Bundesministerium
Innovation, Mobilität
und Infrastruktur

Open Building Automation Modelling - Offene Modellierung der Gebäudeautomation über den gesamten Gebäudelebenszyklus (openBAM)

Berichte aus Energie- und Umweltforschung 87/2025

Wien, 2025

Impressum

Medieninhaber, Verleger und Herausgeber:
Bundesministerium für Innovation, Mobilität und Infrastruktur,
Radetzkystraße 2, 1030 Wien
Verantwortung und Koordination: Abteilung III/3 - Energie und Umwelttechnologien
Leitung: DI (FH) Volker Schaffler, MA, AKKM
Autorinnen und Autoren:
Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Bednar, Dipl.-Ing. Sabine Sint, Noah Fritscher
Forschungsbereich Bauphysik, TU Wien
Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner, Dipl.-Ing. Felix Knorr, Dipl.-Ing. Jürgen Pannosch, DDipl.-Ing. Dr.techn. Gernot Steindl, Forschungsbereich Automation Systems, TU Wien
Ing. Jürgen Kromp, Zsombor Jarosi SAUTER Meß- und Regeltechnik Ges.m.b.H
Wien, 2025. Stand: September 2024

Ein Projektbericht gefördert im Rahmen von

Rückmeldungen:
Ihre Überlegungen zu vorliegender Publikation übermitteln Sie bitte an iii3@bmimi.gv.at.

Rechtlicher Hinweis

Dieser Ergebnisbericht wurde von die/der Projektnehmer:in erstellt. Für die Richtigkeit, Vollständigkeit, Aktualität sowie die barrierefreie Gestaltung der Inhalte übernimmt das Bundesministerium für Innovation, Mobilität und Infrastruktur (BMIMI) keine Haftung.

Mit der Übermittlung der Projektbeschreibung bestätigt die/der Projektnehmer:in ausdrücklich, über sämtliche für die Nutzung erforderlichen Rechte – insbesondere Urheberrechte, Leistungsschutzrechte sowie etwaige Persönlichkeitsrechte abgebildeter Personen – am bereitgestellten Bildmaterial zu verfügen.

Die/der Projektnehmer:in räumt dem BMIMI ein unentgeltliches, nicht ausschließliches, zeitlich und örtlich unbeschränktes sowie unwiderrufliches Nutzungsrecht ein, das übermittelte Bildmaterial in allen derzeit bekannten sowie künftig bekannt werdenden Nutzungsarten für Zwecke der Berichterstattung, Dokumentation und Öffentlichkeitsarbeit im Zusammenhang mit der geförderten Maßnahme zu verwenden, insbesondere zur Veröffentlichung in Printmedien, digitalen Medien, Präsentationen und sozialen Netzwerken.

Für den Fall, dass Dritte Ansprüche wegen einer Verletzung von Rechten am übermittelten Bildmaterial gegen das BMIMI geltend machen, verpflichtet sich die/der Projektnehmer:in, das BMIMI vollständig schad- und klaglos zu halten. Dies umfasst insbesondere auch die Kosten einer angemessenen rechtlichen Vertretung sowie etwaige gerichtliche und außergerichtliche Aufwendungen.

Vorbemerkung

Der vorliegende Bericht dokumentiert die Ergebnisse eines Projekts aus dem Forschungs- und Technologieprogramm „Stadt der Zukunft“ des Bundesministeriums für Innovation, Mobilität und Infrastruktur (BMIMI). Dieses Programm baut auf dem langjährigen Programm „Haus der Zukunft“ auf und hat die Intention, Konzepte, Technologien und Lösungen für zukünftige Städte und Stadtquartiere zu entwickeln und bei der Umsetzung zu unterstützen. Damit soll eine Entwicklung in Richtung energieeffiziente und klimaverträgliche Stadt unterstützt werden, die auch dazu beiträgt, die Lebensqualität und die wirtschaftliche Standortattraktivität zu erhöhen. Eine integrierte Planung wie auch die Berücksichtigung aller betroffener Bereiche wie Energieerzeugung und -verteilung, gebaute Infrastruktur, Mobilität und Kommunikation sind dabei Voraussetzung.

Um die Wirkung des Programms zu erhöhen, sind die Sichtbarkeit und leichte Verfügbarkeit der innovativen Ergebnisse ein wichtiges Anliegen. Daher werden nach dem Open Access Prinzip möglichst alle Projektergebnisse des Programms in der Schriftenreihe des BMIMI publiziert und elektronisch über die Plattform www.NachhaltigWirtschaften.at zugänglich gemacht. In diesem Sinne wünschen wir allen Interessierten und Anwender:innen eine interessante Lektüre.

Inhaltsverzeichnis

1 Kurzfassung

Ausgangssituation/Motivation

Hocheffiziente Gebäude, also Gebäude deren gesamter Energieverbrauch (Gebäudebetrieb und Nutzung) durch lokale Energiegewinnung abgedeckt werden kann, sind wesentliche Elemente von klimaneutralen Städten bzw. Stadtteilen. Diese Plus-Plus-Energie-Gebäude wurden bereits in unterschiedlichen Ausprägungen in der Praxis umgesetzt, jedoch zeigte sich bei vielen Projekten, dass deren tatsächlicher Energieverbrauch höher ist als der ursprünglich geplante Verbrauch. Einer der Gründe dafür lässt sich darauf zurückführen, dass die Gebäudeautomation erst im Zuge der Inbetriebnahme der Gebäude detailliert geplant wird. Deswegen und mangels Zeit und Budget wird in den seltensten Fällen das volle Energieeinsparungspotenzial, das die Gebäudeautomation erschließen könnte, ausgeschöpft und die Gebäude haben somit einen höheren Energieverbrauch als notwendig.

Inhalte und Zielsetzungen

Da jede Kilowattstunde nicht verbrauchte Energie dazu beiträgt, das Ziel „Klimaneutralität“ zu erreichen, braucht es Gebäude mit sauber durchdachter, funktionierender Gebäudeautomation, die einen optimalen Gebäudebetrieb sicherstellt, da ansonsten es bei hocheffizienten Gebäuden zu einem unnötigen Mehrverbrauch von bis zu 54% kommen kann. Um die Gebäudeautomation – insbesondere die Steuerungs- und Regelungslogik – schon vorab auf einen optimalen Betrieb auslegen zu können, ist es notwendig, dass sie anhand eines digitalen, simulationsfähigen Abbildes (digitalen Zwillings) entwickelt wird. Dabei müssen alle relevanten Aspekte des Gebäudes (Steuerungs- und Regelungslogik, Nutzungsinformationen, Geometrie, Bautechnik und Gebäudetechnik) in unterschiedlichen Detaillierungsgraden abbildbar sein – von stark vereinfacht bis sehr detailliert – um eine zielführende Planung sicherzustellen.

Methodische Vorgehensweise

Im Zuge des Projektes wird eine Methode zur plattformunabhängigen Modellierung der Steuerungs- und Regelungslogik erstellt und in ein offenes Datenmodell integriert. Damit kann die Steuerungs- und Regelungslogik mit den Komponenten und Parametern der Gebäudemodelle bereits in der Planungsphase und über den gesamten Gebäudelebenszyklus verbunden werden. Für einen Proof-of-Concept wird das IT-Ökosystem SIMULTAN verwendet, dass eine Plattform für existierende offenen Datenmodelle wie z.B. IFC (ISO 16739) und BACNET (ISO 16484) darstellt.

Darauf aufbauend werden methodisch Schnittstellen entwickelt um die plattformunabhängige modellierte Steuerungs- und Regelungslogik in plattformspezifisch Lösungen (z.B. Simulationstools, Engineering Tools) überzuführen. Es wird dadurch ermöglicht, dass Planungsideen und Erkenntnisse aus den Simulationen direkt in die Ausführungslogik übernommen werden.

Die Praxistauglichkeit der Methodik wird durch Anwendung in einem Laborszenario nachgewiesen.

Ergebnisse und Schlussfolgerungen

Im Projekt wurde die Planung der Gebäudeautomation durch eine generische, technologieunabhängige Methodik umgesetzt. Der openBAM Editors ermöglicht die Modellierung der Steuerungslogik mithilfe von UML-Zustandsgraphen. Um die Gebäudeautomation über den gesamten Gebäudelebenszyklus zu unterstützen, wurde die Logikabbildung mit dem digitalen Gebäudemodell, welches in einem offenen Datenmodell gespeichert ist, über konsistent gehaltene Datenpunkte verknüpft. Mittels Modelltransformationen werden die Informationen weiterverarbeitet um diese in Simulations- und Engineering Tools zu nutzen. Anhand eines Laborversuchsaufbaus einer Lüftungsanlage wurde gezeigt, dass eine Modellierung beginnend in der digitalen Welt von Simultan, einschließlich der Nutzung einer Simulation, und endend in der Steuerung von SAUTER möglich ist. Die konsistente Modellierung der Regelungslogik wird dabei garantiert. Dadurch ist möglich, dass die Regelungslogik bereits in der frühen Designphase unabhängig vom Zielsystem (SAUTER) entwickelt und durch Simulationen (z. B. IDA ICE) optimiert wird, um eine Grundlage für den späteren Energieverbrauch zu schaffen. Der optimierte Entwurf kann anschließend ohne Informationsverlust in der Ausführungsphase (SAUTER Engineering) umgesetzt werden.

Ausblick

Im Rahmen des Forschungsprojekts wurden Potenziale für zukünftige Entwicklungen identifiziert. Ein wesentlicher Punkt ist, dass bei Änderungen oder Löschungen von Datenpunkten, die in der Logikmodellierung verwendet werden, eine neue Modellierung oder manuelle Anpassungen erforderlich sind. Zukünftig sollten Benachrichtigungsmechanismen und die Nutzung von Repositorien oder Versionskontrollen entwickelt werden, um Änderungen zu verfolgen und Nutzer:innen darauf aufmerksam zu machen. Darüber hinaus wurde festgestellt, dass die Modellierung von komplexer Regelungslogik herausfordernd sein kann, da viele parallele Prozesse involviert sind und die Zustandsgraphen dazu sehr schnell sehr groß werden können. In diesem Bereich ist unter anderem eine Optimierung der Anzeige notwendig.

2 Abstract

Starting point / motivation

Efficient buildings, i.e., buildings where entire energy consumption (building operation and use) can be covered by local energy generation, are essential elements of climate-neutral cities or urban districts. These plus-plus-energy buildings have already been implemented in practice in various forms. Still, many projects have shown that their actual energy consumption is higher than the initially planned consumption. One of the reasons for this can be traced back to the fact that building automation is not planned in detail until the buildings are commissioned. Due to this reason, and due to lack of time and budget, the entire energy savings potential that building automation could provide is rarely realized. Thus, buildings consume more energy than necessary.

Contents and goals

Since every kilowatt-hour of energy not consumed contributes to achieving the goal of “climate neutrality”, buildings with well-designed and perfectly operating building automation systems are needed to ensure optimal building performance. Otherwise, highly efficient buildings can result in unnecessary additional energy consumption of up to 54%. To design the building automation - particularly the control and regulation logic - for optimal operation in advance, it must be developed based on a digital, simulation-capable representation (digital twin). All relevant aspects of the building (control logic, utilization information, geometry, structural engineering, and building services engineering) should be mapped in various degrees of detail - from simplified to very detailed - to ensure target-oriented planning.

Methods

In the project, a method for platform-independent modeling of the control and regulation logic will be created and integrated into an open data model. This allows the control logic to be linked to the components and parameters of building models already in the planning phase and over the entire building life cycle. The IT ecosystem SIMULTAN is used as a proof-of-concept, providing a platform for open data models such as IFC (ISO 16739) and BACNET (ISO 16484).

Based on this, interfaces are methodically developed to transfer the platform-independent modeled control logic into platform-specific solutions (e.g., simulation tools, engineering tools). The practical suitability of the methodology is demonstrated by application in a laboratory scenario.

Results and conclusions

In the project, the planning of the building automation was implemented using a generic, technology-independent methodology. The openBAM Editor enables the modeling of control logic using UML state graphs. To support building automation over the entire building life cycle, the logic mapping was linked to the digital building model, which is stored in an open data model via consistent data points. Model transformations are used to process the information further in simulation and engineering tools. A laboratory test setup of a ventilation system was used to show that modeling is possible, starting in the digital world of Simultan, including the use of a simulation, and ending in the SAUTER control system. Consistent modeling of the control logic is guaranteed.

This allows the control logic to be developed independently of the target system (SAUTER) in the early design phase and optimized using simulations (IDA ICE) to create a basis for subsequent energy consumption. The optimized design can then be implemented in the execution phase (SAUTER Engineering) without any loss of information.

Outlook

The research project identified potential for future developments. A key point is that new modeling or manual adjustments are required when data points used in logic modeling are changed or deleted. In the future, notification mechanisms and repositories or version controls should be established to track changes and inform users of them. In addition, it was found that modeling complex control logic can be challenging, as many parallel processes are involved, and the state graphs can quickly become very large. In this area, among other things, it is necessary to optimize the display.

3 Ausgangslage

3.1. Forschungsfrage und Zielsetzungen

Um das Ziel von klimaneutralen Städten bzw. Stadtteilen zu erreichen, muss ein effizienter Betrieb von Gebäuden ermöglicht werden, wo deren gesamter Energieverbrauch (Nutzung und Gebäudebetrieb) durch lokale Energiegewinnung abgedeckt wird. Diese Plus-Plus-Energie-Gebäude wurden bereits in unterschiedlichen Ausprägungen in der Praxis umgesetzt, jedoch zeigte sich bei vielen Projekten, dass deren tatsächlicher Energieverbrauch höher ist als der ursprünglich geplante Verbrauch. Einer der Gründe ist, dass die Gebäudeautomation erst im Zuge der Inbetriebnahme der Gebäude detailliert geplant wird. Dadurch und wegen mangelnder Zeit und Budget wird in den seltensten Fällen das volle Energieeinsparungspotenzial ausgeschöpft, das die Gebäudeautomation erschließen könnte. Es haben die Gebäude somit einen höheren Energieverbrauch als notwendig. Vergleiche mit dem Plus-Energie-Bürohochhaus der TU Wien [1] in Österreich und dem Elithis Tower [2] in Frankreich zeigen, dass ein Mehrverbrauch bis zu 54% entstehen kann.

Um solche Mehrverbräuche zu verhindern, muss ermöglicht werden, dass schon in der Planungsphase von Gebäuden die Gebäudeautomation modelliert und getestet werden kann. Eine Herausforderung dabei stellt dar, dass je nach Projektphase unterschiedliche Anforderungen an den Detailgrad der Steuerungs- und Regelungstechnik gestellt werden. Die momentan übliche Vorgehensweise in Planungsprojekten ist, dass zuerst eine semantische Beschreibung für die Mess-, Steuer- und Regeltechnik (MSR) erstellt wird, gefolgt von Tests in Planungstools in der Entwurfsplanung. In der Ausführungsplanung werden dann die Ideen der Planung zur MSR in einem Engineering Tool in einer proprietären Software, die meist hardwarenah ist, neu umgesetzt. Es kommt zu einem Bruch zwischen Planung und Ausführung, da keine Überführungsmöglichkeiten gegeben sind. So können Entscheidungen aus der Planungsphase übersehen und schließlich vergessen werden umzusetzen. Nachträglich werden dann im Betrieb Fehler festgestellt und sukzessive durch Ergänzungen und Änderungen behoben. Sollten Mängel im Betrieb allerdings den Nutzer:innen nicht auffallen, kann es passieren, dass diese nie behoben werden, und die Gebäudeautomation nicht optimiert läuft und somit z.B. mehr Energieverbrauch verursacht wird.

Das Projekt bereitet die Grundlage für eine Methode zur plattformunabhängigen Modellierung der Steuerungs- und Regelungslogik und Integration in ein offenes Datenmodell. Über das Datenmodell erfolgt dann der Austausch mit den Simulations- und Engineering-Tools. Es wird dadurch ermöglicht, dass Planungsideen und Erkenntnisse aus den Simulationen direkt in die Ausführungslogik übernommen werden. Somit werden mehrfache Eingaben verhindert und Entscheidungen können nicht verloren gehen.

Im Projekt soll *(1) eine generische, technologieunabhängige Planung der Gebäudeautomation* ermöglichen und (2) *das Gebäudemodell um die MSR relevanten Informationen* erweitern. Es müssen alle relevanten Parameter und Aspekte gespeichert, gehalten und abgerufen werden können. Auf Basis des Gebäudemodells, welches nun auch die MSR-Logik enthält, können Simulationstools angesprochen werden um das Zusammenspiel des Gebäudeautomationssystems mit der Bautechnik und der Gebäudetechnik detaillierter untersuchen zu können und somit energieeffizientere, nachhaltigere Automationslösungen entwickeln zu können. Darüber hinaus wird die geplante

Information dann an Engineering Tools übergeben. Es wird also mit Hilfe von Model Driven Engineering (MDE) Ansätzen die erstellte MSR-Logik in eine plattformspezifische Implementierung übergeführt. Dadurch ist (3) *der Austausch zwischen Tools über das Datenmodell und Schnittstellen* sichergestellt.

Zur Erprobung und Entwicklung des methodischen Ansatzes werden das Datenmodell, die entwickelten Transformationsmethoden zusammen mit dem Simulationstool und dem Engineering Tool anhand eines Use-Cases „Laborversuchsaufbau“ evaluiert. Hierbei wird überprüft, ob die im offenen Datenmodell abgebildete MSR-Logik mittels der zu entwickelnden MSR-Transformationsmethoden fehlerfrei in das Simulations-Tool und das Engineering-Tool übertragen wird. Es wird verglichen, ob die Steuerungs- und Regelungslogik in der Simulation ein gleiches Verhalten widerspiegelt wie im Labor.

3.2. State of the Art

3.2.1. BIM – Building Information Modeling

Der Austausch von Daten über frei offene Schnittstellen, wo ohne Verlust Informationen von Tools ausgetauscht werden können, ist eine der Grundvoraussetzungen für BIM. Die Idee von BIM ist die durchgängige Nutzung von digitalen Gebäudemodellen von der Planung bis zur Realisierung, von der Betriebsphase bis zum Abriss. Bereits in den 70er Jahren veröffentlichte Eastman [3] ein Konzept für die Erstellung und den Einsatz von virtuellen Gebäudemodellen. Im Jahr 1992 wurde der Begriff BIM erstmals von van Needervan und Tolman [4] verwendet. Ein großer Vorteil von digitalen Gebäudemodellen ist der verlustfreie Informationsaustausch, der jedoch nur durch Interoperabilität gegeben ist. Es muss somit das gesamte Modell, einschließlich weiterer relevanter Informationen wie Klimadaten, Nutzungsinformationen, Varianten usw. austauschbar sein.

Die Interoperabilität von BIM-Modellen ist je nach BIM-Typ in unterschiedlichem Maße gegeben. Little BIM bezieht sich auf die Verwendung einer bestimmten BIM-Software durch eine/n einzelne/n Planer:in. In diesem Fall wird BIM ohne externe Kommunikation verwendet [5]. Big BIM bezieht sich auf eine konsistente modellbasierte Kommunikation zwischen allen Beteiligten, die in allen Phasen des Lebenszyklus eines Gebäudes beteiligt sind. Darüber hinaus wird zwischen Closed und Open BIM unterschieden, je nachdem, ob herstellerneutrale Datenaustauschformate verwendet werden oder nicht [6].

Ein bestehender und bereits weit verbreiteter Standard für BIM ist die Definition der Industry Foundation Classes (IFC) [7]. Die IFC-Modelle enthalten sowohl geometrische Daten als auch Metadaten über Gebäudeobjekte und sollen die Interoperabilität unterstützen. Steel et al. [8] untersuchen verschiedene Aspekte der modellbasierten Interoperabilität beim Austausch von Gebäudeinformationsmodellen zwischen verschiedenen Werkzeugen, mit besonderem Fokus auf die Verwendung von IFC. Die Autoren weisen darauf hin, dass eine der größten Herausforderungen im Hinblick auf die Interoperabilität die Inkonsistenz der Modellierungsstile ist. Die Modellierungssprache sollte alle Möglichkeiten klar beschreiben, damit keine unerwarteten Alternativen möglich sind. Auch die Autoren Polit-Casillas und Howe [9] greifen das Thema komplexer Systeme und die Synchronisation verschiedener Informationstypen auf. In ihrer Arbeit kombinieren sie BIM mit einem Systems-Engineering-Ansatz, um einen modellbasierten Engineering-

Ansatz zu erhalten. Dabei konzentrieren sie sich auf die Interoperabilität und Validierung von Informationen in verschiedenen Planungsphasen, von Anforderungen, die in einer Modellierungssprache wie der Systems Modeling Language (SysML) modelliert wurden, bis hin zu Modellen, die z.B. in Computer Aided Design (CAD)-Tools gezeichnet wurden. Paskaleva et al. [10] stellen dar, dass eine der größten Herausforderungen für IFC ist, mit den Aktualisierungen der Bauvorschriften oder mit dem sich ständig erweiternden Stand der Technik bei Simulationswerkzeugen Schritt zu halten. Daher präsentieren sie das Datenmodell SIMULTAN, welches aus verschiedenen Grundtypen besteht, die dann zu komplexeren Modellen kombiniert und wiederum als Typen für andere Modelle verwendet werden können. Dadurch kann jede/r Domänenexperte:in für eine bestimmte Aufgabe eine eigene Datenstruktur erstellen, die automatisch mit der Datenstruktur der anderen Domänenexperten kompatibel ist, die die gleichen Basistypen verwendet [11]. In Bezug auf Gebäudeautomation und Abbildung von Regelungslogik ist in IFC bisher nur möglich, Sensoren und Aktuatoren mit bestimmten Eigenschaften (properties) und Beziehungen zu definieren, aber nicht die Applikations-Logik an sich. In Closed BIM Lösungen gibt es schon rudimentäre sehr vereinfachte Abbildungen, da diese aber „verschlossen“ sind, ist ein Austausch und eine Flexibilität in der Anwendung nicht gegeben.

3.2.2. Gebäudesimulation

Praktisch jedes Gebäude ist ein Unikat – einzigartig in seinem Aufbau, seiner Nutzung oder zumindest seiner Lage. Die Erstellung von Gebäudemodellen im Sinne von Miniaturen während der Vorentwurfs- und Entwurfsphase ist ein traditioneller Arbeitsschritt von Architekt:innen gewesen und ist nach wie vor ein Teil der entsprechenden Ausbildung. Das Verhalten des Gebäudes im Betrieb war dann relativ gut berechenbar, solange einfache Haustechnik eingesetzt wurde und es sich, aus heutiger Sicht, um nicht optimierte Gebäude handelte (= „Altbestand“). Detailliertere Aussagen über das Verhalten modern ausgestatteter Gebäude hingegen, vor allem mit sehr geringem Energieverbrauch, konnten bislang nur nach der Errichtung der Gebäude getroffen werden, d.h., man lernte und lernt nach wie vor in der modernen Haustechnik energieeffizienter Gebäude sehr viel durch „trial and error“. Da Gebäude zu Kulturgütern mit den höchsten Nutzungsdauern zählen, bedeutet diese Vorgangsweise prinzipiell sehr lange Innovationszyklen - abgesehen von den aufgetretenen vermeidbaren Schäden und Reparaturkosten.

Die Möglichkeit der letzten 20 Jahre, Gebäude in ihrem Betriebsverhalten unter Berücksichtigung der haustechnischen Versorgung ausreichend genau zu simulieren, hat große Verbesserungspotenziale eröffnet. Um Gebäude planen zu können, die aus Sicht der Kosten und des Energieverbrauchs optimiert sind, bedarf es Applikationen zur thermischen Gebäudesimulation. Mit steigender Computer-Rechenleistung und fortschreitender Digitalisierung gibt es immer mehr Tools zur Auswahl – ein jedes mit eigenen Vor- und Nachteilen – es gibt kein universelles Framework, das für alle Komplexitätsgrade und Anwendungsfälle perfekt geeignet ist [12]. Ein Visionär wie Joseph Clarke formulierte sogar „If you can't simulate it, don't build it“ und bezog dies vor allem auch auf das haustechnische Verhalten.

Softwarelösungen für die Umsetzung dieser Vision sind im Bereich der Bauphysik definitiv bereits vorhanden, die Entwicklungen simulationstechnischer Grundlagen im Bereich der MSR hingegen hinken noch hinterher. Insbesondere ist die Einbindung in eine BIM-Architektur de facto noch nicht vorhanden, da es zumindest keine weitverbreiteten marktgängigen Lösungen gibt.

Erste Schritte, diese Lücke zu schließen, finden sich in einem der momentan gängigsten Tools für eine integrale Lösung, nämlich IDA-ICE (IDA Indoor Climate and Energy) [13]. In diesem Tool können nicht nur das Gebäude, seine 3D-Geometrie, seine Bauteilaufbauten und seine Nutzung detailliert spezifiziert werden, sondern es kann auch die Gebäudetechnik mitsamt der zugehörigen MSR-Technik abgebildet werden. Die Software bietet einen soliden Baustein dafür, da sie mit den gängigen Control-Blöcken umfangreich vorbestückt ist und über eine ausgereifte Benutzeroberfläche (GUI) verfügt. Sie kann aber auch dazu genutzt werden, eigene komplexe Haustechniksysteme abzubilden, wobei die Informationen in Textdateien (LISP-Dialekt) gehalten werden.

Die objektorientierte Modellierungssprache „Modelica“ und die zahlreichen darauf aufbauenden Tools (z.B. Dymola oder openModelica; weitere Tools siehe [14]) ist eine Alternative zu IDA-ICE. Modelica erhebt den Anspruch, die verschiedensten technischen Domänen (Mechanik, Thermodynamik, Elektrotechnik, Informatik – inkl. MSR) abbilden und miteinander verknüpfen zu können. Aufgrund dieses umfassenden Anspruchs sind die in den verfügbaren Bibliotheken vordefinierten Komponenten in erster Linie ein Abbild einer oder mehrerer grundsätzlicher physikalischer oder logischer Zusammenhänge. D.h., es gibt an sich keine Komponente „Wand“, dafür jedoch Bausteine wie „Wärmeübergang“ oder „Wärmekapazität“, aus denen man die Komponente „Wand“ selbst erstellen könnte. Weiters fehlt in Modelica auch eine Möglichkeit, 3D-Geometrie direkt zu verarbeiten – ein wesentliches Feature für eine Ankopplung an BIM. Kommerzielle Tools, die auf Modelica aufbauen, verfügen zwar über Schnittstellen zu CAD-Programmen (z.B. Dymola kann mit der CAD-Software Catia interagieren), jedoch ist deren Fokus in erster Linie die Automobilindustrie bzw. der Maschinenbau. Weitere namhafte Tools zur Gebäudesimulation sind „Energy Plus“ [15] und „TRNSYS“ [16].

3.2.3. Gebäudeautomation

Gebäudeautomation ist Messen, Steuern, Regeln, Überwachen und Optimieren im Bereich von Anlagen und Einrichtungen der technischen Gebäudeausrüstung. Darunter fallen Systeme der Heizungs-, Lüftungs- und Klimatechnik (HLK) sowie Geräte zur Beleuchtung und Verschattung. Durch optimierte Strategien in diesen energieintensiven Kern-Bereichen werden erhebliche Einsparungen bei den Betriebskosten erzielt, der Energieverbrauch reduziert und gleichzeitig angestrebt, den Komfort der Benutzer zu erhöhen [17]. Aufgrund der räumlichen Gegebenheiten und funktionalen Anforderungen werden verteilte Systeme eingesetzt, wobei sich ein Trend zur weiteren Dezentralisierung und flexibleren Zuordnung von Systemfunktionen zu Geräten abzeichnet. Offene Kommunikationsstandards gewinnen zusehends an Bedeutung, die einer zweischichtigen Systemarchitektur folgen, die lokale Segmente über einen gemeinsamen Backbone verbinden. Einige davon erheben den Anspruch Gewerke-übergreifend agieren zu können (BACnet [18], KNX [19], LonWorks [20]), während andere für spezifische Teilaufgaben vorgesehen sind (z.B. DALI [21], SMI [22]), funkbasierend operieren (z.B. ZigBee [23]) oder im Fokus die Messung des Verbrauchs von Gas, Wärme oder Wasser haben (z.B. M-Bus [24]). Diese Kommunikationsstandards folgen jeweils ihren eigenen Applikationsmodellen, um steuerungs- und regelungstechnische Anwendungen umzusetzen. Technologie-neutrale Richtlinien und Standards, wie VDI 3813 und VDI 3814 des Vereins Deutscher Ingenieure (VDI), enthalten Definitionen für Funktionen, Begriffe und Grundlagen in Gebäude- und Raumautomationssystemen. Eine Liste der Raumautomatisierungsfunktionen findet man beispielsweise in Teil 2 der VDI 3813-Richtlinie [25], während Anwendungsbeispiele für Raumtypen und Funktionsmakros der Raumautomatisierung und -steuerung in Teil 3 vorgestellt werden [26].

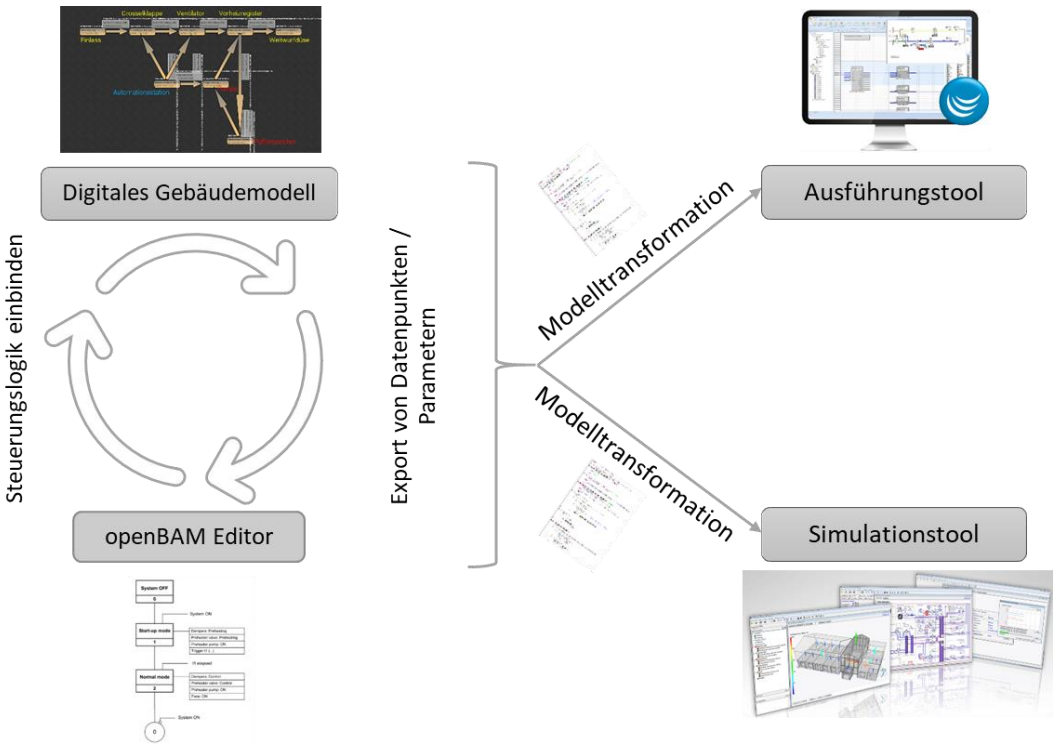
Aufgrund der kontinuierlich ansteigenden Nutzung von 3D-Modellen und der Integration von Informationen in diesen Modellen, müssen allerdings auch Schnittstellen zur Übertragung der Informationen zur Verfügung gestellt werden. Hier gibt es eine Vielzahl an verschiedenen Austauschformaten (z.B. csv, JavaScript Object Notation (JSON)) wie auch die extensible Markup Language (XML), welche sich als plattform- und implementationsunabhängig Sprache etabliert und bewährt hat. Zukünftig könnte dieses Format jedoch durch einen ontologischen Ansatz verdrängt werden. Aussichtsreich sind vermutlich die Standards Resource Description Framework (RDF) oder Web Ontology Language (OWL) des World Wide Web Consortium (W3C), die Begrifflichkeiten und deren Beziehungen beschreiben lassen. Mithilfe von spezifischen Abfragen können die benötigten Informationen an der Schnittstelle abgerufen werden. Die Abfragen könnten zum Beispiel mit SPARQL erfolgen, anhand von bestehenden Brick-Klassen. Zur Integration eines Datenformates in dem Fachbereich der Modellierung und Regelungstechnik wurden schon mehrere Publikationen [27-30] veröffentlicht.

4 Projektinhalt

4.1. Überblick

Im Austausch mit externen Expert:innen mittels Umfrage wurden die Anforderungen an eine Modellierungssprache für MSR erhoben. Auf Basis dessen wurden die generische Abbildung der Logik vorgenommen und mittels Modelltransformationen über das digitale Gebäudemodell an Simulation- und Ausführungstools weitergeleitet. Anhand eines Laboraufbaus einer kleinen Lüftungsanlage werden die entwickelten Modelle und ihre Ergebnisse evaluiert und analysiert. Abbildung 1 zeigt die schematische Darstellung der einzelnen Teile des Projekts openBAM und wie diese ineinandergreifen. Die einzelnen Teile werden in den nachfolgenden Abschnitten nach der Anforderungsanalyse näher beschrieben.

Abbildung 1: Schematische Darstellung der Umsetzung des Projekts openBAM (vgl. [31])



Durch die Fokussierung auf nachfolgende Innovationen hebt sich das Projekt hervor:

(1) Methodische Innovation Im Zuge des Projekts erfolgt eine Konzeption und methodische Erarbeitung einer Modellierungssprache für die Gebäudeautomation, die generisch und plattformunabhängig aufgebaut ist. Die Anforderungen hierzu werden genau analysiert und mit Planer:innen abgestimmt, sodass direkter Nutzen für die Anwender:innen entsteht.

(2) Verknüpfung plattformunabhängige und plattformspezifische MSR Modellierung Da zu Beginn der Planung genaue Implementierungen der Gebäudeautomation noch nicht feststehen, muss hier eine Lösung auf generischer Ebene zur Verfügung gestellt werden. Sobald aber die Planung weiter

fortgeschritten ist und es in Richtung Ausführung geht, soll die schon geplante Arbeit aber für die Umsetzung genutzt werden. Es müssen daher Transformationsmethoden entwickelt werden.

(3) Erweiterung eines generischen offenen Datenmodells Um Informationen erfolgreich und plattformunabhängig austauschen zu können, benötigt es ein offenes Datenmodell, in dem die Daten gehalten werden können. Aufbauend auf dem IT-Ökosystem „SIMULTAN", welches im „Stadt der Zukunft"-Projekt SIMULTAN[1] begonnen wurde zu entwerfen (Proof-of-Concept für das Datenmodell und die konfliktfreie Teamarbeit in einem Planungsteam), wird nun MSR von Gebäudeautomationssystemen abgebildet und gespeichert. Dadurch wird ein weiterer Schritt Richtung holistic open meta data model gesetzt, wo Daten gehalten und mit anderen Tools auf Basis von Datenstrukturen und Modellen wie z.B. IFC, cityGML ausgetauscht werden können. Das Datenmodell dient als Grundlage zur Schnittstelle zwischen Simulations- und Engineering Tools.

(4) Evaluierung der Modelle anhand eines proof-of-concept Use-Case „Laborversuchsaufbau" Bei der Entwicklung der Methode, der generischen Modellierungssprache und der Schnittstellen muss sichergestellt werden, dass die einzelnen Komponenten ineinandergreifen. Deswegen wird das Zusammenspiel in einem Use-Case evaluiert und analysiert. Darüber hinaus werden Stakeholder und Zielgruppen/Nutzer:innen einer zukünftigen Methodenanwendung einbezogen, um die unterschiedlichen Sichten zu berücksichtigen und sicherzustellen, dass die Ergebnisse verständlich und plausibel genutzt werden können.

4.2. Anforderungsanalyse

4.2.1. Stand der Technik der Modellierung MSR

Im Bereich der MSR-Planung gibt es eine Vielzahl unterschiedlicher Standards und Normen. Besonders zu erwähnen seien hier die Standards VDI 3814 – Gebäudeautomation [32], sowie der internationale Standard ISO 16484 - Building Automation and Control Systems [33], der International Standards Organisation (ISO). Daraus ergeben sich folglich auch Inkonsistenzen durch die fehlende Abstimmung der Gremien untereinander. Hinzu kommt, dass in Österreich, im Gegensatz zu Deutschland beispielsweise, keine gesetzliche Verpflichtung zur Befolgung dieser Normen besteht. Dennoch wird im deutschsprachigen Raum vorwiegend auf VDI 3814 gesetzt. Durch die geplante und weitgehend umgesetzte Vereinheitlichung der VDI 3813 – Raumautomation [34] und der VDI 3814 zur neuen VDI 3814 befindet sich der Standard derzeit im Umbruch. Daher sind die genauen Details zu den künftigen Möglichkeiten der MSR-Modellierung noch nicht vollständig spezifiziert. Allerdings werden die grundlegenden Möglichkeiten zur Funktionsbeschreibung bereits in VDI 3814 Blatt 4.3 thematisiert und in früheren Ausgaben des Standards (vgl. VDI 3813-2:2011 und VDI 3814-6:2008) teilweise beschrieben. Sie umfasst die

- textliche Funktionsbeschreibung und/oder die
- GA-Funktionsblockdarstellung (GA-FB) (siehe VDI 3814 Blatt 3.1 und Blatt 3.2 (in Vorbereitung)) und/oder
- GA-Zustandsgraphen und/oder
- GA-Ablaufdiagramme

[1] https://nachhaltigwirtschaften.at/de/sdz/publikationen/schriftenreihe-2020-04-simultan.php

Ein Problem, das auch in der neuen VDI 3814 Norm nicht gelöst ist, ist die einheitliche Datenweitergabe der verschiedenen an Bauprojekte beteiligten Aktoren. So beschreibt der Standard nur wie die einzelnen Diagramme und Schemata auszusehen zu haben, ohne jedoch ein standardisiertes Austauschformat anzubieten. Die in der Praxis verwendeten Tools, sowie das zugehörigen Datenformat sind zumeist proprietär. Dies bedeutet für die einzelnen Akteur:innen erheblichen Mehraufwand durch die Neumodellierung des Projektes in den von ihnen verwendeten Tools. Es sei noch erwähnt, dass während der fortschreitenden Projektlaufzeit Änderungen in der Entwicklung der VDI 3814 vorgenommen wurden, sodass die ursprünglich geplanten Ablaufdiagramme und Zustandsgraphen aus Blatt 4.4 inzwischen nicht mehr eins zu eins vorhanden sind. Nichtsdestotrotz bleibt die einheitliche, maschinenlesbare Struktur für den standardisierten Austausch von Daten von zentraler Bedeutung. Dies wurde auch in den Ergebnissen unserer Umfrage (vgl. Abschnitt 4.2.3) bestätigt, die die Wichtigkeit dieser Struktur unterstrichen.

Anforderungen an MSR-Modellierungssprachen

Neben der Möglichkeit, die in der Gebäudeautomation notwendigen Funktionalitäten erfassen und modellieren zu können, sollte eine Modellierungssprache ein standardisiertes oder zumindest ein offenes Austauschformat besitzen. Sofern möglich, sollte größtmögliche Konformität zu VDI 3814 gewahrt und Open-Source Lösungen bevorzugt werden. Neben einer breiten Unterstützung, inklusive Toolsupport, muss außerdem im Sinne der Akzeptanz darauf geachtet werden, beteiligte Akteur:innen nicht zur Offenlegung ihres gesamten Knowhows zu zwingen.

Anforderungen an MSR-Modellierungssprachen

Um ein (semi-) automatisches Deployment sowie automatisierte Simulationen durchführen zu können, kann es notwendig sein das gewählte MSR-Modellierungsformat in ein anderes Dateiformat zu konvertieren/transformieren. Hierfür bieten sich Modell-zu-Modell (M2M) Transformationen aus dem Bereich MDE an. Wichtig auch für zukünftige Entwicklungen im Bereich der VDI und speziell Weiterentwicklungen der VDI 3814, müssen die gewählten Transformationsmethoden mächtig genug sein, ein gewähltes Model in eine der VDI entsprechende Darstellung zu transformieren, beziehungsweise rücktransformieren.

Auch bei den Transformationsmethoden gilt, dass nach Möglichkeit offene, standardisierte Formate verwendet werden, die eine breite Unterstützung finden und eine flexible und leicht adaptierbare Transformation ermöglichen.

Bestandserhebung und Vergleich etablierter Sprachen

Basierend auf der VDI 3814 bieten sich neben einer rein textuellen Beschreibung drei Beschreibungsmethoden besonders an, nämlich:

- eine Beschreibung mittels Zustandsgraphen,
- eine Beschreibung mittels Ablaufdiagrammen, sowie
- eine Beschreibung mittels Funktionsblöcken

Bereits in der VDI 3814-6:2008 [35] wird eine Zustandsgraphenbeschreibung erwähnt. Hier wird ein Zustandsgraph definiert als „grafische Darstellung von Zuständen, deren Übergängen, Übergangsbedingungen und Aktionen“. In VDI 3814-6:2008 wurden ebenfalls die folgenden möglichen Beschreibungssprachen in Bezug auf die Kriterien Eindeutigkeit, Übersichtlichkeit, Detaillierungsgrad, Strukturierbarkeit, Selbsterklärbarkeit und Aufwand untersucht:

- Ablaufsprache (AS) – nach IEC 61131-3
- Automaten (Zustandsgraph)
- Funktionsbausteinsprache (FBS) – nach IEC 61131-3
- Kontaktplan (KOP) – nach IEC 61131-3
- Petrinetze (PN)
- Procedural Function Charts (PFC)
- Programmablaufgraph (PAG)
- Message Sequence Chart (MSC)
- Specification and Description Language (SDL)
- Hardware Description Language (VHDL)

Basierend auf den erwähnten Kriterien, wurde die Beschreibung mittels Zustandsgraphen ausgewählt. Abbildung 2 zeigt die graphische Darstellung eines Zustands laut VDI 3814-6:2008. Dieser wird mittels eines Rechtecks, in dem sich die Zustandsbeschreibung, sowie die Zustandsnummer befindet, beschrieben. Durch Linien auf der Ober- und Unterseite des Rechtecks ist ein Zustand mit seinen Vor- und Nachfolgezuständen verbunden. Ebenfalls auf der Oberseite des Zustands befinden sich die Übergangsbedingung(en), die erfüllt sein müssen, um vom Vorgängerzustand in den aktuellen Zustand zu wechseln. Geschieht dies, werden die Aktionen rechts des Zustands ausgeführt.

Abbildung 2: Zustand laut VDI 3814-6:2008 [35]

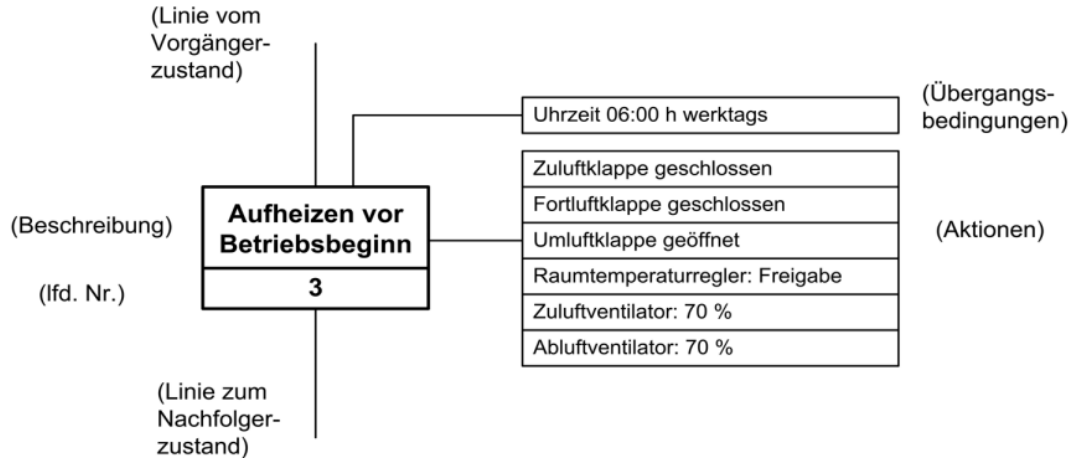
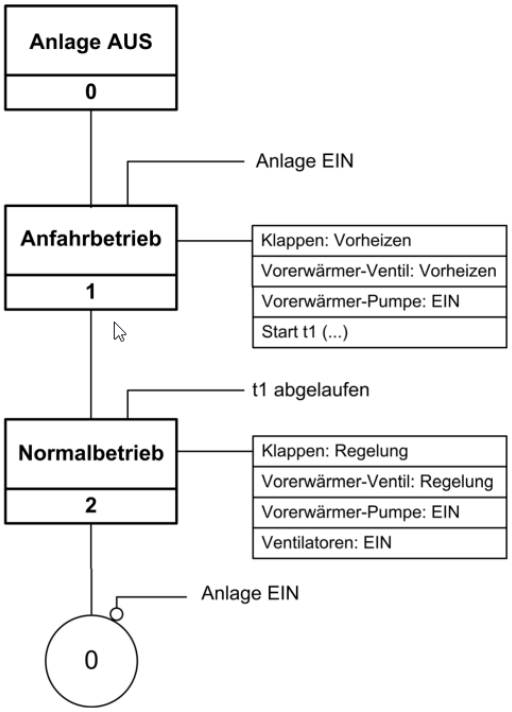


Abbildung 3 zeigt eine vereinfachte beispielhafte Abbildung einer Lüftungsanlage. Initial befindet sich das System im Zustand „Anlage Aus“. Sobald das Gerät eingeschalten und die Übergangsbedingung von Zustand 2 erfüllt wird, wechselt das System in den Anfahrbetrieb. In diesem werden diverse Aktionen, wie zum Beispiel das Vorheizen der Klappen ausgeführt. Zusätzlich wird ein Timer gestartet. Nach Ablauf des Timers wechselt das System in den Normalbetrieb, in dem bis zum Ausschalten verweilt.

Abbildung 3: Beispielhafte Lüftungssteuerung laut VDI 3814-6:2008 [35]



Maschinenlesbare Zustandsgraphendarstellung

Basierend auf den etablierten Sprachen und den Anforderungen an MSR-Modellierungssprachen ergab die Recherche mehrere Möglichkeiten einer maschinenlesbaren Darstellung von Zustandsgraphen. So bietet beispielsweise die Unified Modeling Language (UML) mit ihren Zustandsgraphen eine solche Beschreibung. UML genießt eine breitflächige (open source) Toolunterstützung wie zum Beispiel Papyrus[2] und stellt ein standardisiertes XML Metadata Interchange (XMI) Format für die Darstellung zur Verfügung.

Darüber hinaus hat auch das W3C eine Empfehlung zur XML-Darstellung von Zustandsgraphen, namens State Chart XML (SCXML)[3], publiziert. Hierbei handelt es sich um eine XML-basierte Beschreibung von Harel Zustandstabellen, welche vom Mathematiker David Harel entwickelt wurden und auch in UML 2.3 zum Einsatz kommen. Im Gegensatz zur graphischen Beschreibungssprache UML erlaubt SCXML eine formale Darstellung eines Zustandsgraphen in einem XML-Dialekt. Wie auch UML, wird SCXML von mehreren (Open-Source) Tools, wie zum Beispiel ScxmlGui[4] und QTCreator[5] unterstützt. Daneben existiert noch eine Vielzahl an Bibliotheken und Sprachanbindungen für verschiedenste Programmiersprachen. In [36] präsentiert Andolfato et al. ein Tool zur Konvertierung der UML Zustandsgraphendarstellung nach SCXML.

Auch IEC 61499 [37] bietet mit seinen Execution Control Charts (ECCs) eine formale Beschreibung von Zustandsgraphen. Diese ECCs werden verwendet, um das dynamische Verhalten von Funktionsblöcken zu beschreiben. Es handelt sich dabei um Zustandsmaschinen, die abhängig von der Eingabe und dem aktuellen Zustand bestimmte Aktionen durchführen. Da IEC 61499 eine XML-

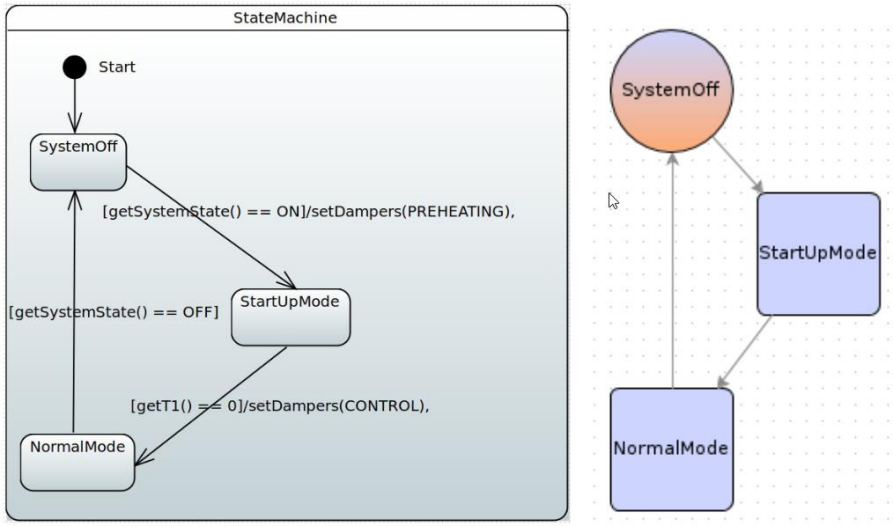
[2] https://www.eclipse.org/papyrus/
[3] https://www.w3.org/TR/scxml/
[4] https://github.com/fmorbini/scxmlgui
[5] https://doc.qt.io/qtcreator/creator-scxml.html

Beschreibung für alle seine Elemente anbietet, existiert auch eine standardisierte Beschreibung der ECCs.

Alternativ existieren noch einige weitere Ansätze zur Modellierung von endlichen Automaten (EA), wie beispielsweise GraphML[6], ein allgemeiner XML-Dialekt zur Beschreibung von Graphen, oder JFMS[7], einer JSON basierten EA-Modellierungssprache. Es ist ebenfalls möglich EAs mittels einfacher Tabellen zu beschreiben.

Nachdem sowohl UML als auch SCXML, im Gegensatz zu den anderen Ansätzen, auf eine breitflächige Toolunterstützung bauen können, wird im Weiteren, der Fokus auf diese beiden Beschreibungsmöglichkeiten gelegt, um passende MSR-Transformationsmethoden evaluieren zu können. Dazu wurde der in Abbildung 3 gezeigte, einfache Anwendungsfall mit Hilfe der Open-Source Tools Eclipse Papyrus und ScxmlGui modelliert. Abbildung 4 zeigt die graphische Darstellung, während der Source Code im Anhang 9.2 gefunden werden kann. UML erlaubt auch ein Datenmodell für Wächter und Aktionen zu modellieren, während SCXML einen zu keinem bestimmten Datenmodell zwingt, sondern Datenmodelle wie XPath und ECMA-Script vorschlägt.

Abbildung 4: Modellierung eines Anwendungsfalls mit Papyrus (links) und ScxmlGui (rechts)



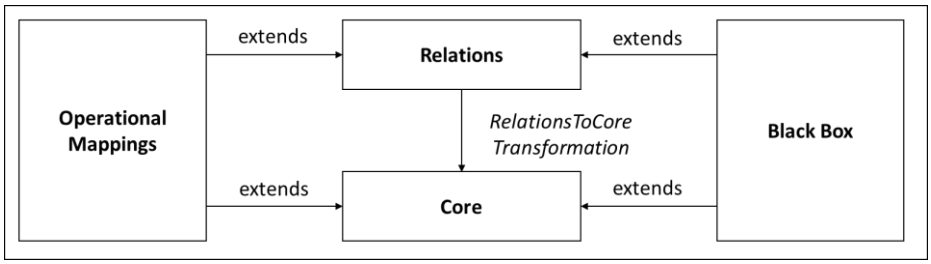
Modell zu Modell Transformation

Sobald Zustandsgraphen maschinenlesbar verfügbar sind, ist es wichtig, diese auch weiterzuverarbeiten. Eine Möglichkeit um Spezifikationen in unterschiedliche Zielformate zu transferieren bieten Modelltransformationen. Im Kontext von MDE [39], versteht man darunter einen automatisierten Prozess für das Ändern und Erstellen von Modellen, um Fehler und Arbeit zu reduzieren. Es gibt verschiedene Möglichkeiten Modelltransformationen zu klassifizieren. So kann beispielsweise zwischen Modell zu Modell (M2M) Transformationen und Model zu Text (M2T) Transformationen unterschieden werden. Während als Eingabe beide Transformationen ein Modell benötigen, liefert eine M2M Transformation wiederum ein Modell als Ausgabe und eine M2T Transformation (strukturierten) Text, beispielsweise in Form von Programmcode.

[6] http://graphml.graphdrawing.org/
[7] https://github.com/ryankurte/jfsm

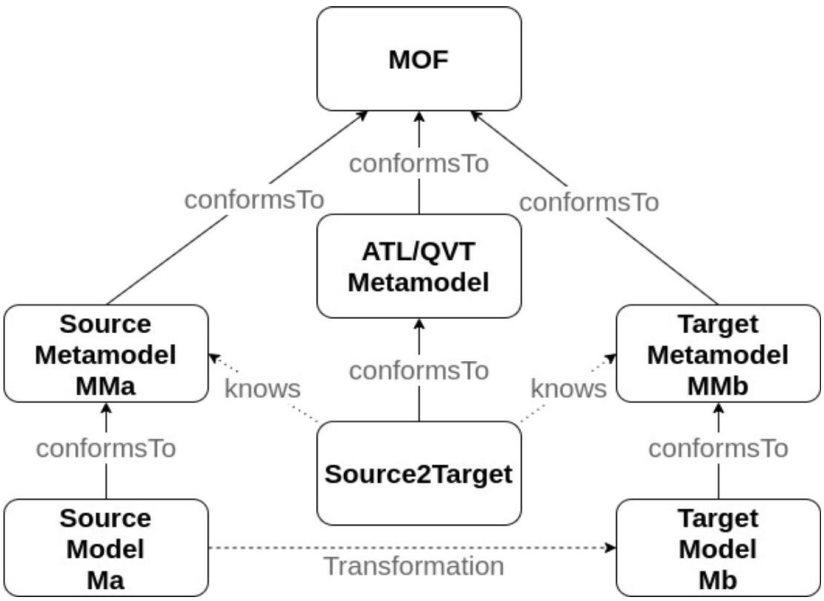
Prinzipiell können Modelltransformationen mit beliebigen (höheren) Programmiersprachen beschrieben werden. Dennoch wurden sogenannte Modelltransformationssprachen (MTLs) eigens für diesen Zweck entwickelt. Beispielsweise hat die Object Management Group (OMG) eine Reihe an MTLs unter dem Namen Query/View/Transformation (QVT) standardisiert [38]. QVT erlaubt nicht nur Modelltransformationen (Transformation), sondern auch die Evaluierung von Ausdrücken über ein Modell (Query), sowie die Ableitung eines Modells unter Aufrechterhaltung der Beziehung untereinander (View). Neben den beiden deklarativen Sprachen QVT-Core und QVT-Relations, bietet QVT auch eine imperative Sprache namens QVT-Operational-Mappings an, welche für unidirektionale Transformationen verwendet werden kann. Zusätzlich ist es möglich Transformationsfunktionalität in beliebigen Sprachen mittels QVT Black Box einzubinden. Abbildung 5 zeigt die Architektur von QVT, sowie die Beziehungen der Komponenten untereinander.

Abbildung 5: QVT Architektur [38]



Neben QVT existiert noch eine Vielzahl weitere MTLs, wie die Epsilon Transformation Language (ETL)[8] und die Atlas Transformation Language (ATL)[9]. ATL wurde auf Basis eines Request For Proposal der OMG entwickelt und hat eine große Nutzerbasis. Der ATL Transformationsprozess funktioniert gleich dem von QVT. Dieser Modelltransformationsprozess ist in Abbildung 6 illustriert.

Abbildung 6: M2M Transformationsprozess [40]



[8] https://www.eclipse.org/epsilon/doc/etl/
[9] https://www.eclipse.org/atl/

Die Basis von Transformationen bilden die Metamodelle der Quell- und Zielmodelle. Sowohl ATL als auch QVT verwenden hierfür das, von der OMG standardisierte Meta-Meta-Modell Meta Object Facility (MOF), welches die Beschreibung von Metamodellen wie jenes von UML und XML erlaubt. Angenommen das Quell Meta-Modell beschreibt einen Teil von UML, dann beinhaltet das Quellmodell die tatsächlichen Daten, strukturiert basierend auf dem Meta-Modell. Die Modelle und Meta-Modelle werden im standardisierten XMI-Format beschrieben. Im Standard ist vorgesehen, dass Daten in diesem Format vorliegen, anderenfalls muss eine Einspeisung (Injection) nach, oder Extraktion (Extraction) aus XMI erfolgen.

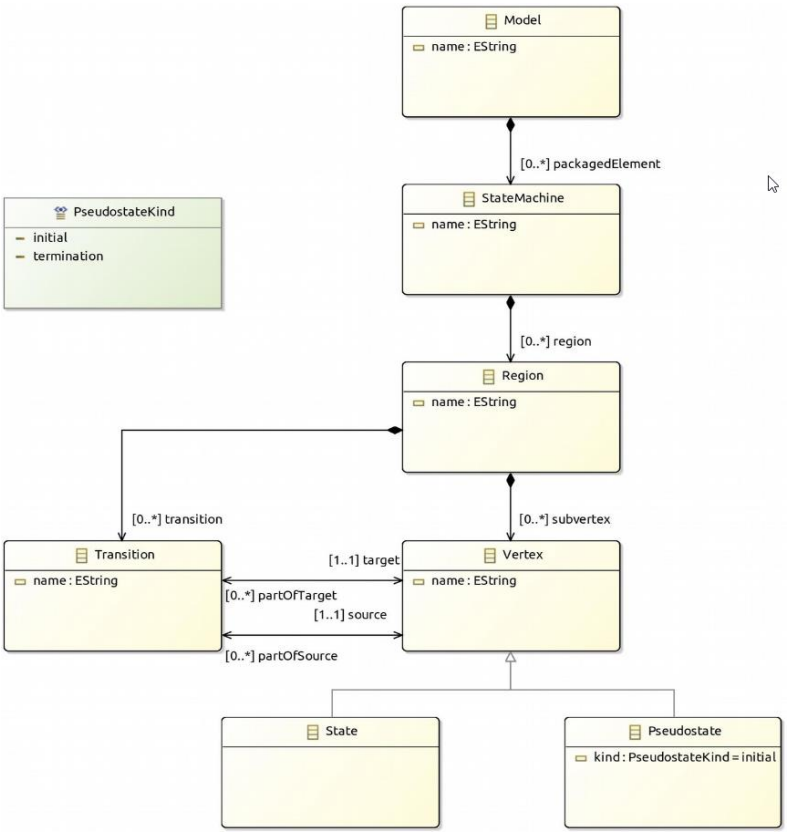
Neben der Implementierung von ATL und QVT bietet das Eclipse Modeling Project auch eine Referenzimplementierung des Essential MOF, einer Teilmenge von MOF, auch Ecore genannt. In dem Framework werden Tools für die automatische Generierung von Meta-Modellen, sowie zur Serialisierung von und nach XMI direkt angeboten. Vereinfacht ausgedrückt besitzt in Ecore eine Klasse neben ihren Namen, Attribute mit ihrem Datentyp und Referenzen mit ihrer Vielfachheit.

Modell zu Modell Transformation am Beispiel UML2SCXML

Basierend auf der Empfehlung von VDI 3814-6:2008 eine Zustandsgraphendarstellung zur MSR-Modellierung zu verwenden, wurden zwei unterschiedliche Modelle UML und SCXML näher untersucht. Als Proof-of-Concept und zur Veranschaulichung des Konzepts wird eine einfache M2M-Transformation von UML-Zustandsdiagrammen nach SGXML mit Eclipse und ATL durchgeführt. In einem ersten Schritt werden die beiden Metamodelle von UML und SCXML mit Ecore modelliert.

Abbildung 7 zeigt eine graphische Darstellung eines vereinfachten UML Metamodells um einfache Zustandsgraphen zu beschreiben. Der Quellcode hierfür findet sich im Anhang 9.2.1.

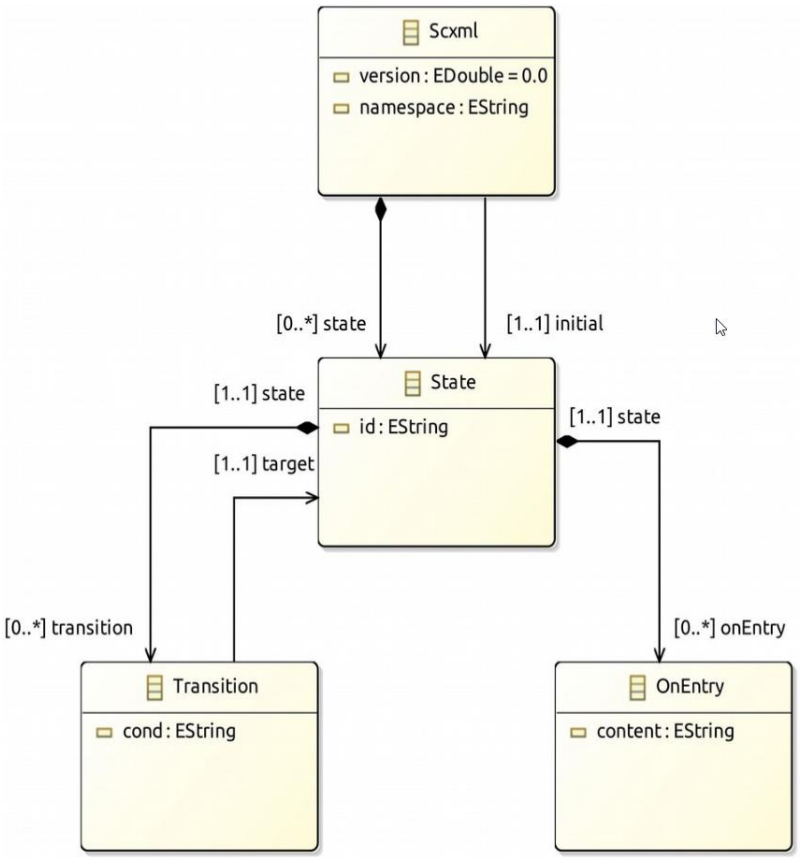
Abbildung 7: Graphische Darstellung eines vereinfachten UML Meta-Modells



Ein Zustandsautomat enthält Regionen, die wiederum Knoten (Pseudozustände und „normale“ Zustände) und Übergänge enthalten. Ein Übergang verbindet zwei Knotenpunkte miteinander. Außerdem kann ein Pseudozustand der Start- oder Endzustand sein.

Abbildung 8 zeigt die grafische Darstellung des vereinfachten SCXML Meta-Modells, das ähnlich aufgebaut ist (siehe auch Anhang 9.2.2). Ein SCXML-Element enthält den Anfangszustand, der einen Aktionsblock (was beim Eintritt zu tun ist) sowie mehrere Übergänge einschließlich der Übergangsbedingung zu anderen Zuständen haben kann.

Abbildung 8: Graphische Darstellung eines vereinfachten SCXML Meta-Modells



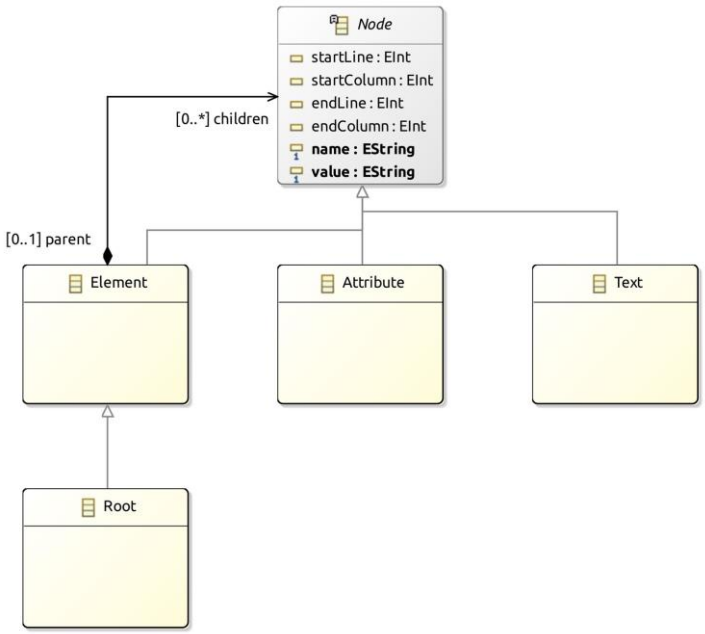
Für die Abbildung von UML auf SCXML werden drei Regeln verwendet:

- Pseudostate (initial) → Scxml, State:
  Für den Anfangszustand des UML-Diagramms wird ein Zustand sowie ein Scxml-Element, das auf diesen Zustand zeigt, erzeugt.
- State → State:
  UML Zustände werden direkt auf Zustände von SCXML abgebildet.
- Transition → Transition, OnEntry:
  Im vereinfachten UML-Modell werden sowohl die Bedingung als auch die Aktion im Namensfeld der Transition gespeichert, getrennt durch das Zeichen '/'. In dieser Regel wird also ein UML-Übergang auf einen SCXML-Übergang abgebildet, der die Bedingung und ein SCXML onEntry-Element mit der Aktion enthält.

Auch hier ist der vollständige Quellcode im Anhang 9.2.5 zu finden. Die Ausgabe dieser Transformation ist dann eine XMI-Datei. Um die entsprechende XML-Datei einfach extrahieren zu

können, ist eine zweite Transformation in ein XML-Modell erforderlich (Anhang 9.2.6). Abbildung 9 zeigt die grafische Darstellung eines vereinfachten XML-Metamodells aus den Eclipse ATL-Beispielen. Ein Element kann andere Elemente, Attribute und Text als Kinder haben. Außerdem wird ein Wurzelelement (Root) definiert, welches das Element spezialisiert.

Abbildung 9: Graphische Darstellung eines vereinfachten XML Meta-Modells



Die Transformation ist im Anhang 9.2.7 zu finden. Das SCXML-Element scxml wird auf das XML-Wurzelelement abgebildet, während die anderen drei SCXML-Elemente auf XML-Elemente abgebildet werden. Außerdem werden die Attribute auf XML-Attribut- oder Textelemente (onEntry) abgebildet. Mit Hilfe einer ATL-Abfrage (aus den Eclipse ATL-Beispielen im Anhang 9.2.8) ist es möglich, eine gültige (SC)XML-Datei zu extrahieren.

Modell zu Text Transformation

M2T-Transformationen ermöglichen es Modelle in beliebigen Text, wie zum Beispiel einer Dokumentation oder einen ausführbaren Programmcode zu transformieren. Um Zustandsgraphen in ein beliebiges Format (nicht XML-ähnlich) transformieren zu können, müssen auch M2T-Transformationen berücksichtigt werden. Wie für M2M hat die OMG auch für M2T eine MTL standardisiert, die MOF-Model to Text Transformation Language (MOFM2T). MOF-basierte Modelle sind daher auch die Basis für diese Transformation. Eclipse Acceleo ist eines der ersten Werkzeuge, das den MOFM2T-Standard implementiert hat und kann als dessen Referenzimplementierung angesehen werden. Daneben gibt es weitere umfangreiche, aber nicht standardisierte Frameworks für M2T-Transformationen wie zum Beispiel das Eclipse Epsilon Projekt. Es sei noch erwähnt, dass Tools, wie zum Beispiel Eclipse XText[10] das Parsen von (strukturiertem) Text, sowie das Umwandeln dieses Textes nach XMI erlauben.

[10] https://www.eclipse.org/Xtext/

Modell zu Text Transformation am Beispiel UML2SCXML

Die Transformationen von der UML in eine SCXML und weiter in ein XML-Modell wurden bereits in den vorherigen Abschnitten gezeigt. Anstatt ATL-Abfragen zu verwenden, ist es auch leicht möglich, Acceleo (MOFM2T) zu verwenden, um eine XML-Datei zu erstellen. Dabei wird für jedes Element das begin-Tag ausgegeben, gefolgt von einer Iteration über die Attribute. Anschließend wird dieser Vorgang für alle untergeordneten Elemente rekursiv wiederholt, um dann das entsprechende End-Tag auszugeben. Der Quelltext ist ebenfalls im Anhang 9.2.9 zu finden.

Synopsis

Sowohl SCXML als auch UML eignen sich für die Beschreibung von Zustandsgraphen, da beide von einer Vielzahl von Tools unterstützt werden. Dennoch ist UML die bevorzugte Wahl, da bereits ein detailliertes Metamodell zur Verfügung steht, welches die Modelltransformationen vereinfacht. Sowohl QVT als auch ATL sind geeignet, ein Ausgangsmodell in ein gewünschtes Zielmodell zu transformieren. Mit MOFM2T ist es dann leicht möglich, einen beliebigen (strukturierten) Text zu erzeugen. Soll dieser Text auch als Input verwendet werden, d.h. ist eine bidirektionale Transformation erforderlich, kann er mit Eclipse Xtext als Parsing-Tool wieder in ein UML-Modell zurückgeführt werden.

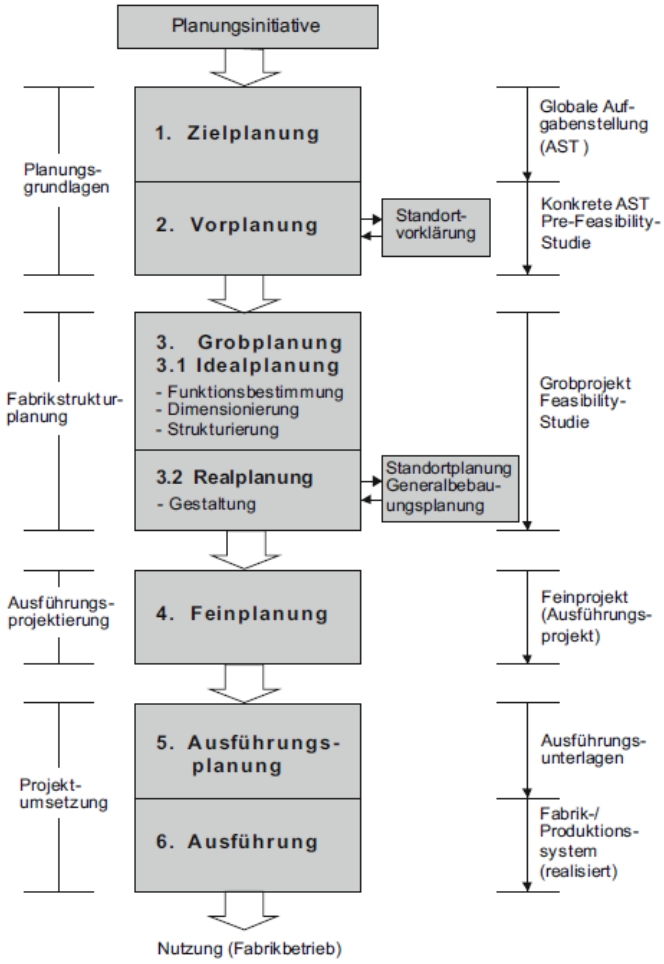
4.2.2. Stand der Praxis

Im vorangegangenen Abschnitt 4.2.1 wurden die Anforderungen an die notwendige Modellierungssprache und deren Umsetzung erörtert. In diesem Abschnitt wird nun auf die wesentlichsten Arbeitsschritte zur Realisierung von Bauprojekten eingegangen. Des Weiteren werden die Stärken und Schwächen des aktuellen Systems behandelt und wie eine zukunftsorientierte Arbeitsweise etabliert werden könnte.

Abbildung 10 zeigt die notwendigen Schritte zur Realisierung eines Bauvorhabens. Grundsätzlich gliedert sich der Bauprozess in sechs Phasen. Je nach Auftragsvolumen sind die ersten vier oder gar fünf Phasen im Auftragsumfang des Planungsunternehmens. Hierbei besteht eine Geschäftsbeziehung zwischen Auftraggeber und Planungsunternehmen (vgl. [41]). Die Schnittstelle zur Ausführungsplanung muss im Startgespräch des Projektes erörtert werden. Das bedeutet, dass die strategische Zielplanung des Vorhabens schon vor Nennung eines Auftragnehmers deklariert sein muss. Die Anforderungen des Auftraggebers müssen vollinhaltlich abgebildet sein.

Bei den Arbeitsschritten handelt es sich um den Idealfall. Leider entstehen während der Ausführungen jedoch Abweichungen, da bestimmte Nutzer:innen-Wünsche identifiziert werden. Es macht somit eine Abweichung von der Theorie notwendig. Auch bauliche Gegebenheiten können es erforderlich machen, dass die Ausführungsplanung adaptiert werden muss. Diese Änderungen müssen unter zeitlichen Druck geplant und nachgeführt werden. Gegenwärtig lassen Bauzeitterminpläne nur wenig Spielraum für diese notwendigen Schritte zu. Zusätzlich muss der kaufmännische Prozess im Zuge von Nachträgen oder Vertragserweiterungen ebenfalls berücksichtigt werden. Hierbei handelt es sich um einen iterativen Prozess, der die meisten Bauvorhaben vor ernsthafte Herausforderungen stellt.

Abbildung 10: 6-Phasen-Modell der Planungsphasen nach Grundig, C.-J. [40, S. 14]



Hinzu kommt, dass es eine Vielzahl an notwendigen Regelwerken gibt, die es im Zuge des Bauprozesses zu berücksichtigen gilt. Die meisten Bauverträge unterliegen der ÖNORM B2110 (Allgemeine Vertragsbestimmungen für Bauleistungen –Werkvertragsnorm) sowie der ÖNORM A2060 (Allgemeine Vertragsbestimmungen für Leistungen –Werkvertragsnorm) als Grundlage. In diesen werden die Vertragsbestandteile sowie Rechte und Pflichten definiert. Bei Errichtung von Nichtwohngebäuden gilt unter anderem auch die „Verordnung über Arbeitsstätten“ (ArbStättV) [42] in der letztgültigen Fassung. Für die gewerkspezifische Ausführung müssen zusätzliche Norm-Reihen berücksichtigt werden. Als Beispiel für die Errichtung von Schaltschränken sei hier die ÖNORM E8001 (Errichtung von elektrischen Anlagen mit Nennspannungen bis ~1000 V und ⎓1500 V) genannt. Somit ist festzuhalten, dass die Prüfung und Überarbeitung der Ausführungsplanung, aufgrund der hohen Komplexität der Richtlinien und Vorgaben, eine komplexe Thematik birgt.

Zum anderen können Informationen nicht automatisiert übernommen werden. So ist der aktuelle Austauschstandard IFC gemäß ISO 16739 (Industry Foundation Classes (IFC) für den Datenaustausch in der Bauindustrie und dem Anlagen-Management) noch nicht vollinhaltlich abbildbar. Die Interoperabilität der Datenaufbereitung kann, zum heutigen Zeitpunkt, nicht gewährleistet werden. Es gibt zum Beispiel einen Informationsverlust, wenn von einer Projektierungssoftware Modelle exportiert und in einer anderen Software importiert werden. Dabei beschränken sich die Möglichkeiten auch primär auf die Gebäudestruktur, -form sowie -daten. Anlageninformationen können partiell exportiert werden. In keiner etablierten Softwarelösung (z.B. Autodesk Revit 2023,

Trimble plancal nova u.Ä.) kann momentan die MSR von Anlagenkomponenten interoperabel abgebildet werden, was die in Abschnitt 4.2.1 formulierten Anforderungen an einen flexiblen Austausch eindeutig unterstreicht.

Projektablauf Mess-, Steuerungs- und Regeltechnik

In diesem Abschnitt wird der Projektablauf eine:s/r Auftragnehmer:s/in, mit Spezialisierung hinsichtlich Regelungstechnik, betrachtet. Auf die kaufmännischen Prozesse wie Angebotslegung, Beauftragung sowie Beschaffungs- und Rechnungswesen wird nicht näher eingegangen.

Die Grundlage zur Projektierung von Regelungstechnischen Projekten bilden folgende Unterlagen:

- Funktionsbeschreibung
- Schematische Darstellung der Anlagen
    - Gewerkspezifische Darstellungen (z.B. Heizung, Lüftung, Klima, Sanitär, Elektro, u.Ä.)
    - Optional: Regelschemata vom Planungsunternehmen
- Grundrisspläne mit eingezeichneten Komponenten
- Auslegungsergebnisse/Berechnungsergebnisse von Fremdgewerken

Bei diesen Unterlagen handelt es sich um die Mindestanforderung, um eine Projektierung durchführen zu können. *„Die Software-Funktionsbeschreibung beschreibt die Handlung, die man mit einer Funktion ausführen kann. Sie fasst dabei nur die Informationen zusammen, die für den Benutzer interessant sind und stellt diese Informationen kurz und übersichtlich dar“* [43, S. 62].

Daraus resultierend stellt die Funktionsbeschreibung einen substanziellen Bestandteil der Projektierung dar. Die schematische Darstellung von Anlagen wird hingegen in regelungstechnische Schemata umgewandelt. In diesen Regelschemen sind die Datenpunkte sowie die Kennlinien der Regelstrategie (Regeldiagramme) sowie der Anlagenaufbau abgebildet. Mithilfe dieser zwei Komponenten können Datenpunktlisten, Gebäudeautomation Funktionslisten (GA-FL), Artikel- und Motorlisten sowie Ventilauslegungen an Auftraggeber:innen zur Verfügung gestellt werden.

Momentan sind die Informationsaustauschmöglichkeiten von bestehenden Planungssystemen in die Projektierungsumgebungen von ausführenden Firmen mangelhaft beziehungsweise schwer umsetzbar, da es keine standardisierte Schnittstelle gibt. So zeigt sich auch hier die Notwendigkeit des Projektziels um ausführende Unternehmen eine Möglichkeit zu bieten, die schon durchgeführte Planung zu importieren und effizient weiter zu verwerten.

Nach vollständigem Abschluss der Projektierungsarbeiten (Engineering-Phase) kann die Parametrierung der Funktionsblöcke – gemäß EN 61131-3 – erfolgen. Der letzte Arbeitsschritt wird durch die Inbetriebnahme-Phase deklariert. In der Inbetriebnahme werden die Komponenten aktiv geschalten und auf Funktion überprüft. Auch eine Prüfung aller Datenpunkte sowie der Sicherheitsfunktionen wird im Zuge der Inbetriebnahmephase durchgeführt. Abschließend kommt ein Probebetrieb zum Einsatz mit einer abschließenden förmlichen Übernahme des Projektes durch die Auftraggeber:innen.

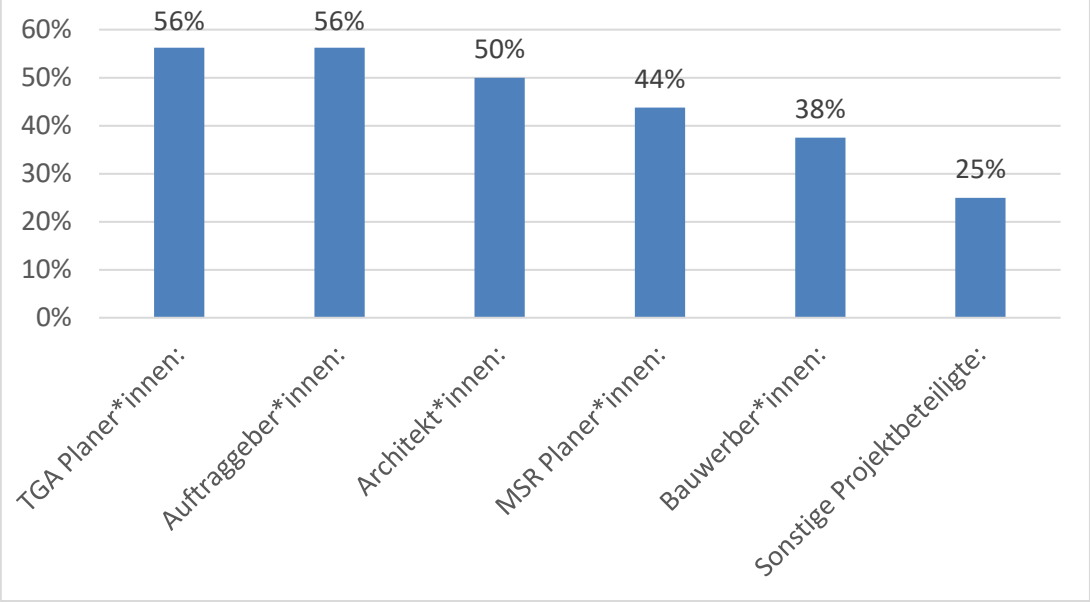
4.2.3. Umfrage im Zuge des Projekts

Im Zuge der Anforderungsanalysephase wurde im Projekt eine Onlineumfrage (Dauer 5 Minuten) von Mai bis Oktober 2022 durchgeführt, wobei von 33 Teilnehmenden rund 42% den Fragebogen

abgeschlossen haben (meist 16 Beantwortungen). Die Teilnehmenden gaben an in der Planung und Ausführung im Bereich MSR (45%), Planung und Ausführung im Bereich Projektentwicklung (inkl. Planung von zukünftigen Projekten auf Basis des Bestands) (39%), sowie in der Planung und Ausführung im Bereich Gebäudeautomation (33%) zu arbeiten, während 8% der Befragten in einem anderen Bereich tätig sind. Die Unternehmensgröße, bei den die Befragten arbeiten, variierte von Einzelunternehmen bis hin zu Unternehmen mit mehr als 100 Mitarbeitenden. Die beruflichen Positionen reichten von Geschäftsführung und Abteilungsleitung über Projektleitung bis hin zu Fachkräften. Zudem erstreckte sich die Berufserfahrung der Befragten von 1-5 Jahren bis zu über 20 Jahren. Mit 67 % stellen Fachhochschulen und Universitäten die häufigste höchste abgeschlossene Schulbildung dar, während 27 % der Befragten die Matura und 6 % eine Berufsschule als höchsten Bildungsabschluss angaben. Die Befragten repräsentierten somit sowohl große als auch kleine Betriebe und deckten ein breites Spektrum an Erfahrungshorizonten ab, von langjähriger bis hin zu kürzerer Berufserfahrung.

In Bezug auf die Technische Gebäudeausrüstung (TGA) zeigen die Ergebnisse der Umfrage klare Herausforderungen auf. So zeigt Abbildung 11 wer in der Regel Vorgaben zur TGA Planung macht, wobei TGA Planer:innen und Auftraggeber:innen am Häufigsten genannt wurden.

Abbildung 11:Wer übermittelt üblicherweise Vorgaben zur TGA Planung? (ausgefüllte Beantwortungen: 16, Mehrfachauswahl zulässig)



Bei der Art und Weise wie diese Vorgaben gemacht werden, lässt sich feststellen, dass in den meisten Fällen eine Kombination von PDF, Excel und DWG/CAD-Dateien verwendet wird, wobei spezifische Anforderungen je nach Rolle und Verantwortungsbereich im Projekt unterschiedlich ausfallen können. Ein paar Mal wurde auch IFC genannt. In Bezug auf Informationsverluste im Planungsprozess gaben die Befragten an, dass dieser vor allem gegenüber Auftraggeber:innen (65%) und TGA Planer:innen (53%) erfolgt. Darüber hinaus erfolgt ebenfalls ein Informationsverlust zu Bauwerber:innen (41 %), Architekt:innen (35%) und MSR Planer:innen (29%). Die restlichen 18% entfallen auf HKLS-Firmen (Heizung, Klima, Lüftung und Sanitär), Nutzer:innen und Betreiber:innen.

Im nächsten Abschnitt der Umfrage ging es um die Planungswerkzeuge und Austauschformate innerhalb der Unternehmen der Befragten im Bereich der TGA Planung. Die Umfrage zeigt eine breite Nutzung unterschiedlicher Softwaretools. In Bezug auf die Microsoft Office Suite verwenden die meisten Befragten Excel (6 Nennungen) und Word (5 Nennungen) oder zusammen (7 Nennungen). Auch Visio und MS Project werden vereinzelt genutzt. Im Bereich CAD 2D bzw. 3D dominieren AutoCAD (10 Nennungen) und WS CAD (3 Nennungen), während vereinzelt auch Programme wie CATS und Brics CAD genannt wurden. Ebenso werden gerne Revit (5 Nennungen) und Trimble (2 Nennungen) in Bezug auf BIM Software genutzt und auch vereinzelt Allplan, Solibri, PlanRadar, ABK3 und Linear. Auch werden Branchen spezifische Tools eingesetzt, so z.B. eigens Firmenintern entwickelte Toolframeworks. Daraus ergeben sich auch die unterschiedlichen Austauschformate, die bei der TGA Planung Verwendung finden, wie in Tabelle 1 zu sehen ist.

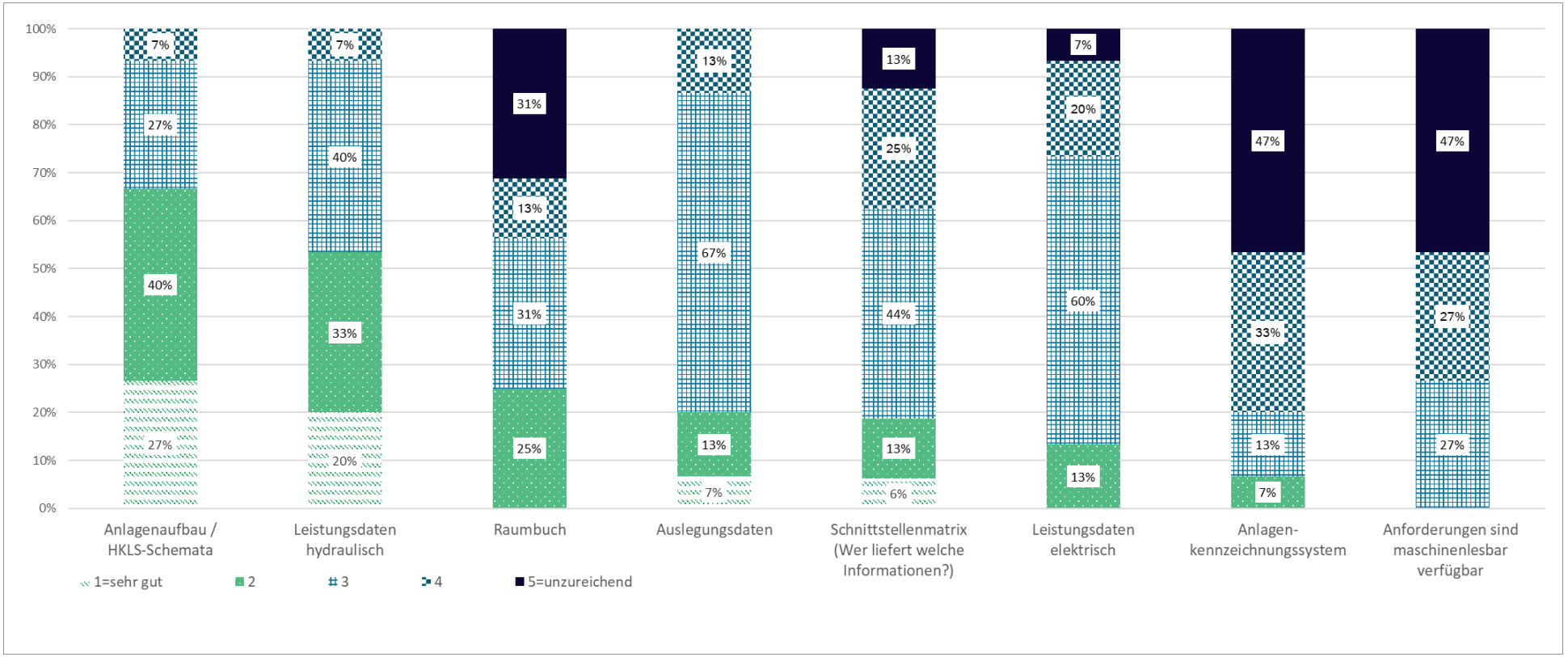
Tabelle 1: Austauschformate in der TGA Planung mit Anzahl der Rückmeldungen aus der Umfrage (ausgefüllte Beantwortungen: 16, Mehrfachauswahl möglich)

Austauschformat	Häufigkeit
.ifc	10
.dwg	6
.xml	5
.xls/.xlsx	4
.pdf	4
.onlv	2
.xmi	1
.dta	1
.csv	1
.dxf	1
.nwd	1
nativ	1
keine dieser Formate	1
keine Angabe	1

Da wir in unserem Projekt auf die Wichtigkeit der Verfügbarkeit und Übertragbarkeit von Daten aus der Designphase in weitere Lebenszyklusphasen aufzeigen wollten, wurde auch nach dem Fortschritt im Planungsstadium zum Zeitpunkt der Ausschreibung anhand bestimmter Aspekte gefragt. Die Rückmeldungen zeigen wie weit der Fortschritt in verschiedenen Bereichen ist (vgl. Abbildung 12). In Bezug auf den Anlagenaufbau / HKLS-Schemata gaben 67 % der Befragten an, dass dieser Aspekt bereits gut bis sehr gut vorangeschritten ist. Bei den hydraulischen Leistungsdaten verzeichneten 53 % der Befragten eine positive Einschätzung (gut bzw. sehr gut). Das Raumbuch wies hingegen einen deutlicheren Rückstand auf, da nur 25 % der Befragten den Fortschritt als gut bewerteten und

31% als unzureichend. Auch die Auslegungsdaten liegen mit nur 20 % der Befragten in den beiden besten Kategorie hinter den erstgenannten Aspekten zurück, allerdings gibt es hier keine Meldung von unzureichend. Die Schnittstellenmatrix (Wer liefert welche Informationen?) erhielt 19 % der Nennungen in den besten Kategorien und die elektrische Leistungsdaten wurden von 13 % der Befragten positiv bewertet. Das Anlagenkennzeichnungssystem erhielt eine überwiegend negative Bewertung, da 47 % der Befragten diesen Bereich als unzureichend einschätzten. Ähnlich wurde der Aspekt „Anforderungen sind maschinenlesbar verfügbar“ mit 47 % negativen Nennungen bewertet.

Abbildung 12: Wie weit ist üblicherweise der Fortschritt im Planungsstadium zum Zeitpunkt der Ausschreibung zu bestimmten Aspekten gegeben? (ausgefüllte Beantwortungen: 16)

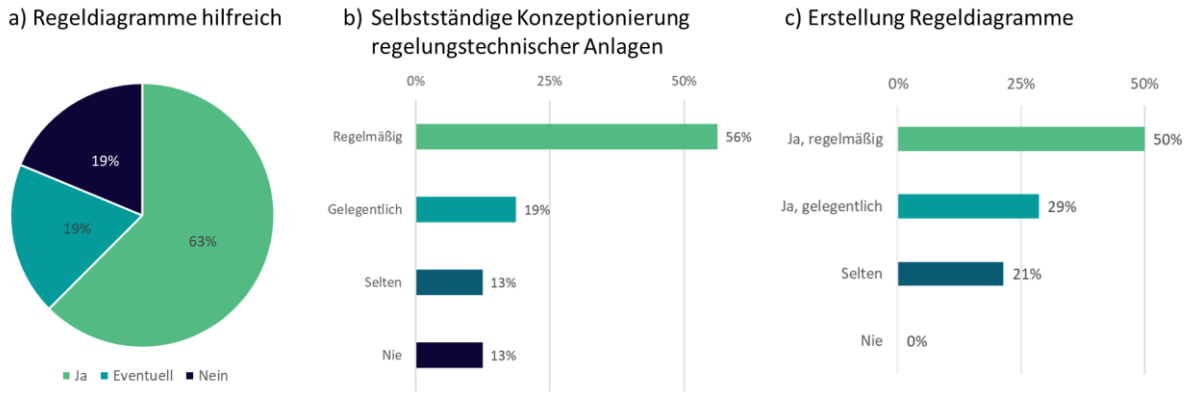


Insgesamt zeigt sich, dass der Fortschritt in vielen Aspekten noch nicht den gewünschten Stand bei der Ausschreibung erreicht hat und zwar insbesondere bei der maschinenlesbaren Verfügbarkeit von Anforderungen und der Kennzeichnungssystematik.

Für die Funktionen der MSR wird überwiegend Prosatexte (50 %) genutzt. Ebenfalls häufig werden Pflichten- und Lastenhefte (44 %), gefolgt von Funktionsplänen (38 %) und Anforderungsdokumenten (31 %) als Form genannt. Auch Zustandsdiagramme werden in 25 % der Fälle verwendet und Anweisungslisten (13 %). Darüber hinaus wurden unter „Sonstiges“ verschiedene weitere Ansätze genannt wie zum Beispiel eigene Erstellung der Funktionsbeschreibungen, mündliche Angaben, Anlagenschema, Regeldiagramm oder Verschiebung der die Erstellung auf die Werks- und Montageplanung. Insgesamt zeigt sich, dass die Dokumentation der Steuerungs- und Regelungslogik vielfach auf eine Mischung aus schriftlichen und schematischen Formaten zurückgreift, wobei Prosatext und Pflichtenhefte die am häufigsten verwendeten Formen darstellen.

Die Untersuchung der strukturierten Darstellung durch Regeldiagramme zeigte, dass 63 % der Befragten die Verwendung von Regeldiagrammen als hilfreich erachten (vgl. Abbildung 13 a). Weitere 19 % sehen dies eventuell als nützlich an, während ebenfalls 19 % der Meinung sind, dass Regeldiagramme nicht hilfreich wären. Bezüglich der Konzeptionierung von regelungstechnischen Anlagen (vgl. Abbildung 13 b) gaben 56 % der Befragten an, diese regelmäßig selbst durchzuführen, 19 % gelegentlich, 13 % selten und 13 % nie. Bei der Frage, ob dabei auch Regeldiagramme erstellt werden (vgl. Abbildung 13 c), gaben 50 % der Befragten an, dies regelmäßig zu tun, 29 % gelegentlich und 21 % selten.

Abbildung 13: Umfrageergebnisse zu Regeldiagrammen (ausgefüllte Beantwortungen: 16) a) Sind Regeldiagramme hilfreich, b) Führen Sie selbstständig Konzeptionierungen für regelungstechnische Anlagen durch, c) Erstellen Sie Regeldiagramme



Die Nutzung von Zustandsgraphendarstellungen nach VDI 3814 ist bei den Teilnehmer:innen eher gering. Die Darstellung wird eher gelegentlich bis nie genutzt. Wenn aber Zustandsgraphen erstellt werden, kommen verschiedene Tools wie zum Beispiel ein bestimmter Revit Aufsatz, WS CAD, MS Designer oder PowerPoint zum Einsatz. Die Beurteilung der Dokumentation von Anlagen Errichter fällt zwischen gut und genügend aus und zeigt daher eine gemischte Zufriedenheit. Prinzipiell wären mehr Unterlagen im Zusammenhang mit z.B. dem Schaltschrankaufbau, den Fremddatenpunktlisten, den Schnittstellenlisten, genaueren Funktionsbeschreibungen und detaillierter Beschreibungen der Regelfunktionen wünschenswert. Änderungen der Ausführung werden in der Regel nachgeführt, wobei es auch die Fälle gibt, wo diese nicht durchgeführt werden. Im Zusammenhang mit der

Automatisierung von Prüfverfahren im Planungsablauf (z.B., wenn sich ein Schema von 23 Volumenstromregler auf 26 ändert) gaben 47% an, dass dies möglich ist und 53% nicht, aufgrund von technischen oder organisatorischen Gründen.

Im letzten Abschnitt der Umfrage untersuchten wir inwieweit die Befragten auch dynamische Gebäudesimulationen im Rahmen der Planung durchführen. Hier war die mehrheitliche Rückmeldung, dass diese gelegentlich bis nie genutzt werden. Wenn sie allerdings zum Einsatz kommt, dann vorrangig für (i) die Konzeption von Funktionalitäten (Heizen, Kühlen, etc.), (ii) die Ermittlung, ob Anforderungen (z.B. an das Raumklima) bei verschiedenen Nutzungsszenarien eingehalten werden können, (iii) die Ermittlung des Energiebedarfs und (iv) die Auslegung von technischen Anlagen und Regelungen. Außerdem werden Planstandänderungen in den Simulationstools fast gänzlich manuell nachgezogen.

Zusammenfassend nennen die Befragten auch mehrere Aspekte, die zur Verbesserung der TGA- und MSR-Planung beitragen könnten. Eine wichtige Forderung ist die Integration von MSR als eigenständiges Gewerk, ähnlich wie in der Architektur, HKLS oder Elektrotechnik. Derzeit wird die Gebäudeautomation oft als nachgeordnetes Element behandelt, was zu einer mangelhaften Wahrnehmung und Planung führt. In diesem Zusammenhang wird auch der wirtschaftliche Faktor betont. Die Honorarordnung für MSR sollte überarbeitet werden, um die Bedeutung dieser Planungsebene widerzuspiegeln. Parallel dazu sollte die MSR-Planung eine höhere Wertschätzung sowohl bei den Planer:innen als auch bei den Ausführenden erhalten, um deren Rolle im Gesamtplanungsprozess zu unterstreichen. Eine bessere Kommunikation und stärkere Konnektivität zwischen Herstellerinformationen würde Planung und Ausführung effizienter gestalten. Zudem wird die Qualität der Ausschreibungsplanung als verbesserungswürdig angesehen, da MSR-Themen oft zu spät und unzureichend behandelt werden.

Es zeigt sich, dass durchaus standardisierte Dokumentationen, die maschinenlesbar sind, gewünscht werden. Um Abläufe zu optimieren und manuelle Nachbearbeitung zu minimieren, sind Schnittstellen ein entscheidender Faktor um einen reibungslosen Austausch von Informationen und Daten sicherzustellen. Besonders im Bereich der MSR-Planung könnte eine frühzeitige und standardisierte Abbildung bereits zu Beginn des Planungsprozesses Qualität und Gesamtplanung optimieren.

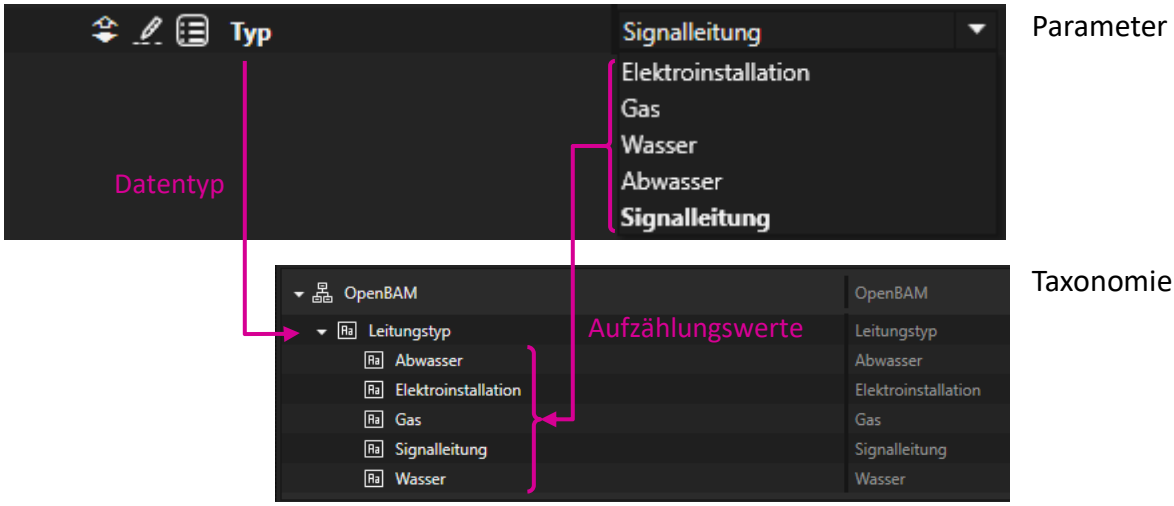
4.3. Integration in ein offenes Datenmodell

Ausgangspunkt für die Methodik und Integration der einzelnen Bestandteile ist das digitale Gebäudemodell (vgl. Abbildung 1). Basierend aus den Erkenntnissen zur Gebäudeautomation und den sich daraus ergebenden Anforderungen (vgl. Abschnitt 4.2) wurde das Meta Datenmodell Simultan (Simultan MDM) dahingehend untersucht, ob alle relevanten Daten der Gebäudeautomation abgelegt und verarbeitet werden können. Während der Entwicklung wurden zwei Bereiche identifiziert, in denen eine Erweiterung des Datenmodells notwendig wurde:

- Abbildung von vordefinierten Optionen für Parameter, Ganzzahlparameter, boolesche Parameter
- Darstellung von Netzwerken

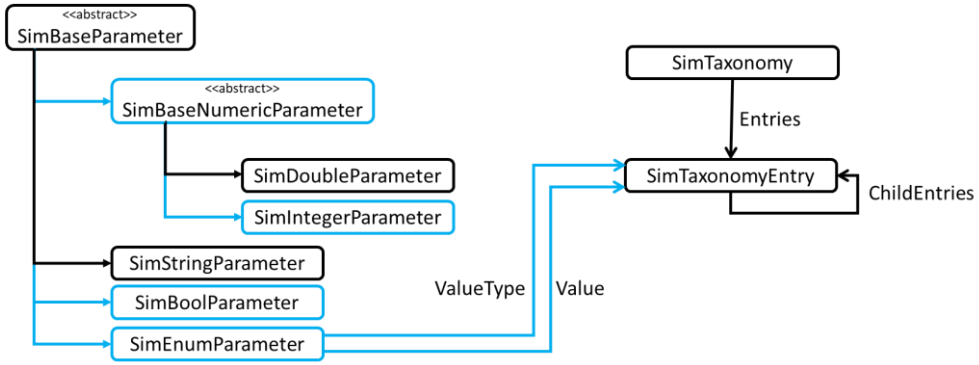
Bisher konnten nur Fließkommazahlen und Texte als Parameter hinterlegt werden, wobei für die Gebäudeautomation auch Parameter mit einer Liste an vordefinierten Optionen notwendig wurde, um eine Zuordnung von Begrifflichkeiten an die relevanten Normen zu garantieren und um die Benutzer:innen dabei zu unterstützen gültige Werte zu wählen. Das Simultan MDM unterstützte bereits Taxonomien [46] bei denen es sich um hierarchische angeordnete Begriffsgruppen handelt. Diese wurden als Basis für die Aufzählungswerte herangezogen, wobei eine Begriffsgruppe als Datentyp des Aufzählungsparameters definiert wird und alle untergeordneten Werte der Gruppe als Werte des Parameters zulässig sind. Abbildung 14 zeigt den Zusammenhang zwischen einer Taxonomie (DP Funktion) und einem Parameter im Simultan Editor. Der Parameter „Typ“ nutzt den Taxonomie Eintrag „Leitungstyp“ als Datentyp. Die untergeordneten Elemente „Abwasser“ bis „Wasser“ können somit als Wert des Parameters ausgewählt werden.

Abbildung 14: Zusammenhang zwischen Aufzählungsparameter und Taxonomie



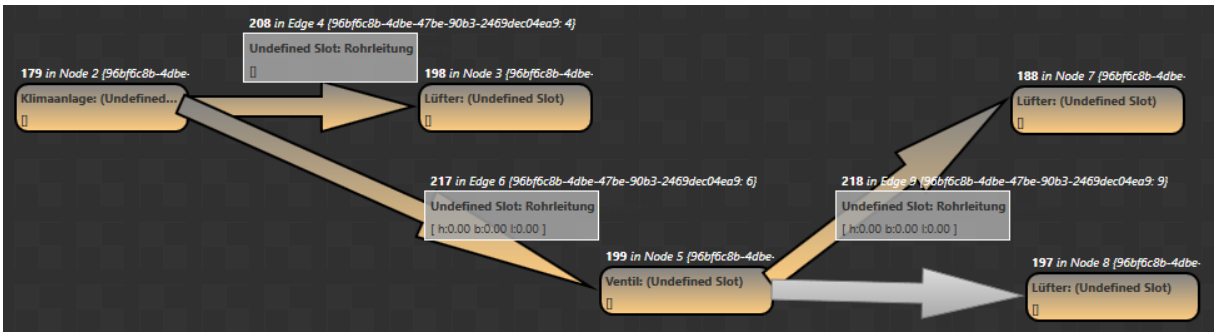
Weitere Parametertypen, die zur Abbildung aller benötigten Informationen notwendig waren, war die Umsetzung von Ganzzahlparameter und Parameter mit booleschen Werten (Wahr/Falsch). Für diese Änderungen musste das Simultan MDM angepasst und erweitert werden. Abbildung 15 zeigt die neuen Parametertypen im Datenmodell, wobei die entwickelten Ergänzungen in blau markiert sind.

Abbildung 15: Klassendiagramm der Parameter. Erweiterungen im Rahmen des Projekts sind blau markiert



Der zweite Bereich, in welchem Erweiterungen notwendig wurden, betrifft die Darstellung von Netzwerken an sich. Im Zuge von Vorprojekten [10] wurde bereits eine vereinfachte Netzwerkdarstellung zur Abbildung der Gebäudetechnik integriert, wobei die Netzwerke dabei aus Knoten und Kanten zusammengesetzt werden. Sowohl Knoten als auch Kanten konnten mit Komponenten verknüpft werden, welche durch ihre Parameter Eigenschaften des Netzwerkelements beschreiben. Abbildung 16 zeigt die ursprüngliche Realisierung.

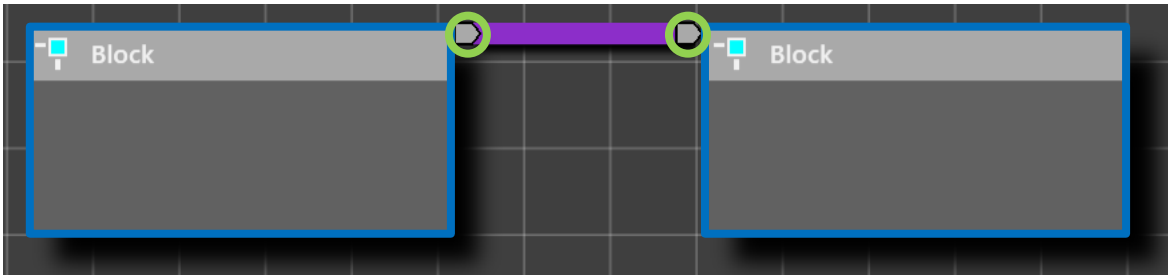
Abbildung 16: Sicht Simultan Netzwerke zum Projektstart



Diese Darstellung ist für eine detaillierte Darstellung der Gebäudetechnik und Automation zu limitierend. Im Speziellen war es zum Beispiel nicht möglich im Vorhinein die erlaubten oder erwarteten Schnittstellen an einem Block zu definieren. Auch die bestehende Verknüpfung mit einer geometrischen 3D Darstellung zur Verortung war nicht ausreichend detailliert, da Konten als Punkte und Kanten als Linien zwischen diesen Punkten dargestellt wurden. Diese Realisierung ermöglicht aber keine exakte Positionierung der Anschlüsse an den Knoten.

Um diese benötigten Anforderungen aber zu erfüllen, wurde die Netzwerkdarstellung komplett überarbeitet. Abbildung 17 zeigt die schematische Darstellung der Umsetzung. Als Grundelement wurden Blöcke (blau) gewählt, welche durch Anschlüsse (Ports, grün) ihre Schnittstellen definieren. Zwischen jeweils zwei Ports kann eine Verbindung (Connection, violett) erstellt werden. Diese Form der Darstellung ist im Bauwesen weit verbreitet und entspricht auch der Implementierung in IFC4.3.2[11].

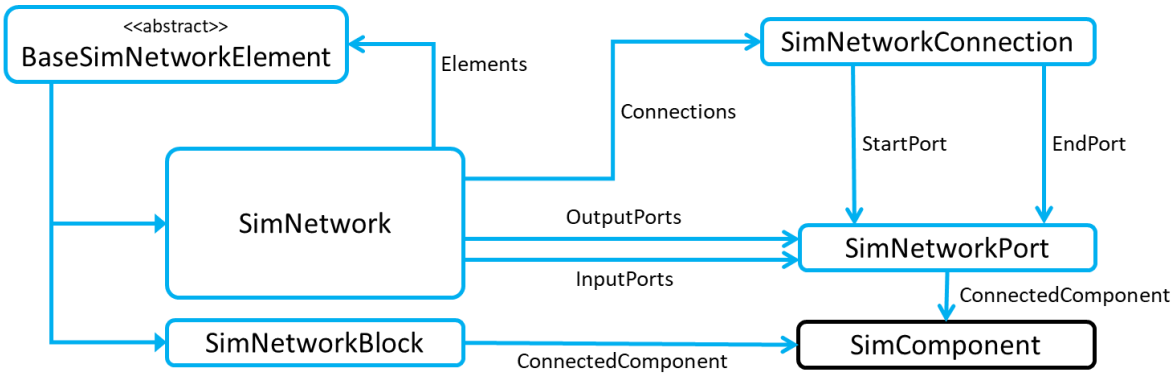
Abbildung 17: Grundelement der Netzwerke. Blöcke (blau), Anschlüsse (grün) und Verbindungen (violett)



Das zugehörige Klassendiagramm der Netzwerke ist in Abbildung 18 dargestellt.

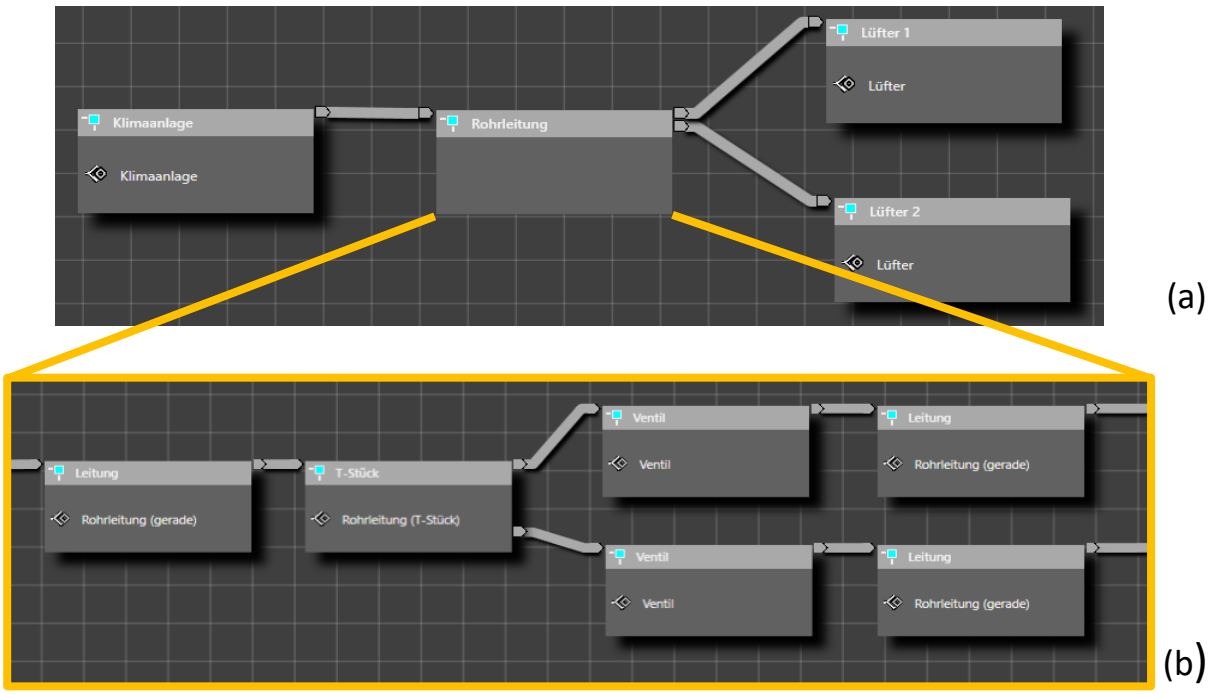
[11] https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcRelConnectsPorts.htm

Abbildung 18: Klassendiagramm der Netzwerke. Erweiterungen im Rahmen des Projekts sind in blau gekennzeichnet



Um einen iterativen Planungsprozess zu unterstützen können Blöcke jederzeit in eigene Sub-Netzwerke umgewandelt werden, in welchen dann wieder Unterblöcke und Verbindungen angelegt werden können. Dies ermöglicht Fachplaner:innen am Anfang der Planung nur eine grobe Struktur einzugeben und diese dann schrittweise zu verfeinern. Beispielsweise kann in einer frühen Planungsphase die Verbindung zwischen einem Klimagerät und zwei Lüftern mit einer abstrakten Leitungsführung eingegeben werden (Abbildung 19 a). In der Detailplanung kann dann der abstrakte Leitungsblock durch ein Sub-Netzwerk mit detaillierter Leitungsführung erweitert werden (Abbildung 19 b).

Abbildung 19: Hierarchische Darstellung von Simultan Netzwerken. Die Grobplanung (a) kann in der Detailplanung um ein Sub-Netzwerk mit konkreter Leitungsführung (b) erweitert werden



Die beschriebene Netzwerkstruktur in Simultan stellt eine rein strukturelle Beschreibung dar, ohne direkt Daten zu halten. Netzwerkelemente können ohne weitere Verknüpfungen existieren, oder sie können mit Komponenten des Simultan MDM verbunden werden, welche dann Daten für das Netzwerkelement halten. Komponenten können entweder mit Blöcken oder mit Anschlüssen im

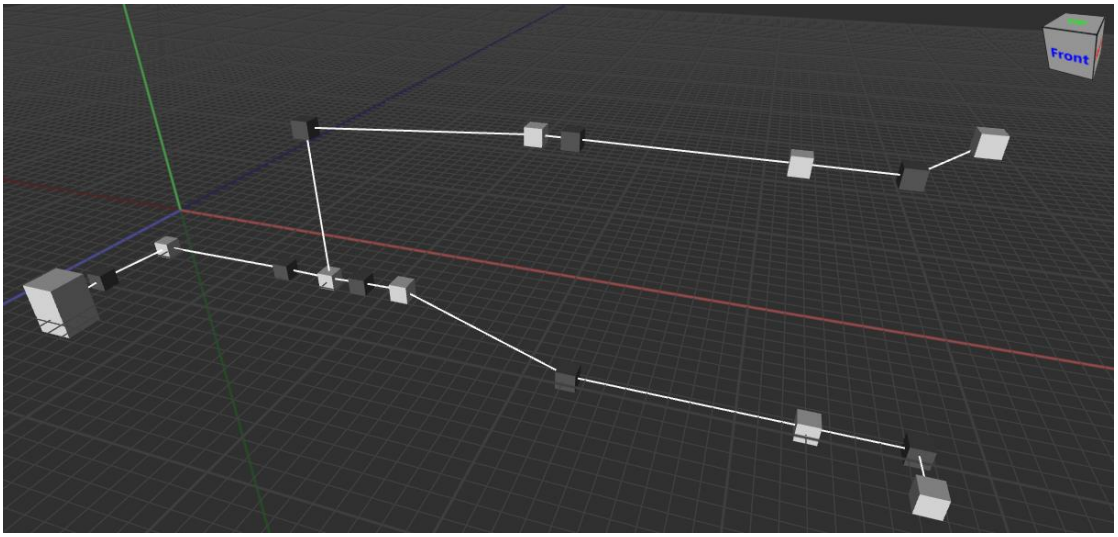
Netzwerk verknüpft werden. Komponenten für Anschlüsse werden als Sub-Komponenten in der Block Komponente hinterlegt, was es erlaubt die erwarteten Schnittstellen eines Blocks bereits in der Komponente zu fixieren. Wenn ein Netzwerkblock mit einer Komponente verknüpft wird, ist dies nur möglich, wenn sowohl der Block als auch die Komponente die gleiche Anzahl an eingehenden und ausgehenden Verbindungen aufweisen.

Netzwerke können auch mit einer geometrischen 3D Repräsentation verknüpft werden, welche die tatsächliche Lage der Netzwerkelemente in einem Gebäude darstellt. Zur Unterstützung der Anwender:innen kann beim Erstellen der 3D Geometrie direkt aus dem Netzwerk eine geometrische Repräsentation erzeugt werden. Diese kann dann von den Anwender:innen an die korrekte Position verschoben werden. Bei der Geometrieerstellung wird ein iterativer Algorithmus verwendet, der ausgehend von einem beliebigen Startblock alle verknüpften Blöcke in Geometrie konvertiert. Für jeden Netzwerkblock wird dabei ein Geometriewürfel erzeugt. Zur Positionierung der Anschlüsse relativ zum Block wurden zwei Varianten umgesetzt:

- Statische Blöcke haben eine vorgegebene Größe, welche in den Komponenten abgespeichert ist. Die Anschlüsse sind in einem definierten Abstand zum Zentrum des Blocks angeordnet. Dies ist vor allem hilfreich, wenn alle Verwendungen einer Komponente die gleiche Größe haben. Maschinen, wie ein spezifisches Modell eines Klimageräts, eignen sich für diese Repräsentation.
- Dynamische Blöcke beschreiben Objekte, bei denen jede Verwendung eine andere Dimensionierung haben kann. Diese Darstellung wird hauptsächlich für Leitungsführungen von z.B.: Elektroleitungen gewählt, da die einzelnen Drähte in beliebiger Länge verbaut werden können. Anschlüsse haben hier keinen fixen Abstand zum Zentrum des Blocks, sondern können vom User beliebig verortet werden.

Die Einschränkungen, die durch dynamische und statische Blöcke vorgegeben sind, werden auch vom Simultan Geometrie Editor berücksichtig. Es wird sichergestellt, dass bei Manipulationen der Geometrie die fixierten Abstände automatisch nachgeführt werden. Abbildung 20 zeigt ein Beispiel für eine Verortung des Netzwerks aus Abbildung 19 (inklusive Sub-Netzwerk) im Geometriemodell. Hellgraue Würfel stellen die Blöcke des Netzwerks dar und dunkelgraue Blöcke zeigen die Anschlusspunkte zwischen den Blöcken.

Abbildung 20: 3D Verortung eines Netzwerks



Alle Komponenten, ihre Parameter und das Gebäudeautomationsnetzwerk können mittels XMI Export einfach in eine gängige Austauschsprache konvertiert werden und somit von zum Beispiel UML basierten Werkzeugen weiterverarbeitet und genutzt werden.

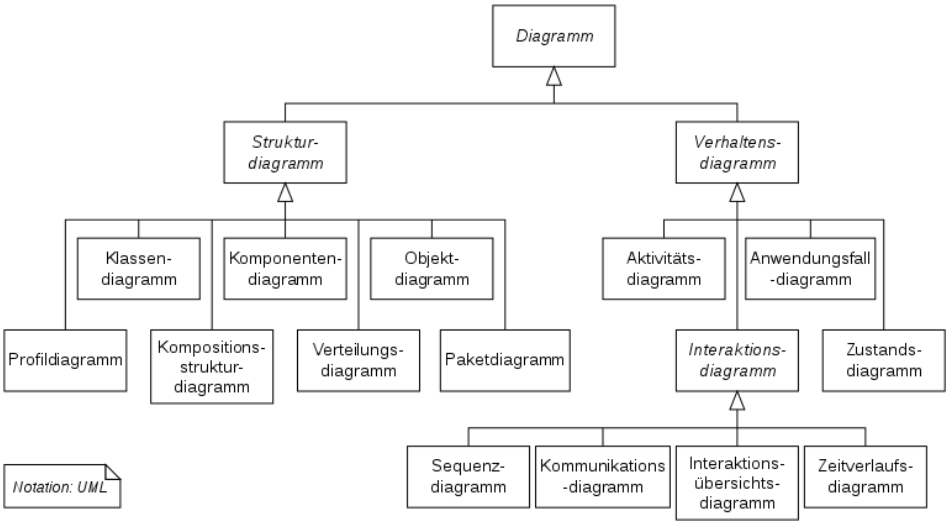
4.4. MSR-Modellierungssprache

Um eine Methode zur Abbildung der verschiedenen Steuerungsebenen in der Gebäudetechnik zu entwickeln, wurde untersucht, wie Logik in anderen Bereichen, wie zum Beispiel bei Speicherprogrammierbaren Steuerungen, abgebildet wird. Bei der Recherche wurden mehrere Ansätze zur Darstellung von Regelungslogik analysiert (vgl. Abschnitt 4.2.1). Die VDI-Richtlinie 3681 [44] bietet Konzepte zur Einordnung und Bewertung der verschiedenen Automatisierungstechniken, welche für die Untersuchung und den Vergleich der Vor- und Nachteile der genannten Technologien genutzt wurden.

Aufgrund dieser Kriterien und der in der VDI 3814-6:2008 [35] genannten Abbildungsmöglichkeit fiel die Wahl der Darstellung auf Zustandsgraphen. Zur Visualisierung werden in der Literatur und Praxis insbesondere zwei Ansätze (UML & SCXML), die bereits im Abschnitt 4.2.1 näher erläutert wurden, genannt. Aufgrund der Verfügbarkeit von Open-Source-Tools für UML wurde diese Sprache als MSR-Modellierungssprache gewählt, wobei auch eine Transformation von UML nach SCXML entwickelt wurde, sodass die Wahl der Darstellung im endgültigen Prozess keine entscheidende Rolle spielt. UML erlaubt darüber hinaus die einheitliche Modellierung von Datenpunkten, Parametern und anderen Metadaten, die für die Modellierung der MSR Logik benötigt werden.

Bei UML handelt es sich um eine von der OMG entwickelten graphischen Modellierungssprache, welche von der ISO unter der Nummer 19505 standardisiert wurde. Sie bietet eine Vielzahl an Diagrammen zur Beschreibung der Struktur und auch des Verhaltens von Systemen, welche in Abbildung 21 ersichtlich sind.

Abbildung 21: Verschiedene Diagrammarten der Modellierungssprache UML



Ein UML Zustandsdiagramm, welches aus einem *Startknoten*, mehreren *Zustandsknoten*, die über *Transitionen* verbunden sind, und optional aus einem oder mehreren *Endknoten* besteht, kann

genutzt werden um Verhalten von Systemen abzubilden. Mittels *Regionen* kann das Diagramm weiter strukturiert werden, sodass der Überblick nicht verloren geht. Außerdem ist es möglich die parallele Abarbeitung zweier Regionen über *Forks* und *Joins* zu realisieren, sowie die Auswahl über *Choice Knoten*. Neben einem *Namen* können für *Zustände Aktivitäten* und *Invarianten* festgelegt werden. Bei *Transitionen* beschreiben *Trigger*, wann diese ausgelöst werden und somit das System von einem Zustand in einen neuen Zustand überführt wird. Ein Beispiel für ein einfaches Zustandsdiagramm zur Abbildung der MSR Logik findet sich dann in Abschnitt 5.2.

Um die Logik der MSR methodisch korrekt abzubilden, bietet UML nicht grundsätzlich alle Möglichkeiten. Allerdgins besitzt UML verschiedene Erweiterungskonzepte, wovon eines das Konzept der Stereotypen ist, womit Elemente aus dem UML-Metamodell erweitert werden. Eine Sammlung solcher Stereotypen kann unter anderem in einem Profil zusammengefasst und in bestehende Modelle eingebunden und genutzt werden [45]. Im Folgenden werden nun die Details der UML-Darstellung näher erläutert. Die Datenpunktlisten der VDI 3814 beschreiben alle Datenpunkte, die zum Messen, Steuern und Regeln der Anlage verwendet werden können. Um eine einheitliche Darstellung dieser Daten in UML zu gewährleisten, muss sich in den Zustandsdiagrammbeschreibungen eine Auflistung aller verwendeter „Signale“ befinden. Hierfür eignet sich das Konzept der *Enumerations*. Mittels *Stereotypen* wurde zur Abbildung ein spezielles *EnumerationLiteral* eingeführt. Dieses stellt neben der eindeutigen Signal-ID/Datenpunkt-ID, noch den jeweiligen Datentyp, sowie ein Feld zur Beschreibung zur Verfügung. In den jeweiligen Aktionen kann dann auf diese Signale referenziert werden. Neben den Signalen können auch *Aktivitäten* für *Zustände* festgelegt und beispielweise über das Element *FunctionBehavior* beschrieben werden. In dem entwickelten Erweiterungsprofil wurden nun zwei *Stereotypes* der Metaklasse *FunctionBehavior* abgeleitet. Erstens wurde die *SubFunctionBehavior* abgeleitet, welche ermöglicht, dass beliebig viele weitere *FunctionBehavior* inkludiert werden können und zweitens die *WriteSignalFunctionBehavior*. Dieser Stereotype erlaubt die Zuordnung eines Wertes zu einem Signal, also das Schreiben eines Datenpunktes. Während die *SubFunctionBehavior* und die *WriteSignalFunctionBehavior* grundlegende Mechanismen zur Erweiterung und Datenzuordnung bieten, spielen *Transitionen* eine zentrale Rolle im Ablauf, da sie durch *Trigger* ausgelöst werden, die wiederum durch das Auslösen eines Events gestartet werden. So kann zum Beispiel im Falle eines Timers auf in UML definierte *TimeEvents* zurückgegriffen werden. Um jedoch auf Datenpunktänderungen reagieren zu können, wurde im entwickelten UML-Profil ein Stereotype *ReadSignalChangeEvent* der Metaklasse *ChangeEvent* abgeleitet. Dieser Stereotype erweitert die Metaklasse um das Signal sowie einen Vergleichswert. Abbildung 22 zeigt die im UML Profil erzeugten Stereotypen.

Abbildung 22: UML Profile für den Zugriff und die Darstellung der Regelungssignale (vgl. [1])

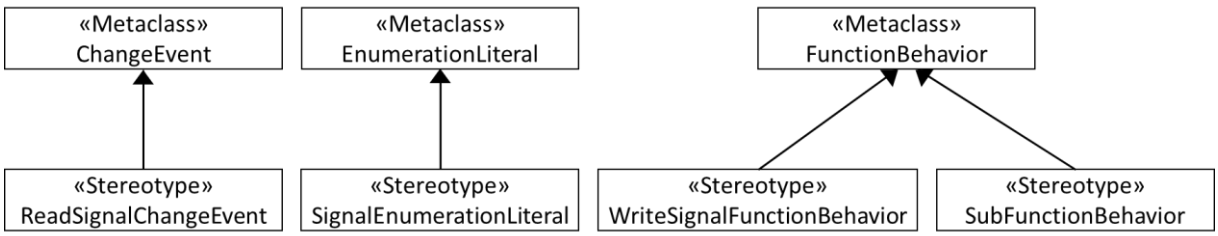


Tabelle 2: Übersicht über die entwickelten Stereotypen

Stereotyp	Metaklasse	Funktion
SignalEnumerationLiteral	EnumerationLiteral	Speicherung eines im Gebäudemodell definierten Signals, einschließlich seiner ID, seines Datentyps und anderer Metadaten
ReadSignalChangeEvent	ChangeEvent	Erweiterung der Definition von Übergangsbedingungen: Ein Signal (signalId) kann gelesen und mit einem Wert verglichen werden
WriteSignalFunctionBehavior	FuctionBehavior	Erweiterung der Aktionen (z.B. beim Erreichen eines neuen Zustands): Bereitstellung des Signals und welcher Wert geschrieben werden soll
SubFunctionBehavior	FunctionBehavior	Erweiterung von Aktionen von Zuständen: Zustände können eine beliebige Anzahl von FunctionBehaviors aufnehmen (standardmäßig nur eine FunctionBehavior in UML möglich)

Tabelle 2 zeigt nochmals einen Überblick der benötigten entwickelten Stereotypen mit ihren jeweiligen Funktionen. Auf Basis dessen ist es nun möglich mittels UML Zustandsdiagramm die MSR Logik plattformunabhängig zu beschreiben. Dafür wurde ein Template entwickelt, das als Grundlage eine UML-State-Maschine mit nur einem Zustand, dem Startzustand, enthält. Das entwickelte UML-Profil wird genutzt um die modellierten Signale (vgl. Abschnitt 4.3) zu lesen, zu schreiben und miteinander vergleichen zu können. Das Template wird durch die im nachfolgenden Abschnitt 4.5 beschriebenen Transformationsmethoden aus dem Simultan XMI Export generiert und kann somit von Anwender:innen zur weiteren Bearbeitung genutzt werden.

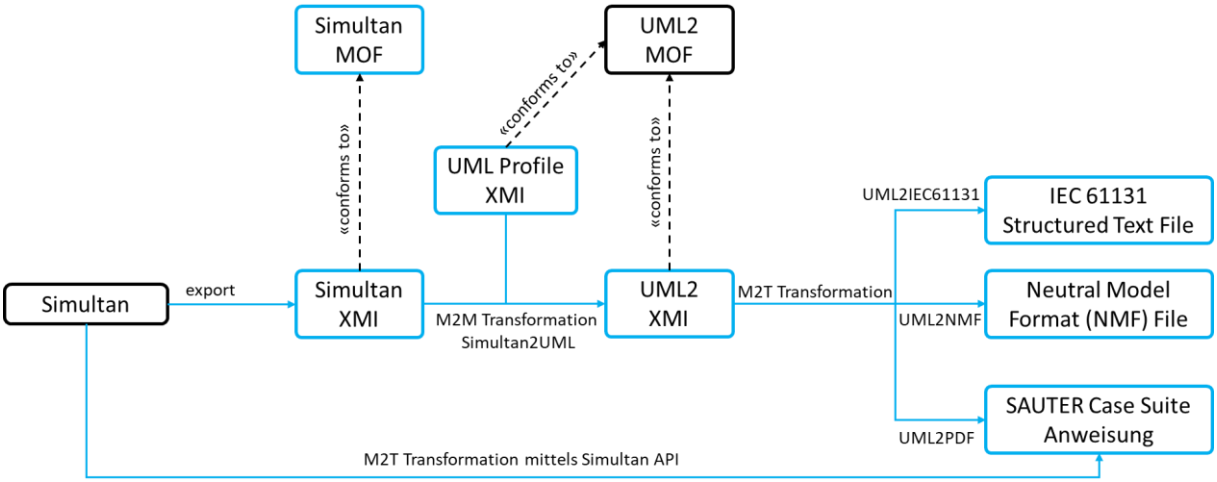
4.5. MSR-Transformationsmethoden

Ziel der entwickelten Methodik ist es die Gebäudeautomation mit der MSR-Logik in den digitalen Lebenszyklusprozess schon in frühen Phasen mit dem digitalen Gebäudemodell zu integrieren und weiter zu nutzen (vgl. Abbildung 1). Ausgehend von der offenen Beschreibung eines Gebäudemodells mit Hilfe des Simultan MDM werden Datenpunkte, Komponenten und Parameter beschrieben. Anschließend werden die Informationen exportiert und in eine „leere“ MSR-Darstellung, das sogenannte UML-Template (vgl. Abschnitt 4.4), transformiert. Nachdem ein Engineer die Logik in diesem Template abgebildet hat, wird diese Logik dem Gebäudemodell als Asset hinzugefügt und ist somit im digitalen Modell vorhanden und abgespeichert. Weiterführung dieses Modells beziehungsweise Transformationen ermöglichen es dann, die Logik in weiteren Applikationen wie zum Beispiel Simulationstools oder Ausführungstools zu nutzen. Durch die konsistente Verknüpfung

der Gebäudeinformationen mit der zugehörigen MSR-Logik kann das Modell über die gesamte Lebensdauer des Gebäudes aktuell gehalten werden.

Die entwickelten Transformationen ermöglichen die MSR-Beschreibung und das Gebäudemodell in das proprietäre Simulationstools IDA ICE und in eine Beschreibung für das Engineering-Tool SAUTER Case Suite überzuführen. Weiters wurde auch die Transformation in offene Engineering-Optionen wie die IEC 61131 Sprachen umgesetzt. Durch Anpassungen/Erweiterungen der entsprechenden Modelltransformationen können auch andere (proprietäre) Engineering- oder Simulationswerkzeuge unterstützt und die Informationen aus den digitalen Gebäudemodellen genutzt werden.

Abbildung 23: Ablauf der einzelnen Transformationsschritte (in blau markiert entwickelte Elemente)



In Abbildung 23 ist der Ablauf der einzelnen Transformationsschritte ersichtlich, wobei die entwickelten Teile in blau markiert. Simultan ermöglicht den Export des erstellten Datenmodells im XMI-Format. Ein zugehöriges Schema (Simultan MOF) beschreibt die Struktur des Modells. Die mittels ATL definierte M2M-Transformation Simultan2UML erstellt aus diesen Informationen zusammen mit dem in Abschnitt 4.4 beschriebenen UML-Profil das UML-Template, welches mit den aus Simultan gewonnenen Metadaten angereichert ist. Der Simultan2UML-Quellcode ist auf GitHub[12] zu finden. Im Wesentlichen funktioniert die Transformation wie folgt: Zunächst wird ein Zustandsautomat mit einer Region, einem Startknoten und einer leeren Aufzählung erzeugt und das UML-Profil importiert. Im nächsten Schritt werden für alle Steuerungskomponenten aus SIMULTAN Aufzählungsliterale mit den entsprechenden im UML-Profil definierten Metadaten erzeugt. Diese stellen die zu verwendenden Datenpunkte dar. Mit Hilfe von Werkzeugen wie Eclipse Papyrus kann dann die Steuerlogik auf der Basis der verfügbaren Datenpunkte beschrieben und umgesetzt werden. Diese Beschreibung wird dann im digitalen Datenmodell zusammen mit den weiterführenden Transformationen als Asset gespeichert, sodass diese Informationen nun eng mit den Gebäudeinformationen verknüpft sind und in einem digitalen Modell vorliegen. Ausgehend von MSR Beschreibung, gibt es drei M2T-Transformationen, wobei UML2IEC61131 und UML2NMF mit Eclipse Accelio und UML2PDF direkt in Simultan als Plugin umgesetzt wurden. UML2IEC61131 erzeugt aus dem UML-Modell eine Applikation in strukturiertem Text (ST) und ermöglicht damit ein Deployment [31]. Diese Transformation erzeugt einen Zustandsautomaten, der in einem Function Block (FB) im strukturierten Text der IEC 61131-3 gekapselt ist. Die Transformation beinhaltet die nachfolgenden

[12] https://github.com/fxknorr/consistent-control-logic-design

Schritte. Funktionen (in Acceleo „Helpers“ genannt) extrahieren die entsprechenden Daten aus der UML-Datei. Nach der Definition der Ein- und Ausgänge wird für jeden Knoten ein Zustand erzeugt. In diesen Zustand werden die Datenpunkte entsprechend der UML-Beschreibung geschrieben. Für jede ausgehende Kante wird dann der Übergang zu den entsprechenden Zuständen erzeugt. Dabei wird zwischen der Startkante, d.h. der Kante vom Startknoten zum ersten Zustand, Änderungsereignissen und Zeitereignissen unterschieden. Während Änderungsereignisse voraussetzen, dass sich ein Eingangsdatenpunkt ändert, bevor der Zustand gewechselt wird, tritt ein Zeitereignis nach einer bestimmten Zeitspanne ein, die ebenfalls im UML-Modell definiert ist. Ein Auszug der Transformationen nach IEC 61131 ist in Abbildung 24 abgebildet.

Abbildung 24: M2T Transformation UML2IEC61131

```
[template public generateElement(aModel : uml::Model)]
[file (aModel.name + '.st', false, 'UTF-8')]

[generateEnum(stateMachine(aModel)) /]

FUNCTION_BLOCK PUBLIC [stateMachine(aModel).name/]

VAR_INPUT
    [inputs(aModel)/];
END_VAR
VAR_OUTPUT
    [outputs(aModel)/];
END_VAR
VAR
    eState : E_States := 0;
    wait : TON;
END_VAR

CASE eState OF
[for (v: Vertex | states(stateMachine(aModel)))]
    E_States.[v.name/]:
        [for (fb: FunctionBehavior | writeBehaviours(v))]
            [fbVariable(fb)/]:= [fbValue(fb)/];
        [/for]
        [for (t: Transition | outgoingTransitions(aModel, v))]
            [if (t.trigger->isEmpty())]
            eState := E_States.[t.target.name/];
            [elseif (isChangeEvent(t))]

            IF ([tVariable(t)/] = [tValue(t)/]) THEN
                eState := E_States.[t.target.name/];
            END_IF
            [elseif (isTimeEvent(t))]

            wait(IN:=TRUE, PT:=T#[getTimeEventTime(t)/]s);

            IF wait.Q THEN
                wait(IN:=FALSE);
                eState := E_States.[t.target.name/];
            END_IF
            [/if]
        [/for]
[/for]
END_CASE
[/file]
```

UML2NMF erzeugt einen Funktionsblockanweisung, die für das Tool IDA ICE genutzt werden kann und somit eine Simulation erlaubt [31]. Da diese Transformation große Ähnlichkeit zu der in Abbildung 24 gezeigten hat, wird nicht näher auf sie eingegangen.

Die Transformation UML2PDF nutzt die Informationen aus Simultan und des Logikfiles um für das proprietäre Tool SAUTER Case Suite Anweisungen zu erstellen, die der Engineer umsetzen kann. Da diese Transformation, wie bereits erwähnt, als eigenständiges Plugin umgesetzt wurde, wird sie näher in Abschnitt 4.6 beschrieben.

Der XMI-Export des Datenmodells (Simultan XMI) exportiert, wie bereits erwähnt, die Komponenten mit ihren Attributen und eindeutigen IDs. Dadurch bleibt beim Austausch die Verknüpfung zur konkreten Instanz erhalten. Außerdem wurden Parameter, Datentypen und Verknüpfungen soweit erweitert (vgl. Abschnitt 4.3), sodass alle für die Abbildung beispielsweise einer Lüftungsanlage notwendigen Informationen gespeichert werden können.

4.6. Transformation in Richtung Ausführung

Dieser Abschnitt befasst sich mit der Komplexität des Datentransfers vom offenen Datenmodell von Simultan zu den proprietären Engineering-Tool SAUTER Case Suite. Die SAUTER Case Suite ist ein umfassendes Software-Ökosystem, das für die Planung von Gebäudeautomationsprojekten, insbesondere für Heizungs-, Lüftungs- und Klimaanlagen (HLK), weit verbreitet ist.

Das Hauptziel dieses Teils war die Entwicklung eines Softwarepakets, das in der Lage ist, Gebäudeautomationsmodelle aus dem Simultan-Datenmodell in Richtung SAUTER zu übertragen. Auf diese Weise können Ingenieure effizient mit der Implementierung von HLK-Systemen beginnen, indem sie detaillierte bereits modellierte Modelle verwenden.

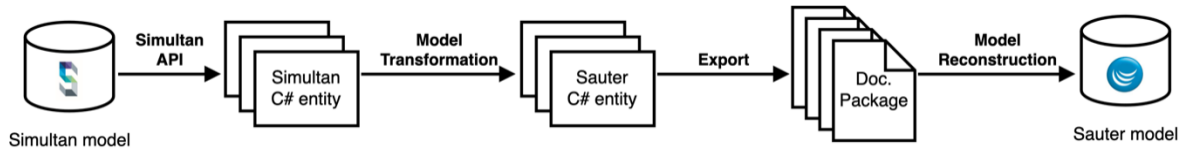
Dabei wurden folgende Schritte umgesetzt:

- Umwandlung des plattformunabhängigen Simultan-Modells in ein SAUTER-spezifisches Format, das mit der Case Suite kompatibel ist.
- Implementierung einer Umsetzung, die die Lücke zwischen dem offenen Datenmodell und der proprietären Entwicklungsumgebung überbrückt.

Bei der Erarbeitung wurde festgestellt, dass die SAUTER Case Suite nicht über ausreichende Importmöglichkeiten für extern generierte Daten, die interne Prozesse ersetzen würden, verfügt. Dies liegt daran, dass der Datenfluss innerhalb des Ökosystems in sich geschlossen ist. Der typische Arbeitsablauf beginnt mit dem Case Builder, wo Modelle von Grund auf erstellt werden, und geht dann zur weiteren Entwicklung und Implementierung in die Case Engine über. Das Nachbauen der proprietären Dateien war keine Option, da dazu zu viele firmeninterne Spezifikationen darin vorhanden sind, die aber nicht für die Modellerstellung und die Projektziele zielführend sind. Deshalb wurde auf die direkte Modellumwandlung verzichtet und ein alternativer Ansatz gewählt. Es wurde ein umfassendes Dokumentationspaket automatisch generiert, das die Ingenieure bei der manuellen Neuerstellung des Modells innerhalb der SAUTER-Konstruktionswerkzeuge optimal unterstützt. Obwohl diese Methode für den Datentransfer auf die Ingenieure angewiesen ist, überbrückt sie die Lücke effektiv, indem sie detaillierte, strukturierte Informationen bereitstellt. Abbildung 25 zeigt den

entwickelten Ablauf um die UML2PDF Transformation inklusive aller notwendigen Komponenten aus dem digitalen Modell in die Anweisung zu überführen.

Abbildung 25: Datenfluss vom plattformunabhängigen Simultan-Datenmodell zum plattformspezifischen SAUTER-Modell



Der Proof-of-Concept wurde in der Programmiersprache C# entwickelt, unter Verwendung der offenen Datenmodell-API von Simultan, die ebenfalls in C# verfügbar ist. Die Untersuchungen der Datenstrukturen des Case-Software-Ökosystems von SAUTER lieferte wertvolle Erkenntnisse darüber, wie bestimmte Komponenten des Modells dargestellt werden. Dies ermöglichte eine Eins-zu-Eins-Darstellung der Teile des SAUTER-spezifischen Modells in C# zu definieren, welche als Bindeglied zwischen dem plattformunabhängigen Modell und dem proprietären plattformspezifischen Modell dient. Diese wird dann in ein für Menschen lesbares Format exportiert. Die Methodik wurde in drei Hauptmodule unterteilt:

1. Datenzugriffsmodul: Dieses Modul bildet über seine API eine Schnittstelle zum plattformunabhängigen Simultan-Modell und extrahiert die notwendigen Datenelemente, die für das werkzeugspezifische Modell benötigt werden.

2. Modul zur Modelltransformation: Hier werden die extrahierten Daten in die zwischengeschaltete C#-Darstellung transformiert, die die Struktur des SAUTER-spezifischen Modells widerspiegelt.

3. UML2PDF Transformation: Dieses Modul übernimmt die Visualisierung und den Export der C#-Darstellung in menschenlesbare Formate, die im Dokumentationspaket enthalten sind.

Die Umsetzung erfolgte mittels Simultan Component Builder Plugin, welches den Simultan Component Builder erweitert. Es fügt eine neue Registerkarte und Elemente der Benutzeroberfläche hinzu, die es dem Benutzer ermöglichen, einzelne Teile oder das gesamte Dokumentationspaket auf einfache Weise zu exportieren. Es bietet eine benutzerfreundliche Schnittstelle für diejenigen, die bereits mit dem Simultan-Ökosystem arbeiten. Darüber hinaus wurde auch eine eigenständige Anwendung, die direkt über das Terminal ausgeführt werden kann, umgesetzt. Es bietet Kommandozeilenzugriff auf die Funktionen der Software und kann somit in automatisierte Arbeitsabläufe integriert werden.

Durch die Entwicklung des umfassenden Dokumentationspakets kann die Lücke zwischen dem offenen Datenmodell von Simultan und der proprietären Entwicklungsumgebung von SAUTER geschlossen werden. Obwohl eine direkte Modelltransformation aufgrund der proprietären Beschränkungen nicht möglich ist, stellt der Ansatz den Ingenieuren die notwendigen Ressourcen zur Verfügung, um die Modelle innerhalb der SAUTER-Software abzubilden.

5 Ergebnisse

5.1. Umsetzung

Die Methodik und einzelnen Schritte auf dem Weg zu einer technologieunabhängigen Logikbeschreibung bis hin zu plattformabhängige Lösungen wurden schon ausführlich in Kapitel 4 dargestellt. Für den Proof-of-Concept und die Überprüfung der Methoden wurde zuerst ein einfaches Beispiel (vgl. Abschnitt 5.2) zur Logiküberprüfung herangezogen. Im Lüftungsanlagen Use Case im Labor (vgl. Abschnitt 5.3) wurde dann der gesamte Workflow evaluiert.

Die in Abbildung 1 einzelnen dargestellten Abschnitte wurden mit C# (Digitales Gebäudemodell: Simultan MDM, User Interfaces, Plugin für die Transformation in Richtung Ausführungstool), Eclipse Modeling Framework, ATL, Accelio und Papyrus (openBAM Editor: UML Profil, Zustandsgraph Template für MSR Logik, Transformation in Richtung nmf für die Simulation und IEC61131) entwickelt. Als Simulationstool wurde IDA ICE genutzt und für die Ausführung wurde SAUTER Case Suite herangezogen.

Ein offenes Gebäudemodell bildet die Grundlage für diesen Prozess. Das Modell umfasst neben der Gebäudestruktur auch Systeme und Geräte, wie z.B. die HLK-Anlage mit ihren Komponenten und den verfügbaren Datenpunkten, Parametern und anderen Metainformationen. Dies sind alle Informationen, die der Modellingenieur benötigt. Nachdem der Modellingenieur die Steuerungslogik in einem offenen Beschreibungsformat entworfen hat, wird sie mit Hilfe einer Modelltransformation in das Gebäudemodell eingebettet. In diesem Fall erfolgt die Beschreibung der Steuerungslogik über einen Zustandsgraphen, wie in der VDI 3814 vorgeschlagen. Ein Export (Modelltransformation) erlaubt es dann, die Anwendung erneut zu bearbeiten und einzubetten. Dabei werden Abhängigkeiten und Inkonsistenzen aktualisiert. Dies ist der Fall, wenn neue Datenpunkte hinzukommen, aber auch wenn Geräte ersetzt oder verändert werden. Ein Beispiel hierfür könnte ein Ventilator sein, der zunächst mit drei diskreten Geschwindigkeitsmodi und später mit einem kontinuierlichen Geschwindigkeitssignal modelliert wurde. In diesem Fall wird der Datenpunkt für die diskreten Geschwindigkeitsmodi entfernt und ein neuer Datenpunkt für das kontinuierliche Geschwindigkeitssignal hinzugefügt. Außerdem hat der neue Datenpunkt einen anderen Datentyp als der alte. Die Änderungen müssen, wenn sie nicht automatisch korrigiert werden, vom Modellierer nachgepflegt werden. Die geänderte Anwendung wird dann wieder in das Gebäudemodell integriert.

5.2. Beispielanwendung zur Methodenprüfung

In diesem Abschnitt wird ein einfacher Anwendungsfall zur Evaluierung des vorgestellten Ansatzes beschrieben. Zu diesem Zweck wird ein vereinfachtes HLK-System verwendet, das in der VDI 3814 vorgestellt wird und in Abbildung 3 beschrieben wurde. In diesem Beispiel besteht das System aus fünf Datenpunkten, wobei der Systemzustand (Aus/Ein) ein Eingang und die Klappen (Aus/Vorheizen/Regelung), die Vorheizpumpe und das Vorheizventil (Aus/Vorheizen/Regelung), sowie der Ventilator (Aus/Ein) Ausgänge sind. Alle diese Datenpunkte werden als Ganzzahlen modelliert. Zu Beginn befindet sich das System im Zustand „System AUS“, in dem alle Ausgänge auf Aus (0) gesetzt sind. Wenn die Übergangsbedingung von Zustand 1 (dem Nachfolgezustand von

Zustand 0 mit der Bezeichnung „Start-up-Modus“) erfüllt ist, nämlich dass das System eingeschaltet ist (Systemzustand = 1), wechselt das System in diesen Zustand. Zusätzlich werden verschiedene Aktionen durchgeführt, wie das Vorheizen der Klappen und das Setzen eines Timers t1. Nach Ablauf dieses Timers schaltet das System in den „Normalbetrieb“ bis das System wieder ausgeschaltet wird (Systemzustand = 0). Im „Normalbetrieb“ werden alle Ausgangsdatenpunkte auf Ein/Regelbetrieb gesetzt. Der Kreiszustand am unteren Rand des Diagramms symbolisiert den bestehenden Zustand 0 („System OFF“). Abbildung 26 zeigt diese vereinfachte beispielhafte Abbildung einer Lüftungsanlage im openBAM Editor, wobei die drei *States* ‘SystemOff’, ‘StartUpMode’ und ‘NormalMode’ die Zustände des Systems beschreiben. Initial befindet sich das System im Zustand ‘SystemOff’, modelliert mittels Startknoten und zugehöriger Transition. Die drei Zustände besitzen als Aktivität jeweils ein *SubFunctionBehavior,* welches sich wiederum aus vier WriteSignalFunctionBehaviors zusammensetzt. In diesem speziellen Fall werden in jedem der Zustände die gleichen Signale auf unterschiedliche Werte gesetzt, wie Tabelle 3 entnommen werden kann.

Abbildung 26: Beispielhafte Lüftungssteuerung laut VDI 3814-6:2008 [35] modelliert in UML

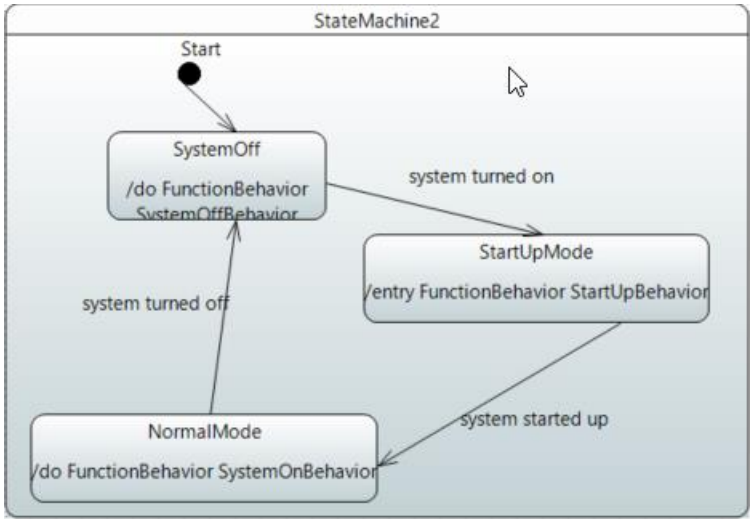


Tabelle 3: Aktivitäten in den Zuständen der beispielhaften Lüftungsanlage

WriteSignalFunction-Behaviour	Signal_id	Wert (SystemOff)	Wert (StartUpMode)	Wert (NormalMode)
WritePreheaterValve	PreheaterValve	0	1	2
WritePreheaterPumps	PreheaterPump	0	1	1
WriteFans	Fans	0	0	1
WriteDampers	Dampers	0	0	1

Bei den Transitionen ist zwischen „system turned on/off“ und „system started up“ zu unterscheiden. Erstere besitzen einen Trigger mit einem *ReadSignalChangeEvent* als Auslöser, bei dem geprüft wird, ob das Signal „SystemState“ auf 1 oder 0 gesetzt wird. Der Trigger der „system started up“-Transition auf der anderen Seite wird durch ein *TimeEvent* ausgelöst, welches überprüft ob 5 Sekunden vergangen sind.

Als erster Proof-of-Concept wurde ein Beispielgebäude in SIMULTAN modelliert, welches die in Abbildung 26 genannten Komponenten enthält. Darüber hinaus wurde der vorgestellte Anwendungsfall in der von der M2M-Transformation „Simultan2UML“ generierten UML-Vorlage implementiert. Die Modellierungssprache und die Transformationen wurden dahingehend überprüft, dass sowohl die IDA ICE-Simulation lauffähig ist, als auch der in IEC 61131 erstellte strukturierte Text kompiliert werden kann. Im Hinblick auf die Abbildung von plattformunabhängigen Modellen in plattformspezifische Beschreibungen mit einem Fokus auf Hardware-Details wurden erste Überlegungen angestellt, wie die im Datenmodell verwendeten Methoden zur Darstellung von Hardware-Informationen in Richtung SAUTER Case Suite transformiert werden können.

5.3. Evaluierung anhand eines Use Cases im Labor

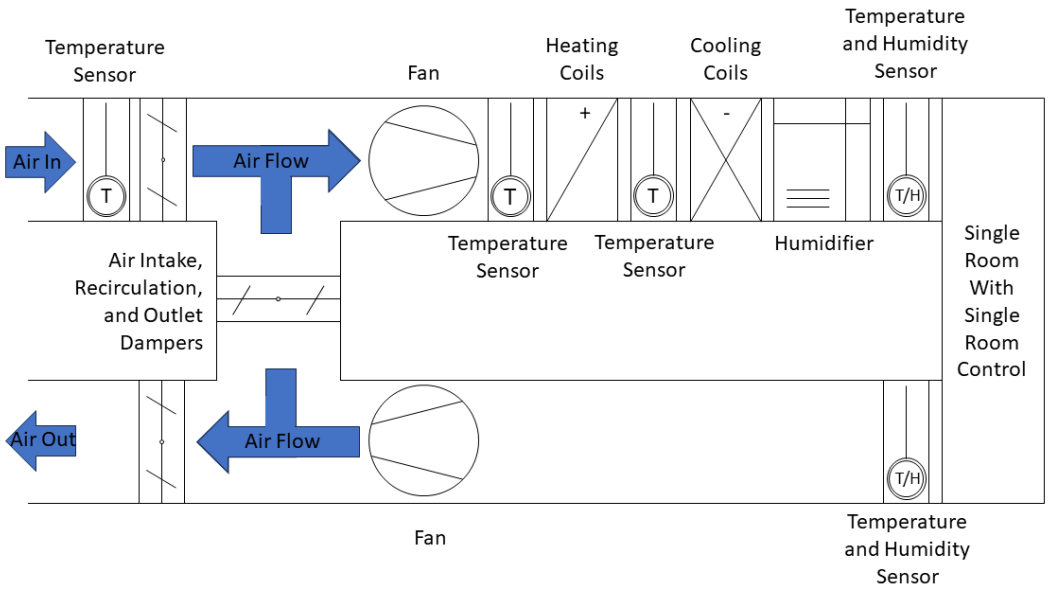
Zur Evaluierung wurde der Laborversuchsaufbau des Forschungsbereichs Automation Systems der TU Wien herangezogen. Es handelt sich beim Laborversuchsaufbau um eine Lüftungsanlage, an der das Modell eines Testraumes angeschlossen ist, der mittels der Anlage belüftet, gekühlt und beheizt werden kann. Darüber hinaus befindet sich in dem Testraum selbst ein Fan Coil, mit dem Raum Wärme zugeführt oder entzogen werden kann – hiermit kann simuliert werden, dass Wärme aus dem Raum abfließt (z.B. aufgrund der Außenwitterung während der kalten Jahreszeit) oder Wärme in den Raum eingebracht wird (z.B. durch innere Lasten). Zur Durchführung von Übungen und Laborversuchen ist der Versuchsaufbau mit mehreren Automationsstationen von unterschiedlichen Herstellern ausgestattet und wurde in diesem Projekt mit Komponenten von SAUTER ausgestattet, wie in Abbildung 27 zu sehen ist.

Abbildung 27: Use Case Lüftungsanlage im Labor

Ziel der Evaluierung ist die Modellierung des Use-Cases mitsamt seiner MSR-Logik im SIMULTAN-Datenmodell und der Übertrag der im Datenmodell gespeicherten Informationen an das

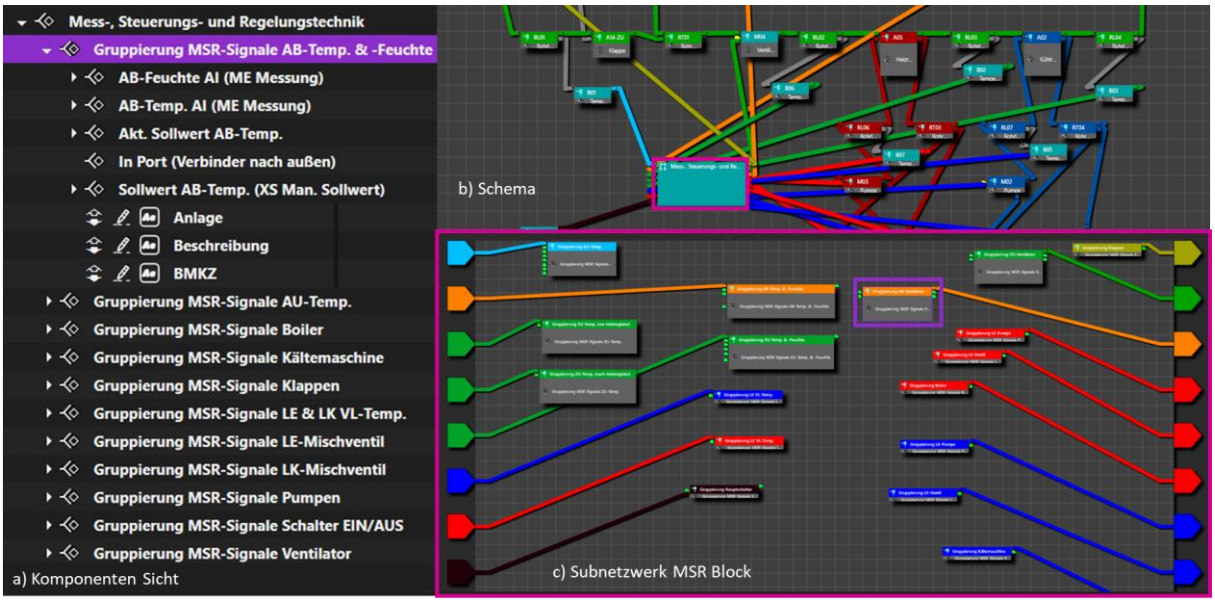
Simulationstool IDA ICE und an das Engineering-Tool von SAUTER. Da die Temperatur im Testraum und etwaige auf den Testraum wirkende Wärmeströme einfach in Form von Zeitreihen festgelegt werden können, wurde beschlossen, dass der Fan Coil im Testraum und seine zugehörige MSR-Logik nicht detailliert modelliert werden. D.h. es wird nur die Lüftungsanlage mit ihrer MSR-Logik im Datenmodell abgebildet. Ausgangspunkt für die Modellierung war das Anlagenschema des Laborversuchsaufbaus, welches in Abbildung 28 dargestellt ist.

Abbildung 28: Schematische Darstellung des HVAC-Testaufbaus [31]



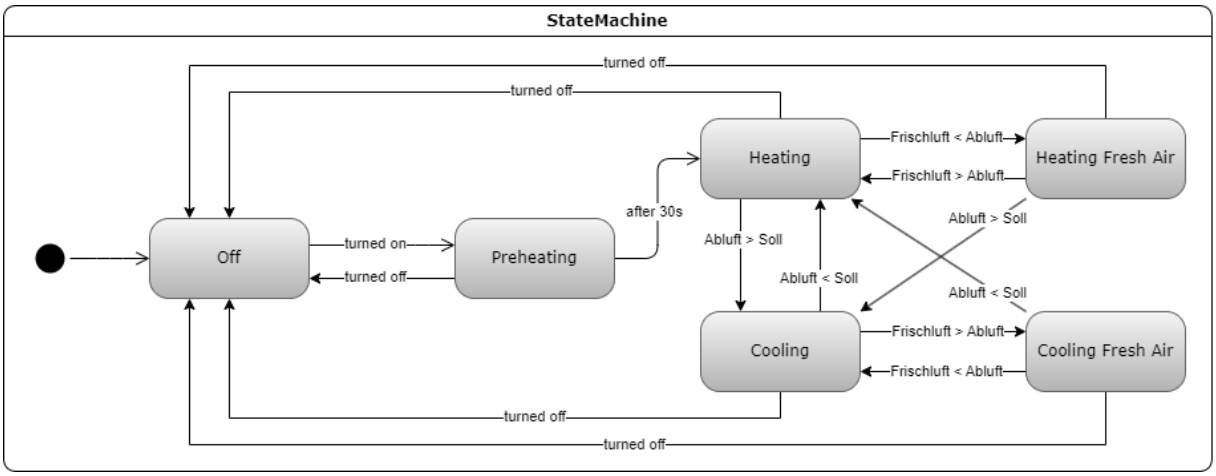
Startpunkt war die Modellierung in Simultan, wo das Datenmodell mit den Datenpunkten, Signalen und allen notwendigen Parametern erstellt wurde. Das Schema selbst wurde im Netzwerkeditor erstellt. Abbildung 29 zeigt einen Ausschnitt der Umsetzung in Simultan. In der Komponentensicht werden alle notwendigen strukturellen Informationen erstellt und gehalten. Zum Beispiel die Informationen für die Ablufttemperatur und Feuchte in der Lüftungsanlage (vgl. Abbildung 29a). Im Netzwerk wird als erstes das Schema der Lüftungsanlage erstellt mit den Rohrleitungen und den mechanischen Komponenten (z.B. Klappe). Dort wird auch ein Block zur Steuerung definiert, der „Mess-, Steuerungs- und Regelungslogik“ Block, damit Logiksignale verarbeitet werden können (vgl. Abbildung 29b). Um den Überblick nicht zu verlieren, wird die detaillierte Modellierung dieses Blocks in ein Subnetzwerk ausgelagert, was den Block näher beschreibt. Dort finden sich dann alle Steuerungssignale wie zum Beispiel Temperaturen (Sollwerte, aktuelle Messwerte, …), die später für die Logikmodellierung genutzt werden können. Dort ist unter anderem auch der Abluftventilator verfügbar, der die Parameter für Ablufttemperatur und Feuchte als Signale verarbeiten kann (vgl. Abbildung 29c). Diese verschiedenen Sichten zeigen somit immer auf dasselbe digitale Modell, wobei je nach Notwendigkeit verschiedene Informationen dem User visualisiert und dargestellt werden. Je weiter ein Planungsprojekt fortgeschritten ist, können die Informationen und Elemente verfeinert werden. Nach Abbildung der strukturellen Informationen, kann nun die Logik des Systems modelliert werden.

Abbildung 29: Ausschnitt aus der Abbildung im digitalen Datenmodell in Simultan (a) Auszug aus der Komponenten Sicht; (b) Schema des Netzwerks; (c) Detailsicht der MSR



Das digitale Gebäudemodell wird mittels XMI Export in den openBAM Editor übertragen, wo mit dem Zustandsgraphen (UML Template) die Logik modelliert werden kann. Abbildung 30 zeigt die MSR-Logik, welche für die Evaluierung modelliert wurde. Vereinfacht gesagt handelt es sich dabei um eine Kühl- und Heizsteuerung mit optionaler Frischluftzufuhr, sofern die Außentemperatur unter beziehungsweise über der Solltemperatur liegen. Dies kann im Sommer beispielsweiße nachts der Fall sein und hilft Energie zu sparen.

Abbildung 30: UML-Darstellung der MSR Logik des Use-Cases im Labor



Für die Modellierung werden die Signale aus dem digitalen Modell genutzt um die Steuerung zu modellieren. In diesem Fall werden die nachfolgenden Signale verwendet.

- Außentemperatursensor (°C)
- Zulufttemperatursensor (°C)
- Ablufttemperatursensor (°C)
- Anlagenschalterzustand (Ein/Aus)

Eingänge

- Sollwert Abluft
- Sollwert Zuluft (°C)
- Zuluftventilator (0 - 100%)
- Abluftventilator (0 - 100%)
- Klappenzustand (0 - 100%)

Ausgänge

Der Ablauf im Einzelnen erfolgt nun folgendermaßen: Wird der Anlagenschalter betätigt, also das System eingeschaltet, werden für eine halbe Minute die Außenklappen geschlossen, sodass die Luft nur im Inneren der Anlage zirkuliert, der Sollwert der Zuluft auf 20°C gesetzt und die Ventilatoren auf voller Leistung laufen gelassen. Im Anschluss nimmt das System in Abhängigkeit der Ablufttemperatur und des zugehörigen Sollwertes (Abluftregelung) entweder den Heiz- oder Kühlzustand (Heating/Cooling) ein und setzt abhängig von den beiden Werten entsprechend den Sollwert der Zulufttemperatur. Sofern gekühlt wird und die Außentemperatur kleiner als die Zulufttemperatur ist (CoolingFreshAir), werden die Außenklappen vollständig geöffnet, ansonsten wird nur 20% der Außenluft bezogen. Gleiches gilt im Heizzustand, sofern die Außentemperatur größer als die Zulufttemperatur ist (HeatingFreshAir). Wird die Anlage ausgeschaltet, so werden alle Ventilatoren ausgeschaltet und die Außenklappen geschlossen.

Diese Modellierung wurde nun in einem ersten Schritt für Simulationen in IDA-ICE nach NMF transformiert. Ein Auszug des generierten NMF Codes ist in Abbildung 31 sichtbar. Dieser wurde automatisch mittels der in Abschnitt 4.5 beschriebenen UML2NMF Transformation erzeugt. Mit Hilfe von dem graphischen Tool IDA-Translator können solche Anweisungen dann in Blöcke für die Simulationssoftware IDA ICE übergeführt werden. Dieser Schritt wird hier nicht näher ausgeführt, da dies ein IDA ICE Spezifikum ist, um eigene NMF Blöcke in der Simulation verwenden zu können. Andere Simulationen, die NMF direkt verwenden können, benötigen diesen Schritt nicht.

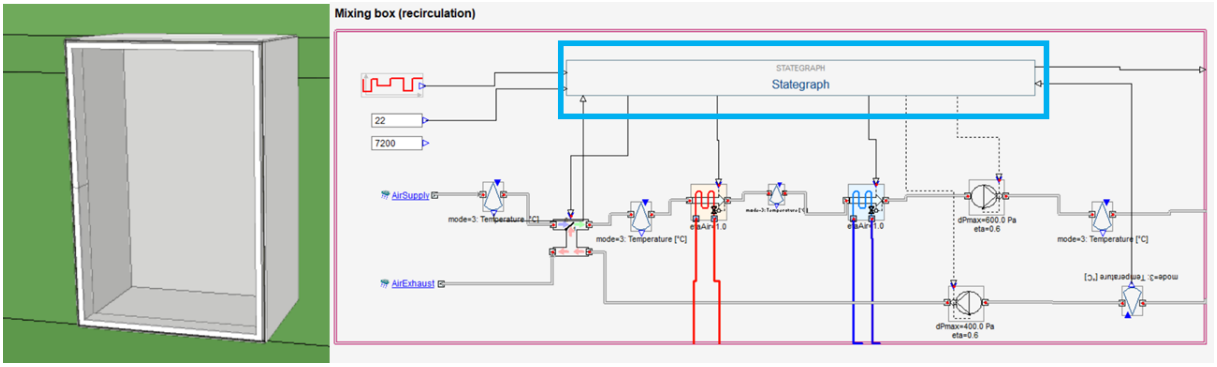
Abbildung 31: Auszug aus dem generierten NMF Code

```
CONTINUOUS_MODEL StateGraph
ABSTRACT "ExtendedUsecase"
EQUATIONS
M021SB01 = IF state == 1 THEN 0 ELSE_IF state == 2 THEN 0 ELSE_IF state == 3 THEN 1 ELSE_IF state == 4
  THEN 1 ELSE_IF state == 5 THEN 1 ELSE_IF state == 6 THEN 1 ELSE_IF state == 7 THEN 1 ELSE 0 END_IF;
M011SB01 = IF state == 1 THEN 0 ELSE_IF state == 2 THEN 0 ELSE_IF state == 3 THEN 1 ELSE_IF state == 4
  THEN 1 ELSE_IF state == 5 THEN 1 ELSE_IF state == 6 THEN 1 ELSE_IF state == 7 THEN 1 ELSE 0 END_IF;
Y102YB01 = IF state == 1 THEN 0 ELSE_IF state == 2 THEN 0 ELSE_IF state == 3 THEN 1 ELSE_IF state == 4
  THEN 0.2 ELSE_IF state == 5 THEN 0.2 ELSE_IF state == 6 THEN 1 ELSE_IF state == 7 THEN 1 ELSE 0 END_IF;
SW01XC01 = IF state == 1 THEN 0 ELSE_IF state == 2 THEN 20 ELSE_IF state == 3 THEN 20 ELSE_IF state == 4
  THEN SW01XC02+100/10*(SW01XC02-B231ME01) ELSE_IF state == 5 THEN SW01XC02-100/10*(B231ME01-SW01XC02) ELSE_IF state == 6
  THEN SW01XC02+100/10*(SW01XC02-B231ME01) ELSE_IF state == 7 THEN SW01XC02-100/10*(B231ME01-SW01XC02) ELSE 0 END_IF;
SimL_Time := Sim_Time + timeStep;

IF state == 1 THEN
    state := 2;
    timer_value := 0;
ELSE_IF state == 2 AND SW01SB01 == 1 THEN
    state := 3;
    timer_value := 0;
ELSE_IF state == 3 AND SimL_Time > timer_value THEN
    state := 4;
    timer_value := 0;
...
END_IF;
Sim_Time := SimL_Time;
LINKS
GENERIC IN_B231ME01  B231ME01;
...
GENERIC OUT_M021SB01  M021SB01  (output);
...
VARIABLES
GENERIC state A_S 0 0 7 "internal state"
GENERIC Sim_Time A_S 0 0 BIG "Integrated time"
...
PARAMETERS
GENERIC pt S_P 5. 0 BIG ""
EXTENSIONS
...
END_EXTENSIONS
END_MODEL
```

In IDA ICE wird dann der Block als Modellierungsblock hinzugefügt, wie in Abbildung 32 blau umrandet zu sehen ist. Dieser Block wird mit den Input und Output Signalen der einzelnen Komponenten (z.B. Ablufttemperaturfühler) verbunden. Der Testraum wurde mit den Abmessungen des realen Raums nachmodelliert (Abbildung 32, links). Nach Zuordnung eines Klimafiles für die Simulation des Wetters kann die Anlage simuliert werden. In unserem Evaluierungsfall wurde eine gleichbleibende Temperatur für den Fall Sommer bzw. Winter ausgewählt, da dies dem Labor Gegebenheiten entspricht. Die Ergebnisse der Simulation werden in Abschnitt 5.3.1 näher gezeigt.

Abbildung 32: 3D Testraumabbildung (links) und Steuerung von Heiz- und Kühlregister (rechts) in IDA ICE



In einem nächsten Schritt wurde wieder das digitale Gebäudemodell in Simultan zusammen mit der MSR-Logik hergenommen um die Transformation in Richtung Dokumentationspaket durchzuführen. Das Dokumentationspaket besteht aus mehreren Schlüsselelementen, die den Ingenieuren alle notwendigen Informationen liefern sollen.

Das Komponentendiagramm mit Datenpunkten ist ein Übersichtsdiagramm, das alle Systemkomponenten, ihre Verbindungen und die zugehörigen Datenpunkte darstellt. Dieses dient als Brücke zwischen den Simultan- und SAUTER-Modellen und bietet ein plattformunabhängiges Verständnis des gesamten Systems auf hohem Niveau. Abbildung 33 zeigt diese Schemadarstellung für den Use Case, wobei ähnlich zu dem Simultan Modell die Komponenten dargestellt sind. Darüber hinaus werden aber auch Datenpunkte und ihre Datentypen direkt angezeigt.

Als nächstes wird der Komponenten-Schaltplan dargestellt, der ein detailliertes, SAUTER-spezifisches Schema des HLK-Systems mit Komponenten, Datenpunkten und Luftstromwegen darstellt. Dadurch werden Ingenieur:innen bei der genauen Nachbildung des Modells in der SAUTER-Software durch eine vertraute Visualisierung unterstützt. Abbildung 34 zeigt diese überführte Darstellung für den Evaluierungsfall.

Neben den strukturellen Diagrammen ist auch das Steuerungslogik-Diagramm essentiell, welches in Abbildung 35 dargestellt ist. Dieses veranschaulicht die im openBAM Editor modellierte Steuerlogik des Systems, einschließlich Zustände, Übergänge und Bedingungen für die Übergänge. Dadurch wird die effektive Implementierung in der SAUTER-Umgebung ermöglicht.

Darüber hinaus wird eine Datenpunkt-Liste generiert, die detaillierte Informationen zu den Datenpunkten, die für die Systemimplementierung wichtig sind, enthält. Die Funktionsgruppenbeschreibungen geben dann detailliert Auskunft über die Hardwarekomponenten mit einer Schritt-für-Schritt-Anweisungen für die Modellrekonstruktion, wie in Abbildung 36 zu sehen

ist. Dadurch werden Ingenieur:innen bei der Erstellung des Modells, einschließlich der Schaltpläne und der spezifischen Datenpunkte im Zusammenhang mit den Funktionsgruppen, im Case Builder unterstützt.

Abbildung 33: Komponentendiagramm, das alle Systemkomponenten, ihre Verbindungen und Datenpunkte anzeigt passend zum Use Case

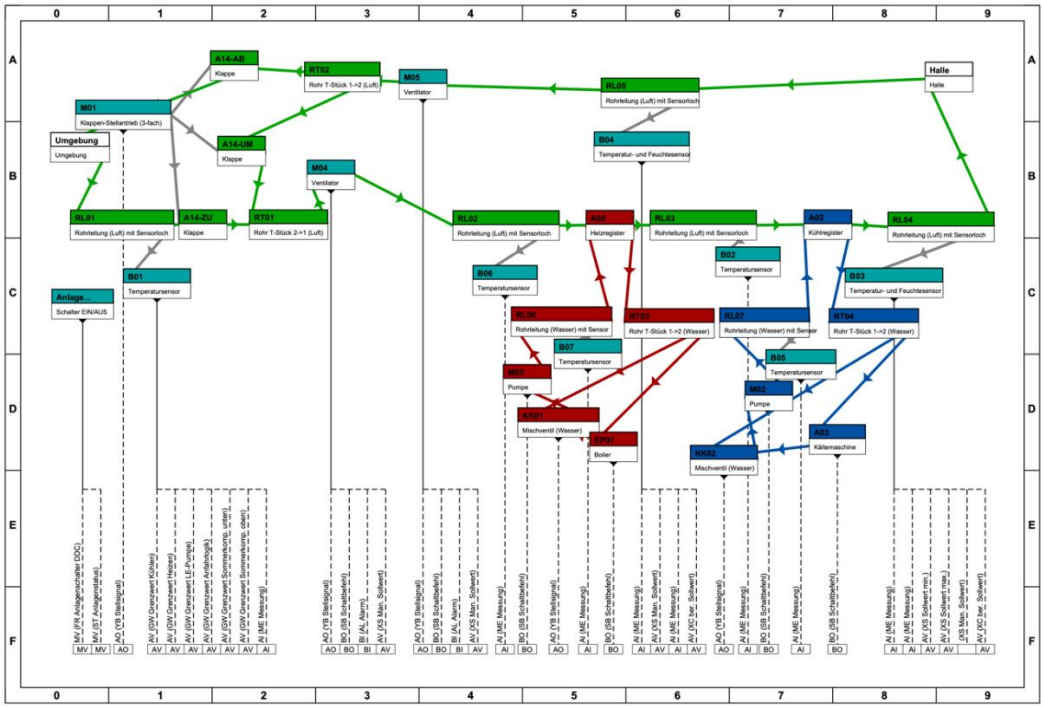


Abbildung 34: Detailliertes SAUTER-spezifisches HVAC-Schema mit Komponenten, Datenpunkten und Luftströmungspfaden

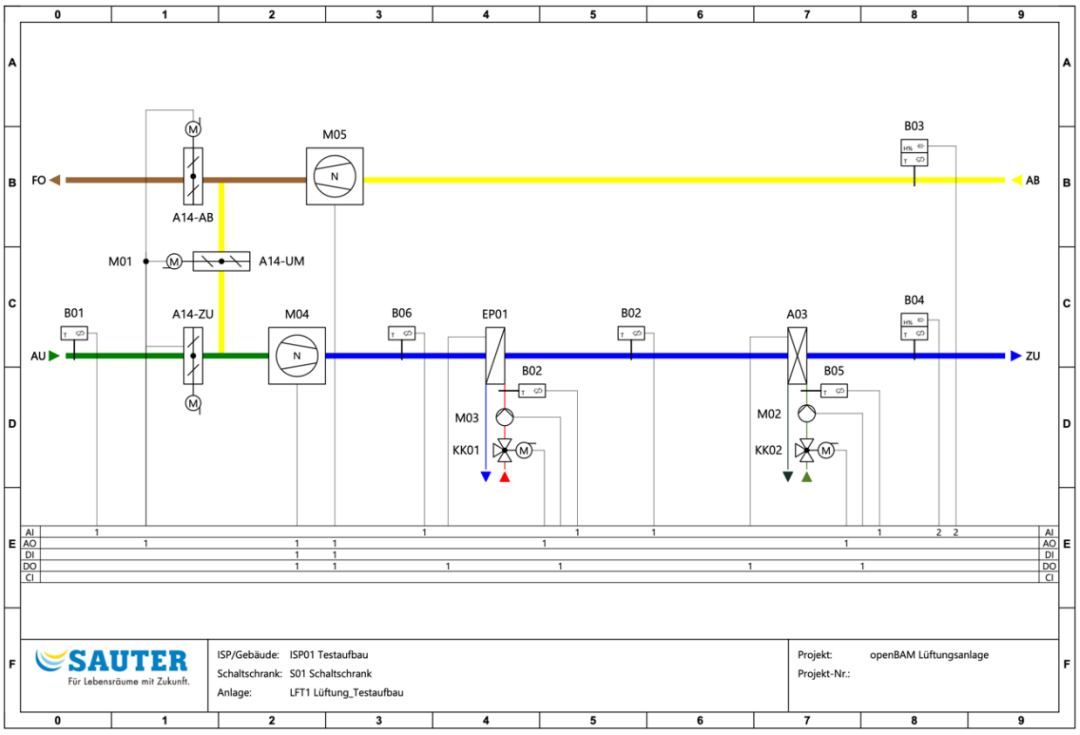


Abbildung 35: Steuerungslogikdiagramm, das Systemzustände, Übergänge und Bedingungen veranschaulicht

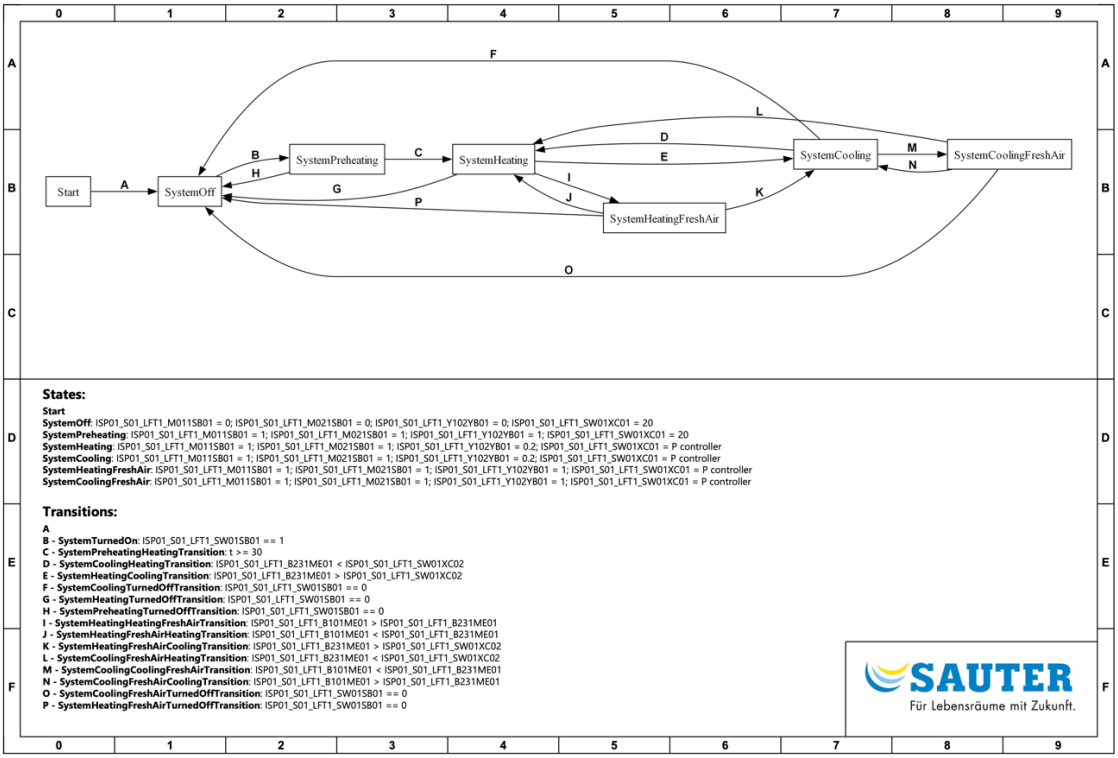
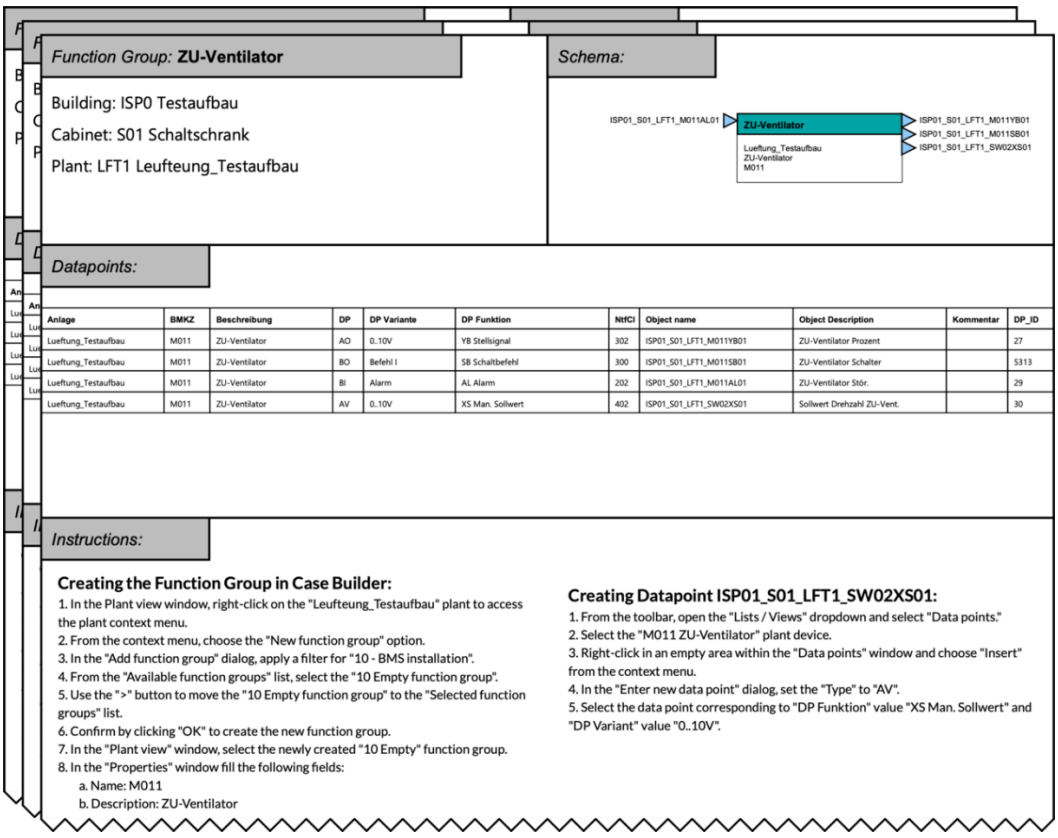
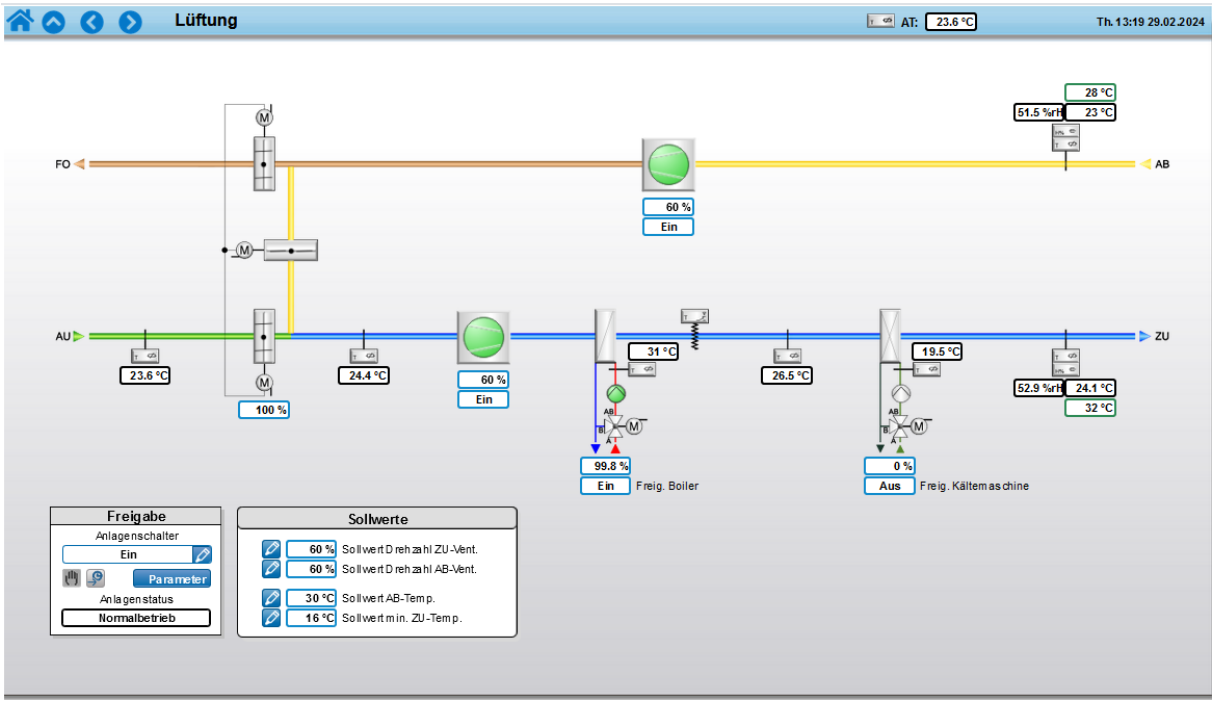


Abbildung 36: Funktionsgruppenbeschreibungen und Anleitungen mit detaillierter Hardware-Dokumentation und Schritt-für-Schritt-Anleitung für die Modellrekonstruktion in Case Builder, einschließlich Schaltplänen und spezifischen Datenpunkten



Nach erfolgreicher Überführung in den Case Builder kann das System auch in der Ausführung fertig umgesetzt werden und wurde vor Ort installiert. Abbildung 37 zeigt die Weboberfläche zur Steuerung der installierten Anlage im Laborversuchsaufbau.

Abbildung 37: SAUTER Steuerungsoberfläche des Laborversuchsaufbaus



5.3.1. Vergleich von Mess- und Simulationsdaten

Abschließend wurden Simulationen und Messungen miteinander verglichen um Festzustellen, ob die MSR-Logik erfolgreich in beide Systeme gleich überführt wurde. Es sei angemerkt, dass es darum geht, ob das Verhalten gleich ist, nicht aber ob die Werte ident sind. Die Evaluierung soll zeigen, dass die Überführung von plattformunabhängiger in plattformabhängige Tools konsistent möglich ist.

Für diese Evaluierung wurde ein Vergleich zwischen den gemessenen Daten und den mit IDA ICE simulierten Ergebnissen durchgeführt. Im Sommer soll mit Hilfe des Lüftungssystems auf 20°C gekühlt werden und im Winter auf 28°C geheizt (diese Werte ergeben sich aufgrund der Raumumluft des Prüfstands) . Um die Vergleichbarkeit von Messung und Simulation sicherzustellen, wurde der Mittelwert der gemessenen Außentemperaturen (Sommer: 25,98 °C, Winter: 20,75 °C) als konstante Außentemperatur in der jeweiligen Klimadaten-PNR-Datei angepasst und für die Simulation verwendet.

Ein weiterer wichtiger Schritt war die Entfernung des Eingangsparameters der Solareinstrahlung in den Klimadaten, da die durch das Fenster auf den Testraum einfallende Sonnenwärme die Simulationsergebnisse stark verfälschen kann.

Zusätzlich wurden Windrichtung, Windgeschwindigkeit, sowie die solare Diffusstrahlung auf 0 gesetzt, um realistische Randbedingungen zu schaffen. Die Standard-U-Werte für Innenwände, Decke und Boden wurden leicht angepasst, indem die Betonschichten auf einen U-Wert von 0,25 W/(m²K)

festgelegt wurden. Unter Berücksichtigung der unveränderten Standardwerte der anderen Bauteilkomponenten ergaben sich die folgenden U-Werte:

- Innenwände: 0,6187 W/(m²K)
- Decke und Boden: 0,175 W/(m²K)
- Plexiglasfenster: 2,0 W/(m²K).

Die betrachtete Zeitperiode umfasst die erste Stunde, in der die Kühlung oder Heizung aktiviert wird. Die Zu- und Abluftmengen wurden auf 10 L/(s·m²) festgelegt, da die voreingestellten Luftwechselwerte für größere Räume ausgelegt sind. Die luftseitige Wirksamkeit des Kühlregisters wurde auf 23 % festgelegt, während die des Heizregisters bei 100 % belassen wurde. Was beim Heizregister jedoch verändert wird, ist die Vorlauftemperatur, welche von 20°C auf 70°C gehoben wird.

Um die thermische Trägheit zu berücksichtigen, numerische Fehler zu minimieren und realistische Anfangsbedingungen zu gewährleisten, wurde die Einschwingzeit auf 14 Tage festgelegt.

Kühlung im Sommer

Abbildung 38 zeigt die Temperaturen der Zuluft, Abluft, Außenluft und der Solltemperatur über den Messzeitraum in °Celsius für einen Sommerfall. Abbildung 39 zeigt dieselben Temperaturen für die Simulation des Sommerfalls, welche in IDA ICE durchgeführt wurden. Die Zeitskalen sind von der Uhrzeit her nicht ident, können aber da ein stationäres Wetter gerechnet wurde, trotzdem miteinander verglichen werden.

Abbildung 38: Temperaturmessdaten des Testraumes (Zuluft, Abluft, Außen, Soll) - Sommer

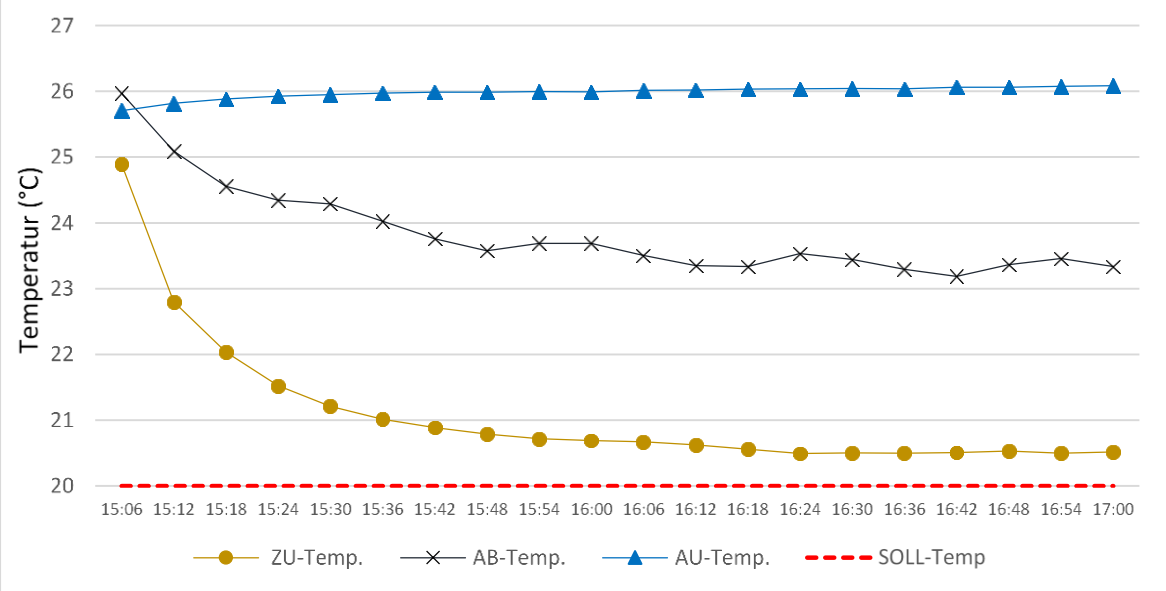
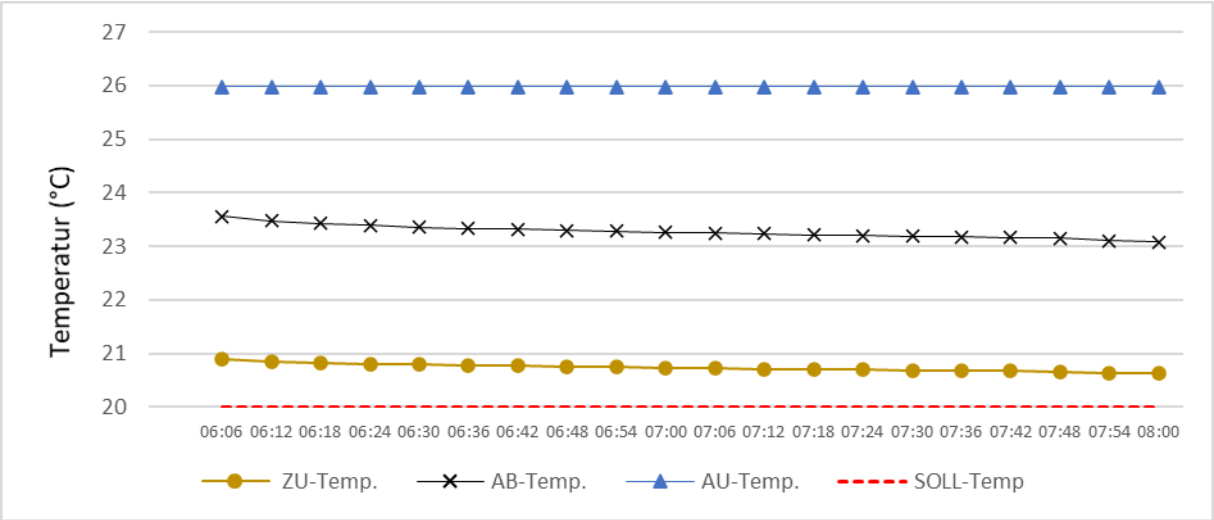


Abbildung 39: Temperaturen der Simulation des Testraumes (Zuluft, Abluft, Außen, Soll) - Sommer



Die Ergebnisse zeigen eine hohe Übereinstimmung zwischen der Kühlung von Messung und Simulation. Dennoch weist die Simulation in Abbildung 39 eine höhere Glätte auf als die Messung in Abbildung 38. Die Verläufe sind linearer und sichtbar weniger schwankend, als in den realen Messungen. Dies lässt sich darauf zurückführen, dass in der Simulation idealisierte Bedingungen vorliegen, insbesondere durch die konstante Außentemperatur und die direkte Anpassung der Zulufttemperatur ohne Vorlaufzeit. Im Gegensatz dazu können in den realen Messungen dynamische Schwankungen durch tatsächliche äußere Einflüsse sowie durch die zeitverzögerte Regelung des Kühlregisters entstehen.

Heizung im Winter

Abbildung 40 zeigt die Temperaturen der Zuluft, Abluft, Außenluft und der Solltemperatur über den Messzeitraum in °Celsius für den Winterfall, wo geheizt werden muss. Abbildung 41 zeigt wiederum auch diese Arten an Temperaturen für die Simulation des Winterfalls. Auch bei dieser Simulation ist zu beachten, dass eine konstante Außentemperatur angenommen wurde, wodurch kleinere Schwankungen geglättet werden.

In der Simulation von Ida Ice in Abbildung 41 sieht man einerseits, dass die Zulufttemperatur sich erst einpendeln muss und mit einem hohen Wert startet und dann schnell auf eine Temperatur, ähnlich wie in der Messung (vgl. Abbildung 40), sinkt. Im realen Fall steigt die Zulufttemperatur an und pendelt sich bei 32°C ein. Da die Simulation und die Messung im Heizfall in den Abbildungen verschiedene Temperaturskalen auf der Y-Achsen haben, ist der Vergleich schwierig. Da die MSR Logik aber die Regelung der Abluft steuert wird in Abbildung 42 nur Ablufttemperatur des Winterfalls mit der Solltemperatur (Zieltemperatur) dargestellt. Im direkten Vergleich zeigt sich, dass sich die Ablufttemperatur bei der Messung mit kleineren Gradienten als in der Simulation an die Solltemperatur annähert. Die Simulation hat auf Grund des Einpendeln am Start der Simulation einen höheren Temperaturanstieg im Raum, kommt aber in dem Zeitintervall weniger nah an die Solltemperatur als im realen Fall. Ein genauer Vergleich wie im Sommer bei der Kühlung ist nicht möglich, da durch die 14 Tage Simulationsvorlaufzeit sich die Raumtemperatur (Ablufttemperatur) nicht genau an die Außentemperatur angleicht und somit einen höheren Startwert besitzt.

Abbildung 40: Temperaturmessdaten des Testraumes (Zuluft, Abluft, Außen, Soll) - Winter

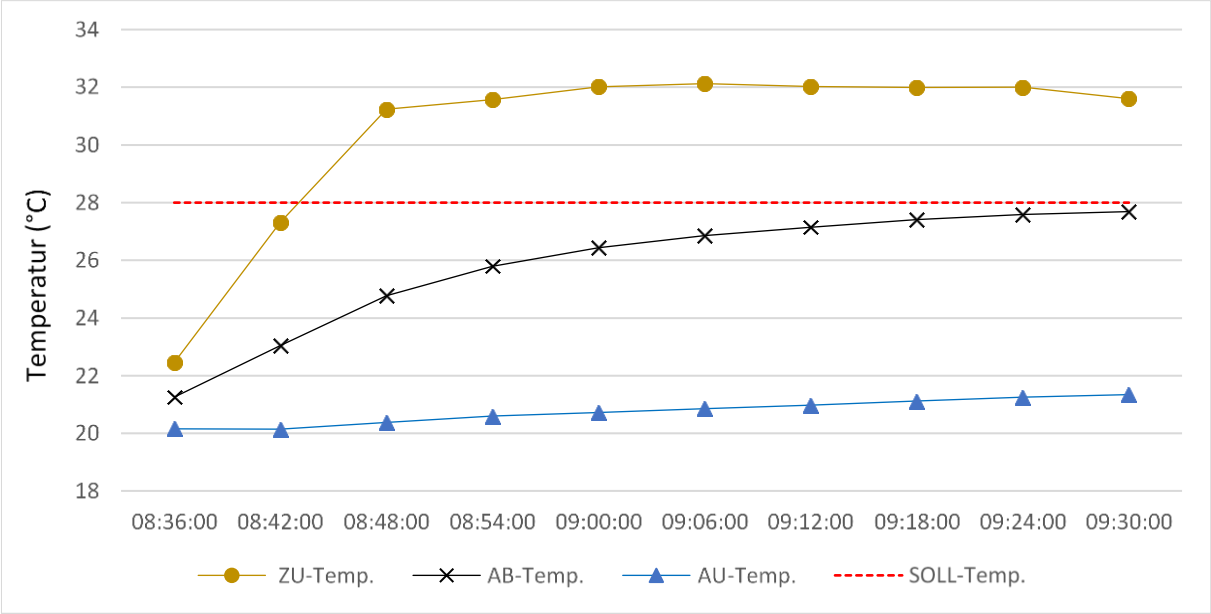


Abbildung 41: Temperaturen der Simulation des Testraumes (Zuluft, Abluft, Außen, Soll) - Winter

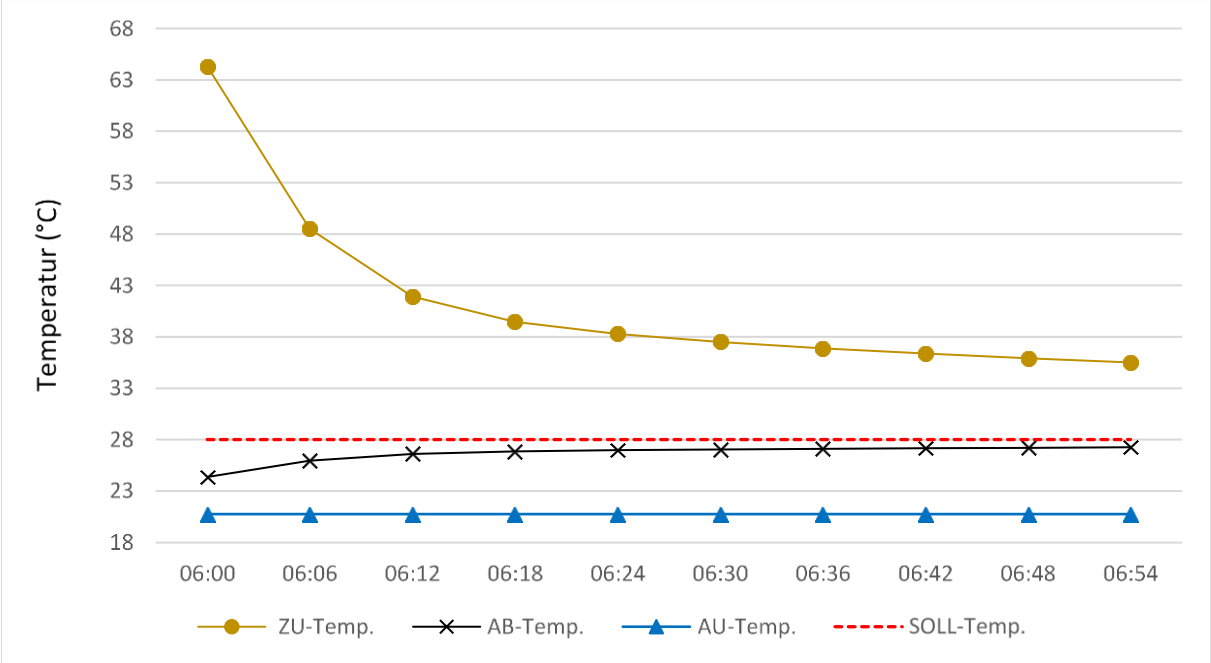
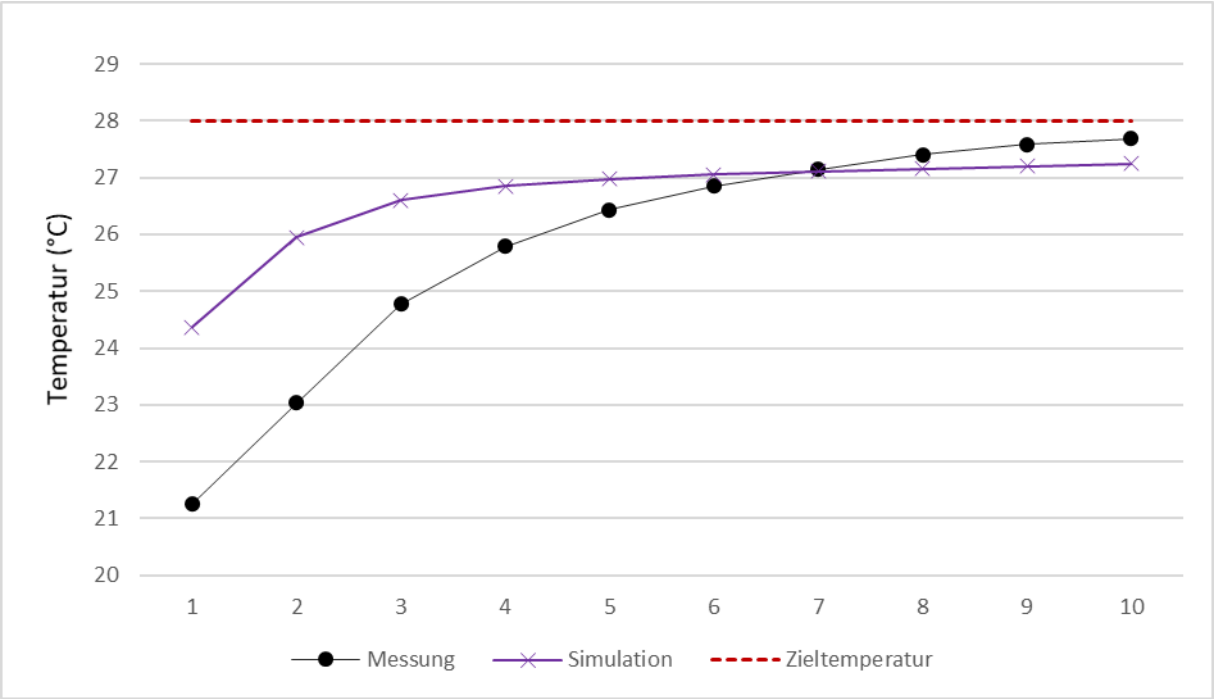


Abbildung 42: Gegenüberstellung der Ablufttemperaturen der Messung und der Simulation zusammen mit der Zieltemperatur (Sollwert) für den Heizungsfall im Winter



Die Evaluierung zeigt, dass das Verhalten von der Simulation und der Messung ähnlich ist und somit dieselben Zustände angestrebt werden, wie es für eine konsistente Modellierung gewünscht ist.

5.4. openBAM Projekt im „Stadt der Zukunft“ Programm

openBAM war im Bereich Digitales Planen, Bauen und Betreiben beim Thema Systemintegration und -kombination von Digitalem Planen, Bauen und Betreiben der 8.Stadt der Zukunft Ausschreibung angesiedelt. Die Umsetzung des Projekts setzt auch einen Startpunkt zu den in der Ausschreibung genannten Zielen. Nachfolgend sind mehrere Aspekte angeführt, wie das Projekt zur digitalen Integration von Prozessen und der Verbesserung der Effizienz im Lebenszyklus von Gebäuden und Quartieren beiträgt.

- Einbettung digitaler Technologien und Methoden in den Lebenszyklus von Gebäuden:
  Durch die Verwendung einer plattformunabhängigen Beschreibung der MSR-Logik (UML Zustandsdiagramm) und deren Verknüpfung mit weiteren Informationen des digitalen Modells zu Gebäuden (über die offene Datenschnittstelle „Simultan“) wird eine durchgängige digitale Modellierung und Vernetzung ermöglicht. Die Verknüpfung sorgt für eine durchgehende Datenkonsistenz, da mittels Transformationen diese Informationen auch bei Simulation und Ausführungsentwicklung konsistent genutzt werden kann.
- Optimierung und Verknüpfung von Abläufen und Arbeitsweisen:
  Die Methode erlaubt es, die Regelungslogik unabhängig vom Zielsystem (im Fall des Projekts SAUTER Engineering) zu entwickeln und in der Simulation zu testen und zu optimieren. Dies führt zu einer verbesserten Planung der Gebäudeautomation und somit zu einem optimierten Betrieb, da die Logik konsistent modelliert wird und auch in der Simulation auf ihre Effizienz hinsichtlich des späteren Energieverbrauchs überprüft werden kann.

- Reduktion von Planungs- und Schnittstellenrisiken:
  Die Methode sorgt für eine klare und konsistente Modellierung, die unabhängig vom Zielsystem entwickelt und daher schon frühzeitig in der Planung eingesetzt werden kann. Durch die automatisierte Transformation reduziert sich Möglichkeit, dass Entwicklungen nicht in die Ausführung gelangen können. Außerdem können Erweiterungen der Transformation in weitere Tools auf Basis von standardisierten Möglichkeiten von MDE umgesetzt werden. Es verringern sich somit entlang des Lebenszyklus mögliche Missverständnisse und Fehlerquellen.

Zusammengefasst ermöglicht die Methode durch die Verknüpfung des digitalen Gebäudemodells, der Abbildung der MSR Logik und der Transformation in Richtung Simulation und Ausführungstool eine verbesserte Datenintegration und Optimierung des Betriebs. Sie kann Risiken beim Datentransfer verringern und die Ressourceneffizienz im Gebäudelebenszyklus zu fördern.

6 Schlussfolgerungen

6.1. Erkenntnisse des Projektteams

Das Projekt konnte zeigen, dass die Modellierung der MSR bis zu einem gewissen Grad vom Hersteller unabhängig erfolgen kann. Das erzeugte Modell kann zur Simulation verwendet werden und somit können Energieverbräuche im Betrieb besser abgeschätzt werden. Weiters ist es möglich Optimierungen an der Regelstrategie integriert mit dem digitalen Gebäudemodell schon in der Designphase vorzunehmen. Die Simulation ermöglicht es, Szenarien durchzuspielen und die Logik so zu verfeinern, dass sie möglichst effizient arbeitet, bevor sie in die tatsächliche Ausführung überführt wird. Weiters kann dadurch langfristig auch die Reduzierung des Energieverbrauchs in realen Anwendungen erreicht werden. Da die Logikentwicklung in der Designphase plattformunabhängig mitgedacht werden kann, können schneller Anpassungen durchgeführt werden. Durch automatisierte Transformationen ist es möglich, dann das Zielsystem auszuwählen, welches für die Ausführung genutzt wird.

Zusammenfassend stellen diese Ergebnisse den Start für weitere Verbesserungen am Weg zum vollwertigen Digitalen Zwilling dar, wo digitale Modelle nicht nur strukturelle Informationen, sondern auch Verhaltensinformationen beinhalten. Durch die technologieunabhängige Modellierung der MSR ist es möglich während der Planung die Gebäudeautomation schon abzubilden. Für die Ausführung kann dann die optimierte Lösung je nach Vergabe automatisiert in das Zielsystem transformiert werden. Es wird sichergestellt, dass die Daten konsistent bleiben.

Das Projekt zeigte außerdem mehrere Herausforderungen im Zusammenhang mit proprietärer Software und offenen Datenformaten auf. Proprietären Systemen fehlen oft standardisierte Methoden für den Datenimport und -export, was die Interoperabilität mit externen Tools erschwert. Diese Systeme sind auf die Lösung spezifischer Probleme in einem bestimmten Bereich ausgerichtet und berücksichtigen möglicherweise nicht den plattformübergreifenden Datenaustausch, was die Bemühungen um eine branchenweite Interoperabilität behindert. Um diese Herausforderungen abzumildern und eine bessere Zusammenarbeit über verschiedene Plattformen und Disziplinen hinweg zu fördern, könnten Unternehmen ermutigt werden, in die Unterstützung offener Datenformate für Import- und Exportfunktionalitäten zu investieren. Diese Anreize könnten durch die Einführung von Industriestandards, die Umsetzung rechtlicher Rahmenbedingungen oder durch Marktanreize geschaffen werden. Das Eintreten für die Annahme standardisierter Datenstrukturen und -formate innerhalb der Branche ist von wesentlicher Bedeutung. Die Standardisierung würde Integrationshindernisse abbauen und einen nahtloseren Datenaustausch zwischen proprietären und Open-Source-Tools ermöglichen, was die Interoperabilität insgesamt verbessern würde. Es verdeutlicht die Notwendigkeit von branchenweiten Bemühungen zur Unterstützung offener Datenformate und gemeinsamer Standards.

Der Forschungsbereich Bauphysik arbeitet daran weiter, dass der Prozess des Austausches von digitalen Modellen und weiterführenden Tools automatisiert erfolgen kann und die manuellen Schritte reduziert werden. Ein weiterführender Schritt wird die Kopplung von den Simulationsergebnissen und Messergebnissen sein, sodass auch hier der Überblick nicht mehr verloren geht. Im Forschungsbereich Automation Systems ist ein Ziel, dass der Grad der

Automatisierung in der Gebäudeautomation weiterwächst und gleichzeitig dabei auch die Qualität der Tools verbessert wird. Ein nächster Schritt wird darin bestehen, die Herausforderungen zu untersuchen, die auftreten, wenn Änderungen am digitalen Zwilling vorgenommen werden. Hierbei stellt sich die Frage, wie mit diesen Änderungen umgegangen werden kann: Muss die Simulation neu erstellt werden, oder kann sie automatisiert angepasst werden? Zudem ist die Erstellung der Regelstrategien ein weiterer wichtiger Punkt, an dem in Zukunft weitergearbeitet wird. Bei SAUTER werden die Ergebnisse genutzt um die Notwendigkeit von Austauschmöglichkeiten im Fachverband der Elektro- und Elektronikindustrie (FEEI) noch weiter voranzutreiben.

Auf Basis der gewonnen Erkenntnisse in diesem Projekt wurde auch ein weiterführendes Projekt beim AI Ökosysteme 2024: AI for Tech & AI for Green Call eingereicht: Hybrid AI for Fault Detection and Explanation in HVAC Systems (HyFADE-HVAC). Das Forschungsprojekt zielt darauf ab, symbolische Ansätze (regelbasierte Logik) mit großen Sprachmodellen (LLMs) zu kombinieren, um semantisch angereicherte BIM-Daten zu erstellen, die als Grundlage für ein hybrides AFDD-System dienen. Dadurch sollen HVAC-Systeme effizienter betrieben und technische Erklärungen für Nutzer bereitgestellt werden, was das Vertrauen in die Technologie stärkt.

6.2. Relevanz für Zielgruppen

Schon während der Projektlaufzeit gab es durch den Firmenpartner SAUTER einen Austausch mit dem FEEI, wo über die Erkenntnisse und die aktuellen Fortschritte im Forschungsprojekt berichtet wurde. Die gewonnen Erkenntnisse zeigen auch dem Fachverband auf, dass offene APIs relevant sind um die Konzepte breit nutzbar zu machen. Fakt ist, dass die Normierung in Richtung BIM und zur inline Verwendung von Planungsdaten der Schritt in eine digital transformierte MSR sind. Der Weg wird hierfür weiter im FEEI beschritten.

Da der menschliche Eingriff zum gegenwärtigen Zeitpunkt in Projekten bei der Planung und der Gebäudeautomation stark ausgeprägt ist, weist diese Form der Projektabwicklung eine gewisse Anfälligkeit auf. Die Fehlerquote durch Übertragfehler und veraltete Planunterlagen variiert je nach Projekt und den verantwortlichen Personen. Aus diesem Grund kann die Fehleranfälligkeit nicht pauschal bewertet werden, sondern muss jeweils für das konkrete Projekt geprüft werden. Grundsätzlich bleibt jedoch festzuhalten, dass ein höherer Automatisierungsgrad und die Übernahme von Informationen aus Modellen schon zur Designphase einen erheblichen Mehrwert für die gesamte Branche der Regelungstechnik bieten würden.

6.3. Bisherige Verwertung, Anwendungs- und Marktpotenziale

Die Projekterkenntnisse und Ansätze stießen beim Endworkshop im September 2024 auf großes Interesse der Teilnehmer:innen des FEEI. Auf Basis dessen wurde auch diskutiert, wie die Entwicklung und der Ausbau einer offenen Austauschschnittstelle zukünftig weitergeführt werden kann. Diese Schnittstelle könnte den interdisziplinären Dialog und die Vernetzung innerhalb der Branche weiter fördern. Die Diskussion stellte weitere gemeinsame Projekte (z.B. Collective Research) und Kooperationen in Aussicht.

Durch die Publikationen, welche am Ende dieses Abschnitts angeführt werden, gab es auch einen guten Austausch mit der wissenschaftlichen Community. Auf verschiedenen Fachkonferenzen

wurden unsere Entwicklungen präsentiert und intensiv diskutiert, was nicht nur die Relevanz unterstreicht, sondern auch die Grundlage für zukünftige interdisziplinäre Kooperationen bildet. Durch einen solchen Austausch bei der BauSIM 2024 ergab sich die Möglichkeit mit einer Forschungsgruppe der Universität Innsbruck in intensiveren Kontakt zu treten und weitere Treffen zu vereinbaren. Da wird momentan an einer gemeinsamen Projektidee gearbeitet, welche die Automatisierung noch weiter vorantreibt.

Im Bereich der Lehre nutzen wir die gewonnenen Erkenntnisse, um Studierenden die Relevanz und Notwendigkeit von Modellen, Automatisierung und frühzeitiger Simulation näherzubringen und ihre praktischen Anwendungsmöglichkeiten aufzuzeigen. Durch die Einbindung dieser Themen in die akademische Ausbildung möchten wir nicht nur die Fachkompetenz der Studierenden fördern, sondern ihnen auch die Relevanz und Notwendigkeit moderner Technologien in der Praxis näherbringen. Dies soll sie optimal auf zukünftige Herausforderungen und Entwicklungen in der Branche vorbereiten und einen Beitrag zur langfristigen Weiterentwicklung des Fachgebiets leisten.

Entstandene Publikationen und Präsentationen

- Jürgen Pannosch und Wolfgang Kastner: Reference Model for Building Automation. In: ETFA 2023: 1-4
- Felix Knorr und Wolfgang Kastner: Towards a Uniform Exchange Format for Home and Building Automation using VDI 3814. In: ETFA 2023: 1-4
- Felix Knorr, Jürgen Pannosch und Gernot Steindl: A Model-Driven Approach for Consistent Building Automation Control Logic Design and Deployment. In: ICCAD 2024: 1-6
- Sabine Sint, Felix, Knorr, Jürgen Pannosch, Jürgen Kromp und Thomas Bednar: Towards Open Modeling of Building Automation over the entire building life cycle. In: BauSIM 2024 Companion Proceedings : 10te Konferenz von IBPSA-DACH, TU Wien, Österreich, 2024, S. 34–34), https://doi.org/10.34726/7560
- Álvaro Sicilia, Thomas Bednar, Rolando Biere und Gloria Calleja: Round table 3 - Improving building assessment through data integration [Presentation]. In: TIMEPAC final conference, Barcelona, Spain, 2024, https://doi.org/10.5281/zenodo.14523693

7 Ausblick und Empfehlungen

7.1. Forschungs- und Entwicklungsbedarf

Während der Bearbeitung und Evaluierung im Forschungsprojekt ergaben sich einige Potentiale für weitere Entwicklungen, die den Rahmen dieses Forschungsprojekts gesprengt hätten.

In der momentanen Umsetzung des Workflows werden Änderungen innerhalb des Datenmodells schon verarbeitet. Sollte aber ein Datenpunkt geändert oder sogar gelöscht werden, der in der Logikmodellierung genutzt wird, würde das zu der Notwendigkeit einer neuen Modellierung führen oder manueller Ausbesserungen. Zukünftig müssten also Entwicklungen in Richtung von Notifikation überlegt werden, die Änderungen durchpropagieren und den User darauf aufmerksam machen. Es könnten die Möglichkeiten von Repositorien und Versionskontrollen für einen Entwicklungsansatz genutzt werden.

Ein weiterer Schritt wäre in Richtung Vollautomatisierung des Workflows. Der momentane Workflow beinhaltet automatisierte Abschnitte, welche aber noch manuell angestoßen und dann in die jeweiligen Tools überführt werden müssen.

In der Modellierung der Gebäudeautomation selbst wäre die Erarbeitung komplexerer Zustandsgraphen mit Verzweigungen und parallelen Abläufen sowie deren Evaluierung ein nächster Schritt. In diesem Zusammenhang könnte dann auch realisierte komplexe Systeme (z.B. Lüftungsanlagen) in realen Gebäuden nachmodelliert und analysiert werden.

Neben den Optimierungsmöglichkeiten innerhalb der Forschungsergebnisse der Methodik von openBAM gibt es auch weiterführenden Forschungs- und Entwicklungsbedarf. So könnten mit Hilfe von Systematiken der künstlichen Intelligenz bzw. auf Basis der Erkenntnisse von Betreiber:innen von Systemen automatisierte Grundeinstellungen zu einem ersten Entwurf von MSR-Logik Modellen führen. Ein Beispiel hierfür wäre, wenn eine Lüftungsanlage eine bestimmte Kennzahl (Raumluftqualität, Temperatur, Feuchtigkeit, etc.) erfüllen soll, dann benötigt die Steuerung dies oder jenes. Dadurch könnte das Logikmodell schon vorab bestimmte Blöcke und Signale verarbeiten und nicht als leeres Template zur Verfügung gestellt werden.

Darüber hinaus könnte eine Weiterführung des Ansatzes auch die Möglichkeit von einer Fehlererkennung anhand von Kennzahlen beinhalten. Während des Designs werden Simulationen genutzt um das Systemverhalten vorab genau zu analysieren. Durch den Vergleich von Kennzahlen während des Betriebs mit der Simulation könnten Warnungen aufgrund von größeren Abweichungen frühzeitig gemeldet werden. Es könnten also Ausfälle verringert werden und vielleicht notwendige Wartungen zeitnah erkannt werden.

7.2. Potenzial für Demonstrationsvorhaben

Aus den Ergebnissen und Erkenntnissen des Projekts ergibt sich ein Potenzial für Demonstrationsvorhaben vor allem in der praktischen Anwendung und Integration der digitalen Technologien. Durch die Weiterführung der gezeigten Methodik und die komplette Automatisierung

des Workflows können auch die digitalen Gebäudemodelle in Richtung digitale Zwillinge weiter ausgeführt werden.

Die Abbildung der Gebäudeautomation schon früh im Designprozess ermöglicht verschiedene Varianten zu simulieren und Planungsfehler schon frühzeitig zu erkennen (z.B. Heizen und Kühlen gleichzeitig). Es kann dadurch eine Optimierung und Verbesserung des Lebenszyklus Managements erreicht werden. Durch die Verbindung von digitalen Zwillingen und automatisierten Workflows könnte ein durchgängiger, transparenter Prozess geschaffen werden. In einem Der offene Austausch über ein Datenmetamodell (Simultan) bietet die Flexibilität verschiedene Softwarelösungen miteinander zu nutzen. Diese plattformübergreifende Kommunikation zwischen unterschiedlichen Softwarelösungen und Geräten ist auch im Bereich der Gebäudeautomation von Vorteil und fördert den Austausch von Daten zwischen verschiedenen Komponenten und Systemen (z.B. MSR-Technik und Gebäudemodelle). Die konsistente Modellierung der Regelungslogik garantiert eine einheitliche und zuverlässige Entwicklung der Steuerung.

Demonstrationsvorhaben könnten solche integrierten Workflows an größeren aus der Praxis gegriffenen Demonstrationsobjekten visualisieren und testen, um zu zeigen, wie die Automatisierung und die frühzeitige Einbindung von Gebäudeautomation zu geringeren Fehlerquoten und einer schnelleren Umsetzung führt. Darüber hinaus könnte in diesem Vorhaben auch die Erfassung und Mitführung der Echtdaten im digitalen Zwilling weitere Optimierungsmöglichkeiten in der MSR aufzeigen.

8 Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] A. David, T. Bednar: Transition towards an energy-efficient scientific Office Setup in Austria's largest (Plus-)Plus-Energy Office Building - An Analysis. In: E3S Web Conf., 2020

[2] O. Hernandez: Elithis Tower in Dijon, France. In: REHVA JOURNAL, Volume 48: Nearly Zero Energy Buildings, 2016

[3] C. Eastmann: The Use of Computers Instead of Drawings. In: AIA Journal, Volume 63, Number 3, 1975, S. 46-50
[4] G. van Nederveen und F.P. Tolman: Modelling multiple views on buildings. In: Automation in Construction, Volume 1, Number 3, 1992, S. 215-224
[5] F. E. Jerniga: BIG BIM little bim - The practical approach to Building Information Modeling - Integrated practice done the right way!. 4 Site Press, Salisbury (MD), USA, 2nd ed., 2008
[6] A. Borrmann, M. König, C. Koch und J. Beetz: Building Information Modeling. Technologische Grundlagen und industrielle Praxis. Springer, Wiesbaden, 2015
[7] buildingSMART, "IFC4.3.x," http://ifc43-docs.standards.buildingsmart.org/, 2024, (abgerufen am 16.7.2024 10:13)
[8] J. Steel, R. Drogemuller und B. Toth: Model Interoperability in Building Information Modelling. In: Software and Systems Modeling, 11(1), 2012, S. 99-109
[9] R. Polit-Casillas und A.S. Howe: Virtual Construction of Space Habitats: Connecting Building Information Models (BIM) and SysML. In: Proceedings of the AIAA Space 2013 Conference and Exposition, San Diego, 2013
[10] G. Paskaleva, S. Wolny und T. Bednar: Big-open-real-BIM Data Model - Proof of Concept. In: IBPC2018 - Healthy, Intelligent and Resilient Buildings and Urban Environments, 2018, S. 1083 - 1088
[11] G. Paskaleva, A. Mazak-Huemer, S. Sint und T. Bednar: Standardized Data Integration in the AEC Domains – What does it take to succeed?. In: IOP Conference Series: Earth and Environmental Science, https://doi.org/10.34726/3551, 2022
[12] J. Sousa: Energy Simulation Software for Buildings: Review and Comparison. In: 1st Int. Workshop on Information Technology for Energy Applications, 2012
[13] IDA Indoor Climate and Energy: https://www.equa.se/de/ida-ice, (abgerufen am 21.9.2024 14:53)
[14] Modelica Tools: https://www.modelica.org/tools, (abgerufen am 21.9.2024 15:30)
[15] EnergyPlus: https://energyplus.net/, (abgerufen am 10.9.2024 14:20)
[16] TRNSYS: https://www.trnsys.de/, (abgerufen am 26.9.2024 10:05)
[17] W. Kastner und G. Neugschwandtner: Datenkommunikation in der verteilten Gebäudeautomation. Bulletin SEV/VSE 17:9–14, 2006
[18] ISO 16484-5: Systeme der Gebäudeautomation - Teil 5: Datenkommunikationsprotokoll, 2017
[19] ISO/IEC 14543-x: Information technology - Home electronic system (HES) architecture, 2006
[20] ISO/IEC 14908-x: Information technology -- Control network protocol, 2012
[21] IEC 62386-x: Digital addressable lighting interface, 2020
[22] Standard Motor Interface SMI: https://standard-motor-interface.com/, (abgerufen am 26.9.2024 10:05)
[23] ZigBee Alliance: https://zigbeealliance.org/, (abgerufen am 27.9.2024 09:05)
[24] DIN EN 13757-x: Kommunikationssysteme für Zähler, 2015
[25] VDI 3813 2: Gebäudeautomation (GA) - Raumautomationsfunktionen (RA-Funktionen), 2011
[26] VDI 3813-3: Gebäudeautomation (GA) - Anwendungsbeispiele für Raumtypen und Funktionsmakros in der Raumautomation, 2015
[27] G. F. Schneider, P. Pauwels, und S. Steiger: Ontology-Based Modeling of Control Logic in Building Automation Systems. In: IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, Volume 13, No. 6, 2017 pp. 3350–3360.

[28] W. Kastner, A. Fernbach, und W. Granzer: Secure and Semantic Web of Automation, https://energiewelten.tuwien.ac.at/fileadmin/t/eu/extern/Dokumente/Veranstaltungen/20150928_Blickpunkt_Forschung/A3_Kastner_Fernbach.pdf, (abgerufen am 31.1.2021 09:00)

[29] C. Reinisch, W. Granzer, F. Praus und W. Kastner: Integration of Heterogeneous Building AutomationSystems using Ontologies. In: 34th Annual Conference of IEEE Industrial Electronics, Orlando, FL, USA, 2008, S. 2736-2741

[30] I. Pelesi, A. Fernbach, W. Granzer und W. Kastner: Semantic Integration in Building Automation A Case Study for KNX, oBIX and Semantic Web Applications. In: Proceedings of the KNX Scientific Conference, 2016

[31] F. Knorr, J. Pannosch und G. Steindl: A Model-Driven Approach for Consistent Building Automation Control Logic Design and Deployment. In: 2024 International Conference on Control, Automation and Diagnosis (ICCAD), Paris, France, 2024, S. 1-6, doi: 10.1109/ICCAD60883.2024.10553923

[32] VDI 3814-1: Building automation and control systems (BACS) – Fundamentals, The Association of German Engineers, Beuth Verlag, 2019

[33] DIN EN ISO 16484-1: Building automation and control systems (BACS) - Part 1: Project specification and implementation (ISO 16484-1:2010). Beuth Verlag, 2011

[34] VDI 3813-1: Building automation and control systems (BACS) - Fundamentals for room control, Beuth Verlag, 2011

[35] VDI 3814-6: Building automation and control systems (BACS) - Graphical description of logic control tasks, The Association of German Engineers, Beuth Verlag, 2008

[36] L. Andolfato, G. Chiozzi, und C.A. Morales: A platform independent framework for statecharts code generation, 2011

[37] IEC 61499-1: Function blocks - Part 1: Architecture. International Electrotechnical Commission: Geneva, Switzerland, 2012

[38] M.A. Mukhtar, A.B. Abdullah und A.G. Downe: Preliminary overview about relations QVT: Query/View/Transformation model transformation language. In: 2011 National Postgraduate Conference, 2011, S. 1-5

[39] I. Ayala: Model Driven Development of Agents for Ambient Intelligence. PhD Thesis, Malaga, Spain, 2013

[40] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev: ATL: A model transformation tool. In: Science of Computer Programming, Volume 72, Issues 1–2, 2008, S. 31-39

[41] M. W. Peter Römisch: Projektierungspraxis Verarbeitungsanlagen. Springer Fachmedien Wiesbaden GmbH, 2014

[42] Bundesministerium für Finanzen: Gesamte Rechtsvorschrift für Arbeitsstättenverordnung. https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10009098, (abgerufen am 20.03.2023 11:05)

[43] D. Juhl: Technische Dokumentation. Praktische Anleitungen und Beispiele. Springer, Berlin, 2005

[44] VDI 3681: Einordnung und Bewertung von Beschreibungsmitteln aus der Automatisierungstechnik, The Association of German Engineers, Beuth Verlag, 2005

[45] S. A. Sinan: Guide to applying the UML. Springer, ISBN 978-0-387-95209-3, 2002

[46] C. Bauer und A. Sarkany: Endbericht Ganzheitliche Gebäudesimulation (FFG Projektnr: 899637), 2024

Abkürzungsverzeichnis

Abk.	Abkürzung
ATL	Atlas Transformation Language
BIM	Building Information Modeling
csv	Comma-separated-values
EA	endliche Automaten
ECC	Execution Control Chart
ETL	Epsilon Transformation Language
FEEI	Fachverband der Elektro- und Elektronikindustrie (feei.at)
HKLS	Heizung, Klima, Lüftung und Sanitär
ISO	International Standards Organisation
JSON	JavaScript Object Notation
MDE	Model Driven Engineering
MOF	Meta Object Facility
MOFM2T	MOF-Model to Text Transformation Language
MSR	Mess-, Steuer- und Regeltechnik
MTL	Modelltransformationssprache / Model Transformation Language
M2M	Modell zu Modell / Model-to-Model
M2T	Modell zu Text / Model-to-Text
NMF	Neutral Model Format
OMG	Object Management Group
OWL	Web Ontology Language
QVT	Query/View/Transformation
RDF	Resource Description Framework
SCXML	State Chart XML
Simultan MDM	Meta Datenmodell Simultan
TGA	Technische Gebäudeausrüstung
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieure
W3C	World Wide Web Consortium

XMI	XML Metadata Interchange
XML	extensible Markup Language

9 Anhang

9.1. Data Management Plan (DMP)

Das Projekt openBAM hat für die Evaluierung der Methodik Programmiercode erzeugt. Nachfolgend sind die Repositorien angegeben, wo die Daten verfügbar sind.

1: Datenerstellung und Dokumentation
Das Simultan MDM (Tabelle 4) und die graphische Benutzeroberfläche von Simultan (Tabelle 5) wurden erweitert. Dabei wurden auch Entwicklungen aus Vorprojekten genutzt, wobei sich die Version, aber nicht das hier angeführte Repository geändert hat.

Tabelle 4: Simultan Data Model Repository mit dem Source-Code des Simultan Meta Datenmodells

Attribut	**Beschreibung**
ID	SIM1
Titel	Simultan Data Model Repository
Zusammenfassung	GitHub Repository mit Open-Source Code des Simultan Meta Datenmodells
Kurzbezeichnung	Simultan Data Model Repository
Zeitraum und Referenzjahr	Entwickelt seit 2017
Institution	TU Wien, Instituts für Werkstofftechnologie, Bauphysik und Bauökologie, Forschungsbereich Bauphysik https://www.tuwien.at/cee/mbb/bph
Kontakt	Bernhard Steiner (bernhard.steiner@tuwien.ac.at)
Mitwirkende und Rolle	Thomas Bednar: Konzeption Datenmodell Galina Paskaleva: Konzeption Datenmodell und Softwareentwicklung Bernhard Steiner: Softwareentwicklung Jenö Pazmandi: Softwareentwicklung Jakob Pernsteiner: Softwareentwicklung
Kommentar	Das Simultan Datenmodel wurde gemeinsam mit dem Simultan ComponentBuilder (Editor) im Rahmen mehrerer Forschungsprojekte seit 2017 entwickelt. Im Rahmen des OpenBAM Projekts wurden neue Features eingefügt und einige bestehende Features erweitert
Erstellungsdatum	2017-heute
Datentyp	Text/Source Code sowie Dokumentation
Zugang	Öffentlich (MIT Lizenz) https://github.com/bph-tuwien/SIMULTAN

Tabelle 5: Simultan Component Builder Repository für die graphische Bearbeitung der Daten

Attribut	**Beschreibung**
ID	SIM2
Titel	Simultan ComponentBuilder Repository
Zusammenfassung	GitHub Repository mit Source Code für den Simultan Component Builder (Editor)
Kurzbezeichnung	Simultan ComponentBuilder Repository
Zeitraum und Referenzjahr	Entwickelt seit 2017
Institution	TU Wien, Instituts für Werkstofftechnologie, Bauphysik und Bauökologie, Forschungsbereich Bauphysik https://www.tuwien.at/cee/mbb/bph
Kontakt	Bernhard Steiner (bernhard.steiner@tuwien.ac.at)
Mitwirkende und Rolle	Thomas Bednar: Konzeption Datenmodell Galina Paskaleva: Konzeption Datenmodell und Softwareentwicklung Bernhard Steiner: Softwareentwicklung Jenö Pazmandi: Softwareentwicklung Jakob Pernsteiner: Softwareentwicklung
Kommentar	Das Simultan Datenmodel wurde gemeinsam mit dem Simultan ComponentBuilder (Editor) im Rahmen mehrerer Forschungsprojekte seit 2017 entwickelt. Im Rahmen des OpenBAM Projekts wurden neue Features eingefügt und einige bestehende Features erweitert
Erstellungsdatum	2017-heute
Datentyp	Text/Source Code sowie Dokumentation, Tools zur Generierung und Verteilung von Releases
Zugang	Nicht öffentlich

Tabelle 6 zeigt die erstellten Dateien, welche für die Evaluierung erstellt wurden. Tabelle 7 zeigt alle Informationen zum erstellten Repository im Zuge der Entwicklung des openBAM Editors.

Tabelle 6: Erstellte Dateien für die Abbildung der MSR und des digitalen Datenmodells

ID	**Beschreibung**	**Typ**	**Art**	**Verantwortlich**	**Zugang**
UML-Zustands-diagramm	Zustandsgraph der modellierten Regelung	Text	UML	Forschungsbereich Automation Systems – Jürgen Pannosch	nicht öffentlich
IEC61131	Zustandsgraph exportiert in Structural Text nach IEC61131	Structural Text	Text	Forschungsbereich Automation Systems – Jürgen Pannosch	nicht öffentlich
NMF	Zustandsgraph exportiert nach NMF um es in Simultan für die Simulation zu nutzen	Code	NMF	Forschungsbereich Automation Systems – Jürgen Pannosch	nicht öffentlich

Labor_Simultan	Simultan File mit dem Netzwerk und der Abbildung des Laborversuchs	Code	simultan	Forschungsbereich Bauphysik - Sabine Sint	nicht öffentlich
Engineering_Beschreibung	Anweisungsbeschreibung für die Umsetzung in der Case Suite	Text	pdf	SAUTER – Jürgen Kromp	nicht öffentlich

Tabelle 7: openBAM Editor Repository

Attribut	**Beschreibung**
ID	openBAM
Titel	openBAM Editor
Zusammenfassung	Repository mit Source Code für die MSR Modellierungssprache, Modelltransformationen sowie Umsetzung des Laborversuchs
Kurzbezeichnung	openBAM Editor
Zeitraum und Referenzjahr	Entwickelt seit 2021
Institution	TU Wien, Forschungsbereich Automation Systems
Kontakt	Jürgen Pannosch (juergen.pannosch@tuwien.ac.at)
Mitwirkende und Rolle	Gernot Steindl: Konzeption Jürgen Pannosch: Konzeption, Softwareentwicklung Felix Knorr: Softwareentwicklung
Kommentar	Der openBAM Editor wird genutzt zur Abbildung der MSR Logik auf Basis eines Simultan XMI Files. Dieser wurde im Rahmen des Projekts entwickelt.
Erstellungsdatum	2021-2024
Datentyp	Text/Source Code sowie Evaluierungsbeispiele
Zugang	Nicht öffentlich

2: Ethische, rechtliche und Sicherheitsaspekte

Die vorliegenden Daten unterliegen keinen Persönlichkeitsrechten, da keine personenbezogenen Daten genutzt wurden. Die Evaluierung wurde anhand einer Lüftungsanlage im Labor durchgeführt und somit gibt es auch keine Rückschlüsse auf real betriebene Anlagen.

3: Datenspeicherung und -erhalt

Der Datenaustausch zwischen den Projektpartnern erfolgt über eine TU-interne Cloud-Lösung („TU Pro Cloud"). Die Forschungsdaten werden am Server des Forschungsbereichs Bauphysik gesichert. Die Repositorien werden einerseits auf GitHub und andererseits direkt an der TU Wien bei dem Forschungsbereich Automation Systems gehostet. Beim Server wird täglich ein Backup erstellt, verantwortlich ist der Administrator des Forschungsbereichs Automation Systems.

4: Wiederverwendbarkeit der Daten

Das Simultan Meta Datenmodell ist frei verfügbar und kann genutzt werden. Zugriff zum openBAM Editor und zum Component Builder kann nach Rücksprache gewährt werden. Die Umsetzung in Richtung Funktionsbeschreibung für die Case Suite von SAUTER ist nicht frei verfügbar, da es sich hier um eine proprietäre Lösung für eine Firmeneigene Software handelt.

9.2. Modelle und Transformationen Source Code

9.2.1. UML XMI Modell

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="20131001" xmlns:xmi="http://www.omg.org/spec/XMI/20131001"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" xmlns:uml="http://www.eclipse.org/uml2/5.0.0/UML"
xmi:id="_OvvPkIBGEeymzYHjTTooxg" name="Example">
 <packageImport xmi:type="uml:PackageImport" xmi:id="_O03AIIBGEeymzYHjTTooxg">
  <importedPackage xmi:type="uml:Model"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#_0"/>
 </packageImport>
 <packagedElement xmi:type="uml:StateMachine" xmi:id="_UYbWYIBGEeymzYHjTTooxg" name="StateMachine">
  <region xmi:type="uml:Region" xmi:id="_UuftAIBGEeymzYHjTTooxg" name="Region1">
   <transition xmi:type="uml:Transition" xmi:id="_dBpO8IBGEeymzYHjTTooxg" name="[getSystemState() ==
ON]/setDampers(PREHEATING),&#xA;setPreheaterValve(PREHEATING),&#xA;setPreheaterPump(ON),&#xA;triggerT1
()" source="_XL9N8IBGEeymzYHjTTooxg" target="_Zg2BsIBGEeymzYHjTTooxg"/>
   <transition xmi:type="uml:Transition" xmi:id="_pLPeAIBGEeymzYHjTTooxg" name="[getSystemState() == OFF]"
source="_nmREQIBGEeymzYHjTTooxg" target="_XL9N8IBGEeymzYHjTTooxg"/>
   <transition xmi:type="uml:Transition" xmi:id="_uwoZMIBGEeymzYHjTTooxg" name="[getT1() ==
0]/setDampers(CONTROL),&#xA;setPreheaterValve(CONTROL),&#xA;setPreheaterPump(ON),&#xA;setFans(ON)"
source="_Zg2BsIBGEeymzYHjTTooxg" target="_nmREQIBGEeymzYHjTTooxg"/>
   <transition xmi:type="uml:Transition" xmi:id="_yf9xsIBGEeymzYHjTTooxg"
source="_WhEd4IBGEeymzYHjTTooxg" target="_XL9N8IBGEeymzYHjTTooxg"/>
   <subvertex xmi:type="uml:Pseudostate" xmi:id="_WhEd4IBGEeymzYHjTTooxg" name="Start"/>
   <subvertex xmi:type="uml:State" xmi:id="_XL9N8IBGEeymzYHjTTooxg" name="SystemOff"/>
   <subvertex xmi:type="uml:State" xmi:id="_Zg2BsIBGEeymzYHjTTooxg" name="StartUpMode"/>
   <subvertex xmi:type="uml:State" xmi:id="_nmREQIBGEeymzYHjTTooxg" name="NormalMode"/>
  </region>
 </packagedElement>
</uml:Model>
```

StateMachineUML.xmi

9.2.2. SCXML Modell

```
<scxml initial="SystemOff" version="1.0" xmlns="http://www.w3.org/2005/07/scxml">
 <state id="SystemOff">
  <transition cond="getSystemState() == ON" target="StartUpMode"></transition>
 </state>
 <state id="StartUpMode">
  <onentry>
   setDampers(PREHEATING)
   setPreheaterValve(PREHEATING)
   setPreheaterPump(ON)
   triggerT1()
  </onentry>
  <transition cond="getT1() == 0" target="NormalMode"></transition>
 </state>
 <state id="NormalMode">
  <onentry>
   setDampers(CONTROL)
   setPreheaterValve(CONTROL)
   setPreheaterPump(ON)
   setFans(ON)
  </onentry>
  <transition cond="getSystemState() == OFF" target="SystemOff"></transition>
 </state>
</scxml>
```

StateMachineSCXML.xml

9.2.3. UML Ecore Modell

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="uml"
nsURI="http://www.eclipse.org/uml2/5.0.0/UML" nsPrefix="uml">
 <eClassifiers xsi:type="ecore:EClass" name="StateMachine">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="region" upperBound="-1"
     eType="#//Region" containment="true"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Region">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="subvertex" upperBound="-1"
     eType="#//Vertex" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="transition" upperBound="-1"
     eType="#//Transition" containment="true"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Vertex">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="partOfSource" upperBound="-1"
     eType="#//Transition" eOpposite="#//Transition/source"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="partOfTarget" upperBound="-1"
     eType="#//Transition" eOpposite="#//Transition/target"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Transition">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="source" lowerBound="1"
     eType="#//Vertex" eOpposite="#//Vertex/partOfSource"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="target" lowerBound="1"
     eType="#//Vertex" eOpposite="#//Vertex/partOfTarget"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Pseudostate" eSuperTypes="#//Vertex">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="kind" eType="#//PseudostateKind"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="State" eSuperTypes="#//Vertex"/>
 <eClassifiers xsi:type="ecore:EEnum" name="PseudostateKind">
  <eLiterals name="initial"/>
  <eLiterals name="termination"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Model">
  <eStructuralFeatures xsi:type="ecore:EReference" name="packagedElement" upperBound="-1"
     eType="#//StateMachine" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
 </eClassifiers>
</ecore:EPackage>
```

UML.ecore

9.2.4. SCXML Ecore Modell

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="scxml"
nsURI="http://www.w3.org/2005/07/scxml" nsPrefix="scxml">
  <eClassifiers xsi:type="ecore:EClass" name="Scxml">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" unique="false"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="namespace" unique="false"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="state" upperBound="-1"
       eType="#//State" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="initial" unique="false"
       lowerBound="1" eType="#//State"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="State">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="id" unique="false" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="onEntry" upperBound="-1"
       eType="#//OnEntry" containment="true" eOpposite="#//OnEntry/state"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="transition" upperBound="-1"
       eType="#//Transition" containment="true" eOpposite="#//Transition/state"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="OnEntry">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="content" unique="false"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="state" unique="false" lowerBound="1"
       eType="#//State" eOpposite="#//State/onEntry"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Transition">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="cond" unique="false" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="target" unique="false"
       lowerBound="1" eType="#//State"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="state" unique="false" lowerBound="1"
       eType="#//State" eOpposite="#//State/transition"/>
  </eClassifiers>
</ecore:EPackage>
```

SCXML.ecore

9.2.5. UML2SCXML ATL Transformation

```
-- @path Uml=/Uml2Scxml/Metamodels/UML.ecore
-- @path Scxml=/Uml2Scxml/Metamodels/SCXML.ecore

module Uml2Scxml;
create OUT : Scxml from IN : Uml;

helper context Uml!Transition def: containsAction(): Boolean =
        if self.name.toString().endsWith('/') or self.name.toString().indexOf('/') = -1 then
                false
        else
                true
        endif;
```

```
helper context Uml!Transition def: action(): String =
	if self.name.indexOf('/') < 0 then
		''
	else
		self.name.substring(self.name.indexOf('/') + 2, self.name.size())
	endif;

helper context Uml!Transition def: condition(): String =
	if self.name.indexOf('/') < 0 then
		self.name.substring(1, self.name.size())
	else
		self.name.substring(1, self.name.toString().indexOf('/'))
	endif;

rule PseudoState2Scxml {
	from
		s: Uml!Pseudostate(s.kind = #"initial")
	to
		t: Scxml!State (
			id <- s.name
		),
		t2: Scxml!Scxml (
			namespace <- 'http://www.w3.org/2005/07/scxml',
			version <- 1,
			initial <- s,
			state <- Scxml!State.allInstances()
		)
}

rule State2State {
	from
		s: Uml!State
	to
		t: Scxml!State (
			id <- s.name
		)
}

rule Transition2Transition {
	from
		s: Uml!Transition(not s.source.oclIsTypeOf(Uml!Pseudostate))
	to
		t: Scxml!Transition (
			cond <- s.condition(),
			target <- s.target,
			state <- s.source
		),
		t2: Scxml!OnEntry (
			content <- s.action(),
			state <- s.source
		)
}
```

UML2SCXML.atl

9.2.6. XML Ecore Modell

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="xml" nsURI="http://www.w3.org/2005/07/xml"
nsPrefix="exml">
 <eClassifiers xsi:type="ecore:EClass" name="Node" abstract="true">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="startLine" ordered="false"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="startColumn" ordered="false"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="endLine" ordered="false"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="endColumn" ordered="false"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" lowerBound="1"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" ordered="false"
    lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="parent" ordered="false"
    eType="#//Element" eOpposite="#//Element/children"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Attribute" eSuperTypes="#//Node"/>
 <eClassifiers xsi:type="ecore:EClass" name="Text" eSuperTypes="#//Node"/>
 <eClassifiers xsi:type="ecore:EClass" name="Element" eSuperTypes="#//Node">
  <eStructuralFeatures xsi:type="ecore:EReference" name="children" upperBound="-1"
    eType="#//Node" containment="true" eOpposite="#//Node/parent"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Root" eSuperTypes="#//Element"/>
</ecore:EPackage>
```

XML.ecore

9.2.7. SCXML2XML ATL Transformation

```
-- @path Scxml=/Uml2Scxml/Metamodels/SCXML.ecore
-- @path Xml=/Uml2Scxml/Metamodels/XML.ecore

module Scxml2Xml;
create OUT : Xml from IN : Scxml;

helper context Scxml!Transition def: state() : Scxml!State =
        Scxml!State.allInstances() -> select(e|e.transition.includes(self))->first();

helper context Scxml!OnEntry def: state() : Scxml!State =
        Scxml!State.allInstances() -> select(e|e.onEntry.includes(self))->first();

rule Scxml2XmlRoot {
        from
                s: Scxml!Scxml
        to
                a1: Xml!Attribute(
                        name <- 'xmlns',
                        value <- s.namespace
                ),
                a2: Xml!Attribute(
                        name <- 'version',
                        value <- s.version.toString()
```

```
		),
		a3: Xml!Attribute(
			name <- 'initial',
			value <- s.initial.id
		),
		t: Xml!Root(
			name <- 'scxml',
			children <- Sequence { a1, a2, a3 }
		)
}

rule State2XmlElement {
	from
		s: Scxml!State
	to
		a: Xml!Attribute(
			name <- 'id',
			value <- s.id
		),
		t: Xml!Element(
			name <- 'state',
			children <- Sequence { a },
			parent <- Xml!Root.allInstances().first()
		)
}


rule Transition2XmlElement {
	from
		s: Scxml!Transition
	to
		a1: Xml!Attribute(
			name <- 'cond',
			value <- s.cond
		),
		a2: Xml!Attribute(
			name <- 'target',
			value <- s.target.id
		),
		t: Xml!Element(
			name <- 'transition',
			children <- Sequence { a1, a2 },
			parent <- Xml!Attribute.allInstances() -> select(e|e.name='id' and
e.value=s.state().id)->first().parent
		)
}

rule OnEntry2XmlElement {
	from
		s: Scxml!OnEntry
	to
		a: Xml!Text(
			value <- s.content
		),
		t: Xml!Element(
			name <- 'onentry',
			children <- Sequence { a },
```

```
                        parent <- Xml!Attribute.allInstances() -> select(e|e.name='id' and
e.value=s.state().id)->first().parent
                )
}
```

SCXML2XML.atl

9.2.8. XML ATL Abfrage (Query)

```
-- @path XML=/Uml2Scxml/Metamodels/XML.ecore

query XML2Text = XML!Root.allInstances()
                ->asSequence()
                ->first().Root().writeTo('/Uml2Scxml/Models/xml.xml');

helper context XML!Root def: Root() : String =
        '<?xml version="1.0"?>'+'\n'
        + self.toString2('');


helper context XML!Element def: toString2(indent : String) : String =
        let na : Sequence(XML!Node) =
                self.children->select(e | not e.oclIsKindOf(XML!Attribute)) in
        let a : Sequence(XML!Node) =
                self.children->select(e | e.oclIsKindOf(XML!Attribute)) in
        indent + '<' + self.name +
        a->iterate(e; acc : String = '' |
                acc + ' ' + e.toString2('')
        ) +
        if na->size() > 0 then
                '>'
                + na->iterate(e; acc : String = '' |
                        acc +
                        if e.oclIsKindOf(XML!Text) then
                                ''
                        else
                                '\r\n'
                        endif
                        + e.toString2(indent + '  ')
                ) +
                if na->first().oclIsKindOf(XML!Text) then
                        '</' + self.name + '>'
                        else
                                '\r\n' + indent + '</' + self.name + '>'
                endif
        else
                '/>'
        endif;


helper context XML!Attribute def: toString2(indent : String) : String =
        self.name + '=\"' + self.value + '\"';


helper context XML!Text def: toString2(indent : String) : String =
        self.value;
```

XML2Text.atl

9.2.9. SCXML2XML M2T Transformation

```
[comment encoding = UTF-8 /]
[module generate('http://www.w3.org/2005/07/xml')]


[template public generateScxml(root : Root)]
[comment @main /]
[file ('scxml.xml', false, 'UTF-8')]
<?xml version="1.0"?>
[generateElement(root)/]
[/file]
[/template]


[template public generateElement(element : Element)]
<[element.name/] [for (elemAttribute: Attribute | element.children->filter(Attribute)) separator('
')][elemAttribute.name/]="[elemAttribute.value/]"[/for]>
[for (elemElement: Element | element.children->filter(Element)) separator('')]
   [generateElement(elemElement)/]
[/for]
[for (elemText: Text | element.children->filter(Text)) separator('')]
   [elemText.value/]
[/for]
</[element.name/]>
[/template]
```

GenerateScxml.mtl