eigenentwicklung MiMed),
- einem Funktionsmodul „NeuroControl“ (Eigenentwicklung MiMed, Abschnitt 5.7.4) und
- einem Funktionsmodul „NaviMerge“ (Eigenentwicklung MiMed).

Um die Ausdruckskraft der Beschreibungsmethode und der Verification Toolbox zu verdeutlichen, genügt es den MN, den Fußschalter, die Saug-/Spülpumpe, den US-Dissektor und das NeuroControl-Modul detailliert zu beschreiben.

5.7.1. Modellierung des Fußschalters

Der Fußschalter der Firma steute am Münchener Demonstrator verfügt über zwei Pedale, welche jeweils Analog-Werte von 0 (nicht gedrückt) bis 100 (durchgedrückt) erzeugen. Zudem übermittelt jedes Pedal diskrete Werte der Menge $\{0, 1, 2\}$, je nachdem, wie stark es betätigt wird. Der Fußschalter wird als System nebenläufiger TEFAs modelliert – zu modellieren sind der physikalische Prozess (siehe Abschnitt 4.3.2), der EPL-Prozess (Abschnitt 4.3.1) und die jeweiligen Generatoren-Prozesse (Abschnitt 5.6.4) für das stufenlose und das diskrete Signal beider Pedale.

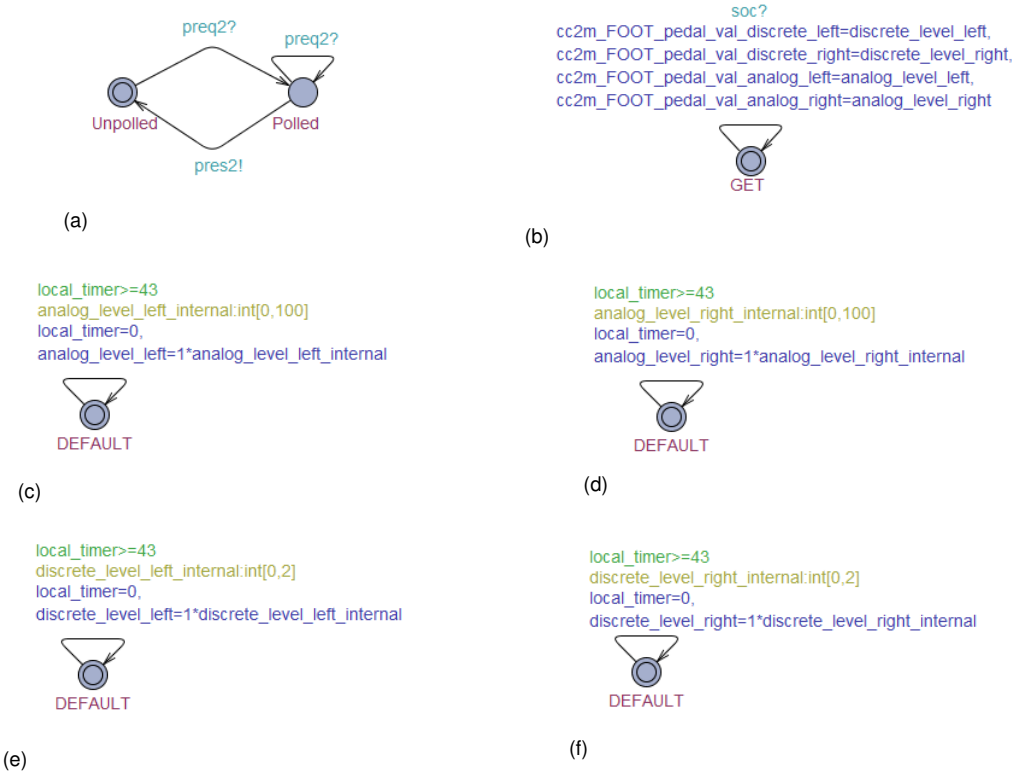


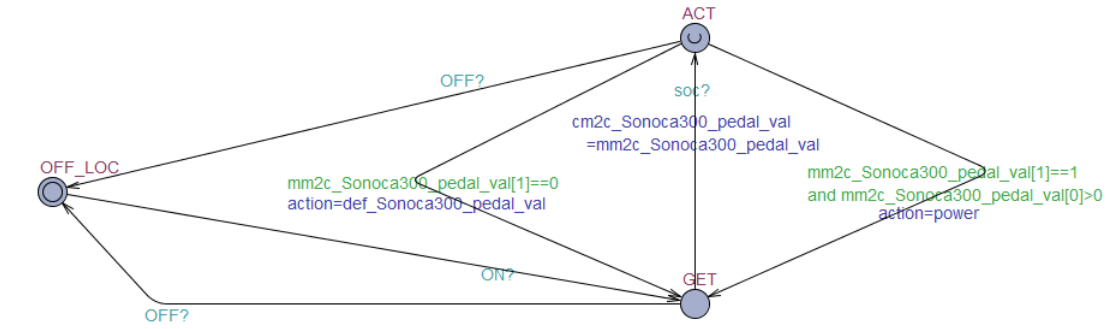
Abbildung 45 Nebenläufige Prozesse des zweipedaligen Fußschalters. In Abb. 45a ist der EPL-Prozess zu sehen, der sich nur anhand des Index 2 von anderen Geräten unterscheidet. Der physikalische Prozess 45b reagiert auf das Start-of-Cycle-Signal (soc?) und übermittelt alle analogen und diskreten Werte der beiden Pedale. In Abb. 45c bis 45f sind die Signalgeneratoren, die durch die Verification Toolbox automatisch erzeugt werden, zu sehen. Beispielhaft wurde hier eine maximale Änderungsfrequenz von 23 Hz angegeben, was durch eine automatisch generierte Uhr modelliert wird, die erst ab 43 ms eine Änderung des Wertes zulässt. In der gelben Zeile wird zufällig eine Wert zwischen 0 und 100 (analog) bzw. 0 und 2 (diskret) gewählt und – nach einem Reset der Uhr – als Fußpedalwert übernommen (letzte Zeile).

5.7.2. Modellierung des USD's

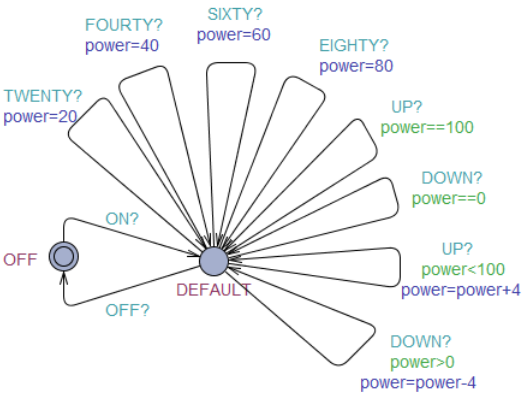
Der US-Dissektor (genauer: sein US-Generator) verfügt über Druckknöpfe zum Ein- und Ausschalten, sowie zum Einstellen der prozentualen US-Leistung für die Dissektion. Diese kann absolut über Knöpfe in den Stufen 20%, 40%, 60% und 80%, oder relativ zur aktuellen Leistung über zwei Knöpfe (+ und -) in 4er-Prozentschritten eingestellt werden. Zusätzlich zur Dissektion kann mithilfe des Generators auch gespült und Gewebe abgesaugt werden. Diese Funktionen wurden hier aus Platzgründen nicht modelliert, sie unterscheiden sich nur geringfügig von denen der modellierten Saug-/Spülpumpe. Ebenso wurde auf die Modellierung der Display-Anzeige, sowie der Lautstärkeregelung und weiteren Nebenfunktionen des US-Dissektor verzichtet. In dieser Arbeit wird die Funktion des US-Dissektor durch zwei nebenläufige TEFAs modelliert: Ein TEFA modelliert die „Hauptfunktion", nämlich die Verarbeitung und Umsetzung des Fußschalter-Wertes, der andere TEFA modelliert das Setup – hierzu wird die Einstellung der prozentualen US-Leistung abhängig vom aktuellen Wert und den eingehenden Signalen modelliert. Hinzu kommen wie zuvor der EPL-Modus und die Signalgeneratoren für alle Druckknöpfe. Das Ergebnis aller modellierten Untermodule ist in Abb. 46 festgehalten.

5.7.3. Modellierung der Saug-/Spülpumpe

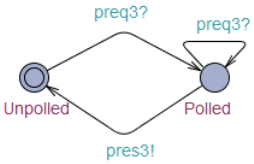
Die Funktionalität der Saug-/Spülpumpe wird – ähnlich wie beim US-Dissektor– in Haupt- und Nebenfunktionen aufgeteilt und diese gesondert als nebenläufige TEFAs modelliert. Die Saug-/Spülpumpe verfügt über zwei Betriebsmodi – den *Normal Mode* und den *High Mode* – in denen sie jeweils unterschiedliche, voreingestellte Fördermengen spült bzw. absaugt. Zwischen den Modi kann durch einen zweistufigen Fußschalter, wie er in Abb. 45 modelliert ist, umgeschaltet werden. Anders als der US-Dissektor, muss die Saug-/Spülpumpe vor der Benutzung per Knopfdruck *initialisiert* werden. Ist dies geschehen, so wechselt sie in den Zustand *READY* und beginnt, die Fußschalter-Signale umzusetzen. Nebenläufig zu dieser Hauptfunktion kann im Setup-Prozess, modelliert durch einen eigenen TEFA, die Soll-Förderrate für beide Modi eingestellt werden. Ein Druckknopf auf der Bedienoberfläche der Saug-/Spülpumpe ermöglicht hierbei das Umschalten zwischen den Modi. Zusätzlich zu Haupt- und Setup-Prozess werden erneut EPL-Prozess und die Signalgeneratoren automatisch mit modelliert – das Ergebnis ist in Abb. 47 zu sehen.



(a)



(b)



(c)

(d) (e) (f) (g)

(h) (i) (j) (k)

Abbildung 46 Nebenläufige Prozesse des US-Dissektors. Die obere Abbildung zeigt den Hauptprozess, in dem nach Einschalten auf das soc-Signal gewartet und anschließend – je nachdem, ob der Wert des Fußschalters gültig ist (1) oder nicht (0) – der Soll-Wert (power) umgesetzt (action). In Abb. 46b ist der Setup-Prozess modelliert, der die Soll-Leistung auf Knopfdruck speichert. Abb. 46d bis 46g zeigen die Generatoren für Knopfdrucksignale der absoluten Werte, Abb. 46h und 46i diejenigen der relativen Werte. Die restlichen Generatoren modellieren das Ein- und Aus-Signal.

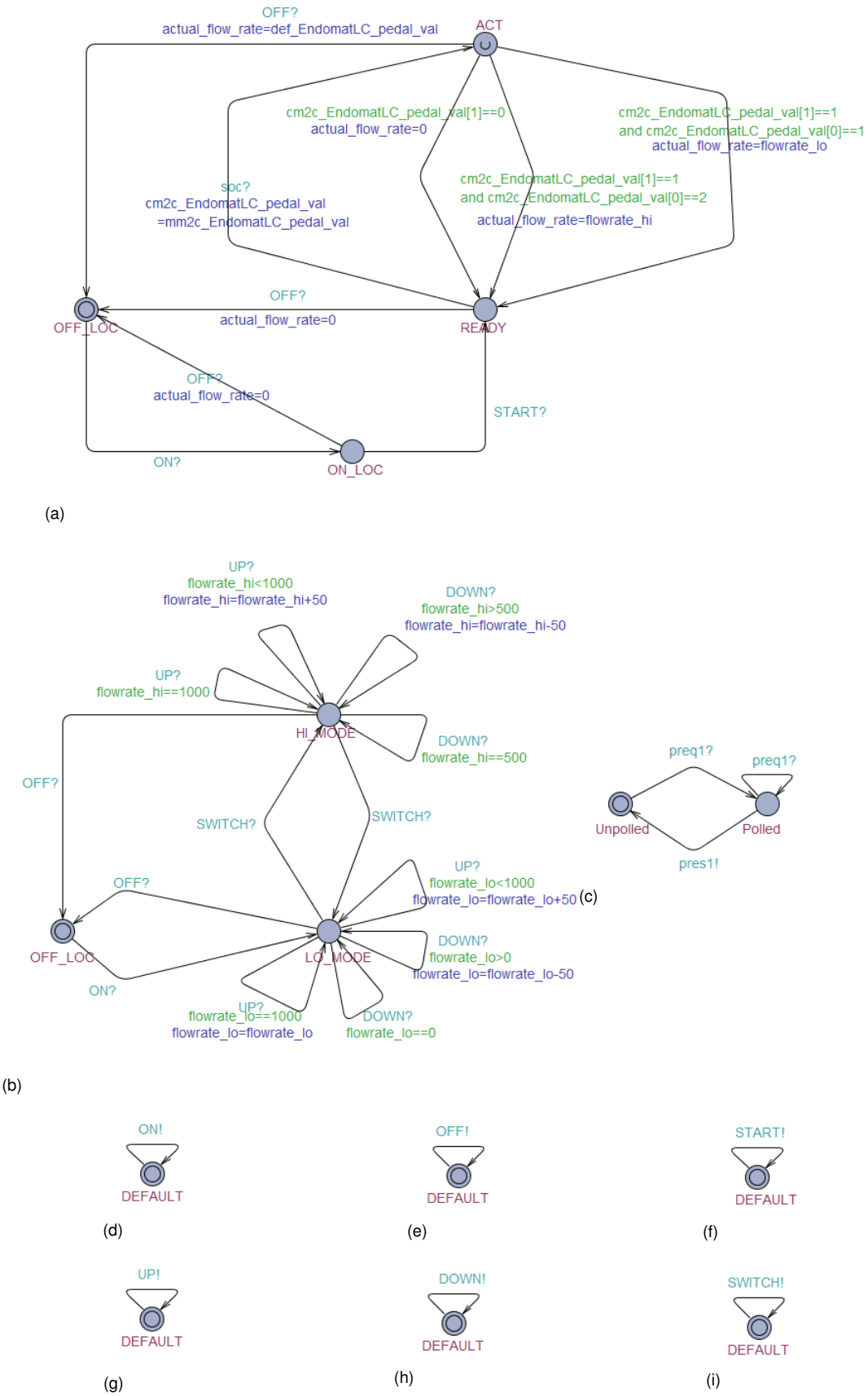


Abbildung 47 Nebenläufige Prozesse der Saug-/Spülpumpe. In Abb. 47a ist der Hauptprozess modelliert. Dieser startet in der OFF-Location (OFF_LOC) und wechselt nach dem Einschaltsignal ON und der Initialisierung START in den Modus READY. Nach Eingang des soc-Signals realisiert der Hauptprozess je nach Fußschalter-Wert voreingestellte Förderraten. In Abb. 47b ist der Setup-Prozess abgebildet. Hier wird zwischen den Modi LO_MODE und HI_MODE unterschieden – der Wechsel wird durch das Switch-Signal ausgelöst. Aufbauend auf (Dingler et al., 2017).

5.7.4. Modellierung von NeuroControl

Das Funktionsmodul NeuroControl am Münchener Echtzeit-Demonstrator dient dazu, Signale eines Neuromonitors zu verwerten und auf dieser Basis die Steuersignale eines Fußschalters für einen US-Dissektor entweder durchzuleiten, oder nicht. In der derzeitigen Implementierung sind zwei binäre Werte für die Entscheidung bedeutend:

Der Wert `current_confirm` der angibt, ob die Neuromonitoring-Sonde korrekt angebracht ist und der Wert `peak_detection`, der angibt, ob eine kritische Nervenantwort vorliegt. Der Wert des Fußschalters wird exakt dann durchgeleitet, wenn `current_confirm` $== 1$ und `peak_detection` $== 0$ – andernfalls wird der Wert auf null gesetzt. Das NeuroControl-Modul wird als System nebenläufiger TEFAs modelliert – um die Modellierung einfach zu halten, wird in dieser Arbeit angenommen, dass die beiden binären Werte zufällig erzeugt werden – in der Realität stammen diese vom Neuromonitor. Zu modellieren sind daher der physikalische Prozess (siehe Abschnitt 4.3.2), der EPL-Prozess (Abschnitt 4.3.1) und die jeweiligen Generatoren-Prozesse (Abschnitt 5.6.4) für `current_confirm` und `peak_detection`. Das Resultat der Modellierung ist in Abb. 48 zu sehen.

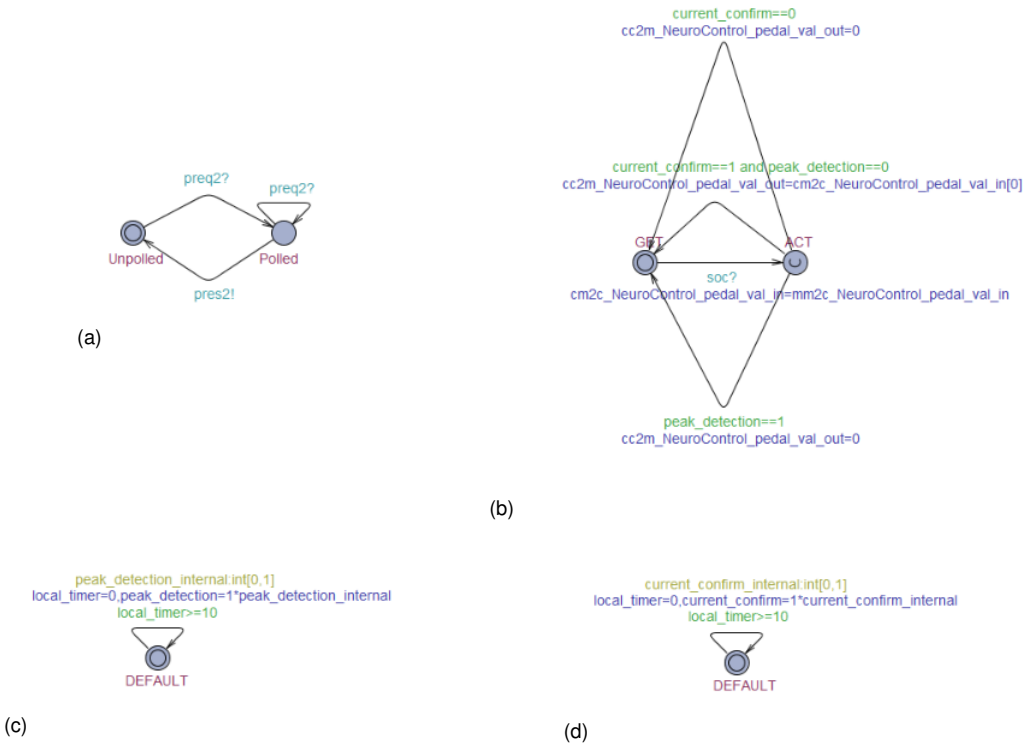


Abbildung 48 Nebenläufige Prozesse von NeuroControl. In Abb. 48a ist der EPL-Prozess zu sehen. Dieser unterscheidet sich nur anhand des Index 2 von anderen Geräten. Der physikalische Prozess 48b reagiert auf das Start-of-Cycle-Signal (soc?) und empfängt den diskreten Wert des Fußschalters. In Abb. 48c und 48d sind die Generatoren für die binären Werte `current_confirm` und `peak_detection`, die durch die Verification Toolbox automatisch erzeugt werden, zu sehen. Beispielhaft wurde hier eine maximale Änderungsfrequenz von 100 Hz angegeben, was durch eine automatisch generierte Uhr modelliert wird, der erst ab 10 ms eine Änderung des Wertes zulässt.

6. Drei beispielhafte Szenarien

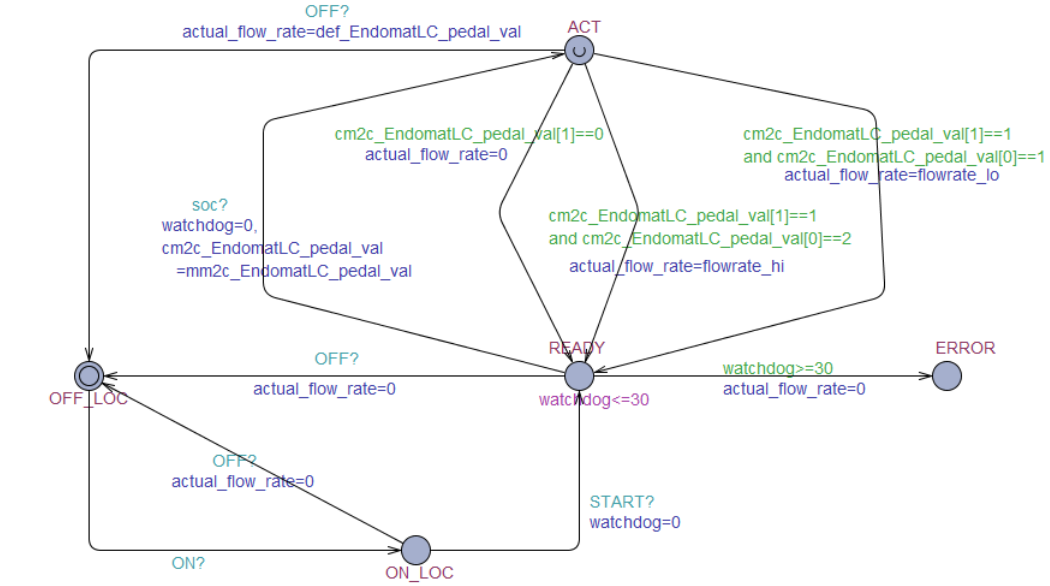
In diesem kurzen Kapitel werden exemplarische Anwendungsfälle für die Verification Toolbox vorgestellt. Dazu werden typische Fragestellungen des OR.Net-Echtzeit-Demonstrators herangezogen, an dem im Rahmen des BMBF-Projekts OR.Net die Machbarkeit der offenen Übertragung echtzeitkritischer Daten in der Medizintechnik nachgewiesen wurde. Zu den typischen Funktionalitäten, die über das Echtzeit-Bussystem realisiert werden können, gehören insbesondere die Auslösung und Regelung chirurgischer Instrumente sowie die Übertragung von Sensordatenflüssen wie Navigations-und Neuromonitoringdaten. Für die nun folgenden Abschnitte werden durchgehend die Gerätemodelle aus Abschnitt 5.7 verwendet.

6.1. Einhaltung zeitlicher Anforderungen – Anwenderanforderung

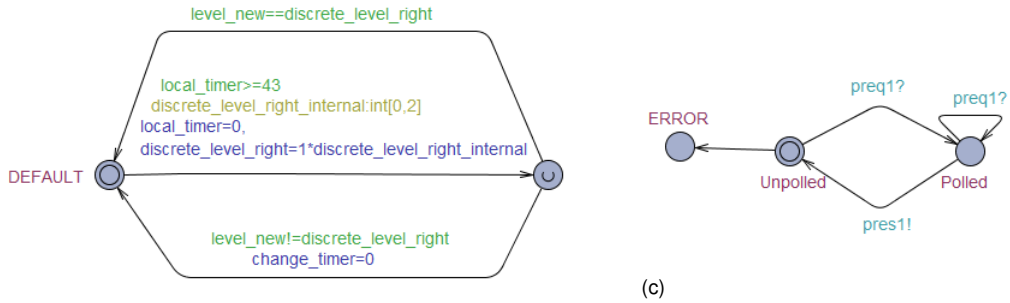
Der erste Fall der automatisierten Verifikation (angelehnt an (Dingler et al., 2017)) beschäftigt sich mit der Überschreitung von Echtzeitbedingungen. In einem Eintrag in die *U.S. Food and Drug Administration* (FDA) Manufacturer and User Facility Device Experience *(MAUDE)*-Datenbank schildert ein klinischer Anwender, dass ein HF-Gerät wiederholt für 3-4 Sekunden weiter Energie absonderte, nachdem er vom Fußschalter gestiegen war (FDA, 2016a). Für den Demonstrator wurde daher folgendes Experiment durchgeführt:
Fragestellung: Schaltet die Saug-/Spülpumpe verlässlich innerhalb von 50 ms ab, wenn der Anwender vom Fußschalter steigt?
Beteiligte Akteure: Rechtes Fußpedal (diskrete Werte), Saug-/Spülpumpe (alle Module), MN (alle Module).
Anforderungsformel: $\forall\square\left[(\text{change\_timer} > 50) \wedge (\text{pedal\_val} == 0) \Rightarrow (\text{flow\_rate} = 0)\right]$.
Analyseergebnis: Die Anforderung konnte **nicht** verifiziert werden. Die folgende Abfolge an Ereignissen führt zur Nicht-Erfüllung der Anforderung:

1. Der Anwender steige auf den Fußschalter und bleibe auf diesem mindestens bis zum Ende des EPL-Zyklus stehen.
2. Die Saug-/Spülpumpe löst darauf hin aus.
3. Vor Abfrage des nächsten Fußschalter-Wertes (preq): Netzwerkausfall.
4. Der Anwender steige nun vom Fußschalter herunter.
5. Die Saug-/Spülpumpe bleibt unverändert aktiviert.

Statistisches: Verifikationsdauer: 51,8 Sekunden. Spitzenbelastung des RAM: 250 MB.
Folgemaßnahmen: Die obige Abfolge an Ereignissen legt nahe, dass der Ausfall der Kommunikationsschnittstelle der Saug-/Spülpumpe zum unerwünschten Ereignis führen kann. Daher wurde für jedes Modul des Demonstrators eine Uhr (*watchdog*, siehe Abb. 49a) implementiert, welche prüft, wie häufig die Daten aus der Kommunikationsschnittstelle über-



(a)



(c)

(b)

Abbildung 49 Überarbeitete Modelle für das Verifikationsexperiment. Durch die Einführung zusätzlicher Uhren (Fußschalter, Abb. 49b) und Fehlerlocations (Pumpe, Abb. 49c) kann die Anforderung automatisiert geprüft werden. Das obige Bild 49a zeigt die überarbeitete Version der Implementierung des Konnektors der Saug-/Spülpumpe, nachdem die Verifikation ergeben hat, dass die Echtzeit-Anforderung nicht erfüllt ist. Diese Version des Endomat, die nach 30 ms ohne soc!-Signal in den ERROR-Zustand wechselt und die Förderrate auf null setzt, erfüllt nunmehr die im Experiment formulierte Anforderung.

schrieben werden. Wenn eine gegebene Schranke (in diesem Fall 30 ms) überschritten wird, schaltet sich die Saug-/Spülpumpe automatisch ab.

Durch erneute Verifikation konnte die Erfüllung der Anforderung nachgewiesen werden.

Statistisches zur neuerlichen Verifikation: Verifikationsdauer: 701,9 Sekunden. Spitzenbelastung des RAM: 1,5 GB. Der deutlich erhöhte Rechenaufwand gegenüber der ersten Verifikation lässt sich darauf zurückführen, dass erstere nur ein Gegenbeispiel hervorbringen muss, um die Anforderung zu negieren, die Verifikation der „Fehlerfreiheit“ jedoch sämtliche Systemzustände durchschreiten muss.

Bemerkung: Um dieses Experiment durchführen zu können, mussten zusätzliche Locations und Uhren eingeführt werden (siehe Abb. 49). Dadurch wird deutlich, dass der Hersteller eines Moduls die Möglichkeit des Ausfalls eines Geräts schon bei der Modellierung berücksichtigen muss. Es ist jedoch davon auszugehen, dass bei derart komplexen Überlegungen trotzdem noch Ergänzungen des Systems nötig werden.

6.2. Sicherstellung der Konnektivität – Herstelleranforderung

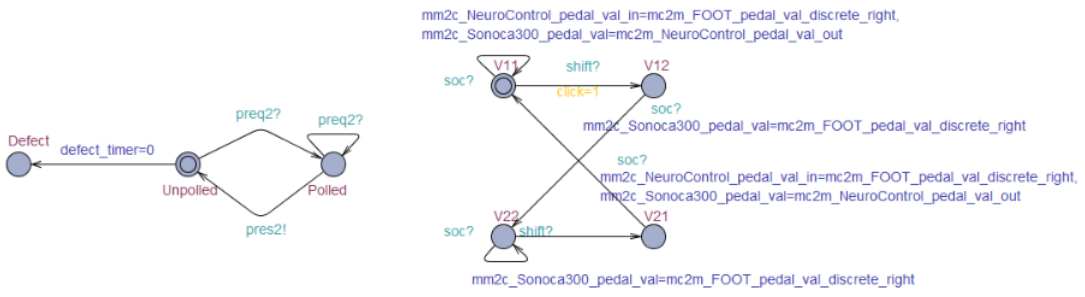
Dieser Anwendungsfall für die automatisierte Verifikation beschäftigt sich mit der Sicherstellung der Konnektivität aller angeschlossenen Geräte. Bei chirurgischen Eingriffen kann die Verfügbarkeit von Medizingeräten kritisch sein. Man stelle sich hierzu beispielsweise einen Eingriff vor, bei dem plötzlich eine starke Blutung eintritt. Ist ein HF-Gerät zur Stillung der Blutung zwar an das Netzwerk angeschlossen (und vermittelt so den Eindruck, einsetzbar zu sein), aufgrund der Netzwerkkonfiguration jedoch nicht aktivierbar (z.B. falsche Verschaltung), kann dies zu schwerwiegenden Komplikationen führen (Anheuser und Steffens, 2011). Prinzipiell ist daher bei der Modellierung der Geräte durch den Hersteller sinnvoll, eine Location zu identifizieren, deren Erreichbarkeit garantiert, dass das Gerät Netzwerkdaten empfangen kann. Die *Reachability* dieser Location (siehe Tabelle 6) sollte dann auf *Possible* gesetzt werden. Für das vorige Setup aus Fußschalter, Saug-/Spülpumpe und US-Dissektor wurde daher folgendes Experiment durchgeführt:
Fragestellung: Werden alle Polled-Locations der EPL-Prozesse zwingend einmal erreicht? Können die Saug-/Spülpumpe und der US-Dissektor jeweils die Location ACT erreichen (siehe Abschnitte 5.7.2, 5.7.3)?
Beteiligte Akteure: Fußschalter (alle Werte), Saug-/Spülpumpe (alle Module), US-Dissektor (alle Module), MN (alle Module).
Anforderungsformeln:

1. $\exists \diamond \, EndomatLC.ACT$
2. $\forall \diamond \, EndomatLC\_EPL.Polled$
3. $\forall \diamond \, Footswitch\_EPL.Polled$
4. $\exists \diamond \, Sonoca300.ACT$
5. $\forall \diamond \, Sonoca300\_EPL.Polled$

Analyseergebnis: Alle Anforderungen konnten verifiziert werden.
Statistisches: Verifikationsdauer: 10,7 Sekunden. Spitzenbelastung des RAM: 83 MB.
Bemerkung: Dieses Experiment wird vollautomatisch durchgeführt. Weder müssen Anforderungen explizit eingetippt werden, noch muss das Modell abgeändert werden. Durch die geeignete Modellierung im Sinne des Herstellers ist es möglich, alle Anforderungen dieser Form direkt aus der Beschreibungsdatei abzuleiten und automatisch in den Anforderungskatalog zu übernehmen. Noch akkurater wäre in diesem Zusammenhang zu fordern, dass die tatsächliche observable-Variable, z.B. *actual_flow_rate*, einmal einen Wert größer als Null annehmen kann.
Gegenwärtig muss diese Anforderung händisch in den Anforderungskatalog aufgenommen werden. Die Ergänzung geeigneter Attribute in Tabelle 8 würde aber mühelos ermöglichen, auch dies zu automatisieren. Diese schärfere Anforderung konnte auch verifiziert werden, allerdings mit beträchtlich höherem Rechenaufwand (siehe Kapitel 7).

6.3. Vermeidung riskanter Neuverschaltungen – NeuroControl

Eine zentrale Herausforderung der Digitalisierung ist der Fertigkeitsverlust klinischer Anwender durch die Gewöhnung an und Verlass auf immer intelligentere Assistenzsysteme. Der Anwender in (FDA, 2016b) etwa schildert, wie er unter Nutzung des Navigationssystems *ENT Navigation* der Firma Brainlab ® aufgrund fehlerhafter Registrierung die Schädelbasis eines Patienten durchbrach und so den Austritt von zerebrospinaler Flüssigkeit verursachte.
Am Lehrstuhl MiMed wurde im Rahmen des Projekts OR.Net ein Funktionsmodul *NeuroControl* entwickelt (Dietz et al., 2017), das Daten eines Neuromonitors nutzt, um aktive Instrumente abzuregeln, wenn die Nervenantwort auf eine drohende Schädigung hindeutet.
Der Verlass auf NeuroControl ist ebenfalls risikobehaftet: Fällt bei einem NeuroControl-gestützten Eingriff das Modul ohne Kenntnis des Anwenders aus und wird das Fußpedal-Signal direkt an das Instrument durchgereicht, so kann durch die Wirkung des US-Dissektors empfindliches Gewebe, z.B. der Fazialisnerv, irreparabel geschädigt werden (Green et al., 1994). Ob dieser Fall eintreten kann, wird im letzten Anwendungsfall geprüft.
Fragestellung: Ist für eine Aktivierung des US-Dissektors nach Ausfall von NeuroControl eine Neuverschaltung durch den Anwender erforderlich?
Beteiligte Akteure: Fußschalter (rechtes Pedal, diskrete Werte), US-Dissektor (Hauptprozess), NeuroControl (alle Module), MN (alle Module).
Anforderungsformel (Variablennamen zugunsten der Lesbarkeit angepasst):
$\forall\Box\,[\text{NeuroControl\_EPL.Defect} \wedge (\text{timer} > 30) \wedge (\text{Sonoca300.action} > 0) \Rightarrow (\text{click} == 1)]$
Analyseergebnis: Die Anforderung konnte verifiziert werden.
Statistisches: Verifikationsdauer: 1987,7 Sekunden. Spitzenbelastung des RAM: 802 MB.
Bemerkung: Für dieses Experiment muss ein DEFECT-State und eine Uhr `defect_timer` für NeuroControl hinzu modelliert werden (siehe Abb. 50a). Die Verifikation wird jedoch erst durch eine zusätzliche Variable „click“ ermöglicht, die genau dann 1 (`true`) ist, wenn die Verschaltung durch den Anwender geändert wurde (siehe Abb. 50b). Bei Ausfall des NeuroControl-Moduls kann so erst nach Neuverschaltung des Echtzeit-Bussystems der US-Dissektor wieder aktiviert werden (obschon NeuroControl in der Defect-Location bleibt).



(a)

(b)

Abbildung 50 Überarbeitete Prozesse zum Zwecke des Experiments. Im EPL-Prozess wurde eine Defect-Location eingefügt, um einen Netzwerkausfall bei NeuroControl zu modellieren (Abb. 50a). Der Mapping Mode wurde um die Variable `click` (in Gelb) angereichert, um festzustellen, wann eine Neuverschaltung vorgenommen wurde (Abb. 50b).

7. Messungen und Experimente

Durch die folgenden beiden Experimentserien soll zum einen ein erster Einblick in den typischen Berechnungsaufwand bei der Verifikation des Echtzeitbussystems gegeben werden und zum anderen sichergestellt werden, dass die von UPPAAL umgesetzte Interpretation den Definitionen aus Kapitel 4 entspricht.

7.1. Beispielhafter Rechenaufwand

Zur Ermittlung des Rechenaufwands bei der Verifikation des Echtzeit-Bussystems sind zwei übergeordnete Größen von besonderer Bedeutung: Der Aufwand der Verifikation unterschiedlich komplexer Aussagen und die Dauer der Verifikation *derselben Anforderung* mit zunehmender Modellkomplexität. Es ist bekannt (siehe (Alur et al., 1993)), dass das Model Checking von TCTL-Formeln, ähnlich wie bei CTL-Formeln (Laroussinie et al., 2001) und LTL-Formeln (siehe (Baier et al., 2008a), Thm. 5.4.8) PSPACE-complete ist. Es gehört somit zu den komplexesten derjenigen Probleme, deren Speicheraufwand polynomiell in der Komplexität des zu verifizierenden Ausdrucks wächst.
Da der Rechenaufwand unterschiedlich komplexer Anforderungen beispielhaft bereits in den Anwendungsfällen aus Kapitel 6 gemessen wurde, sollen in diesem Kapitel gleichbleibende Anforderungen bei variierender System-Konfiguration verifiziert werden. Dabei soll untersucht werden, *wie lange* die Verifikation dauert, *wie viele Zustände* dabei durchschritten werden müssen und *wie viel Speicher* dafür erforderlich ist.
Alle Messungen wurden auf einem Lenovo ThinkPad E460 mit Intel Core i5-6200U CPU, 8 GB RAM und 64-bit Windows 10 Pro Betriebssystem durchgeführt. Die verwendete UPPAAL-Version 4.1.19 wurde in der 32-bit Ausführung benutzt, weshalb UPPAAL nur 4 GB des RAM nutzen kann. In den UPPAAL-Einstellungen wurde wechselweise Breitensuche (BFS) und Tiefensuche (DFS) eingestellt. Durchgängig wurde die stärkste Variante der UPPAAL-integrierten Zustandsraum-Kompression (*aggressive*) verwendet. Diese ist zwar zeitintensiv, aber speicherschonend, was bei der 32-bit UPPAAL Version und komplexen Anforderungen zuweilen nötig ist, um einen Abbruch der Verifikation zu vermeiden. Die Messungen wurden in UPPAAL vorgenommen und anschließend in Matlab R2016b eingelesen und visualisiert.

7.1.1. Rechenaufwand in Abhängigkeit der Modellkomplexität

Für dieses Experiment wurde ein Grundsystem, bestehend aus dem Fußschalter aus Abb. 45 und der Saug-/Spülpumpe aus Abb. 47 schrittweise wie in Tabelle 17 ergänzt, bis das vollständige Modell aus Fußschalter, Saug-/Spülpumpe und US-Dissektor (Abb. 46) mit allen Funktionalitäten erreicht war. Dabei wurden die folgenden zwei Anforderungen verifiziert:

1. Die Saug-/Spülpumpe lässt sich einschalten.
2. Die Saug-/Spülpumpe kann aktiviert werden, d.h. die Ist-Flowrate ist größer als Null.

Für jedes Modell aus Tabelle 17 wurde die Verifikation 10-Mal durchgeführt und anschließend über den Rechenaufwand gemittelt. Diese Vorgehensweise ist gerechtfertigt, da UPPAAL so eingestellt war, dass kein Lerneffekt eintritt (siehe Abb. 51).

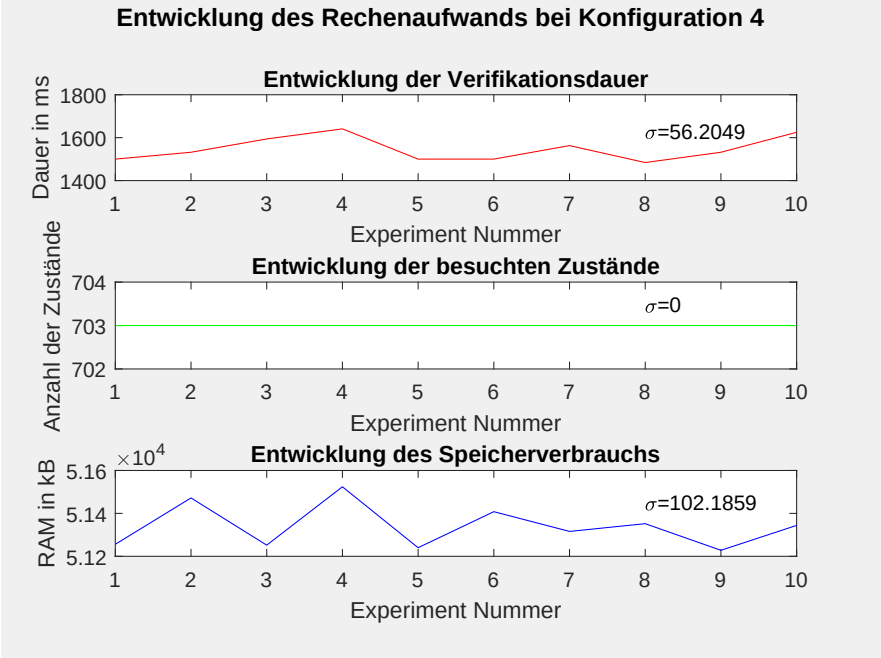


Abbildung 51 Entwicklung des Rechenaufwands (Tiefensuche) für die Verifikation von Anforderung 1 bei Konfiguration 4 aus Tabelle 17. Die Graphen zeigen, dass in keiner Kategorie ein merklicher Lerneffekt vorhanden ist. Daher ist es sinnvoll, über die Zeitreihen zu mitteln.

Verschiedene Modellkonfigurationen zur Verifikation der Anforderungen

Index	Beteiligte TEFAs
1	PollingMode und MappingMode Endomat LC (physikalischer Prozess und EPL-Prozess) Generatoren für Endomat LC-Signale ON, OFF, START TEFA für das Flowrate-Setup des Endomat LC (Abb. 47b). Generatoren für Setup-Signale UP, DOWN, SWITCH (Endomat LC) Fußschalter (physikalischer Prozess und EPL-Prozess) Generatoren für diskrete Fußpedal-Werte
2	Konfiguration 1 und Generator für linkes analoges Fußpedalsignal
3	Konfiguration 2 und Generator für rechtes analoges Fußpedalsignal
4	Konfiguration 3 und US-Dissektor (Hauptprozess, EPL-Prozess)
5	Konfiguration 4 und Setup-Prozess des US-Dissektors (siehe Abb. 46b) Generatoren für UP, DOWN, TWENTY, FOURTY, SIXTY, EIGHTY

Tabelle 17 Verschiedene Konfigurationen, unter denen eine gegebene Anforderung experimentell verifiziert wurde. Mit zunehmendem Index steigt die Modellkomplexität.

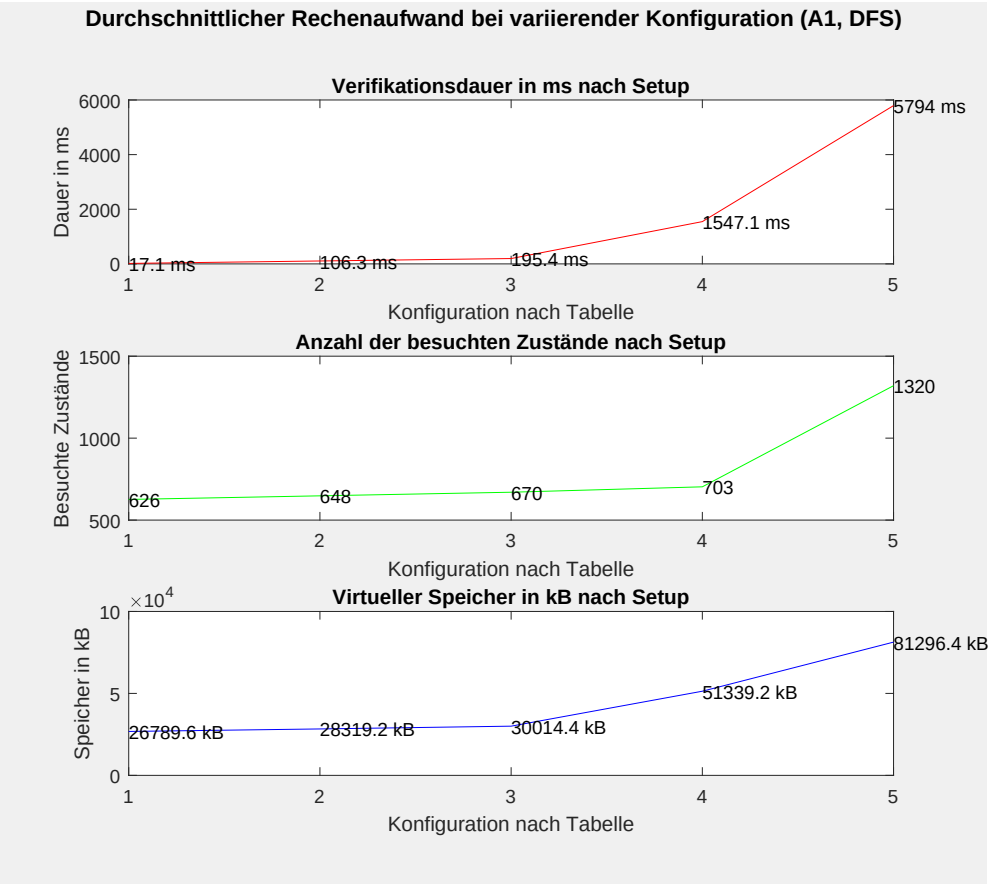


Abbildung 52 Plots des durchschnittlichen Rechenaufwands (Tiefensuche) für die Verifikation von Anforderung 1 entsprechend der Konfiguration nach Tabelle 17, aufgeteilt in Rechendauer, Anzahl der besuchten Zustände und Speicheraufwand.

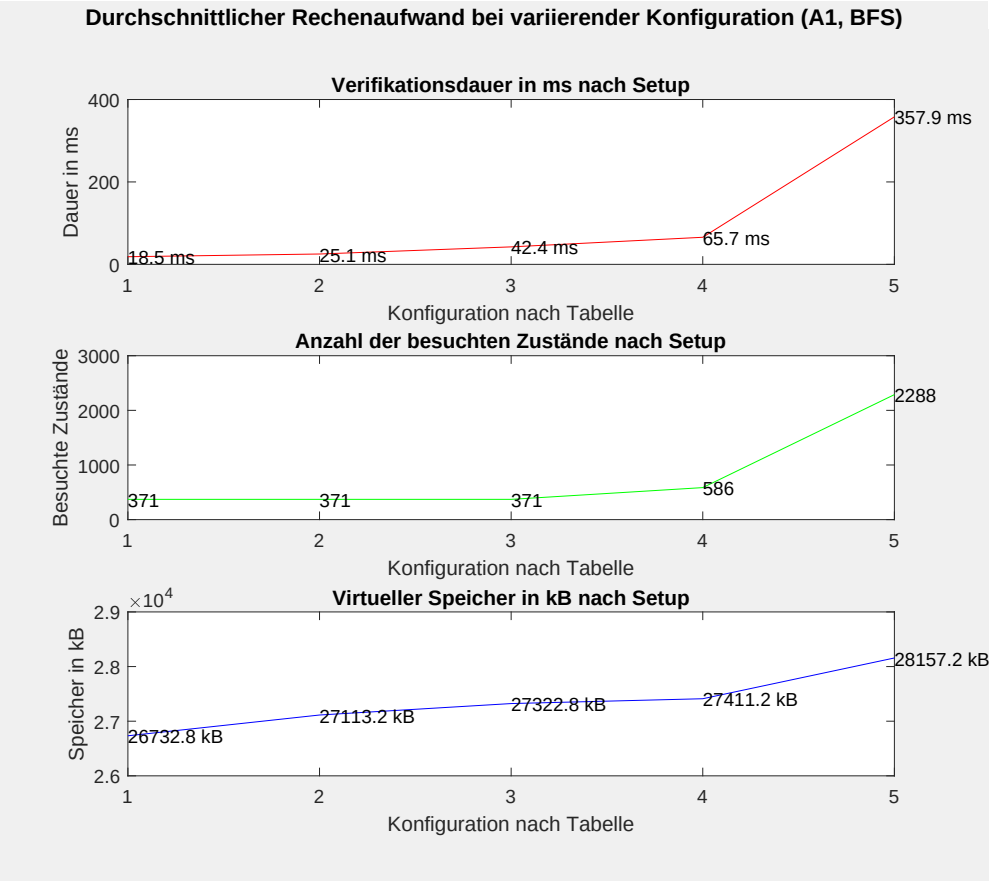


Abbildung 53 Plots des durchschnittlichen Rechenaufwands (Breitensuche) für die Verifikation von Anforderung 1 entsprechend der Konfiguration nach Tabelle 17, aufgeteilt in Rechendauer, Anzahl der besuchten Zustände und Speicheraufwand.

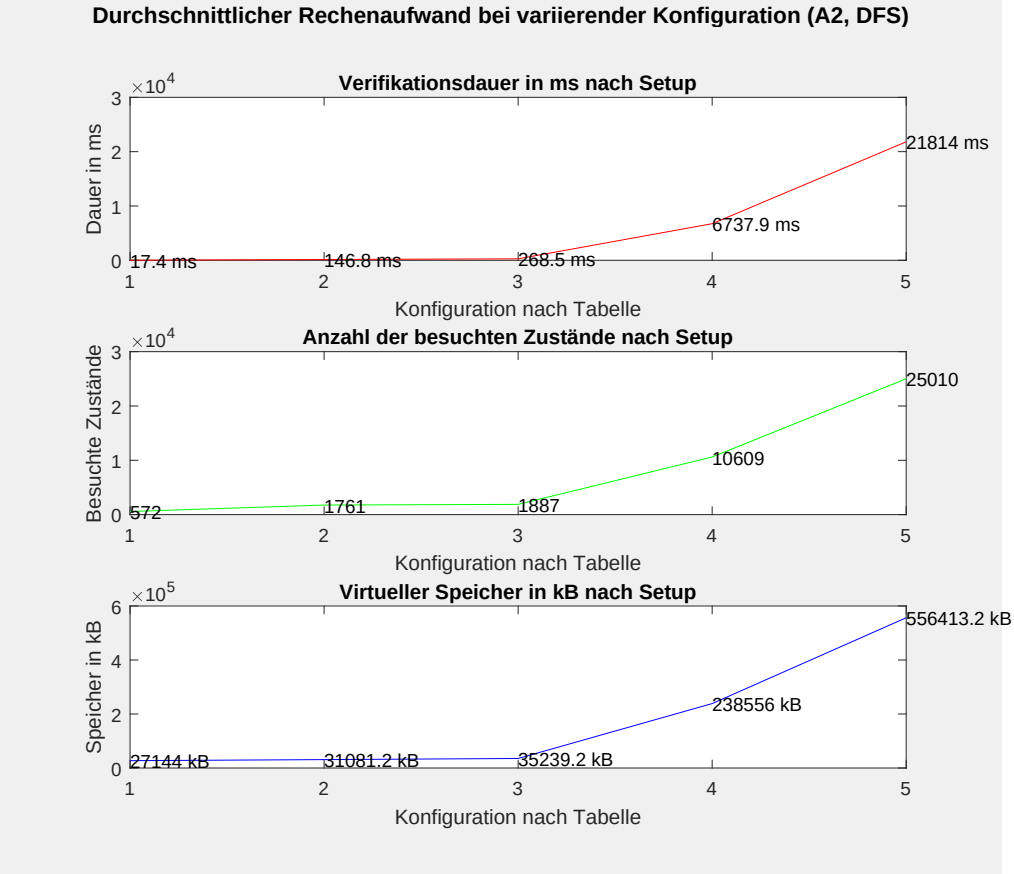


Abbildung 54 Plots des durchschnittlichen Rechenaufwands (Tiefensuche) für die Verifikation von Anforderung 2 entsprechend der Konfiguration nach Tabelle 17, aufgeteilt in Rechendauer, Anzahl der besuchten Zustände und Speicheraufwand.

7.2. Validierung von UPPAAL

Neben der korrekten Modellierung des Systems und der Anforderungen, ist für eine verlässliche Verifikation erforderlich, dass der Model Checker selbst korrekt arbeitet. Leider gibt es hierzu keine praxistauglichen Ansätze (siehe Kapitel 8). Jedoch kann geprüft werden, ob die Semantik, die in den vorausgehenden Kapiteln zugrunde gelegt wurde, durch das Tool in einfachen Fällen umgesetzt wird. Die folgenden Untersuchungen sind nicht als formaler Nachweis zu verstehen, sondern vielmehr als ein Plausibilitätstest, der grobe Abweichungen zwischen erwarteter und (durch UPPAAL) umgesetzter Interpretation ausschließen soll. Beispielhaft werden dazu die beiden komplexesten Konstrukte – die Broadcast Synchronisation und das Variablen-Update – gezeigt.

7.2.1. Broadcast Synchronisation

Im ersten Experiment soll geprüft werden, ob die Broadcast Synchronisation aus Definition 4.1.14 erwartungsgemäß simuliert wird. Hierzu wurde die einfachste Möglichkeit modelliert, um herauszufinden, ob die vorgesehene Semantik durch UPPAAL umgesetzt wird. Dabei ist es wichtig zu beachten, dass unter UPPAAL bei der Einführung eines Broadcast Channels automatisch das Hiding (Definition 4.1.15) mit vorgenommen wird. Modelliert werden drei TEFAs `Send`, `Rec1` und `Rec2` wie in Abb. 19. Gezeigt werden soll, dass die dort entwickelte Lösung durch UPPAAL wie in nachfolgender Abb. 55 umgesetzt wird.

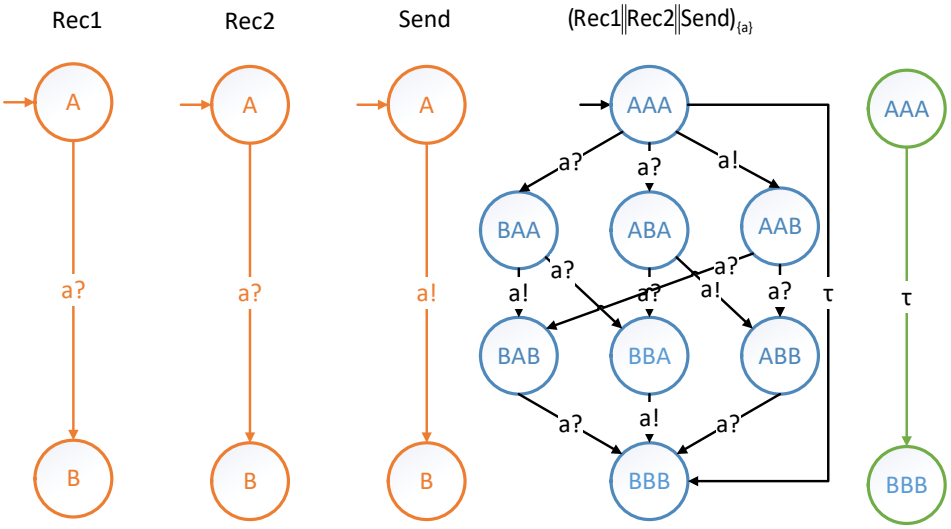
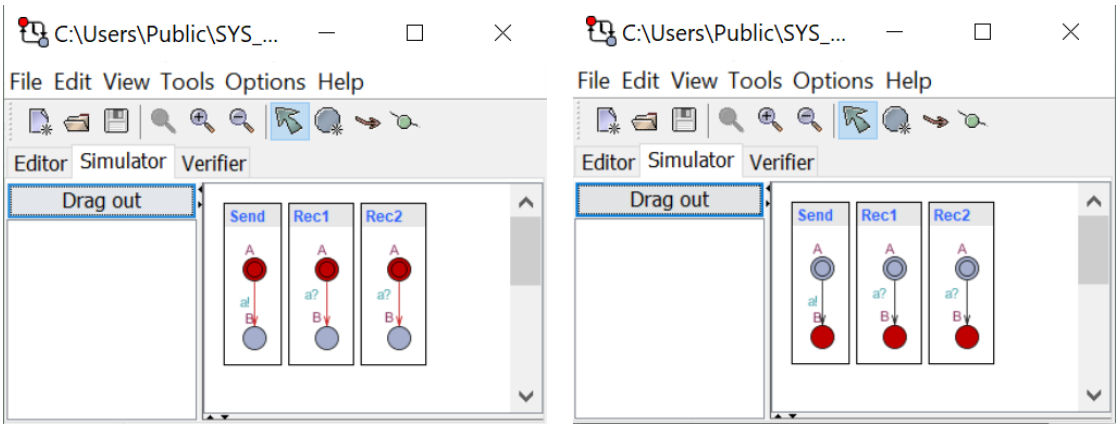


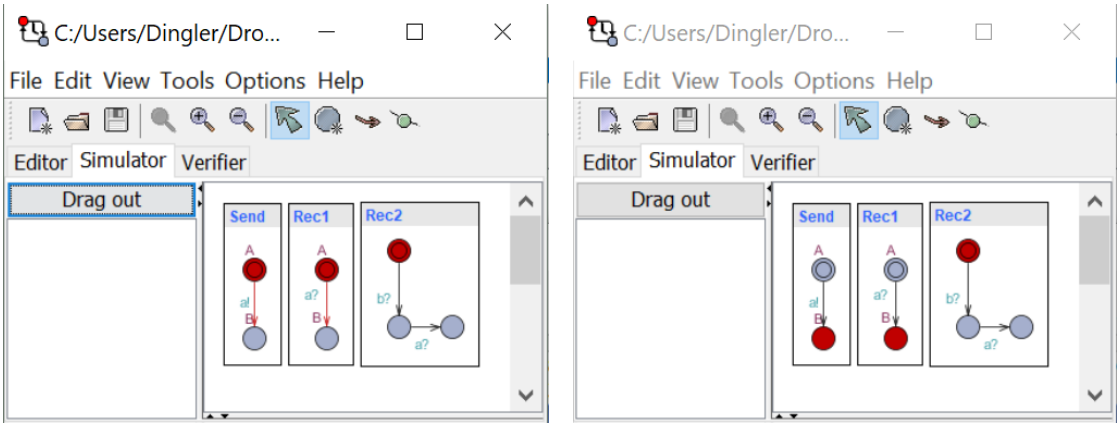
Abbildung 55 Die einfachen TEFAs Rec1, Rec2 und Send (orange) werden erst zum Broadcast-TEFA (blau) vermengt (siehe auch Abb. 19) und anschließend durch Hiding auf den einfachen TEFA (grün) reduziert. Ob dieses Verhalten von UPPAAL widergespiegelt wird, soll in Abb. 56 validiert werden.



(a) (b)

Abbildung 56 Broadcast Synchronisation mit Hiding für einfache 2-Location TEFAs. Der TEFA `Send` sendet das Synchronisationssignal (a!), die beiden TEFAs `Rec1` und `Rec2` empfangen es und alle drei TEFAs nehmen synchron einen Zustandswechsel vor. Vom Startzustand in Abb. 56a zum Endzustand in Abb. 56b geht es in einem Schritt (siehe dazu Definition 4.1.14, letzter Absatz, Unterpunkt c.)

Im Gegensatz zur binären Synchronisation, soll die Broadcast Synchronisation jedoch *nicht blockieren*, wenn ein möglicher Empfänger diese nicht quittieren kann (siehe Definition 4.1.14, 4.c, d). Anders als in Abb. 56, ist nun in Abb. 57 mit `Rec2` ein TEFA Teil des Systems, der auf der ersten Transition nicht das Broadcast Synchronisation a?, sondern ein anderes Signal b? trägt. Entsprechend Definition 4.1.14, Absatz 3, Unterpunkte b und c soll nun `Rec1` zusammen mit `Send` die Transition nehmen, `Rec2` hingegen in seiner Location bleiben. Dass diese Modellierung auch von UPPAAL umgesetzt wird, wird abschließend in der nachfolgenden Abb. 57 gezeigt.

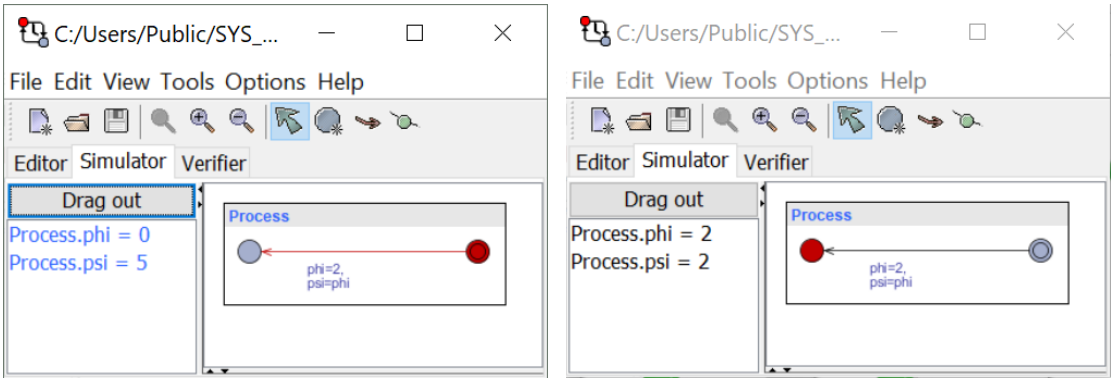


(a) (b)

Abbildung 57 Anders als die binäre Synchronisation, blockiert die Broadcast Synchronisation nicht, wenn ein Teilautomat (hier: `Rec2`) das Broadcast-Signal (hier: `a!`) nicht verarbeiten kann (Abb. 57a). `Rec2` bleibt daher in seiner Location, `Send` und `Rec1` nehmen hingegen simultan einen Zustandsübergang wahr (Abb. 57b).

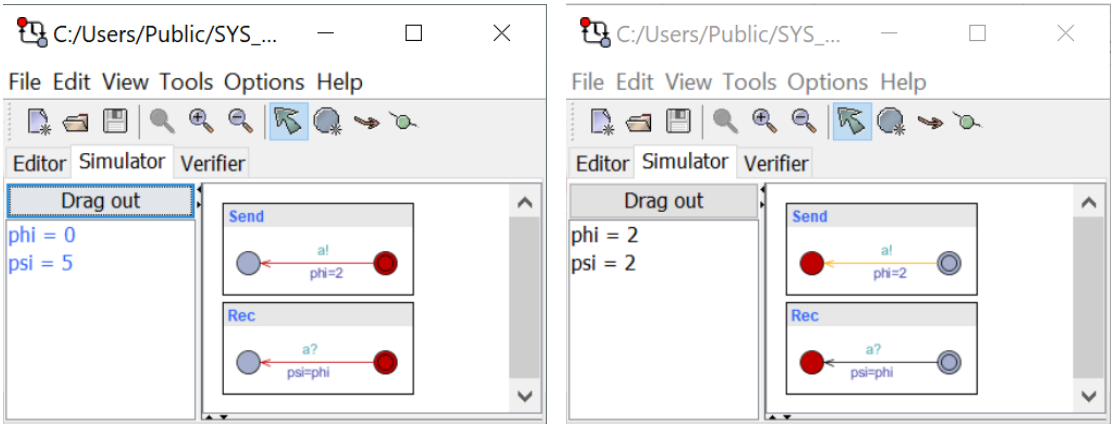
7.2.2. Update der Variablen

Beim Update der Variablen soll beispielhaft nachgewiesen werden, dass die Konstrukte aus den Definitionen 4.1.11 und 4.1.13 zutreffend sind. Speziell ist zu zeigen, dass die Reihenfolge der Variablenbelegung bei einer Transition deren Endwerte beeinflusst. Dass dies sowohl auf einfache Transitionen zutrifft, als auch auf gepaarte Transitionen durch Synchronisation (hier beispielhaft: binäre Synchronisation), wird in Abb. 58 und 59 gezeigt. In beiden Abbildungen werden TEFAs mit den Variablen phi und psi untersucht. Die Variablen werden mit den Werten $phi = 0$ und $psi = 5$ initialisiert. Durch eine Transition (Abb. 58: einfach, Abb. 59 synchron) findet das Update $phi = 2$, $psi = phi$ statt. Würden die Variablen *gleichzeitig* belegt, so wäre nach der jeweiligen Transition $phi = 2$ und $psi = 0$, da phi mit 0 initialisiert wurde. Tatsächlich wird aber das Konstrukt aus Definition 4.1.6 implementiert, das auch in Definition 4.1.11 Verwendung findet.
Die Variablenbelegung findet demnach *sequentiell* statt – zuerst wird $phi = 2$ gesetzt, anschließend $psi = phi = 2$, wie Abb. 58b und Abb. 59b zu entnehmen ist.



(a) (b)

Abbildung 58 Nachweis, dass unter UPPAAL bei einfachen Transitionen die Reihenfolge der Variablenbelegung Einfluss auf deren Werte nach einer Transition hat. Der erste genannte Wert (hier: $phi = 2$) wird zugewiesen und *anschließend* der zweite auf Basis des bereits aktualisierten ersten Werts.



(a) (b)

Abbildung 59 Nachweis, dass unter UPPAAL auch bei binärer Synchronisation die Reihenfolge der Variablenbelegung Einfluss auf deren Werte hat. Derjenige Prozess, der die Synchronisation veranlasst (hier: `Send`) nimmt das Update der Variablen als erster vor, danach folgt der synchronisierte Prozess (hier: `Rec`). Die Variablenbelegung findet demnach auch bei binärer Synchronisation sequentiell statt.

7.3. Zusammenfassung der Experimente

7.3.1. Rechenaufwand

Um den Rechenaufwand zu ermitteln, wurde ein bestehendes System schrittweise erweitert und gleichbleibende Aussagen bei variierendem System mit unterschiedlichen Einstellungen verifiziert. Bei Betrachtung des Rechenaufwands konnte beobachtet werden, dass sowohl der zeitliche Aufwand der Verifikation, als auch der Speicherbedarf ab Setup 4 (Hinzunahme des Ultraschalldissektors) unabhängig von der Komplexität der Anforderung sprunghaft ansteigt. Dies trifft sowohl auf die Breitensuche in Abb. 52 zu, als auch auf die Tiefensuchen in Abb. 53 und 54. Besonderes Gewicht hat zudem der verwendete Suchalgorithmus: Während etwa Anforderung 2 bei einer Tiefensuche (Default-Einstellung) selbst bei vollem Setup noch in rund 22 Sekunden bei einem Speicherbedarf von weniger als 600 MB verifiziert werden konnte, bricht die Verifikation bei einer Breitensuche schon ab Setup 2 wegen Speicherfehlern ab. Dabei ist jedoch hervorzuheben, dass in dieser Arbeit betriebssystembedingt mit der 32-bit Version von UPPAAL verifiziert wurde. Für Mac OS X und Linux sind zwischenzeitlich 64-bit Versionen verfügbar, die jedoch in dieser Arbeit nicht untersucht wurden.

7.3.2. Validierung der Semantik

Beim Vergleich der Semantik der TEFAs in Kapitel 4 und deren Interpretation durch UPPAAL konnten keine grundlegenden Unterschiede festgestellt werden. Um dies zu ermitteln, wurden zweckgerichtete TEFAs entwickelt, anhand derer ausgewählte Elemente der Semantik validiert wurden. Nicht auszuschließen bleibt weiterhin, dass das Programm Fehler enthält – diese können in der Simulation, besonders aber in der Verifikation von Anforderungen auftreten. Angesichts der langen Referenzliste von UPPAAL (siehe Kapitel 8) und mangels Veröffentlichungen von erkannten Fehlern ist jedoch davon auszugehen, dass diese vernachlässigbar selten auftreten.

7.3.3. Hinweise zur Optimierung des Rechenaufwands

Der Autor in (Behrmann et al., 2004) erwähnt Hinweise, mithilfe derer die Performanz von UPPAAL gesteigert werden kann. Dies ist besonders dann wichtig, wenn eine UPPAAL-Verifikation aus Speichergründen abbricht.

Demnach empfiehlt es sich bei unbekannten Verifikationsaufgaben, zunächst die Verifikation ohne Kompression des Zustandsraums zu versuchen, da dies die zeitsparendste Alternative ist. Ergeben sich dabei Speicherprobleme, gibt es die Möglichkeit, die Zustandsraumkompression schrittweise zu intensivieren (in den Stufen *None*, *Conservative*, *Aggressive* und inzwischen auch: *Extreme*). Je stärker die Kompression, desto geringer der Speicherverbrauch und höher die Rechendauer.

Über die Kompression hinaus kann über die Darstellung des Zustandsraums Speicher geschont oder Rechendauer verkürzt werden. Standardmäßig werden zur Darstellung der erreichbaren Zustände *Difference Bounded Matrices* (DBMs) eingesetzt. Bei Systemen mit vielen Uhren gibt es die Möglichkeit, die Dauer der Verifikation zu verkürzen, wenn diese

Darstellung durch *Convex Hull Over-Approximation* ersetzt wird. Um hingegen Speicher zu sparen, empfiehlt sich die Unter-Approximation. In beiden Fällen kann jedoch auftreten, dass die Verifikation mit dem Ergebnis *Property may be satisfied* endet, also das Ergebnis der Verifikation nicht zweifelsfrei feststeht.
Während es keine allgemeingültigen Aussagen gibt, ob sich Breitensuche oder Tiefensuche als Ansatz besser empfiehlt, besteht nach (Behrmann et al., 2004) die Gefahr, dass die Tiefensuche in Kombination mit starker Zustandsraumkompression sich „verirrt“ und beliebig lange Rechenzeiten hervorrufen kann.

8. Zusammenfassung und Ausblick

In dieser Arbeit wurden Lösungen für die Konformitätsbewertung herstellerübergreifend vernetzter Medizingeräte (siehe Kapitel 1) erarbeitet, beispielhaft für das OR.Net Echtzeit-Bussystem implementiert und experimentell untersucht. Sie können von Krankenhausbetreibern im Rahmen der Konformitätsbewertung solcher Netzwerke zur Verifikation der Erfüllung ihres vorgesehenen Zwecks und zur Entdeckung von Gefährdungen eingesetzt werden.
In Kapitel 2 wurde das regulatorische Spannungsfeld, in dem sich die Vernetzung von Medizingeräten bewegt, exakt beschrieben und eingegrenzt, um in Kapitel 3 eine konkrete Aufgabe daraus abzuleiten.
Im Kern der Arbeit, Kapitel 4, wurden die Methoden entwickelt, die in dem abstrakten Modell aus Abschnitt 4.8 münden, mit dem den Herausforderungen der Konformitätsbewertung unter Berücksichtigung aller involvierten Parteien – Hersteller, Betreiber und Anwender – begegnet werden kann. Um dieses zu realisieren, wurden zunächst anforderungsgerechte Beschreibungsmethoden für Medizingeräte und ihre Anforderungen bezüglich deren typischer Funktionsweise gewählt und präzisiert. Für die Beschreibung von Medizingeräten und Zubehör haben sich TEFAs – eine Klasse formaler Automaten, die einen Zeit- und Variablenbegriff haben, als ein geeigneter Formalismus herausgestellt. Dieser wurde in den Abschnitten 4.2 bis 4.4 auf die Eigenheiten des Echtzeit-Bussystems angewandt – insbesondere wurde dabei ein skalierbares Modell für die Netzwerksteuerung und die dynamische Verschaltung von Datenquellen und -senken erarbeitet. Anforderungen, die an das Netzwerk gestellt werden, werden in einer speziellen temporalen Logik, eine Teilsprache von TCTL (siehe Abschnitt 4.5), formuliert. Alle erarbeiteten Komponenten werden in Abschnitt 4.6 zusammengeführt.
Die resultierende *Verification Toolbox* ist in Kapitel 5 beschrieben. Sie basiert auf den in dieser Arbeit entwickelten, standardisierten Spezifikationstemplates aus Abschnitt 5.2, mit deren Hilfe Hersteller die Eigenschaften und Anforderungen ihrer Module beschreiben können und der Verschaltungsspezifikation aus Abschnitt 5.3. Die Matlab-basierte Verification Toolbox liest diese Komponenten ein, generiert ein Modell des so entstehenden Netzwerks und verifiziert mithilfe des UPPAAL-Tools automatisiert Anforderungen (Abschnitt 5.4) an das Netzwerk. Die Anforderungen können dabei von beteiligten Herstellern stammen (und bereits in der Modulbeschreibung enthalten sein), oder aber von der Betreiberorganisation. Einzelheiten der Implementierung und resultierende, beispielhafte Modelle sind in den Abschnitten 5.5 bis 5.7 beschrieben.
Die entwickelte Toolbox wurde in Kapitel 6 auf typische Fragestellungen im OR.Net Echtzeit-Demonstrator angewandt. Dabei wurde deutlich, dass die Verification Toolbox zuverlässig in kurzer Zeit entwicklungsrelevante Erkenntnisse gewinnen kann.
Die Verifikationsdauer und der Speicheraufwand variieren merklich mit der zu verifizierenden Anforderung, der Systemkomplexität und dem verwendeten Suchalgorithmus (vgl. Kapitel 7). Die größte Herausforderung besteht aktuell darin, Anforderungen geeignet zu modellieren und das zu untersuchende System auf die Fragestellung anzupassen. Weitere Herausforderungen bei der Fortentwicklung der Toolbox werden im Folgenden kurz dargestellt.

Diskussion

Das Validierungsproblem und konforme Codegenerierung

Eine zentrale Herausforderung bei der Verifikation formaler Anforderungen bleibt der Nachweis, dass das Verifikationsproblem im Modell mit dem System der Praxis übereinstimmt. Dies ist auch als Validierungsproblem (Baier et al., 2008e) bekannt.

Jede formale Beschreibungsmethode abstrahiert von zahlreichen systeminhärenten Eigenschaften, um eine effiziente Verifikation überhaupt erst zu ermöglichen. Verifiziert wird dabei stets das Modell und nicht das eigentliche System. Besondere Bedeutung kommt daher auch der korrekten Modellierung der Module durch beteiligte Hersteller, sowie die korrekte Interpretation der Anforderungen seitens ihrer Formulierer zu. Zusätzliche Fehler können bei der Verifikation durch einen Model Checker selbst entstehen. Die Verification Toolbox kann daher nur als Indikator für die korrekte Funktionalität dienen und mögliche Fehlfunktionen aufzeigen – im Rahmen des Konformitätsbewertungsprozesses müsste sie jedoch begleitet werden von Tests, welche die Ergebnisse der Verification Toolbox empirisch belegen. Hierzu gibt es bereits Ansätze, die unter den Begriff *Model-Based-Testing* fallen (vgl. (Rösch et al., 2015) für eine aktuelle Übersicht).

In dieser Arbeit wurden die einzelnen Module vom Autor selbst beschrieben, um die Ausdruckskraft des beschrittenen Ansatzes zu verdeutlichen. Ein möglicher Ansatz zur Bewältigung des Validierungsproblems ist das *Model-Based Development*. So schlagen die Autoren in (Lee et al., 2012) einen modellbasierten Entwicklungsprozess vor, welcher UPPAAL zur händischen Modell-Bildung und Verifikation, sowie *TIMES* (Amnell et al., 2002) zur Generierung von Code unter Erhaltung der verifizierten Eigenschaften nutzt.

Die Autoren in (Pajic et al., 2014) hingegen entwickeln zu diesem Zweck das *UPP2SF*-Tool, das UPPAAL-Modelle in mathematisch äquivalente Stateflow-Modelle übersetzt. Die Modelle der Geräte können in UPPAAL verifiziert werden und nach Konvertierung zu Simulations- und Testzwecken in Stateflow aufgerufen werden. Nähere Details über den Erfolg der Implementierung sind aber noch nicht bekannt.

Vollständigkeit der Anforderungen

Im Model Checking werden stets nur die Anforderungen verifiziert, die explizit an das System gestellt werden. In dieser Arbeit können diese vom Hersteller, oder aber von Betreiber und Anwender stammen. Wie in der klassischen Risikoanalyse auch, gibt es jedoch keine Garantie, dass diese Anforderungen *vollständig* sind und die Einhaltung aller Anforderungen den sicheren Systembetrieb gewährleistet. Für Betreiber einer Gesundheitseinrichtung wird es daher eine Herausforderung bleiben, anwendungsspezifische Risiken zu identifizieren und als Anforderung zu formulieren (siehe auch (Fantechi und Gnesi, 2011)). Hinzu kommt, dass nicht alle Anforderungen formalisierbar sind – den Autoren in (Lee et al., 2012) gelang es beispielsweise nur, etwa ein Fünftel eines Satzes von mit der *U.S. Food and Drug Administration* (FDA) erarbeiteten Anforderungen in der Logik aus Abschnitt 4.5 zu formalisieren.

Rechenaufwand

Wie in Kapitel 7 dargestellt, variiert der Rechenaufwand mit der Modell- und Anforderungskomplexität. Schon bei vergleichsweise einfachen Systemen, wie etwa dem Fußschalter mit zwei steuerbaren Endgeräten (Tabelle 17), kann der Zustandsraum sehr groß und die Berechnungsdauer, sowie der Speicherbedarf stark ansteigen (siehe Abb. 52 – 54). Dies ist als das *State-explosion-problem* bekannt – ein Phänomen, dem derzeit nur durch schrittweise Approximation des Zustandsraums und dessen kompakte Darstellungsweise begegnet werden kann.
Gleichwohl bereitet die reine Anzahl beteiligter Automaten keine Schwierigkeiten: Nach (Kristoffersen, 1998) kann UPPAAL für bis zu 1400 nebenläufige Automaten benutzt werden. Beachtet man, dass typische endoskopische Eingriffe mit weniger als 10 Medizingeräten zu bewältigen sind (Liehn et al., 2014a), und jedes Gerät in dieser Arbeit durch weniger als 10 nebenläufige Automaten beschrieben werden konnte, ist dies keine nennenswerte Einschränkung.

Korrektheit des *Model Checkers*

Neben der Akkuratesse der Beschreibungsmethode stellen potenzielle Fehler im Model Checker (insbesondere UPPAAL) selbst eine weitere Fehlerquelle dar. Im Besonderen ist von Interesse, ob das Ergebnis der Verifikation bei korrekter Modellierung selbst korrekt ist.
Ansätze hierzu sind in (Fantechi und Gnesi, 2011) beschrieben. Demnach greift der RTCA-Standard DO-178B aus der Avionik diese Frage unter dem Schlagwort *Tool Qualification* auf und nennt als Qualifikationskriterium den Nachweis, dass die Software nach einem methodisch einwandfreien Vorgehen getestet wurde. Nach diesem Standard wurde bisher jedoch laut den Autoren in (Fantechi und Gnesi, 2011) kein Model Checker zertifiziert.
Als alternatives Qualifikationskriterium nennen die Autoren das Stichwort *proven in use* aus dem Standard EN 50128. Demnach ist ein Tool qualifiziert, wenn es eine hinreichend lange *Referenzliste* an Anwendungsfällen ohne bekanntes Fehlverhalten gibt. Dies wäre im Fall von UPPAAL wohl wie bei keinem anderem Model Checker gegeben (Larsen et al., 1995a,b; Bengtsson et al., 1996; Larsson et al., 1997; Havelund et al., 1997; D'Argenio et al., 1997; Aceto et al., 1998; Bowman et al., 1998a,b; Lindahl et al., 1998; Havelund et al., 1999; David und Yi, 2000; David et al., 2009; Ravn et al., 2010, 2011). Da zugleich kein einziger publizierter Fall eines Fehlverhaltens von UPPAAL bekannt ist, scheint das Tool prinzipiell ein geeigneter Kandidat für die Verifikationsaufgabe zu sein.

Ausblick

Den Herausforderungen, die in den vorausgehenden Absätzen aufgezeigt wurden, kann auf unterschiedliche Weise begegnet werden. Besonders Einschränkungen in der Ausdruckskraft der Anforderungsspezifikation können durch den Ausbau der Verification Toolbox adressiert werden. Parallel zur Verifikation von TCTL-Formeln in UPPAAL ist vorstellbar, weitere temporale Logiken und etablierte Tools, wie etwa SPIN oder SMV für die Verifikation einzusetzen. Vorbereitend darauf wurden in dieser Arbeit bereits in der Modul-Beschreibung Elemente aufgenommen, die erst durch den Ausbau der Verification Toolbox verifizierbar werden (z.B. die beliebig häufige Wiederkehr in eine Location).
Verifikationsabbrüchen aufgrund von begrenztem Speicher bei der derzeitigen Toolversion kann durch Nutzung alternativer Betriebssysteme entgegengewirkt werden.
Die letzte UPPAAL-Version 4.1.19 ist für MAC OS-X und Linux bereits als 64-bit Version verfügbar, mit der mehr RAM als in der 32-bit Version adressiert werden kann.
Wichtiger jedoch ist es, die Beschreibungsmethode, insbesondere was die Realisierung der Beschreibungsdatei und der darin formulierten Anforderungen angeht, weiter auszubauen, um

- weitere Anforderungen automatisiert zu generieren,
- die Kniffe, die in Kapitel 6 angewendet werden mussten, überflüssig zu machen und so
- die Verantwortung des Risikomanagements weiter Richtung Hersteller auszulagern, um
- den Betreiber bei der Konformitätsbewertung medizinischer IT-Netzwerke zu unterstützen.

In dieser Arbeit wurde ein erster Schritt in diese Richtung unternommen.

Glossar

action
Ausgabesignal eines Automaten.

AIS
Lehrstuhl für Automatisierung und Informationssysteme der TU München.
Leitung: Prof. Dr.-Ing. Birgit Vogel-Heuser.

Alphabet
Nicht-leere, endliche Menge. Häufig wird der Begriff Alphabet im Zusammenhang mit Automaten gewählt und beschreibt dabei die Menge an Ein- und Ausgabesignalen des Automaten.

Anwender
Personen, die ein Medizinprodukt entsprechend seiner Zweckbestimmung an einem Patienten anwenden. Personen, die lediglich unter Aufsicht eines Anwenders ein Medizinprodukt bedienen, fallen dabei nicht unter die Definition, da sie dabei nicht eigenverantwortlich handeln (Jäkel, 2016).

asynchron
Nicht zeitgleich. Häufig wird Asynchronität im Zusammenhang mit Kommunikation verwendet. Asynchrone Kommunikation bezeichnet einen Modus der Kommunikation, bei dem Versand und Empfang von Daten zeitlich versetzt und ohne Blockieren von Sender oder Empfänger stattfindet.

Ausgabevariable
Ausgabevariablen bezeichnen Variablen einer Netzwerkkomponente, welche die Komponente selbst manipuliert (Anzeige, Wirkung, weitere Komponenten). Siehe Abschnitt 5.2.4.

Betreiber
Betreiber im juristischen Sinne (vgl. Urteil des Bundesverwaltungsgerichts 16.12.2003, 3C 47/02) ist derjenige, der die Sachherrschaft über das Gerät ausübt. Umstritten ist dabei, ob dafür die tatsächliche Sachherrschaft (Gebrauch und Benutzung, ggf. ohne Eigentumsrechte), oder die rechtliche Sachherrschaft (Eigentümerstellung, ggf. ohne Besitz) ausreicht. Nach allgemeiner Auffassung sind Betreiber beispielsweise Klinikträger, Inhaber einer Arztpraxis oder ein Pflegedienst. Der Betreiberbegriff sollte mit dem Entwurf des 3. MPG-Änderungsgesetzes 2005 rechtlich verbindlich definiert werden. Das Vorhaben ist jedoch fallen gelassen worden. Das Bundesgesundheitsministerium plant derzeit Änderungen der MPBetreibV, in welcher der Betreiberbegriff dann festgehalten werden soll (Jäkel, 2016; Euteneier, 2015).

Binäre Synchronisation
Synchrone Daten- oder Signalübertragung zweier Akteure.

Broadcast Synchronisation
Eine Broadcast Nachricht ist eine Nachricht, die an alle beteiligten Module des Netzwerks geht, die diese empfangen können. In dieser Arbeit ist ein wichtiges Broadcast-Signal das soc-Signal, mit dem der *Managing Node* (MN) allen CNs mitteilt, dass ein neuer Zyklus

begonnen hat.

co-action
Eingabesignal eines Automaten.

DICOM
Digital **I**maging and **Co**mmunications in **M**edicine. Ein offener Standard für den Austausch und die Archivierung medizinischer Bilddaten. Siehe (Mildenberger et al., 2002).

DIN EN 80001-1
Standard für das Risikomanagement von *Medizinisches IT-Netzwerk* (MIT-Netzwerk)en. Siehe Literaturverzeichnis → (DIN EN 80001-1)

DIN EN ISO 13485
Standard für Qualitätsmanagementsysteme für Hersteller von Medizinprodukten. Siehe Literaturverzeichnis → (DIN EN ISO 13485).

DIN EN ISO 14971
Standard für die Anwendung des Risikomanagements auf Medizinprodukte. Siehe Literaturverzeichnis → (DIN EN ISO 14971).

Echtzeit-Bussystem
In dieser Arbeit bezeichnet das Echtzeit-Bussystem ein an der TU München entwickeltes, echtzeitfähiges Bussystem für die Integration von Medizingeräten und Zubehör.

Eigenhersteller
Erzeuger von Medizinprodukten aus Eigenherstellung. Diese sind “Medizinprodukte einschließlich Zubehör, die in einer Gesundheitseinrichtung hergestellt und angewendet werden, ohne dass sie in den Verkehr gebracht werden oder die Voraussetzungen einer Sonderanfertigung [...] erfüllen” (MPG).

Eingabevariable
Eingabevariablen bezeichnen Variablen einer Netzwerkkomponente, die von außen (Anwender, weitere Komponenten) manipuliert werden. Siehe Abschnitt 5.2.4.

epl_get
Globale Eingabevariable eines TEFA. Siehe Abschnitt 5.2.4.

epl_set
Globale Ausgabevariable eines TEFA. Siehe Abschnitt 5.2.4.

Funktionsmodul
Funktionsmodule sind *Netzteilnehmer* am Echtzeit-Bussystem, die auf Basis standardisierter Netzwerk-Daten dedizierte Funktionen wahrnehmen (automatische Leistungssteuerung, Verrechnung von Navigationsdaten etc.). Siehe (Lüth et al., 2016).

Globale Variable
In dieser Arbeit wird eine Variable als globale bezeichnet, die über das Echtzeit-Bussystem propagiert wird. Siehe Abschnitt 5.2.4.

Grammatik
Eine *Grammatik* G ist ein Tupel $G = (N, \Sigma, P, S)$ mit einer endlichen Menge N (genannt: *Nichtterminale*), einem Alphabet Σ, einer endlichen Menge $P \subseteq ((N \cup \Sigma)^* \backslash \Sigma^*) \times (N \cup \Sigma)^*$

(genannt: *Ableitungsregeln*) und einem Startsymbol $S \in N$. Siehe Definition 4.1.2.

Grundlegende Anforderungen

Anforderungen an Medizinprodukte für den europäischen Wirtschaftsraum nach Richtlinie 93/42 EWG, Anhang I (MDD).

Gültigkeits-Flag

Binäre Variable, die einem Datum zugehörig ist. Der Wert der Variable deutet auf die Gültigkeit (*true*) bzw. Ungültigkeit (*false*) des zugehörigen Datums hin.

Harmonisierte Norm

Eine von der Europäischen Kommission und EFTA mandatierte Norm, deren Fundstelle von der Europäischen Kommission im EU-Amtsblatt bekannt gegeben wurde.

Hersteller

Die “[...] natürliche oder juristische Person, die für die Auslegung, Herstellung, Verpackung und Etikettierung eines Produkts im Hinblick auf das Inverkehrbringen im eigenen Namen verantwortlich ist, unabhängig davon, ob diese Tätigkeiten von dieser Person oder stellvertretend für diese von einer dritten Person ausgeführt werden.” (Art. 1 Abs. 2f MDD).

Initial Location

Eindeutige Start-Location eines Timed Extended Finite Automaton.

Integrierter Operationssaal

Monolithisches, informationstechnisches Netzwerk von Medizingeräten und Zubehör mit einer zentralen Bedienstation, welche Einstellungen an den angeschlossenen Geräten erlaubt.

Interleaving

Eine Form der nebenläufigen Ausführung mehrerer TEFAs. Beim Interleaving werden alle Pfade der beteiligten TEFAs unter Erhaltung ihrer Reihenfolge miteinander vermischt. Daneben können simultane Zustandsübergänge realisiert werden, die durch ein synchronisierendes Signal ausgelöst werden. Siehe Definition 4.1.13.

Invariante

In der Theorie getimter Automaten ist eine Invariante eine Bedingung an eine Uhr, die auf eine Location bezogen ist (Henzinger et al., 1994).

Inverkehrbringer

Im Umfeld der Medizinprodukte bezeichnet das Inverkehrbringen die "‘[...] erste entgeltliche oder unentgeltliche Überlassung eines Produkts, das nicht für klinische Prüfungen bestimmt ist, im Hinblick auf seinen Vertrieb und/oder seine Verwendung inner-halb der Gemeinschaft, ungeachtet dessen, ob es sich um ein neues oder ein als neu aufbereitetes Produkt handelt."’ (Art. 1, Abs. 2h MDD).

isochronen

Gleich lange dauernd. Im Umfeld von *Ethernet-Powerlink* (EPL) ist damit gemeint, dass sowohl die Zyklen eine feste, gleichbleibende Dauer haben, als auch die Zeitslots innerhalb eines Zyklus, die einem *Controlled Node* (CN) eingeräumt werden.

Key

Siehe *Key-Value-Pair*.

Key-Value-Pair

Als Key-Value-Pairs werden in dieser Arbeit Paare aus zwei Bezeichnern genannt: Der Key steht dafür, was beschrieben werden soll (z.B. "'Name"'), der Value spezifiziert dann den konkreten Wert des Keys, z.B. "'Peter"'. Siehe Abschnitt 5.2.

Kleene-Abschluss

Sei A ein Alphabet. Die Menge $A^* := \bigcup_{k=0}^{\infty} A^k$ mit $A^0 := \{\epsilon\}$ heißt der Kleene-Abschluss von A. Siehe Definition 4.1.1.

Konkatenation

Seien A und B *Alphabete*. Die Konkatenation ist eine Abbildung

$$\cdot : A^* \times B^* \to (A \cup B)^*, (a, b) \mapsto ab,$$

welche zwei Strings aus A^* und B^* aneinanderreiht. Siehe Definition 4.1.1.

Konnektor

Hardware-Modul zur nachträglichen Integration bestehender Medizingeräte in das OR.Net Echtzeit-Bussystem (Pfeiffer et al., 2015).

Liveness

In der Automatentheorie wird als Liveness die Einhaltung einer Liveness-Anforderung bezeichnet. Liveness-Anforderungen gehen von der Auszeichnung mancher Zustände des Zustandsraums des Automaten als *good* aus. Liveness ist dabei die Einhaltung der Anforderung, dass ein Zustand der Klasse *good* zwingend erreicht wird (Alur, 2015).

Location

In der Theorie der Timed Extended Finite Automata ist eine Location der erste Eintrag eines Zustands-Tupels (siehe Definition 4.1.11).

Lokale Variable

In dieser Arbeit wird eine Variable als lokal bezeichnet, wenn sie nicht über das Echtzeit--Bussystem propagiert wird, sondern nur einem beteiligten CN bekannt ist. Siehe Abschnitt 5.2.4.

manual_get

Lokale Eingabevariable eines TEFA. Siehe Abschnitt 5.2.4.

Mapping Mode

Prozess des Ethernet Powerlink Managing Nodes, welcher die Zuordnung von Datenquellen zu Datensenken innerhalb eines gegebenen Netzwerks vornimmt. Der Prozess wird nebenläufig zum Polling Mode ausgeführt.

MDD

Richtlinie 93/42/EWG des Rates über Medizinprodukte. Die MDD wird mit Wirkung zum 26. Mai 2020 aufgehoben (Artikel 122 (MDR)) und durch die MDR ersetzt. Die in dieser Arbeit verwendeten Passagen sind in ähnlichem Wortlaut in der neuen Verordnung zu finden. Siehe Literaturverzeichnis → (MDD).

MDIB

Medical **D**evice **I**nformation **B**ase. Bezeichnung für die Gerätespezifikation aus der ISO 11073.

MDR

Verordnung (EU) 2017/745 des europäischen Parlaments und des Rates vom 05. April 2017 über Medizinprodukte. Die MDR ersetzt mit Wirkung zum 26. Mai 2020 die MDD (Artikel 122 (MDR)). Siehe Literaturverzeichnis → (MDR).

Medizinprodukt

"'[...] alle einzeln oder miteinander verbunden verwendeten Instrumente, Apparate, Vorrichtungen, Software, Stoffe oder anderen Gegenstände, einschließlich der vom Hersteller speziell zur Anwendung für diagnostische und/oder therapeutische Zwecke bestimmten und für ein einwandfreies Funktionieren des Medizinprodukts eingesetzten Software, die vom Hersteller zur Anwendung für Menschen für folgende Zwecke bestimmt sind:

- Erkennung, Verhütung, Überwachung, Behandlung oder Linderung von Krankheiten;
- Erkennung, Überwachung, Behandlung, Linderung oder Kompensierung von Verletzungen oder Behinderungen;
- Untersuchung, Ersatz oder Veränderung des anatomischen Aufbaus oder eines physiologischen Vorgangs;
- Empfängnisregelung,

und deren bestimmungsgemäße Hauptwirkung im oder am menschlichen Körper weder durch pharmakologische oder immunologische Mittel noch metabolisch erreicht wird, deren Wirkungsweise aber durch solche Mittel unterstützt werden kann."' (Art. 1 Abs. 2a MDD).

Middleware

Software-Schicht, die "Dienstleistungen für die Integration in einer verteilten, heterogenen Umgebung erbringt." (Riehm und Vogler, 1996).

MiMed

Lehrstuhl für Mikrotechnik und Medizingerätetechnik der TU München.

Leitung: Prof. Dr. rer. nat. Dipl.-Ing. Tim C. Lueth.

MITI

Forschungsgruppe für minimal-invasive interdisziplinäre therapeutische Intervention der TU München.

Leitung: Prof. Dr. Hubertus Feußner.

Model Checker

Software, die Methoden des Model Checking implementiert.

Model Checking

Eine Technik zur automatisierten Prüfung der Gültigkeit einer Aussage in einem Modell, das durch endlich viele Zustände beschrieben wird (Baier et al., 2008e).

Modul

Medizingerät und/oder Zubehör, das an das Echtzeit-Bussystem angeschlossen ist und über dieses Informationen austauscht.

monolithisch

Monolithische Systeme zeichnen sich dadurch aus, dass ihre Bestandteile vor dem Inverkehrbringen durch einen Hersteller vollständig spezifiziert sind. Dies können Integrierte Operationssäle größerer Hersteller (siehe Abs. 1.1.3) sein, oder aber auch herstellerge-

bundene Insellösungen (siehe Abs. 1.1.4).

MPBetreibV

Medizinprodukte-Betreiberverordnung. Siehe Literaturverzeichnis → (MPBetreibV).

MPG

Medizinproduktegesetz. Siehe Literaturverzeichnis → (MPG).

Netzteilnehmer

Medizingeräte und Zubehör, die über das Echtzeit-Bussystem kommunizieren.

observable

Lokale Ausgabevariable eines TEFA. Siehe Abschnitt 5.2.4.

Payload

Im Bereich der Kommunikation bezeichnet Payload die Nutzdaten eines übertragenen Datenpakets.

Permutation

Sei A ein Alphabet. Eine Permutation auf A ist eine bijektive Abbildung $\rho : A \to A$. Siehe Definition 4.1.8.

Polling

Zyklisches Abfragen von Werten. Im EPL-Umfeld werden CNs, z.B. ein Fußschalter, in jedem Zyklus gepollt, damit sie ihre neuen Werte mitteilen.

Polling Mode

Prozess des Ethernet Powerlink Managing Nodes, welcher alle Controlled Nodes sukzessive abfragt und mit Daten versorgt. Der Prozess wird nebenläufig zum Mapping Mode ausgeführt.

proprietäres Kommunikationsprotokoll

Ein proprietäres Kommunikationsprotokoll ist firmengebunden und in der Regel nicht öffentlich einsehbar, so dass die Einbindung von Systemen von Drittanbietern nicht ohne Absprache mit dem entsprechenden Hersteller möglich ist.

Realisierung

Das Tupel $\mathbf{MAP} = (u, w) : C \times V \to [0, \infty) \times \mathbb{Z}$ aus Zeitpunkt und Wertzuweisung heißt Realisierung. Siehe Abschnitt 4.1.10.

RES

Lehrstuhl für Echtzeitsysteme und Robotik der TU München.
Leitung: Prof. Dr.-Ing. habil. Alois Christian Knoll.

Risikomanagement

Die "'[...] systematische Anwendung von Managementstrategien, Verfahren und Praktiken auf die Aufgaben der Analyse, Bewertung, Beherrschung und Überwachung von Risiken"' (DIN EN ISO 14971).

Safety

In der Automatentheorie wird als Safety die Einhaltung einer Safety-Anforderung bezeichnet. Safety-Anforderungen gehen von der disjunkten Unterteilung des Zustandsraums des

Automaten in *safe* und *unsafe* aus. Safety ist dabei die Einhaltung der Anforderung, dass ein Zustand der Klasse *unsafe* niemals erreicht wird (Alur, 2015).

Shared Memory

Gemeinsamer Speicherbereich für nebenläufige Prozesse.

Stateflow

Eine Matlab-Umgebung für die Modellierung, Simulation und Verifikation kombinatorischer und sequenzieller Entscheidungslogik auf Grundlage von Zustandsautomaten und Flussdiagrammen. Siehe Abschnitt 5.6.3.

String

Sei A ein Alphabet. Elemente der Form

$$x_1 x_2 \dots x_k := \left(x_1, x_2, \dots x_k\right) \in A^k \ (k \in \mathbb{N})$$

heißen Strings der Länge k. Siehe Definition 4.1.1.

Surgical Deck

Prototyp eines vernetzten HNO- Operationssaals, der Elemente unterschiedlicher Vernetzungsarchitekturen vereint, um Vorteile in der Ergonomie und Betriebssicherheit zu erzielen (Strauss et al., 2013).

synchron

Zeitgleich. Häufig wird Synchronität im Zusammenhang mit Kommunikation verwendet. Synchrone Kommunikation bezeichnet einen Modus der Kommunikation, bei dem Versand und Empfang von Daten zeitgleich stattfindet. Im Gegensatz zur asynchronen Kommunikation blockieren Sender und Empfänger bis die Datenübertragung stattfindet.

TEFA

Timed Extended Finite Automaton.

Transition

Zustandsübergang eines Automaten.

u_0

Nullabbildung.

UPPAAL

Ein Tool zur automatisierten Verifikation von TEFAs. Siehe Abschnitt 5.6.2.

Value

Siehe Key-Value-Pair.

Vereinfachtes Konformitätsbewertungsverfahren

Eine “[...]Konformitätserklärung des Herstellers, deren Zulässigkeit bei den Behörden durch das Bestehen von Rechtsvorschriften über die Produkthaftung einfacher geworden ist.” (Council of the European Communities, 2016). Im Zusammenhang mit Medizinprodukten wird damit häufig das Verfahren nach Medizinprodukteverordnung §7 Abs. 9 bezeichnet, das für Medizinprodukte aus Eigenherstellung zulässig ist.

Verschaltung

Zuordnung von Datenquellen zu Datensenken.

Wertzuweisung

Eine `Wertzuweisung` $w \in \mathbb{Z}^D$ ist eine Abbildung, die jeder Variable eine ganze Zahl zuweist. Siehe Abschnitt 4.1.10.

Zeitpunkt

In der Theorie der Timed Extended Finite Automata ist ein Zeitpunkt eine Abbildung, die jeder Uhr eine nicht-negative reelle Zahl zuordnet.

Akronyme

Aesculap
Aesculap AG

BfArM
Bundesinstitut für Arzneimittel und Medizinprodukte
BFS
Breitensuche

CFC
Continuous Function Charts
CN
Controlled Node
CT
Computer-Tomographie
CTL
Computation Tree Logic
CTL*
Branching Time Logic

DDS
Data Distribution Service
DFS
Tiefensuche
DPWS
Devices Profile for Web Services

EPL
Ethernet-Powerlink
Ergosurg
Ergosurg GmbH

FDA
U.S. Food and Drug Administration
FESS
funktionalen endoskopischen Sinus-Chirurgie
FMEA
Failure Mode and Effects Analysis
FTA
Fault Tree Analysis

gdw
genau dann, wenn

GUI
Graphical User Interface

HF-Gerät
Hochfrequenz-Gerät

HML
Hennessy Milner Logic

inomed
inomed Medizintechnik GmbH

IONM
intraoperatives Neuromonitoring

Karl Storz
KARL STORZ GmbH & Co. KG

KIS
Klinische Informationssysteme

KLS Martin
Gebrüder Martin GmbH & Co. KG

LTL
Linear Temporal Logic

MAUDE
Manufacturer and User Facility Device Experience

MIT-Netzwerk
Medizinisches IT-Netzwerk

MN
Managing Node

MRT
Magnetresonanztomographie

NDI
Northern Digital Inc.

OSCP
Open Surgical Communication Protocol

p.a.
im Jahr

PACS
Picture Archiving and Communication System

PHA
Preliminary Hazard Analysis

POR
Partial Order Reduction

preq
Poll Request

pres
Poll Response

PROMELA
Protocol Meta Language

ROBDD
Reduced Ordered Binary Decision Diagrams

SMV
Symbolic Model Verifier

SoC
Start of Cycle

Söring
Söring GmbH

steute
steute Schaltgeräte GmbH & Co. KG

TCTL
Timed Computation Tree Logic

TDMA
Time Division Multiple Access

TEFA
Timed Extended Finite Automaton

US-Dissektor
Ultraschall-Dissektor

XML
Extensible Markup Language

Symbolverzeichnis

$|$

Logische Disjunktion für die verkürzte Notation einer Grammatik. Siehe Bemerkung 2.

$\Rightarrow$

Implikation. Ist eine Aussage φ hinreichend für eine Aussage ψ, so wird dies durch die Zeichensequenz $\varphi \Rightarrow \psi$ beschrieben. In diesem Fall heißt ψ *notwendig* für φ.

$\Leftrightarrow$

Äquivalenz. Ist eine Aussage φ hinreichend und notwendig für eine Aussage ψ, so wird dies durch die Zeichensequenz $\varphi \Leftrightarrow \psi$ beschrieben. In dem Fall heißt ψ *äquivalent* zu φ.

$\cup$

Vereinigung. Für beliebige Mengen A, B ist $A \cup B := \{c : c \in A \text{ oder } c \in B\}$.

$\dot{\cup}$

Disjunkte Vereinigung. Für beliebige Mengen A, B schreibt man $A \dot{\cup} B := A \cup B$, um zusätzlich zu verdeutlichen, dass $A \cap B = \emptyset$.

$\vee$

Logische Disjunktion. Der Ausdruck $A \vee B$ ist wahr, wenn A, oder B, oder beide wahr sind.

$\bigwedge_{k=1}^{m-1} \varphi_k$

Multiple Konjunktion. Es ist $\bigwedge_{k=1}^{m-1} \varphi_k := \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$. Im Allgemeinen ist dieser Ausdruck abhängig von der Reihenfolge der $\left(\varphi_k\right)_{k=1,\ldots n}$.

$\wedge$

Logische Konjunktion. Der Ausdruck $A \wedge B$ ist wahr, wenn sowohl A, als auch B wahr ist.

$\models$

Erfüllung einer Bedingung. Siehe Definition 4.1.4.

2^A

Die Potenzmenge von A. Sie besteht aus allen Teilmengen der Menge A.

$|A|$

Kardinalität der Menge A.

$\mathcal{A}(A)$

Sprache der affinen Zuordnungen. Siehe Definition 4.1.6.

$\mathcal{A}_0(A)$

Sprache der Resets. Siehe Definition 4.1.6.

a!

Action. Ausgabesignal eines TEFA. Siehe Bemerkung 10.

a?

Co-Action. Eingabesignal eines TEFA. Siehe Bemerkung 10.

A^*

Kleene-Abschluss des Alphabets A. Diese Menge enthält genau alle Strings endlicher Länge, die aus dem Alphabet A geformt werden können.

$B \backslash A$

Komplement von A in B. Diese Menge besteht aus den Elementen, die in B, aber nicht in A enthalten sind.

B^A

Menge der Funktionen, welche die Menge A auf die Menge B abbilden.

ϵ

Leerer String in der Theorie formaler Sprachen.

$f|_C : C \to B$

Die Einschränkung der Funktion f auf die Teilmenge C ihres Definitionsbereichs. $f|_C$ stimmt auf der Teilmenge C mit der Funktion f überein; es gilt also $f|_C(x) := f(x)$ für $x \in C$.

Grammatik

Eine Grammatik G ist ein Tupel $G = (N, \Sigma, P, S)$ mit einer endlichen Menge N (genannt: *Nichtterminale*), einem Alphabet Σ, einer endlichen Menge $P \subseteq ((N \cup \Sigma)^* \backslash \Sigma^*) \times (N \cup \Sigma)^*$ (genannt: *Ableitungsregeln*) und einem Startsymbol $S \in N$.

$[k]$

Die ersten k natürlichen Zahlen, also $[k] := \{1, 2, \ldots k\}$.

$L(G)$

Die aus der Grammatik G erzeugte Sprache. Siehe Definition 4.1.2.

$\Phi_M(A)$

Sprache der Zwangsbedingungen über A und M. Siehe Definition 4.1.3.

$[\varphi]\mu$

Induzierte Abbildung. Siehe Definition 4.1.7.

$\prod_{i=1}^{n} S_i$

Kartesisches Produkt der Mengen $S_1, \ldots S_n$. Es gilt

$$\prod_{i=1}^{n} S_i := \left\{ \left(s_1, \ldots s_n \right) : s_1 \in S_1, \ldots, s_n \in S_n \right\}.$$

S_t

Enthaltene Zustände im Pfad t. Siehe Definition 4.1.12.

$[[\mathcal{T}]]$

Pfad eines TEFA $\mathcal{T}$. Siehe Definition 4.1.12.

$\mathcal{T}_1 \parallel \mathcal{T}_2 \parallel \ldots \parallel \mathcal{T}_n$

Interleaving-Automat der TEFAs $\mathcal{T}_1, \ldots, \mathcal{T}_n$. Siehe Definition 4.1.13.

$\left(\mathcal{T}_1 \parallel \mathcal{T}_2 \parallel \ldots \parallel \mathcal{T}_n\right)_{\mathbb{B}}$

Der durch Broadcast-Synchronisation erzeugte Automat der TEFAs $\mathcal{T}_1, \ldots, \mathcal{T}_n$. Siehe Definition 4.1.14.

$x\left[\tilde{x}_j/x_j\right]$

Derjenige Vektor, der an allen Einträgen bis auf den j-ten mit x übereinstimmt und dessen j-ter Eintrag $\tilde{x}_j$ ist, also $x\left[\tilde{x}_j/x_j\right] := \left(x_1, x_2, \ldots, \tilde{x}_j \ \ldots, x_n\right)$.

$x_1 x_2 \ldots x_k$

String. Aneinanderreihung der Elemente $x_1, \ldots x_n$.

Literaturverzeichnis

[Aceto et al. 1998] ACETO, Luca ; BURGUENO, Augusto ; LARSEN, Kim G.: Model Checking via Reachability Testing for Timed Automata. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 1998, S. 263–280

[Ahlbrandt et al. 2013] AHLBRANDT, Janko ; RÖHRIG, Rainer ; DEHM, Johannes ; WREDE, Christian ; IMHOFF, Michael: Risikomanagement für medizinische Netzwerke in der Intensiv-und Notfallmedizin. Gemeinsames Positionspapier zur Norm IEC 80001-1. In: *GMS Medizinische Informatik, Biometrie und Epidemiologie* 9 (2013), Nr. 3

[Althoff 2010] ALTHOFF, Matthias: *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*, Technische Universität München, Dissertation, 2010

[Alur 2015] ALUR, Rajeev: *Safety Requirements*. Kap. 3, S. 65–124. In: *Principles of Cyber-Physical Systems*, MIT Press, 2015

[Alur et al. 1990] ALUR, Rajeev ; COURCOUBETIS, Costas ; DILL, David: Model-Checking for Real-Time Systems. In: *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, 1990, S. 414–425

[Alur et al. 1993] ALUR, Rajeev ; COURCOUBETIS, Costas ; DILL, David: Model-Checking in Dense Real-Time. In: *Information and Computation* 104 (1993), Nr. 1, S. 2–34

[Alur und Dill 1994] ALUR, Rajeev ; DILL, David L.: A Theory of Timed Automata. In: *Theoretical computer science* 126 (1994), Nr. 2, S. 183–235

[Amnell et al. 2002] AMNELL, Tobias ; FERSMAN, Elena ; MOKRUSHIN, Leonid ; PETTERSSON, Paul ; YI, Wang: TIMES - A Tool for Modelling and Implementation of Embedded Systems. In: *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. London, UK : Springer-Verlag, 2002 (TACAS '02), S. 460–464

[Andersen et al. 2016] ANDERSEN, Björn ; KASPARICK, Martin ; GOLATOWSKI, Frank ; INGENERF, Josef ; OTHERS: Extending the IEEE 11073-1010X Nomenclature for the Modelling of Surgical Devices. In: *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)* IEEE (Veranst.), 2016, S. 244–247

[Anheuser und Steffens 2011] ANHEUSER, Petra ; STEFFENS, Joachim: *Komplikationen der Inkontinenzchirurgie, Prolapschirurgie, Prothetik*. Kap. 12, S. 196–223. In: *Risiken und Komplikationen in der Urologie: systematisch-praxisnah-präventiv*, Georg Thieme Verlag, 2011

[Arney et al. 2014] ARNEY, David ; PLOURDE, Jeff ; SCHRENKER, Rick ; MATTEGUNTA, Pratyusha ; WHITEHEAD, Susan F. ; GOLDMAN, Julian M.: Design Pillars for Medical Cyber-Physical System Middleware. In: *MCPS*, 2014, S. 124–132

[Aßmann et al. 2014] ASSMANN, Olga ; MEIN, Silke ; KÖPCKE, Jens: Orthopädie und Traumatologie. In: (Liehn et al., 2014b), Kap. 8, S. 211–261

[Baier et al. 2008a] BAIER, Christel ; KATOEN, Joost-Pieter ; LARSEN, Kim G.: Linear Temporal Logic. In: *Principles of Model Checking*. (Baier et al., 2008d), Kap. 5, S. 229–312

[Baier et al. 2008b] BAIER, Christel ; KATOEN, Joost-Pieter ; LARSEN, Kim G.: Linear-Time Properties. In: *Principles of Model Checking*. (Baier et al., 2008d), Kap. 3, S. 89–150

[Baier et al. 2008c] BAIER, Christel ; KATOEN, Joost-Pieter ; LARSEN, Kim G.: Modelling Concurrent Systems. In: *Principles of Model Checking*. (Baier et al., 2008d), Kap. 2, S. 19–88

[Baier et al. 2008d] BAIER, Christel ; KATOEN, Joost-Pieter ; LARSEN, Kim G.: *Principles of Model Checking*. Cambridge, MA : MIT Press, 2008

[Baier et al. 2008e] BAIER, Christel ; KATOEN, Joost-Pieter ; LARSEN, Kim G.: System Verification. In: *Principles of Model Checking*. (Baier et al., 2008d), Kap. 1, S. 1–18

[Baier et al. 2008f] BAIER, Christel ; KATOEN, Joost-Pieter ; LARSEN, Kim G.: TCTL Model Checking. In: *Principles of Model Checking*. (Baier et al., 2008d), Kap. 9, S. 673–744

[Büchel et al. 2014] BÜCHEL, Dirk ; SCHERRER, Martin ; MATERN, Ulrich: Festlegung der Bewertungskriterien für den Risikomanagementplan. In: CAPANNI, Felix (Hrsg.) ; MATERN, Ulrich (Hrsg.) ; STEFFEN, Alexander (Hrsg.) ; STOCKHARDT, Jörg (Hrsg.): *Der CE-Routenplaner. Medizinprodukte planen, entwickeln, realisieren*. Berlin : TÜV Media GmbH, 2014, S. 335–356

[Behrmann et al. 2003] BEHRMANN, Gerd et al.: *Data Structures and Algorithms for the Analysis of Real Time Systems*, Department of Computer Science, the Faculty of Engineering and Science, Aalborg University, Dissertation, 2003

[Behrmann et al. 2002] BEHRMANN, Gerd ; BENGTSSON, Johan ; DAVID, Alexandre ; LARSEN, Kim G. ; PETTERSSON, Paul ; YI, Wang: Uppaal Implementation Secrets. In: *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, Springer, 2002, S. 3–22

[Behrmann et al. 2004] BEHRMANN, Gerd ; DAVID, Alexandre ; LARSEN, Kim G.: A Tutorial on UPPAAL. In: *Formal Methods for the Design of Real-Time Systems*. Springer, 2004, S. 200–236

[Bengtsson 2002] BENGTSSON, Johan: *Clocks, DBMs and States in Timed Systems*, Department of Information Technology, Uppsala University, Dissertation, 2002

[Bengtsson et al. 1996] BENGTSSON, Johan ; GRIFFIOEN, W.O. D. ; KRISTOFFERSEN, Kåre J. ; LARSEN, Kim G. ; LARSSON, Fredrik ; PETTERSSON, Paul ; YI, Wang: Verification of an Audio Protocol with Bus Collision Using Uppaal. In: ALUR, Rajeev (Hrsg.) ; HENZINGER, Thomas A. (Hrsg.): *Computer-Aided-Verification 96*, Springer–Verlag, 1996 (Lecture Notes in Computer Science 1102), S. 244–256

[Bengtsson und Yi 2004] BENGTSSON, Johan ; YI, Wang: Timed Automata: Semantics, Algorithms and Tools. In: *Lectures on Concurrency and Petri Nets*. Springer, 2004, S. 87–124

[Beyer 2002] BEYER, Dirk: *Formale Verifikation von Realzeit-Systemen mittels Cottbus Timed Automata*, Brandenburgische Technische Universität Cottbus, Dissertation, 2002

[Bowman et al. 1998a] BOWMAN, Howard ; FACONTI, Giorgio ; KATOEN, Joost-Pieter ; LATELLA, Diego ; MASSINK, Mieke: Automatic Verification of a Lip-Synchronisation Algorithm Using Uppaal-Extended Version. In: *FMICS'98, Third Internatinoal Workshop on Formal Methods for Industrial Critical Systems*, CWI, 1998, S. 97–124

[Bowman et al. 1998b] BOWMAN, Howard ; FACONTI, Giorgio P. ; MASSINK, Mieke: *Specification and Verification of Media Constraints using UPPAAL*. S. 261–277. In: *Design, Specification and Verification of Interactive Systems '98: Proceedings of the Eurographics Workshop in Abingdon, UK, June 3–5, 1998*, Springer Vienna, 1998

[Brookes et al. 1984] BROOKES, Stephen D. ; HOARE, Charles A. ; ROSCOE, Andrew W.: A Theory of Communicating Sequential Processes. In: *Journal of the ACM (JACM)* 31 (1984), Nr. 3, S. 560–599

[Bryant 1986] BRYANT, Randal E.: Graph-based algorithms for boolean function manipulation. In: *Computers, IEEE Transactions on* 100 (1986), Nr. 8, S. 677–691

[Bundesanstalt für Arbeitsschutz 2014] BUNDESANSTALT FÜR ARBEITSSCHUTZ: *Arbeitsmedizin (BAuA) (1998) Technische Regel für Gefahrstoffe 525–Umgang mit Gefahrstoffen in Einrichtungen zur humanmedizinischen Versorgung (TRGS 525). 7*. 2014

[Clarke und Emerson 1981] CLARKE, Edmund M. ; EMERSON, E A.: Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic. In: *Workshop on Logic of Programs*, Springer, 1981, S. 52–71

[Clarke und Emerson 1982] CLARKE, Edmund M. ; EMERSON, E. A.: *Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic*. Kap. 4, S. 52–71. In: *Logics of Programs*. Yorktown Heights, New York : Springer Berlin Heidelberg, 1982

[Clarke et al. 1986] CLARKE, Edmund M. ; EMERSON, E A. ; SISTLA, A P.: Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 8 (1986), Nr. 2, S. 244–263

[Clarke et al. 1995] CLARKE, Edmund M. ; GRUMBERG, Orna ; HIRAISHI, Hiromi ; JHA, Somesh ; LONG, David E. ; MCMILLAN, Kenneth L. ; NESS, Linda A.: Verification of the Futurebus+ Cache Coherence Protocol. In: *Formal Methods in System Design* 6 (1995), Nr. 2, S. 217–232

[Clarke et al. 1999] CLARKE, E.M. ; GRUMBERG, O. ; MINEA, M. ; PELED, D.: State space reduction using partial order techniques. In: *International Journal on Software Tools for Technology Transfer* 2 (1999), Nr. 3, S. 279–287

[Council of the European Communities 2016] COUNCIL OF THE EUROPEAN COMMUNITIES: *Bekanntmachung der Kommission. Leitfaden für die Umsetzung der Produktvorschfiten der EU 2016 ("Blue Guide").* 2016

[D'Argenio et al. 1997] D'ARGENIO, Pedro R. ; KATOEN, J-P ; RUYS, Theo C. ; TRETMANS, Jan: The Bounded Retransmission Protocol Must be on Time! In: *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 1997, S. 416–431

[David 2003] DAVID, Alexandre: *Hierarchical Modeling and Analysis of Timed Systems*, Department of Information Technology, Uppsala University, Dissertation, 2003

[David et al. 2009] DAVID, Alexandre ; ILLUM, Jacob ; LARSEN, Kim G. ; SKOU, Arne: Model-Based Framework for Schedulability Analysis Using UPPAAL 4.1. In: *Model-based design for embedded systems* 1 (2009), Nr. 1, S. 93–119

[David und Yi 2000] DAVID, Alexandre ; YI, Wang: Modelling and Analysis of a Commercial Field Bus Protocol. In: *Proceedings of the 12th Euromicro Conference on Real Time Systems*, IEEE Computer Society, 2000, S. 165–172

[Daws et al. 1996] DAWS, Conrado ; OLIVERO, Alfredo ; TRIPAKIS, Stavros ; YOVINE, Sergio: The Tool KRONOS. In: *Hybrid Systems III*. Springer, 1996, S. 208–219

[Dickhaus und Metzner 2017] DICKHAUS, Hartmut ; METZNER, Roland: OP-Planung und OP-Unterstützung. In: (Kramme, 2017), Kap. 48, S. 867–878

[Dietz et al. 2016] DIETZ, Christian ; LÜDDEMANN, Tobias ; DINGLER, Max E. ; LUETH, Tim C.: Automated Risk Detection for Medical Device Networks with Hard Real Time Requirements. In: *2016 IEEE International Symposium on System Integration (SII)* IEEE (Veranst.), 2016, S. 471–476

[Dietz et al. 2017] DIETZ, Christian ; PFEIFFER, Jonas H. ; DINGLER, Max E. ; LÜTH, Tim C.: Neuro Control – Using Neuro Monitoring Data to Control Networked Active Instruments. In: *Current Directions in Biomedical Engineering* 3 (2017), Nr. 2, S. 449–452

[DIN EN 80001-1] : *DIN EN 80001-1:2011-11. Anwendung des Risikomanagements für IT-Netzwerke, die Medizinprodukte beinhalten - Teil 1: Aufgaben, Verantwortlichkeiten und Aktivitäten (IEC 80001-1:2010); Deutsche Fassung EN 80001-1:2011*

[DIN EN ISO 13485] : *DIN EN ISO 13485:2012 + AC:2012, Medizinprodukte - Qualitätsmanagementsysteme - Anforderungen für regulatorische Zwecke (ISO 13485:2003 + Cor. 1:2009); Deutsche Fassung EN ISO 13485:2012 + AC:2012*

[DIN EN ISO 14971] : *DIN EN ISO 14971:2013. Medizinprodukte - Anwendung des Risikomanagements auf Medizinprodukte (ISO 14971:2007, korrigierte Fassung 2007-10-01); Deutsche Fassung EN ISO 14971:2012*

[Dingler et al. 2017] DINGLER, Max ; DIETZ, Christian ; LÜTH, Tim: Verification of on-demand medical device networks. In: *Current Directions in Biomedical Engineering* 3 (2017), Nr. 2, S. 445–448

[Dingler et al. 2015] DINGLER, Max E. ; DIETZ, Christian ; PFEIFFER, Jonas H. ; LUEDDEMANN, Tobias ; LUETH, Tim C.: A Framework for Automatic Testing of Medical Device Compatibility. In: *Telecommunications (ConTEL), 2015 13th International Conference on*. Graz : IEEE, 2015, S. 1–8

[Dingler et al. 2016] DINGLER, Max E. ; ORTIZ, Diego ; DIETZ, Christian ; LUETH, Tim C.: An Approach Towards Compositional Behavior Specification of Medical Device Networks. In: *System Integration (SII), 2016 IEEE/SICE International Symposium on*, IEEE, 2016, S. 483–489

[Ericson 2016] ERICSON, Clifton A.: *Hazard Analysis Techniques for System Safety*. 2. John Wiley & Sons, 2016

[Ernst et al. 2015a] ERNST, Hartmut ; SCHMIDT, Jochen ; BENEKEN, Gerd: *Grundkurs Informatik. Grundlagen und Konzepte für die erfolgreiche IT-Praxis – Eine umfassende, praxisorientierte Einführung*. Springer Fachmedien Wiesbaden, 2015

[Ernst et al. 2015b] ERNST, Hartmut ; SCHMIDT, Jochen ; BENEKEN, Gerd: Höhere Programmiersprachen und C. In: *Grundkurs Informatik. Grundlagen und Konzepte für die erfolgreiche IT-Praxis – Eine umfassende, praxisorientierte Einführung*. (Ernst et al., 2015a), Kap. 14, S. 597–666

[Ethernet POWERLINK Standardisation Group 2013] ETHERNET POWERLINK STANDARDISATION GROUP: *Ethernet POWERLINK Communication Profile Specification. EPSG DS 301 V1.2.0.* 2013

[Euteneier 2015] EUTENEIER, Alexander (Hrsg.): *Handbuch klinisches Risikomanagement. Grundlagen, Konzepte, Lösungen - medizinisch, ökonomisch, juristisch*. 1. Springer-Verlag Berlin Heidelberg, 2015

[Fantechi und Gnesi 2011] FANTECHI, Alessandro ; GNESI, Stefania: On the Adoption of Model Checking in Safety-Related Software Industry. In: *LNCS 6894*, Springer-Verlag Berlin Heidelberg, 2011, S. 383–396

[FDA 2016a] FDA: *MAUDE Report Number 1226420-2016-00087*. online. 2016

[FDA 2016b] FDA: *MAUDE Report Number 8043933-2016-00028*. online. 2016

[Fehnker 2002] FEHNKER, A: *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking*, Radboud University of Nijmegen, Dissertation, 2002

[Fersman 2003] FERSMAN, Elena: *A Generic Approach to Schedulability Analysis of Real-Time Systems*, Department of Information Technology, Uppsala University, Dissertation, 2003

[Fokkink 2007] FOKKINK, Wan: Protocol Verifications. In: FOKKINK, Wan (Hrsg.): *Introduction to Process Algebra*. 2. Springer-Verlag Berlin Heidelberg New York London Paris Tokyo Hong Kong Barcelona Budapest, 2007, Kap. 6, S. 71 – 96

[Gödel 1931] GÖDEL, Kurt: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. In: *Monatshefte für Mathematik und Physik* 38 (1931), Nr. 1, S. 173–198

[Green et al. 1994] GREEN, J D. ; SHELTON, Clough ; BRACKMANN, Derald E.: Iatrogenic Facial Nerve Injury During Otologic Surgery. In: *The Laryngoscope* 104 (1994), Nr. 8, S. 922–926

[Gregorczyk et al. 2014] GREGORCZYK, David ; FISCHER, Stefan ; BUSSHAUS, Timm B. ; SCHLICHTING, Stefan ; PÖHLSEN, Stephan: An Approach to Integrate Distributed Systems of Medical Devices in High Acuity Environments. In: *MCPS*, 2014, S. 15–27

[Gärtner 2017] GÄRTNER, Armin: Kommunizierende medizinische Systeme und Netzwerke. In: (Kramme, 2017), Kap. 41, S. 759–765

[Haas und Kuhn 2017] HAAS, Peter ; KUHN, Klaus: Krankenhausinformationssysteme: Ziele, Nutzen, Topologie, Auswahl. In: (Kramme, 2017), Kap. 42, S. 767–794

[Hackenberg et al. 2014] HACKENBERG, Georg ; CAMPETELLI, Alarico ; LEGAT, Christoph ; MUND, Jakob ; TEUFL, Sabine ; VOGEL-HEUSER, Birgit: *Formal Technical Process Specification and Verification for Automated Production Systems*. S. 287–303. In: AMYOT, Daniel (Hrsg.) ; CASAS, Pau Fonseca i (Hrsg.) ; MUSSBACHER, Gunter (Hrsg.): *System Analysis and Modeling: Models and Reusability: 8th International Conference, SAM 2014, Valencia, Spain, September 29-30, 2014. Proceedings*. Cham : Springer International Publishing, 2014

[Harel 1987] HAREL, David: Statecharts: A Visual Formalism for Complex Systems. In: *Science of Computer Programming* 8 (1987), Nr. 3, S. 231–274

[Harer 2013] HARER, Johann (Hrsg.): *Anforderungen an Medizinprodukte. Praxisleitfaden für Hersteller und Zulieferer*. 1. München : Carl Hanser Verlag, 2013

[Hatcliff et al. 2012] HATCLIFF, John ; KING, Andrew ; LEE, Insup ; MACDONALD, Alasdair ; FERNANDO, Anura ; ROBKIN, Michael ; VASSERMAN, Eugene ; WEININGER, Sandy ; GOLDMAN, Julian M.: Rationale and Architecture Principles for Medical Application Platforms. In: *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems* (2012), S. 3–12

[Havelund et al. 1999] HAVELUND, Klaus ; LARSEN, Kim ; SKOU, Arne: Formal Verification of a Power Controller Using the Real-Time Model Checker Uppaal. In: *Formal Methods for Real-Time and Probabilistic Systems* (1999), S. 277–298

[Havelund et al. 2001] HAVELUND, Klaus ; LOWRY, Mike ; PENIX, John: Formal Analysis of a Space-Craft Controller Using SPIN. In: *IEEE Transactions on Software Engineering* 27 (2001), Nr. 8, S. 749–765

[Havelund et al. 1997] HAVELUND, Klaus ; SKOU, Arne ; LARSEN, Kim G. ; LUND, Kristian: Formal Modeling and Analysis of an Audio/Video Protocol: An Industrial Case Study Using UPPAAL. In: *Proceedings of the 18th IEEE Symposium on Real-Time Systems*, IEEE, 1997, S. 2–13

[Hennessy und Milner 1985] HENNESSY, Matthew ; MILNER, Robin: Algebraic Laws for Nondeterminism and Concurrency. In: *Journal of the ACM (JACM)* 32 (1985), Nr. 1, S. 137–161

[Henzinger et al. 1994] HENZINGER, Thomas A. ; NICOLLIN, Xavier ; SIFAKIS, Joseph ; YOVINE, Sergio: Symbolic Model Checking for Real-Time Systems. In: *Information and Computation* 111 (1994), Nr. 2, S. 193–244

[Hofmann und Lange 2011] HOFMANN, Martin ; LANGE, Martin: *Logik und Automaten*. Springer Heidelberg Dordrecht London New York, 2011

[Holzmann 1990] HOLZMANN, Gerard J.: *A Protocol Simulator*. Kap. 12, S. 243–296. In: *Design and Validation of Computer Protocols*. Englewood Cliffs, New Jersey 07632 : Prentice-Hall, Inc., 1990

[Holzmann 1994] HOLZMANN, Gerard J.: The Theory and Practice of A Formal Method: NewCoRe. In: PEHRSON, Björn (Hrsg.) ; SIMON, Imre (Hrsg.): *Technology and Foundations - Information Processing '94, Volume 1, Proceedings of the IFIP 13th World Computer Congress, Hamburg, Germany, 28 August - 2 September, 1994* Bd. A-51, North-Holland, 1994, S. 35–44

[Irion und Leonhard 2017] IRION, Klaus ; LEONHARD, Martin: Endoskopie. In: (Kramme, 2017), Kap. 22, S. 387–410

[Jensen 1999] JENSEN, Henrik E.: *Abstraction-Based Verification of Distributed Systems*, Aalborg University, Dissertation, 1999

[Jäkel 2016] JÄKEL, Christian: Medizinprodukterecht. In: (Schmola und Rapp, 2016), Kap. 16, S. 477–498

[Kasparick et al. 2015a] KASPARICK, Martin ; SCHLICHTING, Stefan ; GOLATOWSKI, Frank ; TIMMERMANN, Dirk: Medical DPWS: New IEEE 11073 Standard for Safe and Interoperable Medical Device Communication. In: *Proceedings of the 2015 IEEE Conference on Standards for Communications and Networking (CSCN'15), Tokyo, Japan*, 2015, S. 223–228

[Kasparick et al. 2015b] KASPARICK, Martin ; SCHLICHTING, Stefan ; GOLATOWSKI, Frank ; TIMMERMANN, Dirk: New IEEE 11073 Standards for Interoperable, Networked Point-of-Care Medical Devices. In: *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE* IEEE (Veranst.), 2015, S. 1721–1724

[Katz und Peled 1989] KATZ, Shmuel ; PELED, Doron: *An Efficient Verification Method for Parallel and Distributed Programs*. S. 489–507. In: BAKKER, J. W. de (Hrsg.) ; ROEVER, W. P. de (Hrsg.) ; ROZENBERG, G. (Hrsg.): *Linear Time, Branching Time and Partial Order in*

Logics and Models for Concurrency: School/Workshop, Noordwijkerhout, The Netherlands May 30 – June 3, 1988, Springer Berlin Heidelberg, 1989

[Kenngott et al. 2015] KENNGOTT, H. G. ; WAGNER, M. ; NICKEL, F. ; WEKERLE, A. L. ; PREUKSCHAS, A. ; APITZ, M. ; SCHULTE, T. ; REMPEL, R. ; MIETKOWSKI, P. ; WAGNER, F. ; TERMER, A. ; MÜLLER-STICH, Beat P.: Computer-assisted abdominal surgery: new technologies. In: *Langenbeck's Archives of Surgery* 400 (2015), Nr. 3, S. 273–281

[Kindler und Menge 2017] KINDLER, Manfred ; MENGE, Wolfgang: Vorschriften für Medizinprodukte. In: (Kramme, 2017), Kap. 4, S. 35–54

[King et al. 2015] KING, Andrew L. ; FENG, Lu ; PROCTER, Sam ; CHEN, Sanjian ; SOKOLSKY, Oleg ; HATCLIFF, John ; LEE, Insup: Towards Assurance for Plug & Play Medical Systems. In: KOORNNEEF, Floor (Hrsg.) ; GULIJK, Coen van (Hrsg.): *Computer Safety, Reliability, and Security* Bd. 9337. Delft, the Netherlands : Springer-Verlag, 2015, S. 228–242

[King et al. 2013] KING, Andrew L. ; FENG, Lu ; SOKOLSKY, Oleg ; LEE, Insup: A Modal Specification Approach for On-Demand Medical Systems. In: *Foundations of Health Information Engineering and Systems*. Springer, 2013, S. 199–216

[Kneissler et al. 2003] KNEISSLER, M ; HEIN, A ; MATZIG, M ; THOMALE, U W. ; LUETH, T C. ; WOICIECHOWSKY, C: Concept and Clinical Evaluation of Navigated Control in Spine Surgery. In: *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on* Bd. 2, 2003, S. 1084–1089 vol.2

[Koulechov und Lueth 2004] KOULECHOV, Kirill ; LUETH, Tim: A New Metric for Drill Location for Navigated Control in Navigated Dental Implantology. In: *International Congress Series* Bd. 1268 Elsevier (Veranst.), 2004, S. 1220–1225

[Kozen 1983] KOZEN, Dexter: Results on the Propositional μ-Calculus. In: *Theoretical Computer Science* 27 (1983), Nr. 3, S. 333–354

[Kramme 2017] KRAMME, Rüdiger (Hrsg.): *Medizintechnik. Verfahren - Systeme - Informationsverarbeitung*. 5. Berlin : Springer, 2017

[Kristoffersen 1998] KRISTOFFERSEN, Kare J.: *Compositional Verification of Concurrent Systems*, Aalborg University, Institute of Electronic Systems, Dissertation, 1998

[Kruber 2015a] KRUBER, Kurt: Aufgaben der Medizintechnik. In: (Euteneier, 2015), Kap. 15, S. 173–184

[Kruber 2015b] KRUBER, Kurt: Risikoreduzierung durch Medizintechnik und IT. In: (Euteneier, 2015), Kap. 34.3, S. 630–634

[Kühn und Leucker 2014] KÜHN, Franziska ; LEUCKER, Martin: OR.NET: Safe Interconnection of Medical Devices. In: GIBBONS, Jeremy (Hrsg.) ; MACCAULL, Wendy (Hrsg.): *Foundations of Health Information Engineering and Systems*. Springer, 2014, Kap. 13, S. 188–198

[Kwiatkowska et al. 2002] KWIATKOWSKA, Marta ; NORMAN, Gethin ; PARKER, David: PRISM: Probabilistic Symbolic Model Checker. In: *Computer Performance Evaluation/TOOLS* 2324 (2002), S. 200–204

[Langkafel 2015] LANGKAFEL, Peter: Informationstechnologie und Risikomanagement. In: (Euteneier, 2015), Kap. 14, S. 163–171

[Laroussinie et al. 2001] LAROUSSINIE, Francois ; MARKEY, Nicolas ; SCHNOEBELEN, Philippe: Model Checking CLT^+ and FCTL Is Hard. In: *International Conference on Foundations of Software Science and Computation Structures*. Genova, Italy : Springer, 2001, S. 318–331

[Larsen et al. 1995a] LARSEN, Kim G. ; PETTERSSON, Paul ; YI, Wang: Diagnostic Model-Checking for Real-Time Systems. In: *Proc. of Workshop on Verification and Control of Hybrid Systems III*, Springer–Verlag, 1995 (Lecture Notes in Computer Science 1066), S. 575–586

[Larsen et al. 1995b] LARSEN, Kim G. ; PETTERSSON, Paul ; YI, Wang: Model-Checking for Real-Time Systems. In: *Proc. of Fundamentals of Computation Theory*, 1995 (Lecture Notes in Computer Science 965), S. 62–88

[Larsen et al. 1997] LARSEN, Kim G. ; PETTERSSON, Paul ; YI, Wang: UPPAAL in a Nutshell. In: *International Journal on Software Tools for Technology Transfer* 1 (1997), Nr. 1-2, S. 134–152

[Larsson et al. 1997] LARSSON, Fredrik ; LARSEN, Kim G. ; PETTERSSON, Paul ; YI, Wang: Efficient Verification of Real-Time Systems: Compact Data Structures and State-Space Reduction. In: *Proceedings of the 18th IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, 1997, S. 14–24

[Lee et al. 2012] LEE, Insup ; SOKOLSKY, Oleg ; CHEN, Sanjian ; HATCLIFF, John ; JEE, Eunkyoung ; KIM, BaekGyu ; KING, Andrew ; MULLEN, Margaret ; PARK, Soojin ; ROEDERER, Alexander ; VENKATASUBRAMANIAN, Krishna K.: Challenges and Research Directions in Medical Cyber-Physical Systems. In: *Proceedings of the IEEE* 100 (2012), S. 75–90

[Liehn et al. 2011] LIEHN, M. ; BRUNKEN, M. ; WEISSFLOG, M. ; GUDAT, A.: Neurochirurgisches Basiswissen im Operationssaal. In: LIEHN, Margret (Hrsg.) ; STEINMÜLLER, Lutz (Hrsg.) ; DÖHLER, R. (Hrsg.): *OP-Handbuch. Grundlagen, Instrumentarium, OP-Ablauf*. 5. Springer-Verlag Berlin Heidelberg, 2011, Kap. 10.2, S. 507–515

[Liehn et al. 2014a] LIEHN, Margret ; NIETZ, Annegret ; GUDAT, Annett ; FACHINGER, Conny ; KAJDACSY, Alexander von ; HORN, Nadin ; FISCHBACH, Roman: Medizinisch-Technische Geräte. In: (Liehn et al., 2014b), Kap. 2, S. 39–74

[Liehn et al. 2014b] LIEHN, Margret (Hrsg.) ; RICHTER, Heike (Hrsg.) ; KASAKOV, Leonid (Hrsg.): *OTA-Lehrbuch. Ausbildung zur Operationstechnischen Assistenz*. 1. Springer-Verlag Berlin Heidelberg, 2014

[Lind-Nielsen et al. 2001] LIND-NIELSEN, Jørn ; ANDERSEN, Henrik R. ; HULGAARD, Henrik ; BEHRMANN, Gerd ; KRISTOFFERSEN, Kåre ; LARSEN, Kim G.: Verification of Large State/Event Systems Using Compositionality and Dependency Analysis. In: *Formal Methods in System Design* 18 (2001), Nr. 1, S. 5–23

[Lindahl et al. 1998] LINDAHL, Magnus ; PETTERSSON, Paul ; YI, Wang: Formal Design and Analysis of a Gear-Box Controller. In: *Proc. of the 4*th *Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, Springer–Verlag, 1998 (Lecture Notes in Computer Science 1384), S. 281–297

[Lüth et al. 2001] LÜTH, T ; BIER, J ; BIER, A ; HEIN, A: Verfahren und Gerätesystem zum Materialabtrag oder zur Materialbearbeitung. In: *Patent DE* 101 (2001), Nr. 17, S. 403

[Lüth et al. 2016] LÜTH, Tim ; LÜDDEMANN, Tobias ; PFEIFFER, Jonas ; DINGLER, Max ; DIETZ, Christian: *OR.NET Abschlussbericht*. 2016

[Luz et al. 2014] LUZ, M ; MANZEY, D ; MUELLER, S ; DIETZ, A ; MEIXENSBERGER, J ; STRAUSS, G: Impact of Navigated-Control Assistance on Performance, Workload and Situation Awareness of Experienced Surgeons Performing a Simulated Mastoidectomy. In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 10 (2014), Nr. 2, S. 187–195

[McMillan 1993] MCMILLAN, Kenneth L.: Symbolic Model Checking. In: *Symbolic Model Checking*. Springer, 1993, S. 25–60

[MDD] : *Richtlinie 93/42/EWG des Rates über Medizinprodukte vom 14. Juni 1993 (ABI. EG Nr L 169 S. 1) zuletzt geändert durch Artikel 2 der Richtlinie 2007/47 vom 5. September 2007 (Ab. L 247, S. 21) in Kraft getreten am 11. Oktober 2007*

[MDR] : *Verordnung (EU) 2017/745 des europäischen Parlaments und des Rates vom 05. April 2017 über Medizinprodukte, zur Änderung der Richtlinie 2001/83/EG, der Verordnung (EG) Nr. 178/2002 und der Verordnung (EG) Nr. 1223/2009 und zur Aufhebung der Richtlinien 90/384/EWG und 93/42/EWG des Rates*

[Mealy 1955] MEALY, George H.: A Method for Synthesizing Sequential Circuits. In: *Bell Labs Technical Journal* 34 (1955), Nr. 5, S. 1045–1079

[Mildenberger et al. 2002] MILDENBERGER, Peter ; EICHELBERG, Marco ; MARTIN, Eric: Introduction to the DICOM Standard. In: *European radiology* 12 (2002), Nr. 4, S. 920–927

[Mildner et al. 2015] MILDNER, Alexander ; JANSS, Armin ; DELL'ANNA-PUDLIK, Jasmin ; MERZ, Paul ; LEUCKER, Martin ; RADERMACHER, Klaus: Development of Device-and Service-Profiles for a Safe and Secure Interconnection of Medical Devices in the Integrated Open OR. In: *Risk Assessment and Risk-Driven Testing*. Springer, 2015, S. 65–74

[Milner 1980] MILNER, Robin: *A Calculus of Communicating Systems*. Springer, 1980

[Moore 1956] MOORE, Edward F.: Gedanken-Experiments on Sequential Machines. In: *Automata Studies* 34 (1956), S. 129–153

[MPBetreibV] : *Medizinprodukte-Betreiberverordnung in der Fassung der Bekanntmachung vom 21. August 2002 (BGBl. I S. 3396), die zuletzt durch Artikel 3 der Verordnung vom 11. Dezember 2014 (BGBl. I S. 2010) geändert worden ist*

[MPG] : *Medizinproduktegesetz in der Fassung der Bekanntmachung vom 7. August 2002 (BGBl. I S. 3146), das zuletzt durch Artikel 16 des Gesetzes vom 21. Juli 2014 (BGBl. I S. 1133) geändert worden ist*

[MPV] : *Medizinprodukte-Verordnung vom 20. Dezember 2001 (BGBl. I S. 3854), die zuletzt durch Artikel 2 der Verordnung vom 10. Mai 2010 (BGBl. I S. 542) geändert worden ist*

[Nedetzky und Müllner 2013] NEDETZKY, W ; MÜLLNER, P: Entwicklung von Medizinprodukten. In: (Harer, 2013), Kap. 3, S. 43–112

[Nielsen 2000] NIELSEN, Brian: *Specification and Test of Real-Time Systems*, Department of Computer Science, the Faculty of Engineering and Science, Aalborg University, Dissertation, 2000

[Pajic et al. 2014] PAJIC, Miroslav ; JIANG, Zhihao ; LEE, Insup ; SOKOLSKY, Oleg ; MANGHARAM, Rahul: Safety-critical Medical Device Development Using the UPP2SF Model Translation Tool. In: *ACM Trans. Embed. Comput. Syst.* 13 (2014), Nr. 4s, S. 127:1–127:26

[Petri 1962] PETRI, Carl A.: *Kommunikation mit Automaten*, Universität Hamburg, Dissertation, 1962

[Pettersson 1999] PETTERSSON, Paul: *Modelling and Verification of Real-Time Systems Using Timed Automata: Theory and Practice*, Uppsala University, Dissertation, 1999

[Pfeiffer et al. 2015] PFEIFFER, Jonas H. ; DINGLER, Max E. ; DIETZ, Christian ; LUETH, Tim C.: Requirements and Architecture Design for Open Real-Time Communication in the Operating Room. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* IEEE (Veranst.), 2015, S. 458–463

[Plourde et al. 2014] PLOURDE, J ; ARNEY, D ; GOLDMAN, J M.: OpenICE: An Open, Interoperable Platform for Medical Cyber-Physical Systems. In: *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*, 2014, S. 221

[Pnueli 1986] PNUELI, Amir: Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends. In: *Current Trends in Concurrency*. Springer, 1986, S. 510–584

[Pnueli und Manna 1992] PNUELI, Amir ; MANNA, Zohar: The Temporal Logic of Reactive and Concurrent Systems. In: *Springer* 16 (1992), S. 12

[Queille und Sifakis 1982] QUEILLE, J. P. ; SIFAKIS, J.: Specification and Verification of Concurrent Systems in CESAR. In: *International Symposium on Programming: 5th Colloquium Turin, April 6–8, 1982 Proceedings*, Springer Berlin Heidelberg, 1982, S. 337–351

[Ravn et al. 2010] RAVN, Anders P. ; SRBA, Jiří ; VIGHIO, Saleem: A Formal Analysis of the Web Services Atomic Transaction Protocol with UPPAAL. In: *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, Springer, 2010, S. 579–593

[Ravn et al. 2011] RAVN, Anders P. ; SRBA, Jirí ; VIGHIO, Saleem: Modelling and Verification of Web Services Business Activity Protocol. In: *TACAS* Bd. 6605, Springer, 2011, S. 357–371

[Riehm und Vogler 1996] RIEHM, Rainer ; VOGLER, Petra: Middleware: Infrastruktur für die Integration. In: *Middleware* Springer (Veranst.), 1996, S. 25–135

[Rösch et al. 2015] RÖSCH, Susanne ; ULEWICZ, Sebastian ; PROVOST, Julien ; VOGEL-HEUSER, Birgit: Review of Model-Based Testing Approaches in Production Automation and Adjacent Domains—Current Challenges and Research Gaps. In: *Journal of Software Engineering and Applications* 8 (2015), Nr. 09, S. 499

[Sadelfeld und Liehn 2014] SADELFELD, Tanja ; LIEHN, Margret: Neurochirurgie. In: LIEHN, Margret (Hrsg.) ; RICHTER, Heike (Hrsg.) ; KASAKOV, Leonid (Hrsg.): *OTA-Lehrbuch: Ausbildung zur Operationstechnischen Assistenz*, Springer Berlin Heidelberg, 2014, S. 381–389

[Sarnthein et al. 2012] SARNTHEIN, Johannes ; KRAYENBÜHL, Nicolaus ; ACTOR, Benedikt ; BOZINOV, Oliver ; BERNAYS, Richard: Intraoperatives neurophysiologisches Monitoring verbessert das Outcome in der Neurochirurgie. In: *PRAXIS* 101 (2012), Nr. 2, S. 99–105

[Schmola 2016a] SCHMOLA, Gerald: Grundlagen und Instrumente des Risikomanagements. In: (Schmola und Rapp, 2016), Kap. 11, S. 289–340

[Schmola und Rapp 2016] SCHMOLA, Gerald (Hrsg.) ; RAPP, Boris (Hrsg.): *Compliance, Governance und Risikomanagement im Krankenhaus. Rechtliche Anforderungen - Praktische Umsetzung - Nachhaltige Organisation*. 1. Wiesbaden : Springer-Gabler, 2016

[Schmola 2016b] SCHMOLA, Markus: Risk-Management im Operationsbereich. In: (Schmola und Rapp, 2016), Kap. 14, S. 407–434

[Schömig und Heinen 2014] SCHÖMIG, Edgar ; HEINEN, Peter: Risikomanagement für vernetzte medizinische Geräte. In: *KU Gesundheitsmanagement* (2014)

[Statistisches Bundesamt 2015] STATISTISCHES BUNDESAMT: *Gesundheit / Fallpauschalenbezogene Krankenhausstatistik (DRG-Statistik) / Operationen und Prozeduren der vollstationären Patientinnen und Patienten in Krankenhäusern*. 2015

[Steege et al. 2016] STEEGE, Andrea L. ; BOIANO, James M. ; SWEENEY, Marie H.: Secondhand Smoke in the Operating Room? Precautionary Practices Lacking for Surgical Smoke. In: *American Journal of Industrial Medicine* (2016)

[Stoecklein et al. 2015] STOECKLEIN, Veit M. ; FABER, Florian ; KOCH, Mandy ; MATTMÜLLER, Rudi ; SCHAPER, Anika ; RUDOLPH, Frank ; TONN, Joerg C. ; SCHICHOR, Christian: Optional Real-Time Display of Intraoperative Neurophysiological Monitoring in the Microscopic Field of View: Avoiding Communication Failures in the Operating Room. In: *Acta neurochirurgica* 157 (2015), Nr. 11, S. 1843–1847

[Strauss et al. 2013] STRAUSS, G ; GOLLNICK, I ; NEUMUTH, T ; MEIXENSBERGER, J ; LUETH, T C.: Das „Surgical Deck“: Eine neue OP-Generation für die HNO-Chirurgie. In: *Laryngo-Rhino-Otologie* 45 (2013), Nr. 02, S. 102–112

[Strauss et al. 2005] STRAUSS, G ; KOULECHOV, K ; RICHTER, R ; DIETZ, A ; TRANTAKIS, C ; LÜTH, T: Navigated Control in Functional Endoscopic Sinus Surgery. In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 1 (2005), Nr. 3, S. 31–41

[Strauss und Schmitz 2016] STRAUSS, G ; SCHMITZ, P: Standardisierung und Modellierung von Operationsprozessen. In: *Der Chirurg* 87 (2016), Nr. 12, S. 1008–1014

[Tripakis und Courcoubetis 1996] TRIPAKIS, Stavros ; COURCOUBETIS, Costas: Extending Promela and Spin for Real Time. In: *Tools and Algorithms for the Construction and Analysis of Systems* (1996), S. 329–348

[Valmari 1991] VALMARI, Antti: *A Stubborn Attack on State Explosion*. S. 156–165. In: CLARKE, Edmund M. (Hrsg.) ; KURSHAN, Robert P. (Hrsg.): *Computer-Aided Verification: 2nd International Conference, CAV '90 New Brunswick, NJ, USA, June 18–21, 1990 Proceedings*, Springer Berlin Heidelberg, 1991

[Wallwiener et al. 2011] WALLWIENER, D. ; WALLWIENER, M. ; KRÄMER, B. ; ABELE, H. ; ROTHMUND, R. ; BECKER, S. ; ZUBKE, W. ; BRUCKER, S.: Integrierte OP-Systeme (IOPS) als Basis für innovative Operationsverfahren in der Gynäkologie. In: *Der Gynäkologe* 44 (2011), Nr. 3, S. 187–195

[Wardana 2009] WARDANA, Awang Noor I.: *Development of Automatic Program Verification for Continuous Function Chart based on Model Checking*. Bd. 3. kassel university press GmbH, 2009

[Weißmann et al. 2011] WEISSMANN, Markus ; BEDENK, Stefan ; BUCKL, Christian ; KNOLL, Alois: *Model Checking Industrial Robot Systems*. S. 161–176. In: GROCE, Alex (Hrsg.) ; MUSUVATHI, Madanlal (Hrsg.): *Model Checking Software: 18th International SPIN Workshop, Snowbird, UT, USA, July 14-15, 2011. Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011

[Weimer 2010] WEIMER, Tobias: Medizinproduktehaftung der Anwender und Betreiber. In: *intensiv* 18 (2010), Nr. 02, S. 60–61

[Witsch und Vogel-Heuser 2011] WITSCH, Daniel ; VOGEL-HEUSER, Birgit: PLC-statecharts: An approach to integrate UML-statecharts in open-loop control engineering–aspects on behavioral semantics and model-checking. In: *IFAC Proceedings Volumes* 44 (2011), Nr. 1, S. 7866–7872

[Woskowski 2014] WOSKOWSKI, Christoph: A Pragmatic Approach Towards Safe and Secure Medical Device Integration. In: BONDAVALLI, Andrea (Hrsg.) ; DI GIANDOMENICO, Felicita (Hrsg.): *Proceedings of the 33rd International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*. Florence, Italy : Springer, 2014, S. 342–353

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Beispielhafte Anforderungsspezifikation

```
/*
Der Zustand ACT ist erreichbar
*/
E<>EndomatLC_Instance.ACT

/*
Das Gerät EndomatLC wird irgendwann gepollt
*/
A<>EndomatLC_EPL_Instance.Polled

/*
Das Gerät FOOT wird irgendwann gepollt
*/
A<>FOOT_EPL_Instance.Polled

/*
Der Zustand ACT ist erreichbar
*/
E<>Sonoca300_Instance.ACT

/*
Das Gerät Sonoca300 wird irgendwann gepollt
*/
A<>Sonoca300_EPL_Instance.Polled
```

Abbildung 60 Anforderungskatalog für das Anwendungsbeispiel aus Abschnitt 6.2. Die umgangssprachlichen Beschreibungen der Anforderungen sind jeweils in Kommentar-Umgebungen eingefasst. Diese werden durch die Zeichensequenz /* eingeläutet und durch */ abgeschlossen (siehe Abschnitt 5.4).

Beispielhaftes Versuchsprotokoll

```
File generated automatically by MiMed OR.Net Verification Toolbox.
This is the Verification Result for System "system_spec.xml".
The Requirements Specification Filename is "system_spec.q".
Verification Date: 2017/8/1 @ 19:42.

Verification Result:
Options for the verification:
Generating no trace
Search order is depth first
Using aggressive space optimisation
Seed is 1501609370
State space representation uses minimal constraint systems

Verifying formula 1 at line 2
 -- Formula is satisfied.
 -- States stored : 181069 states
 -- States explored : 1320 states
 -- CPU user time used : 5297 ms
 -- Virtual memory used : 81320 KiB
 -- Resident memory used : 38640 KiB

Verifying formula 2 at line 4
 -- Formula is satisfied.
 -- States explored : 1 states
 -- CPU user time used : 47 ms
 -- Virtual memory used : 28552 KiB
 -- Resident memory used : 8232 KiB

Verifying formula 3 at line 5
 -- Formula is satisfied.
 -- States explored : 6 states
 -- CPU user time used : 0 ms
 -- Virtual memory used : 28544 KiB
 -- Resident memory used : 8240 KiB

Verifying formula 4 at line 6
 -- Formula is satisfied.
 -- States stored : 186065 states
 -- States explored : 1344 states
 -- CPU user time used : 5265 ms
 -- Virtual memory used : 82924 KiB
 -- Resident memory used : 39380 KiB

Verifying formula 5 at line 8
 -- Formula is satisfied.
 -- States explored : 22 states
 -- CPU user time used : 63 ms
 -- Virtual memory used : 28888 KiB
 -- Resident memory used : 8420 KiB
```

Abbildung 61 Automatisiert erzeugtes Versuchsprotokoll für Anwendungsfall 2 in Abschnitt 6.2. Das Protokoll wird auf Basis der Ausgabe des UPPAAL Verifiers angefertigt. Zu Beginn werden Versuchsdatum, Systemspezifikation und Anforderungsspezifikation genannt. Anschließend werden Spezifika über den Verifikationsalgorithmus (z.B. Tiefensuche) aufgeführt und zuletzt das Ergebnis der Verifikation mit Rechenaufwand genannt.

Beschreibungsdatei des Fußschalters

(Class:Device//Manufacturer:SteuteGmbH//ModelName:FOOT// ResponseTime:5//udi:101.122.001)

(Class:Variable//Handle:discrete_level_left//DefVal:0//Frequency:23//MetricCategory:manual_get//Resolution:1//LowerRange:0//UpperRange:2)

(Class:Variable//Handle:analog_level_left//DefVal:0//Frequency:23//MetricCategory:manual_get//Resolution:1//LowerRange:0//UpperRange:100)

(Class:Variable//Handle:pedal_val_discrete_left//DefVal:0//LatencyAcceptance:50//Frequency:223//MetricCategory:epl_set//Resolution:1//LowerRange:0//UpperRange:2)

(Class:Variable//Handle:pedal_val_analog_left//DefVal:0//LatencyAcceptance:50//Frequency:223//MetricCategory:epl_set//Resolution:1//LowerRange:0//UpperRange:100)

(Class:Variable//Handle:discrete_level_right//DefVal:0//Frequency:23//MetricCategory:manual_get//Resolution:1//LowerRange:0//UpperRange:2)

(Class:Variable//Handle:analog_level_right//DefVal:0//Frequency:23//MetricCategory:manual_get//Resolution:1//LowerRange:0//UpperRange:100)

(Class:Variable//Handle:pedal_val_discrete_right//DefVal:0//LatencyAcceptance:50//Frequency:223//MetricCategory:epl_set//Resolution:1//LowerRange:0//UpperRange:2)

(Class:Variable//Handle:pedal_val_analog_right//DefVal:0//LatencyAcceptance:50//Frequency:223//MetricCategory:epl_set//Resolution:1//LowerRange:0//UpperRange:100)

(Class:Location//Handle:GET//InitLocation:True)

(Class:Transition//Source:GET//Destination:GET//Sync:soc?//Update:cc2m_FOOT_pedal_val_discrete_left=discrete_level_left,cc2m_FOOT_pedal_val_discrete_right=discrete_level_right,cc2m_FOOT_pedal_val_analog_left=analog_level_left,cc2m_FOOT_pedal_val_analog_right=analog_level_right)

Abbildung 62 Beschreibungsdatei des Fußschalters.

Beschreibungsdatei der Saug-/Spülpumpe– Hauptprozess

```
(Class:Device//Manufacturer:KarlStorzGmbH//ModelName:EndomatLC//ModelNumb
er:20012630//ResponseTime:5//udi:101.122.001)

(Class:Variable//Handle:flowrate_hi//DefVal:0//Frequency:23//MetricCatego
ry:internal_pass//Resolution:1//LowerRange:500//UpperRange:1000)

(Class:Variable//Handle:flowrate_lo//DefVal:500//Frequency:223//MetricCat
egory:internal_pass//Resolution:1//LowerRange:0//UpperRange:1000)

(Class:Variable//Handle:pedal_val//DefVal:0//LatencyAcceptance:50//Freque
ncy:223//MetricCategory:epl_get//Resolution:1//LowerRange:0//UpperRange:2
)

(Class:Variable//Handle:actual_flow_rate//DefVal:0//LatencyAcceptance:50/
/Frequency:223//MetricCategory:observable//Resolution:1//LowerRange:0//Up
perRange:1000)

(Class:Location//Handle:ACT//Urgent:True//Reachability:Possible)
(Class:Location//Handle:OFF_LOC//InitLocation:True)

(Class:Location//Handle:ON_LOC)
(Class:Location//Handle:READY)

(Class:Signal//Category:GET//Handle:START//Frequency:1)

(Class:Signal//Category:GET//Handle:ON//Frequency:1)

(Class:Signal//Category:GET//Handle:OFF//Frequency:1)

(Class:Transition//Source:OFF_LOC//Destination:ON_LOC//Sync:ON?)

(Class:Transition//Source:ON_LOC//Destination:READY//Sync:START?)

(Class:Transition//Source:READY//Destination:ACT//Sync:soc?//Update:cm2c_
EndomatLC_pedal_val=mm2c_EndomatLC_pedal_val)

(Class:Transition//Source:ACT//Destination:READY//Guard:cm2c_EndomatLC_pe
dal_val[1]==0//Update:actual_flow_rate=def_EndomatLC_pedal_val)

(Class:Transition//Source:ACT//Destination:OFF_LOC//Sync:OFF?//Update:act
ual_flow_rate=def_EndomatLC_pedal_val)

(Class:Transition//Source:READY//Destination:OFF_LOC//Sync:OFF?//Update:a
ctual_flow_rate=def_EndomatLC_pedal_val)

(Class:Transition//Source:ON_LOC//Destination:OFF_LOC//Sync:OFF?//Update:
actual_flow_rate=def_EndomatLC_pedal_val)

(Class:Transition//Source:ACT//Destination:READY//Guard:cm2c_EndomatLC_pe
dal_val[1]==1 and
cm2c_EndomatLC_pedal_val[0]==1//Update:actual_flow_rate=flowrate_lo)

(Class:Transition//Source:ACT//Destination:READY//Guard:cm2c_EndomatLC_pe
dal_val[1]==1 and
cm2c_EndomatLC_pedal_val[0]==2//Update:actual_flow_rate=flowrate_hi)
```

Abbildung 63 Beschreibungsdatei der Saug-/Spülpumpe– Hauptprozess.

Beschreibungsdatei der Saug-/Spülpumpe– Setupprozess

(Class:Device//Manufacturer:KarlStorzGmbH//ModelName:EndomatLC_FlowrateMachine//ResponseTime:5//udi:101.111.001)

(Class:Variable//Handle:flowrate_hi//DefVal:0//Frequency:23//MetricCategory:internal_get//Resolution:1//LowerRange:500//UpperRange:1000)

(Class:Variable//Handle:flowrate_lo//DefVal:500//Frequency:223//MetricCategory:internal_get//Resolution:1//LowerRange:0//UpperRange:1000)

(Class:Location//Handle:OFF_LOC//InitLocation:True)

(Class:Location//Handle:HI_MODE)

(Class:Location//Handle:LO_MODE)

(Class:Signal//Category:GET//Handle:UP//Frequency:1)

(Class:Signal//Category:GET//Handle:DOWN//Frequency:1)

(Class:Signal//Category:PASS//Handle:ON//Frequency:1)

(Class:Signal//Category:PASS//Handle:OFF//Frequency:1)

(Class:Signal//Category:GET//Handle:SWITCH//Frequency:1)

(Class:Transition//Source:OFF_LOC//Destination:LO_MODE//Sync:ON?)

(Class:Transition//Source:LO_MODE//Destination:HI_MODE//Sync:SWITCH?)

(Class:Transition//Source:HI_MODE//Destination:LO_MODE//Sync:SWITCH?)

(Class:Transition//Source:LO_MODE//Destination:LO_MODE//Sync:UP?//Update:flowrate_lo=flowrate_lo+50//Guard:flowrate_lo<1000)

(Class:Transition//Source:LO_MODE//Destination:LO_MODE//Sync:UP?//Guard:flowrate_lo==1000)

(Class:Transition//Source:LO_MODE//Destination:LO_MODE//Sync:DOWN?//Update:flowrate_lo=flowrate_lo-50//Guard:flowrate_lo>0)

(Class:Transition//Source:LO_MODE//Destination:LO_MODE//Sync:DOWN?//Guard:flowrate_lo==0)

(Class:Transition//Source:HI_MODE//Destination:HI_MODE//Sync:UP?//Update:flowrate_hi=flowrate_hi+50//Guard:flowrate_hi<1000)

(Class:Transition//Source:HI_MODE//Destination:HI_MODE//Sync:UP?//Guard:flowrate_hi==1000)

(Class:Transition//Source:HI_MODE//Destination:HI_MODE//Sync:DOWN?//Update:flowrate_hi=flowrate_hi-50//Guard:flowrate_hi>500)

(Class:Transition//Source:HI_MODE//Destination:HI_MODE//Sync:DOWN?//Guard:flowrate_hi==500)

(Class:Transition//Source:LO_MODE//Destination:OFF_LOC//Sync:OFF?)

(Class:Transition//Source:HI_MODE//Destination:OFF_LOC//Sync:OFF?)

Abbildung 64 Beschreibungsdatei der Saug-/Spülpumpe– Setupprozess.

Beschreibungsdatei des Ultraschalldissektors – Hauptprozess

(Class:Device//Manufacturer:SoeringGmbH//ModelName:Sonoca300//ResponseTime:5)

(Class:Variable//Handle:pedal_val//InitVal:0//DefVal:0//Frequency:223//LatencyAcceptance:50//MetricCategory:epl_get//Resolution:1//LowerRange:0//UpperRange:2)

(Class:Variable//Handle:action//InitVal:0//DefVal:0//Frequency:223//LatencyAcceptance:50//MetricCategory:observable//Resolution:1//LowerRange:0//UpperRange:100)

(Class:Variable//Handle:power//Frequency:1//InitVal:0//DefVal:0//MetricCategory:internal_pass//Resolution:4//LowerRange:0//UpperRange:100)

(Class:Location//Handle:OFF_LOC//InitLocation:True)

(Class:Location//Handle:GET)

(Class:Location//Handle:ACT//Urgent:True//Reachability:Possible)

(Class:Signal//Category:GET//Handle:ONN//Frequency:1)

(Class:Signal//Category:GET//Handle:OFFF//Frequency:1)

(Class:Transition//Source:GET//Destination:ACT//Sync:soc?//Update:cm2c_Sonoca300_pedal_val=mm2c_Sonoca300_pedal_val)

(Class:Transition//Source:ACT//Destination:GET//Guard:mm2c_Sonoca300_pedal_val[1]==1 and mm2c_Sonoca300_pedal_val[0]>0//Update:action=power)

(Class:Transition//Source:ACT//Destination:GET//Guard:mm2c_Sonoca300_pedal_val[1]==0//Update:action=def_Sonoca300_pedal_val)

(Class:Transition//Source:OFF_LOC//Destination:GET//Sync:ONN?)

(Class:Transition//Source:GET//Destination:OFF_LOC//Sync:OFFF?)

(Class:Transition//Source:ACT//Destination:OFF_LOC//Sync:OFFF?)

Abbildung 65 Beschreibungsdatei des US-Dissektors– Hauptprozess. Die Signalnamen OFFF und ONNN sind bewusst so geschrieben, um Namenskollisionen mit den Signalen der Saug-/Spülpumpe zu vermeiden.

Beschreibungsdatei des Ultraschalldissektors – Setupprozess

(Class:Device//Manufacturer:SoeringGmbH//ModelName:Sonoca300_PowerMachine//ResponseTime:5)

(Class:Variable//Handle:power//Frequency:1//InitVal:0//DefVal:0//MetricCategory:internal_get//Resolution:4//LowerRange:0//UpperRange:100)

(Class:Location//Handle:DEFAULT)

(Class:Location//Handle:OFF//InitLocation:True)

(Class:Signal//Handle:UPP//Category:GET//Frequency:1)

(Class:Signal//Handle:DOWWN//Category:GET//Frequency:1)

(Class:Signal//Handle:ONN//Category:PASS//Frequency:1)

(Class:Signal//Handle:OFFF//Category:PASS//Frequency:1)

(Class:Signal//Handle:TWENTY//Category:GET//Frequency:1)

(Class:Signal//Handle:FOURTY//Category:GET//Frequency:1)

(Class:Signal//Handle:SIXTY//Category:GET//Frequency:1)

(Class:Signal//Handle:EIGHTY//Category:GET//Frequency:1)

(Class:Transition//Source:OFF//Destination:DEFAULT//Sync:ONN?)

(Class:Transition//Source:DEFAULT//Destination:OFF//Sync:OFFF?)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:UPP?//Update:power=power+4//Guard:power<100)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:UPP?//Guard:power==100)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:DOWWN?//Update:power=power-4//Guard:power>0)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:DOWWN?//Guard:power==0)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:TWENTY?//Update:power=20)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:FOURTY?//Update:power=40)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:SIXTY?//Update:power=60)

(Class:Transition//Source:DEFAULT//Destination:DEFAULT//Sync:EIGHTY?//Update:power=80)

Abbildung 66 Beschreibungsdatei des Ultraschalldissektors – Setupprozess. Die Signalnamen UPP und DOWWN sind bewusst so geschrieben, um Namenskollisionen mit den Signalen des US-Dissektors zu vermeiden.