Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Design and Integration of Simulation Models for Industrial Systems

Doctoral Thesis

August 2016

Ing. Petr Novák

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

DESIGN AND INTEGRATION OF SIMULATION MODELS FOR INDUSTRIAL SYSTEMS

Doctoral Thesis

Ing. Petr Novák

Prague, August 2016

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of Study: Artificial Intelligence and Biocybernetics

Supervisor: Ing. Radek Šindelář, Ph.D.
Supervisor-Specialist: —

Acknowledgement

First and foremost, I would like to thank Silvie for her support. Further I would like to thank my family to enable my studies.

I would like to thank my colleagues from the research group "Intelligent Systems for Industry and Smart Distribution Networks" and its predecessor "Intelligent Systems Group" at the Czech Technical University in Prague. Mainly I would like to thank Prof. Vladimír Mařík for his support, doc. Pavel Vrba for coordination of the research group as well as my workmates Petr Kadera and Václav Jirkovský for discussions and cooperations in the areas of performance analysis and big data for industry.

This thesis was significantly influenced by the research and colleagues from the Vienna University of Technology. I would like to thank all colleagues from the research laboratory "Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems" (CDL-Flex) for discussions and feedback. Especially I would like to thank Prof. Stefan Biffl, who taught me how to write scientific papers and who provided me valuable feedback on my research and dissemination of its results. Moreover, I would like to thank Thomas Moser for cooperation in the area of semantic integration, Alois Zoitl impressing me by distributed intelligent control and industrial automation standards, Richard Mordinyi for cooperation in the areas of software engineering and Enterprise Service Buses, and Fajar Juang Ekaputra for common work in the AutomationML area. Last but not least, I would like to thank the external partner of the laboratory Prof. Arndt Lüder, who influenced me with his enthusiasm about AutomationML and its industrial applications.

During the first two years of my PhD studies, I was able to stay in the Rockwell Automation Research and Development Center in Prague. I would like to thank my colleagues and partners there, especially Marek Obitko for feedback from the industrial perspective.

Finally, I would like to thank my supervisors, doc. Petr Horáček for introducing me into research and giving me the opportunity to participate in the CDL-Flex and Radek Šindelář.

The research done behind this thesis has been supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy, Family and Youth, and the National Foundation for Research, Technology and Development – Austria; and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS12/188/OHK3/3T/13.

Petr Novák

Czech Technical University in Prague
Prague, 2016

Abstract

Industrial systems are becoming complex and large-scale. Optimization of their operation and testing of their control systems are done on simulation models frequently, because simulated experiments are faster, cheaper, and repeatable compared to experiments done on real industrial plants. However, design and re-design of simulation models are difficult and time-consuming tasks. In addition, integration of simulation models within industrial automation systems is not satisfactory nowadays. This thesis is aimed at improving the design and integration phases of the simulation model life-cycle.

In the area of the simulation model design, especially a component-based approach for simulation model creation is investigated and improved in this thesis. It assumes that engineering systems consist of atomic components that are connected into topologies of real industrial plants. The proposed method supports assembling simulation models from simulation components, which can be reused from previous simulation projects. Each real device can be simulated by one of the available implementations of the component, representing this device. The proposed solution is based on the utilization of the bond-graph theory to guarantee the compatibility of the interfaces of the connected component implementations and to support their selection. In addition, the bond-graph theory is used to support splitting a simulation model into a set of simulation modules and their integration into a simulation workflow. For all of these types of tasks, the bond-graph theory was enhanced with an explicit description of component interfaces and a new causality assignment algorithm was designed. This algorithm can be used not only for generation of simulation models, but also for verifications on a conceptual planning level, whether specific sets of simulation component implementations are sufficient to model particular plants.

In the area of the simulation model integration, two research threads are followed. The first one is related to formalizing, capturing, and integrating knowledge about the real industrial plant, input and output tags, parameters of devices, and mappings of all these entities to simulation model components, variables, and parameters. Such engineering knowledge is used to support simulation model design and maintenance of existing simulation models when a real plant is changed. The second thread in the integration area is focused on interoperability of simulation modules on the level of the supervisory control and data acquisition of the automation pyramid. This task covers the access of simulations to runtime data, improved parameter setting, and version-control of simulation modules.

This thesis contributes to the areas of the simulation modeling, knowledge representation, and distributed system integration. The most important results are (i) adaptation of the bond graph theory for non-traditional applications including selection of explicitly specified component implementations as well as a new causality assignment algorithm supporting this approach, (ii) utilization of ontologies for supporting simulation model design and integration, and (iii) improved simulation model integration.

Anotace

Průmyslové systémy se stávají komplexními a rozsáhlými. Optimalizace jejich provozu a testování jejich řídicích systémů jsou typicky podporovány simulačními modely, protože experimenty v simulovaném prostředí jsou oproti experimentům na skutečných průmyslových systémech rychlejší, levnější a opakovatelné. Avšak návrh a přebudování simulačních modelů jsou obtížnými a časově náročnými úlohami. Rovněž integrace simulačních modelů v rámci průmyslových automatizačních systémů dnes není dostatečná. Tato disertační práce se zaměřuje na zlepšení fází návrhu a integrace simulačních modelů v rámci jejich životního cyklu.

V oblasti návrhu simulačních modelů je v této práci diskutován a zlepšen zejména přístup založený na komponentách. Předpokládá se, že technické systémy se skládají z atomických komponent, které jsou navzájem propojeny do topologií skutečných průmyslových systémů. Navržená metoda podporuje skládání simulačních modelů ze simulačních komponent, které mohou být znovu použity z předchozích simulačních projektů. Každé skutečné zařízení může být simulováno jednou z dostupných implementací komponenty, reprezentující toto zařízení. Navržené řešení je založeno na použití teorie vazebních výkonových grafů. Tento typ grafů zajišťuje kompatibilitu rozhraní spojených implementací komponent a usnadňuje jejich výběr. Teorie vazebních výkonových grafů je rovněž použita pro podporu rozdělení simulačního modelu na množinu simulačních modulů a jejich integraci do simulačního celku. Pro všechny tyto typy úloh byla teorie vazebních výkonových grafů rozšířena o explicitní popis rozhraní komponent a byl navržen nový algoritmus pro přiřazení kauzalit. Tento algoritmus může být použit nejen pro generování simulačního modelu, ale také pro verifikaci na úrovni konceptuálního plánovaní, zda je daná množina implementací simulačních komponent dostačující pro modelování konkrétního systému.

V oblasti integrace simulačních modelů probíhal výzkum dvěma směry. První z nich souvisí s formalizací, uchováním a integrací znalostí o skutečném průmyslovém systému, vstupních a výstupních tagách, parametrech zařízení a mapovaní všech těchto entit na simulační komponenty, proměnné a parametry. Takovéto inženýrské znalosti jsou použity pro podporu návrhu simulačních modelů a údržby existujících simulačních modelů, když je reálný systém změněn. Druhý směr v oblasti integrace je zaměřen na interoperabilitu simulačních modulů na úrovni supervizorového řízení a sběru dat v rámci automatizační pyramidy. Tento úkol zahrnuje přístup k provozním datům, vylepšené nastavování parametrů a verzovaní simulačních modulů.

Tato disertační práce přispívá do oblastí simulačního modelování, reprezentace znalostí a integrace distribuovaných systémů. Nejdůležitějšími výsledky jsou (i) adaptace teorie vazebních výkonových grafů pro netradiční použití zahrnující výběr explicitně specifikovaných implementací komponent, stejně jako nový algoritmus pro přiřazení kauzalit umožňující tento přístup, (ii) použití ontologií pro podporu návrhu a integrace simulačních modulů a (iii) vylepšená integrace simulačních modelů.

Contents

1 Introduction **1**

1.1 Simulation Models for Industrial Processes 2
1.2 Integration of Simulation Models within Industrial Automation Systems . . 3
1.3 Goals of this Thesis from the High-Level Perspective 6
1.4 Research Issues Addressed in this Thesis 7
1.5 Structure of the Thesis 8

2 Current Status of Design and Integration of Simulation Models **9**

2.1 Dynamic Systems 9
2.2 Simulation Models 10
2.3 Bond Graphs 11
2.3.1 Signal Analogies 11
2.3.2 Component Analogies 12
2.3.3 Connection Analogies 13
2.3.4 Creating Bond Graphs 14
2.3.5 Tool Support for Bond Graph Modeling 16
2.4 Functional Mockup Interface 19
2.5 Architectures of Industrial Automation Systems 20
2.6 Current Status Summary Motivating the Thesis 21

3 Related Work **23**

3.1 Automated and Semantic Simulation Model Design 23
3.2 Bond Graphs for Simulation Model Design 24
3.3 Integration of Simulation Models 25
3.4 Integration of Industrial SCADA Systems 26
3.5 Current Trends in System Integration 27
3.6 Enterprise Service Bus for System Integration 28
3.7 Semantic and Technical Levels of Integration 30
3.8 Semantic Web 30
3.9 Ontologies for Knowledge Bases 31
3.9.1 Ontologies and Description Logics 32
3.9.2 Ontology Languages 33

3.9.3 Querying of Ontologies . . . 34
3.9.4 Tool Support for Ontologies . . . 34
3.10 Existing Ontologies for Knowledge Representation . . . 35
3.11 Process Data Representation and Big Data . . . 36
3.12 Industrial Standards for Integration and Communication in Automation . . 36
3.13 Multi-Agent and Holonic Systems . . . 38
3.14 Semantic Technologies in Building Automation . . . 40

4 Knowledge Models for Improved Simulation Model Design and Integration 41
4.1 Engineering Disciplines and Engineering Plans . . . 41
4.2 Design of the Knowledge Base . . . 42
4.3 Requirements on the Ontology Model . . . 43
4.4 Automation Ontology . . . 44
4.4.1 Domains of the Automation Ontology . . . 44
4.4.2 Real Plant Domain . . . 45
4.4.3 Variable and Tag Domain . . . 45
4.4.4 Parameter Domain . . . 47
4.4.5 Simulation Domain . . . 47
4.4.6 Bond Graph Domain . . . 48
4.4.7 Summary and Evaluation of the Automation Ontology . . . 49
4.5 Software Prototype of the Ontology Tool . . . 50

5 Extended Bond Graphs for Object-Oriented Simulation Model Design 51
5.1 Design of Simulation Models . . . 51
5.1.1 Simulation Design Scenario when a Simulation Library is not Available 52
5.1.2 Simulation Design Scenario Based on an Available Simulation Library with Simulation Blocks . . . 53
5.2 Motivation for a New Method Supporting Multi-Level Object-Oriented Simulation Modeling . . . 53
5.3 Motivation for a Simulation Block Selection . . . 54
5.4 Simulation Block Selection for SISO Blocks and Serial Connections . . . 56
5.5 Motivation for the Use of the Bond-Graph Theory . . . 57
5.6 Motivation for a New Causality Assignment Algorithm . . . 58
5.7 Extended Bond Graphs Enhanced with Explicit Simulation Block Support . 59
5.7.1 Formal Specification of the Simulation Model Design Task . . . 60
5.7.2 Extended Bond Graph Method . . . 60
5.7.3 Proposed Method in an Algorithmic Way . . . 62
5.7.4 Output of the Extended Bond Graph Method . . . 64
5.8 Electrical Circuit Example . . . 65
5.9 Verification of the Generated Simulation Model for the Electrical Circuit . . 68

5.10 Evaluation of the Proposed Method: Benefits and Weak Points 74
5.11 Semi-Automated Generation of Simulation Module Interfaces Using Extended Bond Graphs 76
5.11.1 Prerequisites of the Simulation Splitting Support 78
5.11.2 Cuts on the Junction Level 78
5.11.3 Cuts on the Power Bond Level 79
5.11.4 Example of Integrating Junctions and Evaluation 80
5.12 Execution of Complex Coupled Simulations at Runtime 81
5.13 Optimization of Complex Simulation Model Execution 83
5.14 Developed Tool Support for the Simulation Model Generation Based on Extended Bond Graphs 88

6 Improved Integration of Simulation Models 91
6.1 Requirements and Challenges on Integrated Automation Systems 91
6.2 Proposed Architecture of the Integrated SCADA Level of Automation Systems 92
6.3 Engineering Tool Domain 94
6.3.1 Connector to Microsoft Visio 95
6.3.2 AutomationML Connector 98
6.4 Simulation Domain 101
6.4.1 MATLAB-Simulink Connector 102
6.4.2 Other Simulation Tool Connectors 104
6.5 SCADA System Domain 104
6.5.1 SCADA Systems – HMI Domain 104
6.5.2 ScadaBR Tool Connector 105
6.5.3 Promotic Tool Connector 105
6.5.4 SCADA Systems – Data Acquisition Domain 106
6.5.5 SCADA Systems – Multi-Agent System Domain 106
6.6 Processes for Simulation Design and Integration 107
6.7 Integration of Simulations and SCADA Systems from the Process Perspective 109

7 Use-Cases and Experiments 113
7.1 Passive House Simulation Use-Case 113
7.1.1 Motivation for the Passive House Simulation Use-Case 113
7.1.2 Passive House Standard 114
7.1.3 Measuring and Control in Passive Houses 114
7.1.4 Simulation Modeling of Houses 114
7.1.5 Experimental Passive House 117
7.1.6 Semi-Automated Design of Simulation Models for a Passive House . 120
7.1.7 Lessons Learned and Evaluation of the Passive House Use-Case . . . 121
7.2 Hydraulic Network Use-Case 123
7.2.1 Simulation Library for Hydraulic Systems 123

7.2.2 Simulation Models for the Tank Model 125
7.2.3 Generation of the Lists of Simulation Parameters and Tags 130
7.2.4 Simulation Model Testing and Comparison of Measured and Simulated Experiments 130
7.2.5 Lessons Learned and Evaluation of the Reached Results 132

8 Conclusions and Future Work **135**
8.1 Fulfillment of the Thesis Goals 135
8.2 Scientific Contributions Reached in the Thesis 136
8.3 Future Work 137

Biblioghraphy **139**

A Application Example of the Traditional Bond Graph Method for Simulation Design **I**

B Simulation Blocks of the Mechatronic Library **VII**

C Screenshots of the Tool Support **XXXV**

D List of the Author's Publications **XXXIX**

Chapter 1

Introduction

Current industrial systems are becoming complex and large-scale. Design and testing of industrial plants including their automation and control systems are thus getting difficult and time-consuming tasks that can no longer rely on manual work only.

Computer simulation of the behavior of industrial plants is becoming an important part of system engineering as simulations facilitate industrial plant testing and optimization. This thesis contributes to improvements of the design phase of simulation models and their better integration into industrial automation environments, which are weak points of current simulations and their use.

In a broad context, "virtualization" is the term related to the upcoming Factories of the Future [49] as well as the fourth industrial revolution changing current industrial facilities to be more flexible and better integrated. This movement is referred as "Industry 4.0" or "Industrie 4.0" in the original German transcription. In conjunction with process simulations, the virtualization is frequently referred in terms of virtual commissioning of industrial plants. It is based on the utilization of simulation models to inspect, to test and to optimize the behavior of real industrial systems [44]. Since this thesis is focused on improving simulation model design and integration, it contributes to the area of virtual commissioning of industrial plants as well.

To reduce repeating manual work needed for engineering automation systems and simulation models as well as for their integration, knowledge representation and data integration are becoming important aspects related to modern industrial automation systems as well as to engineering processes of industrial plants and simulation models for these plants. Although the term integration is one of the key terms in software engineering for several decades, sharing knowledge and data in automation systems engineering is still an emerging topic that needs improvements.

The weak integration is most likely coming from the fact that current industrial plants have a mechatronic nature frequently. Mechatronic systems are featured with engineering based on collaborative work of several engineering disciplines [5, 51]. At the design phase of the automation system life-cycle, engineers of various engineering disciplines utilize diverse software tools. These tools are hereinafter called engineering tools. Their purpose is to support describing the real system from the perspective of the specific engineering discipline. Nowadays, the engineering tools are not integrated properly. The design phase of mechatronic systems can be thus expressively summarized as a kind of "*Engineering Polynesia*" having islanded tools with interfaces that do not fit seamlessly and an "*Engineering Baby-*

lon", where engineering artifacts are represented in various ways in engineering tools [34]. The engineering data sharing between engineering disciplines in mechatronic projects is needed, but it has not been met satisfactorily. When cooperating between several engineering disciplines and sharing knowledge, important pieces of information get lost in current industrial automation projects, which causes unwanted delays in automation systems engineering projects [50].

The integration problems do not emerge in the design phase only (i.e., as it has been introduced above), but the position of simulations is similar at automation system runtime as well. Typically, simulation models are not integrated within the remainder of the automation system, which causes barriers for their efficient utilization, such as for training process operators or supporting decision making at industrial plant runtime. At best, it is possible to visualize simulated data in a standard human-machine interface (HMI) as a part of a Supervisory Control and Data Acquisition (SCADA) system and to import runtime samples into the simulation. However in current industrial systems, software architectures enabling these tasks are either missing or they are neither satisfactory nor general enough. The existing architectures for industrial simulation integration are difficult to maintain and their scopes are only partial in terms of limited access to data sources, initial conditions setting, and others. Since the simulation model structure adopts the structure of the real plant or its sub-part, the design and integration of simulations are strongly coupled issues. Hence, this thesis handles both parts as crucial issues of the complete simulation model life-cycle.

1.1 Simulation Models for Industrial Processes

In the past, the behavior of real systems including their control systems was analyzed mathematically. Unfortunately, analytical methods cannot be used for large-scale cyber-physical systems efficiently because of the high number of heterogeneous components, tags, and parameters. Due to security and cost reasons, experiments should not be done on real systems directly. Moreover, experiments on real systems need not be repeatable (due to changes of boundary and initial conditions), and they can be very time-consuming in many cases. Therefore, simulation models are useful test-beds, simulating the real industrial systems under typical, extreme, or other measured or artificial conditions.

Simulations are useful tools for key tasks in the manufacturing value chain [81]. They can be used to improve the sustainable operation of real plants, to reduce waste, or to save energy. However, current simulation approaches suffer from (i) a complicated design phase and (ii) a problematic integration with other systems related to the design and tool integration for industrial plants. Even though companies and researchers focused on industrial automation emphasize the need for increasing the integration and reuse of codes, algorithms, and other engineering artifacts, such needs are not met in existing simulations. The engineering process of simulation models should be improved in order to bring the simulation benefits into daily industrial practice and into our daily lives.

The method proposed in this thesis should cover not only one specific simulation environment, but it should support all types of process models including dynamic [13], event-based [33], or rule-based [7] models and simulations. However, bridging all these types of simulation environments implies several research challenges that could not be fully addressed in this single thesis. The majority of the presented considerations and experiments

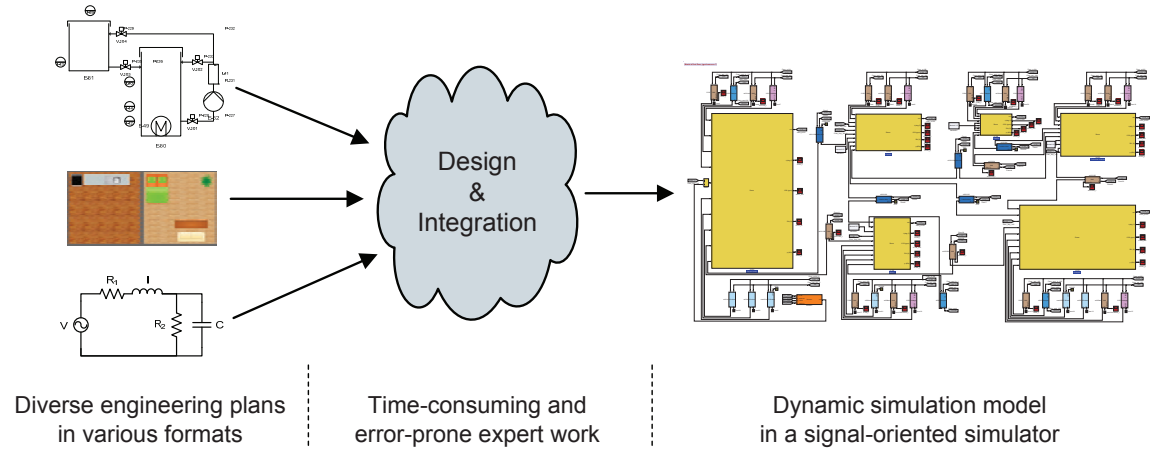

Figure 1.1: Conceptual high-level overview of the proposed improvement of the simulation model design phase.

has been done in the area of signal-based simulations of dynamic industrial systems.

Current design of simulations is based on manual merging pieces of information from various engineering plans, such as electrical plans, piping and instrumentation diagrams [92], information from SCADA systems, etc. To improve the design of simulation models, it is beneficial to integrate knowledge from various kinds of sources (e.g., plans or spreadsheets) and engineering tools. The integrated knowledge from the industrial plant engineering can be consequently reused for the specification of the simulation model structure as well as for the implementation of the model itself. A simplified process of designing integrated simulation models for industrial plants is depicted in Fig. 1.1. In this vision, the engineer focuses on the decision making in the engineer's area of interest and the repeating manual work is eliminated. The integrated computer-aided design and integration contributes to the avoidance of many kinds of errors and inconsistencies, which occur in current projects.

It is not efficient to develop monolithic simulation models for these systems any more. A current trend or in many cases a need is the distribution of simulation models into a set of inter-linked simulation modules. Input data as well as partial results have to be shared among these modules at runtime. The graph expressing the modules and the data flows between them is called a simulation workflow [127]. The executed simulation workflows are called coupled simulations. The simulation workflow is the description of modules and data exchange for coupled simulations. The modularization of simulations requires proper integration of simulation models on the levels of simulation modules and automation system data.

1.2 Integration of Simulation Models within Industrial Automation Systems

Simulation models can no longer be designed and operated in an islanded mode from the perspective of entire industrial automation systems. They have to be integrated at runtime, i.e., the connection of runtime data from other tools to simulation models have to be established as well as other tools have to be supported to read simulation results. This

integration cannot be done ad-hoc by pairing long lists of variable names[1] manually. To be efficient, the integration should be semi-automated and driven by knowledge about the real system.

The very same simulation model should be used and reused for various runtime scenarios required for safe and efficient operation of the real plant:

- Design and testing of automation systems
- Operation analysis and optimization
- Training of human operators
- Estimation of unmeasured variables
- Decision-making support
- Job planning
- Model-based control
- Model-based fault detection

The basic distinction between the aforementioned operation scenarios is related to the way how simulations are integrated within industrial automation systems in terms of the usage of data. In other words, the very same simulation model can be used for various tasks in industrial automation. The introductory schematic requirement on simulation model integration is depicted in Fig. 1.2. A simulation model (in the figure represented as a model in MATLAB-Simulink[2]) should exchange tag values with the SCADA HMI and this data should be version-controlled. The crucial issue is the problem of timing and synchronization of this tag exchange. Since simulators are strongly influenced by the numerical stability of the model itself as well as relative and absolute precisions, the simulation time flows in different time steps. To improve the integration of simulations, it is needed to provide an infrastructure supporting data exchange between simulations and automation systems that automate the behavior and access to the industrial plants.

The operation of industrial plants is automated by automation systems. They have a hierarchically layered architecture, which is frequently called an automation pyramid. Many particular versions of the pyramid exist; one of its representations can be found in [60]. Although research effort as well as needs in industry tend to flatten the pyramid into a flexible dynamically reconfigurable middle-ware as a part of the Industry 4.0 movement, the solutions being used in industry nowadays still rely on the hierarchical structuring. Due to this fact, the classical layered architecture of automation systems is assumed in this thesis.

The automation pyramid depicted in Fig. 1.3 represents the view on the data architecture in automation systems considered in this thesis, which is described in details in Sec. 2.5. The figure includes the position of the contribution proposed by this thesis, which is depicted by dash-dot lines. The proposed simulation model design and integration is related to the third level of the pyramid, which represents a SCADA system [29]. A SCADA system is a system that is intended to provide access to industrial plants, both for human operators and for upper software systems. In this thesis, it is proposed to be extended with simulations.

[1] Variable names are called "tags" frequently in industrial automation.

[2] http://www.mathworks.com/products/simulink/

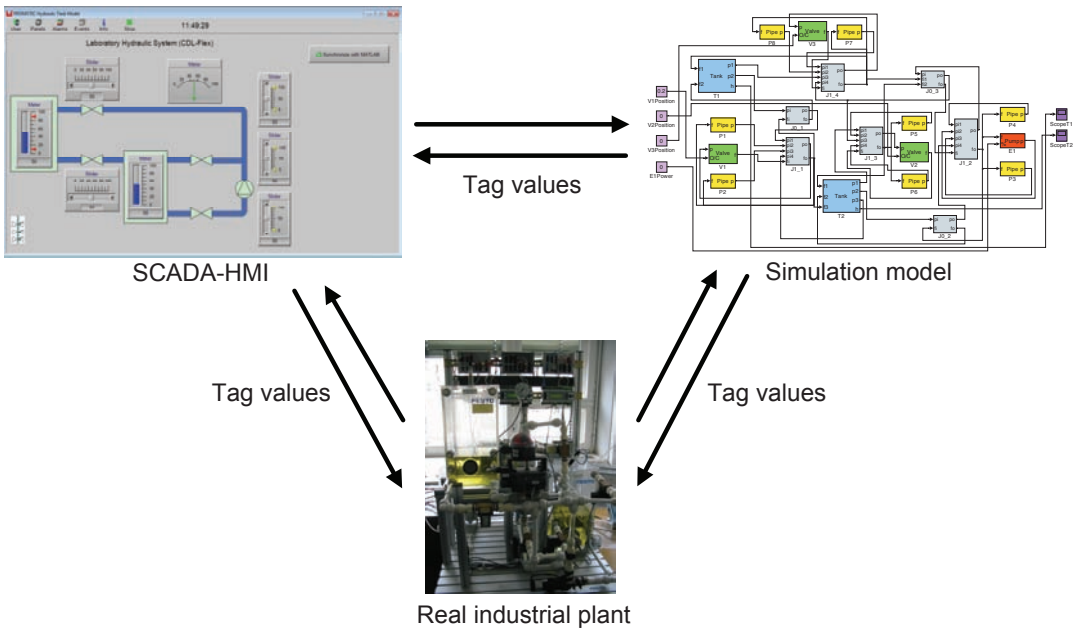

Figure 1.2: The basic idea of the integration of simulation models and SCADA systems at runtime. The utilized SCADA–HMI is called Promotic and it is discussed in Sec. 6.5.3, the specific simulation model is implemented in MATLAB-Simulink, which is addressed in Sec. 6.4.1, and the real industrial plant is an educational hydraulic tank model at the Vienna University of Technology [101, 120].

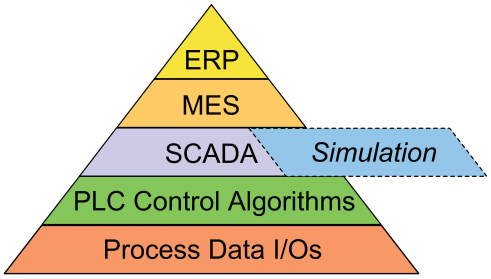

Figure 1.3: Automation pyramid enhanced with the integrated process simulation as it is investigated and proposed in this thesis.

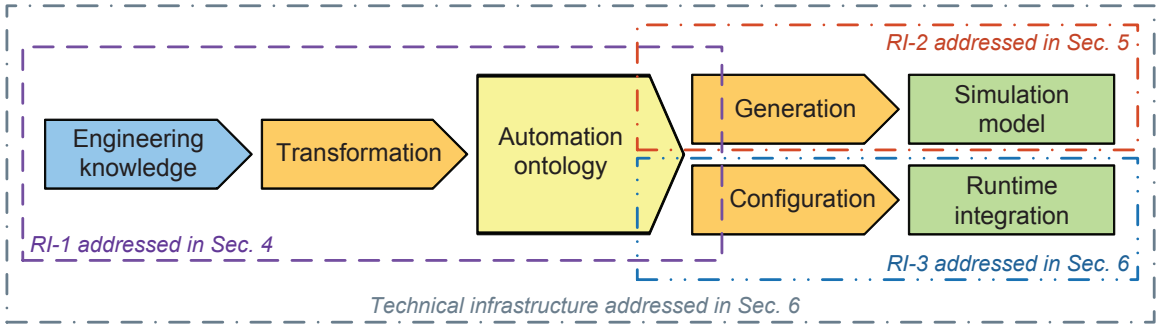

Figure 1.4: High-level overview of the design and integration processes, which are consequently addressed in this thesis.

The manual approach for integration of software systems and automation system tools is not sustainable for modern complex systems. Thus the proposed solution is based on the use of a knowledge base facilitating management of engineering knowledge. The knowledge base inter-relates facts about the structure of a real plant, a simulation model, as well as other automation tools, and knowledge about their interfaces. The knowledge base has to provide information in a computer-understandable form, i.e., the algorithms have to understand the semantics of the captured data. The knowledge base is not intended to be accessed by engineers directly, but it is encapsulated by an ontology tool, providing the access and the functionalities in a user-friendly form. It utilizes domain-specific languages (DSLs), i.e., each domain expert uses a terminology which is normal in the expert's discipline, and the mapping between particular DSLs is captured in the ontology.

The presented work is inter-disciplinary and it adopts methods from cybernetics, system theory, artificial intelligence, description logics, and simulation and control engineering. The research presented in this thesis is not isolated, but it is expected to be utilized in the "simulation integration framework" [127]. It is an emerging generic environment for seamless integration of simulation models within industrial automation systems being developed by industrial and research partners.

1.3 Goals of this Thesis from the High-Level Perspective

The overview of the proposed methodology from the knowledge and user points of view is depicted in Fig. 1.4. The goal of engineers is a running and integrated automation system including simulation models. The addressed process starts with capturing engineering knowledge (i.e., plans, information from SCADA systems, parameter databases, etc.) into a knowledge base. Consequently, required knowledge is retrieved in the appropriate form and used for the support of simulation model design as well as for the configuration of the technical level. The benefits of the proposed methodology are decreasing development and deployment time and costs, improving safety of solutions and making re-design and reuse of simulation models and other industrial tools or knowledge more flexible.

The research done within this thesis addresses the following high-level goals including various research and development problems:

- **G-1**: Representation of engineering knowledge for simulation design and integration

  Flexible design and integration of simulations requires proper classes, properties and

individuals in the automation ontology dedicated to these tasks. The first goal is related to the design, implementation and verification of the data model of the automation ontology, including considerations on a real project level.

- **G-2**: Object-oriented design of simulation models

  Simulation models are designed ad-hoc and they are difficult to maintain, modify and reuse nowadays. This goal targets proposing methodologies supporting simulation model design, which will be applicable in real industrial cases. An algorithm creating a simulation model based on available simulation libraries according to a real plant structure should be designed and implemented. The selection should be based on the compatibility of signals and the generalized physical behavior that is approximated by the simulation blocks.

- **G-3**: Design of simulation workflows consisting of simulation modules

  Simulation design process can be semi-automated and performed much faster than nowadays. This goal is focused on reusing the available knowledge to support splitting simulations into a set of simulation modules and specification of signals to be transferred between modules.

- **G-4**: Integration of simulations within SCADA systems

  Simulations are required to be integrated with measured data and human operators' environments frequently. This research goal covers integration with different subsystems of SCADA including HMIs, or data acquisition.

1.4 Research Issues Addressed in this Thesis

The goals of this thesis summarized in Sec. 1.3 include various implementation issues, considerations on the current state-of-the-art level, as well as research issues. Such research issues have to be investigated, addressed, and disseminated in relevant research communities. The following research issues have been identified and addressed in this thesis:

- **RI-1**: Development of a common model to capture knowledge in simulation projects

  Simulation of mechatronic systems and its engineering are complicated issues utilizing complex models and data models in current large-scale engineering projects. The approach discussed in this thesis should not be limited to particular data models, but it should provide foundations for supporting simulation/engineering projects in general.

- **RI-2**: Extension of the bond-graph theory for supporting explicitly pre-defined simulation components and required simulation modules

  The bond-graph theory is a paradigm for creating simulation models for mechatronic systems manually. However, current computer-centric approaches incorporating various engineering tools are not compliant with the bond-graph theory. Therefore, this research issue is focused on adapting the well-proven bond-graph theory for the needs of current engineering projects and tools.

- **RI-3**: Design of a model-based support for integration of simulations and SCADA systems

  The integration of simulation models within industrial SCADA systems is an important enabler for utilizing simulations effectively and efficiently. This research issue is focused on researching and developing model-based support for the configuration of simulation and SCADA system integration, which has to be tool-independent.

1.5 Structure of the Thesis

To address the high-level goals stated in Sec. 1.3 as well as the specific research issues formulated in Sec. 1.4, the remainder of this thesis has the following structure. Sec. 2 summarizes the current state-of-the-art in the areas of the simulation model design and industrial automation system architectures that are utilized in the industrial practice. This summary also practically motivates the challenges addressed in this thesis later.

Sec. 3 summarizes related work in various areas that are relevant for further descriptions of the author's contributions. Sec. 4 describes the author's contribution into the area of data modeling for mechatronic system engineering and simulation. It discusses the structure, use, and limitations of the proposed automation ontology in details. This section thus provides a solution for the research issue RI-1.

The most fundamental contributions of this thesis are included in the subsequent Sec. 5, which proposes an innovative use of bond graphs. Bond graphs are extended in order to support gray-box components, as well as to support separation of models into a set of simulation modules. The proposed methodology is a solution for the research issue RI-2. The section also shows how the proposed method facilitates design of module interfaces and their integration with glue modules for seamless integration modules into coupled simulations described by simulation workflows.

Sec. 6 describes the integration of simulations, industrial SCADA systems, and engineering tools. It discusses the author's contributions to the technical infrastructure utilizing the Engineering Service Bus as a specific and enhanced implementation of the Enterprise Service Bus concept. The section addresses the integration support both from the technical infrastructure point of view as well as from the perspective of processes making the design and integration of simulations more effective and efficient. Hence Sec. 6 addresses the research issue RI-3.

Later on, Sec. 7 illustrates the designed methodologies when using them for three practical use-cases covering design and integration of simulation models for various automation problems. Finally, Sec. 8 evaluates the reached results and the efficiency of the proposed methodologies, as well as it proposes promising topics for further work.

Chapter 2

Current Status of Design and Integration of Simulation Models

Contemporary design and integration of simulation models are inefficient tasks typically based on mathematical-physical description of the system or on measured responses of the real system. The integration of simulation models requires a manual configuration of signals to be transferred between stakeholders. This chapter discusses the problem of contemporary design and integration of simulation models in details as well as it provides foundations for the further explanation of aspects addressed in this thesis.

2.1 Dynamic Systems

Industrial plants are typically dynamic systems, i.e., they "*have a response to an input that is not instantaneously proportional to the input or disturbance and that may continue after the input is held constant*" [173]. More formally, continuous-time finite-dimensional dynamic systems are described by the following equations [8, 122]:

$$\begin{aligned} \dot{x}_i &= f_i(t, x_1, ..., x_n, u_1, ..., u_m) \quad i = 1, ..., n \\ y_j &= g_j(t, x_1, ..., x_n, u_1, ..., u_m) \quad j = 1, ..., p \end{aligned} \tag{2.1}$$

where u_k, $k = 1, ..., m$, denote inputs or stimuli; y_j, $j = 1, ..., p$, denote outputs or responses; x_i, $i = 1, ..., n$, denote state variables; t denotes time; $\dot{x}_i$ denotes the time derivative of x_i; f_i, $i = 1, ..., n$, are real-valued functions of $1 + n + m$ real variables; and g_j, $j = 1, ..., p$, are real-valued functions of $1 + n + m$ real variables [8, 122]. In the sense of Fig. 2.1, the inputs are a set of u_k, $k = 1, ..., m$. The outputs are a set of y_j, $j = 1, ..., p$. Hereinafter, we assume that the functions f_i, $i = 1, ..., n$ as well as g_j, $j = 1, ..., p$, can be parameterized with mathematical parameters that are real-valued constants. Such a parametrization supports adaptation of created simulation models or their parts to a wider class of problems without complicated re-design of the internal implementation of the model. As each of the functions can have an arbitrary set of parameters, we get a set of constant parameters c_l, $l = 1, ..., q$, where q is a number of parameters. A complete description of dynamic systems requires a set of initial conditions $x_i(t_0) = x_{i0}$, $i = 1, ..., n$, where t_0 denotes initial time [8, 122]. Initial conditions are considered as a special set of simulation model parameters in this thesis.

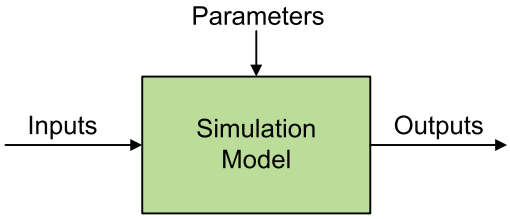

Figure 2.1: Simulation model interface: inputs, outputs, and parameters.

When the dynamic system is linear and time-invariant, it is frequently characterized by a transfer function. The transfer function describes the relationship between input and output signals of the system. The aforementioned state-space description can be transformed to the transfer function of the complex variable s in terms of the Laplace transform according to the equation:

$$G(s) = C(sI - A)^{-1}B + D \tag{2.2}$$

This expression will be used later as one of the ways, how a mathematical description can be transformed to the signal-oriented simulation form.

The transformation of the aforementioned differential equations to difference equations for the discrete-time finite-dimensional systems is straightforward. It can be found in [8], thus it is not discussed in more details here. The proposed method is intended for both, continuous-time and discrete-time finite-dimensional dynamical systems. The use-cases in Sec. 7 are continuous-time.

2.2 Simulation Models

Simulation models for industrial plants are software representations of mathematical models of the real plant behavior. "*Mathematical models for dynamic systems are derived from the conservation laws of physics and the engineering properties of each system component*" [173]. Converting mathematical models to executable simulation models typically relies on manual work of simulation experts. Such work can have diverse nature as various types of simulations exist. Each simulation model is typically executed by a simulation solver, which is a core part of a simulation engine implementing a specific numerical method. The simulation solver performs the simulation models under a given simulation time and satisfying relative and absolute precisions. The very same simulation model can be used for diverse tasks, the difference is how inputs and outputs of the simulation model are used.

Simulation models have inputs and outputs, which are variables in the mathematical sense, and parameters that are mathematical constants specifying model dynamics, see Fig. 2.1. In compliance with [173], the inputs are "*functions of the independent variable of the differential equation, the excitation, or the forcing function to the system*" [173]. The outputs are "*the dependent variables of the differential equation that represent the response of the system*" [173]. In practice, inputs and outputs are sampled, hence their values are time-series with discrete time. If the system is affected by disturbances, we assume that these variables are included as parts of inputs. According to a number of inputs and outputs, simulation artifacts are frequently categorized as SISO (i.e., single-input single-output) or MIMO (i.e., multiple-input multiple-output) simulation components, models, or

modules. The parameters cover the three following sets: (i) constants of the differential equations parameterizing the model dynamics, (ii) settings of the simulation solver, and finally (iii) initial conditions of the simulation model. In the following text, these interfaces of simulation models are described in details.

To calculate outputs of the aforementioned equations (i.e., the time series of variables y_j, $j = 1, ..., p$), a simulation environment utilizes an internal or external simulation solver. The configuration of a simulation solver includes solver-dependent and solver-independent parameters such as required relative and absolute precision, minimal and maximal time-step, maximal number of zero-crossings, and others. We denote these parameters as simulation solver parameters s_z, $z = 1, ..., r$. In the sense of Fig. 2.1, parameters are a set of constants involving (i) sets of mathematical parameters c_l, $l = 1, ..., q$, (ii) simulation solver parameters s_z, $z = 1, ..., r$, and (iii) initial conditions x_{i0}, $i = 1, ..., n$.

2.3 Bond Graphs

Bond graphs are aimed at a unified and systematic way for mathematical description and modeling of physical systems. The bond graph theory is complex, quite easy to use, and supports many physical phenomena. The theory is already well-proven; it origins from the late 1950s, and it was being widely studied and published in 1960s, see for example [83].

Since physical systems are balancing distribution of energy inside by transferring power and tending to reach the highest entropy of the energy distribution, the crucial variables describing the behavior of systems are those physical variables that affect the energy distribution within the system. As the rate of energy transfer is power, it is the power that has the fundamental role in modeling with bond graphs as well as in the method proposed in this thesis.

Bond graphs are focused on describing power flows within systems. In a popular way, "*power is the universal currency of physical systems*" [52]. Power is the rate of energy flow and mathematically, energy is the time-integral of power. Power flows from sources, it can be temporarily stored in specific components (such as capacitors or inductors in an electrical circuit) and it is dissipated (such as in resistors, where electrical power is transformed into heat) [24].

The bond-graph theory is based on the following three types of analogies, which are subsequently described in more details: (i) signal analogies, (ii) component analogies, and (iii) connection analogies.

2.3.1 Signal Analogies

In physical systems of various nature, equations describing the system behavior have very similar forms. This phenomenon was observed by Lord Kelvin and James C. Maxwell in 19th century, but Henry M. Paynter described this systematically later in 1950s [52]. Signals are in this context considered as any quantities having variations in time and conveying information about systems. The high importance for simulation modeling have such signals that are relevant for describing energy respectively power transport in the system as well as signals that are used to control or to influence the status of the specific dynamic system. Due to the correspondences between signals in systems of different engineering disciplines, the bond-graph theory defines signals that are abstract in terms of system-type independence.

In order to introduce a unified approach to describe diverse types of systems, the bond-graph theory defines two generic variables:

- Effort $e(t)$
- Flow $f(t)$

These variables are called "power variables" as their product is power:

$$p(t) = e(t)f(t)$$

Furthermore, the bond-graph theory utilizes two integrated variables, which are useful for component description:

- Integrated effort:

$$p(t) = \int e(\tau)d\tau = p_0 + \int_{t_0}^{t} e(\tau)d\tau \tag{2.3}$$

- Integrated flow:

$$q(t) = \int f(\tau)d\tau = q_0 + \int_{t_0}^{t} f(\tau)d\tau \tag{2.4}$$

Although mathematically it would be feasible to differentiate these equations and to express flow (respectively effort) as a derivative of integrated flow (respectively integrated effort), these equations would not be causal in dynamic systems. The derivative operator needs to know future behavior, which is not possible. These issues are reflected in the concept of "causality", which is an important feature of bond graphs described later in Sec. 2.3.4.

2.3.2 Component Analogies

The above stated definition of generic signals is useful not only for the expression of signal relationships among systems of various physical nature, but also in order to define component interfaces and basic generic components. Component analogies are the second cornerstone of the bond-graph theory.

A pair of the effort and flow variables is called shortly "port" or more accurately "energy port" in the bond-graph theory. A real connection between devices is equivalent to a connection of energy ports of the two components representing these devices. Each component can have 1 to n ports, denoting the number of possible power connections of this component.

More formally, the **power port** is defined as follows:
"*The connection points of a bond graph node that enable the energy exchange with other nodes across a power bond are called power ports*" [28].

The physical interpretation of ports can be easily seen in case of electrical systems, where each energy port corresponds to a pair of single-port connectors. For example, an electrical resistor is the one-port component as it has one pair of single-ports. The selection of the electrical domain as an example is not coincidental, but the names of the abstract components defined by the bond-graph theory are inspired by the electrical domain.

Table 2.1: Signal analogies expressing adequate signals for various types of systems and their bond-graph representations.

Bond Graph	*Effort*	*Flow*	*Integrated effort*	*Integrated flow*
Electrical system	Voltage	Current	Lines of flux	Charge
Hydraulic system	Pressure	Flow	Momentum per unit area	Volume
Translation system	Force	Velocity	Momentum	Position
Rotation system	Torque	Angular velocity	Angular momentum	Angle
Thermal system	Temperature	Entropy flow	–	Entropy
Chemical system	Concentration	Molar flow	–	Molar mass

The bond-graph theory defines the following one-port components:

1. *Source of effort* (SE) is an ideal source of effort.
2. *Source of flow* (SF) is an ideal source of flow.
3. *Resistor* (R) is a component, which relates effort and flow by a static function, which can be non-linear in general.
4. *Capacitor* (C) is a component accumulating energy and having a static function between effort and integrated flow. This function can be non-linear in general.
5. *Inductor* (I) is a component accumulating energy and having a static function between flow and integrated effort.

All these components are called one-port components, which means that each component relates one pair of effort and flow signals. One of these variables is input and the second one is output. Only in the case of sources, it is done by definition, which one is output; in the other cases, it is determined by the bond graph. The bond-graph theory also supports components having more than one ports.

Formally, the theory defines the term **multiport** as follows:
"*A bond graph node is called a multiport if it has more than one port*" [28].

Examples of basic electrical two-port (as a specific case of multiport) components are a transformer (TF) or a gyrator (GY). Having two pairs of single-port connectors defining two (internally coupled) power flows, they frequently transform or convert energy between various engineering domains. The detailed description is not crucial for understanding of this thesis; it can be found in numerous literature such as [28]. A typical example of n-port components are junctions that connect n components.

2.3.3 Connection Analogies

Having the generic components, the simulation model schema should be created by interconnecting these components. The connections are called power bonds in the language of bond graphs and they are pairs of power variables. The third analogy tackles the problem of connecting devices in series or in parallel.

The typical approach used for electrical circuit analysis is based on Kirchhoff's laws. In systems consisting of a high number of components, it is difficult to determine how to combine these laws in order to avoid underdetermined or overdetermined mathematical

models. To face this problem, the bond-graph theory introduces an abstraction of connection types:

1. The *0-junction* is a junction having the same value of effort on all connected power bonds and the sum of the (oriented) flows is zero:

$$e_1(t) = e_2(t) = ... = e_n(t) \tag{2.5}$$

$$f_1(t) + f_2(t) + ... + f_n(t) = 0 \tag{2.6}$$

2. The *1-junction* is a junction having the sum of effort equal to zero and having the same flow for all connected power bonds:

$$e_1(t) + e_2(t) + ... + e_n(t) = 0 \tag{2.7}$$

$$f_1(t) = f_2(t) = ... = f_n(t) \tag{2.8}$$

The type of junction to use depends on the type of the physical system as follows.

- *Non-mechanical systems*: In non-mechanical systems, a 1-junction is a serial connection of components, whereas a 0-junction represents a parallel connection.
- *Mechanical systems*: In the case of mechanical systems, the assignment is vice-versa, i.e., 1-junctions represent parallel connections, whereas 0-junctions represent serial ones.

2.3.4 Creating Bond Graphs

A bond graph is a graph containing components, junctions, connections, directions of power flows, and causality strokes. We have already discussed the fundamental issues related to components, junctions and connections. In the further text, we will focus on the power direction, causality and the entire method of creating bond graphs.

The power direction defines the positive direction of power through each bond. This direction is not crucial in terms of the mathematical description, but it is an important feature for understanding the sign convention, i.e., what a positive or a negative value means for each bond. The theory recommends specific rules for assigning the direction as follows. These rules for the power direction assignment are frequently summarized as follows:

1. Positive direction of power is oriented out of sources SE and SF;
2. Positive direction of power is oriented into 1-port components C, I, and R;
3. Power direction remains in the same direction through 2-port components TF, GY;
4. Power is directed out in case of at least one power bond connected to $0-$ and $1-$ junctions;
5. Power direction in cycles directly powered by a source is in the same direction;
6. Power direction of power bonds leaving out of cycles is arbitrary.

An important aspect of bond graphs is the causality, declaring which of the variables flow and effort are the dependent and independent variables for each power bond (i.e., which of the variables is considered as a given one and which one is calculated in each connected component or junction). A source of effort has effort as a given output, whereas the flow depends on the rest of the system. The second type of source, the source of flow, has a causality vice-versa. As it has been already mentioned, one of the requirements on simulation schemas is to calculate integrated variables by discrete summing and not calculating the flow or effort as a derivative of the integrated flow, respectively the integrated effort. Another requirement on the causality arises in the area of junctions. In the case of 0-junctions, the common effort is given by exactly one component, whereas in the case of 1-junctions, the common flow is given by exactly one component. The causality is denoted by adding a short bar to the end of a bond [24], which is called a causality stroke. In case of 0-junction, the causality mark should be located on exactly one bond near the specific 0-junction. In the case of 1-junction, the causality mark should be on all bonds except one.

For determining the causality, the bond-graph theory proposes the following recipe defining the assignment order:

1. Causality of sources and their directly affected nodes;
2. Integral causality of components C and I, if the integral causality is possible;
3. Causality for other remaining nodes in such a way that the definitions of the nodes are satisfied. In the case of resistors, the causality is arbitrary;
4. In case of causality collisions, we either use a differential causality or solve this collision according to causality-collision rules, which are proposed in this theory.

To create a bond graph, a reference junction has to be excluded from the graph. For example, it is the ground voltage in case of electrical systems. The bond-graph theory defines several types of bond graph reductions. As they do not affect the results of the proposed method, they have not been implemented and will not be discussed in this thesis.

The next step of the simulation model design is going through the created bond graph and writing down mathematical equations that model the behavior of the system. For this task, the power direction plays the role in assigning signs to all signals distributing power within the system. The assigned causality determines the causality relationship of signals flow and effort for all components and junctions. The mathematical modeling of components themselves is done by their physical behavior. Its mathematical expression is affected by the causality assigned. This process step of simulation model design with the bond graph theory is time-consuming and error-prone, especially in terms of confusing sign conventions of specific variables within the system.

As the result of the previous step, we have a mathematical model of the system. This mathematical model has to be transformed into a form that is appropriate for the simulator to be used. In this thesis, we are working with signal-oriented simulations, the mathematical model created based on the bond graph method thus has to be transformed into the signal-oriented specification. This form is very close to block diagrams, which is widely known by control engineers and cyberneticists.

Summarizing the workflow of the bond graph method, the theory proposes to perform the following process steps to create the simulation model:

1. Generation of nodes representing components and junctions;
2. Generation of arcs representing power bonds;
3. Assignment of the power direction;
4. Exclusion of a reference junction;
5. Reduction of the graph;
6. Assignment of the causality strokes;
7. Writing down mathematical equations manually.
8. Transformation of the mathematical model into an executable simulation model.

2.3.5 Tool Support for Bond Graph Modeling

System modeling based on bond graphs is supported by various software tools. In this section, the major tools in this area are summarized and briefly introduced. This introductory list organized in the alphabetical order was created based on literature reviews including mainly [28] and the author's experience with these tools.

- **20-SIM** is the integrated modeling and simulation environment presented in [30]. It was developed at the University of Twente as a successor of the tool TUTSIM from the same university. It is an interactive tool for modeling and simulation of dynamic behavior of engineering systems. It has modeling and simulation parts, it supports data sharing with other simulation packages, and it covers a wide range of techniques. Due to its abilities, it is one of the most widely used tools for bond-graph modeling.
- **ARCHER** was designed at Ecole Centrale de Lille, France. It supports creating bond graphs graphically and exporting the resulting transfer functions and state-space models into a symbolic form. Its benefit is the support for structural analysis, including for example system states in terms of Eq. 2.1.
- **BAPS** is a bond graph preprocessor developed at the Vienna University of Technology. It supports nonlinear constitutive equations and it is equipped with a graphical user interface. The tool assigns the causality to the bond graph and the resulting model can be exported into ACSL or several other supported formats.
- **Bond Graph Add-On Block Library BG V.2.1** is a library for MATLAB-Simulink, which was designed at TU Dresden. It includes nine basic simulation blocks to model systems with the bond graph approach. The library can be used via the standard Simulink GUI, which however causes that it can be utilized mainly by skilled users or for educational purposes. It does not offer any sophisticated model transformations or user-friendly interfaces.
- **Bond Graph Toolbox** is a tool for graphical working with bond graphs. It was developed at the National University of Ireland in Galway. It provides outputs in the form of equation-based system description for Mathematica or in the form of Fortran routines performing the simulation directly. The output model equations are non-causal.

- **BondLab** is a bond graph toolbox supporting integration of bond graphs into MATLAB and Simulink. It was designed at KU Leuven and it offers a graphical editor for bond graphs from which the bond graphs can be exported in several suitable forms to MATLAB and Simulink.

- **CAMBAS** is the abbreviation for "Computer Aided Model Building Automation System". It is a tool for transforming a model of a system to a bond graph and consequently to its linear model. It was developed at the University of Michigan. CAMBAS utilizes a two-level abstraction of simulation modeling. On the higher abstraction level, components have assigned ports but no internal functionality. On the lower level of modeling abstraction, implementations of components are added, which is supported by component libraries. The tool supports to specify the simplest simulation model that is proper for the simulated problem. The benefit is multi-domain orientation from the engineering discipline perspective. On the other hand, the tool is focused especially on linear models.

- **CAMP-G** is a graphical extension of the bond graphs preprocessor called CAMP. It was designed at UC Davis. The graphical extension CAMP-G is user-friendly, but still the users work directly with bond graphs and not with the domain-specific representations of the engineering systems. It does not support hierarchical bond graph models.

- **ENPORT** is one of the first bond-graph tools developed by R. C. Rosenberg. The input for ENPORT is a non-causal description of the bond graph. The basic functionality of the tool is thus the assignment of causality to the given bond graph structure and selecting input and output variables of the components. Due to this fact, such a pioneering tool is intended rather for experts in the simulation domain who are able to create noncausal bond graphs manually. It also supports numeric simulation of the final bond graphs, however, only those models that are linear. The tool is implemented in Fortran language.

- **HybrSim** (also referred as "HyBrSim") is a modeling environment for hybrid bond graphs. This kind of bond graphs extends the traditional theory to support combinations of continuous and discrete systems. In Layman's terms, the original theory is enhanced with an ideal switching element implemented as a special kind of junction, further details can be found for example in [111]. The tool HybrSim provides two toolboxes for modeling and simulation of hybrid systems that are based on (i) bond graphs and (ii) block diagrams. The limitation of this tool is that it supports ideal bond graph elements only. From the mathematical point of view, the tool cannot extract equations, because they are not generated explicitly.

- **Java Applet for BGs** is an educational online application[1] developed at the University of Miskolc. It is applicable only for simple bond graphs with basic components only. Since no inport/export capabilities are available, its usage is strongly university-oriented.

- **Mathematica Bond Graph Toolbox**[2] was designed by N. Venuti. It is a graphical user interface for bond graph modeling in the symbolic environment Mathematica.

[1] http://www.uni-miskolc.hu/ iitbajzi/bond/index.html
[2] http://library.wolfram.com/infocenter/Conferences/4903/

Although Mathematica provides many powerful features, its usage is out of scope of this thesis.

- **MS1** is a powerful integrated modeling and simulation environment developed in France. It supports hierarchical models as well as diverse forms of system modeling including not only bond graphs, but block diagrams or equation models, too. The output of the MS1 environment can be executed in various commercial tools and solvers including ACSL, Maple, Modelica, or MATLAB.

- **MTT** is a tool for transforming models between different model description forms. This is symbolized by the tool name, which is the abbreviation for "Model Transformation Tools". It was developed by Gawthrop at the University of Glasgow.

- **PASION 32** is an object-oriented simulation tool supporting diverse modeling approaches, including bond graphs. It is based on Pascal programming language and utilizes Pascal source codes during runtime as well. Users can enter bond graphs in a simple graphical way. Causality can be assigned manually or decided by tool algorithms. The final bond graph in a graphical form is consequently transformed into textual representation in the form of a proprietary code.

- **SYMBOLS Shakti/Sonata** is an integrated modeling and simulation environment, which was developed at the Indian Institute of Technology Kharagpur. The tool name SYMBOLS is the abbreviation standing for "SYstem Modeling by BOnd-graph Language and Simulation". It has many common features with the tool suite 20-sim. In addition, the SYMBOLS tool introduces the concept of "capsules", which enable hierarchical modeling with respect to an encapsulation of sub-models via explicitly defined interfaces, called glue ports. "*Algebraic loops and derivative causality at storage ports are tolerated*" [28]. Non-linearities can be modeled with non-constant parameters, which are natively supported and can be related, for example, to latest values of specific system states.

- **TUTSIM** is a tool developed at the University of Twente. It is a predecessor of the famous aforementioned tool 20-sim, developed at the same university. Compared to ENPORT that was being used in the coincident era, TUTSIM is slightly more generic and it is better focused on block diagrams rather than on pure mathematical descriptions of ENPORT. It also supports numerical simulation of the bond graphs including nonlinear systems, however, it does not support stiff systems.

The bond graph method can be also used in tools that are not intended to bond graph modeling originally. For this purpose, Modelica language plays a crucial role as it is an equation-based modeling language. It implicitly means that Modelica models are acausal, respectively dependencies between variables are solved by simulation solvers executing the Modelica code. A typical example of simulation environments supporting Modelica language is Dymola. Dymola can be used for bond graph modeling, too, as it is discussed for example in [35]. In general, using Dymola for bond-graph-based modeling is not the desired way of use of this tool originally, yet it is possible.

2.4 Functional Mockup Interface

Since requirements on separation of simulation models into several relatively independent units are emerging or increasing, this topic is addressed in this thesis as well. Functional Mockup Interface[3] (FMI) is a technical solution for the composition of simulation models from simulation modules. Simulation modules in the sense of FMI are called Functional Mockup Units (FMUs).

The basic idea of FMI is to facilitate co-simulation and model exchange. Simulation modules (i.e., functional mockup units) are compiled into an executable platform-independent code. The benefits of FMI/FMU are that (i) it enables bridging diverse simulation languages and platforms, as well as (ii) it hinders revealing details how simulation modules are implemented. The former aspect is important when integrating modules implemented for example in MATLAB-Simulink and Modelica language. These platforms are of different nature and each of them is beneficial for different kinds of simulations. The latter aspect is important in simulation projects covering several stakeholders in industrial consortium, where intellectual property protection plays a significant role. This situation is frequent for example in an automotive industry, where subcontractors deliver products to various competing car manufactures. Examples of the FMI applications in the automotive industry are discussed for example in [154] or [89]. Technically, FMUs are zip files, see [46] for details. Each zip file includes the simulation unit itself (i.e., the simulation module), which has an interface in the C language representation. In addition, each FMU is accompanied with an XML annotation describing the interface of the unit.

Evaluating the FMI approach, it is beneficial in terms of supporting for modular simulations consisting of a set of simulation modules. This technology is well tested, partly adopted by industrial stakeholders, and considered as promising. The intellectual property protection on the FMU level has been already mentioned, which is the next benefit of this technology. On the other hand, the FMI does not provide means how to define the size of units, into which a specific large-scale simulation should be split, or how to specify interfaces of these units. Such issues are in the scope of research, see for example [31] for further details. When used in the co-simulation mode, the FMI requires a master unit, within which the other FMUs are loaded [11]. This characteristic imposes limitations on flexibility at simulation runtime as the master unit cannot be removed and affects the computational stability of the entire federation of units. Last but not least, the FMI is not an open technology, neither on the source-code level, nor on the execution level. Hence if the simulation federation does not work properly, it is difficult to debug. The limited insight of simulation engineers into running simulations within the FMI can pose an important restriction for improving the execution and identifying weak points in simulation models. However, this issue corresponds to the black-box nature of FMUs, which can be seen as a benefit on the other hand.

FMI is partly competing to the utilized simulation integration framework (SIF). Both FMI and SIF are technical infrastructures for running simulation models consisting of a set of simulation modules. FMI is more customer-oriented in terms of each FMU is an encapsulated module which cannot be seen or edited internally from the level of the entire simulation workflow. SIF is more simulation-expert-oriented as various modules can be refined arbitrarily and they can be inspected during simulation. In addition, SIF supports

[3] https://www.fmi-standard.org

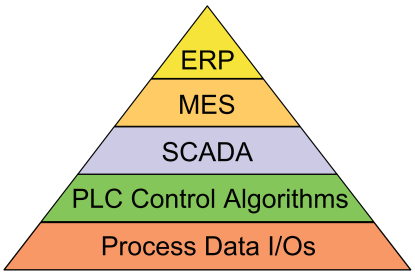

Figure 2.2: Automation pyramid.

access to diverse data sources and arbitrary choice for simulation solvers in case of each simulation module. On the other hand, FMU is a commercial off-the-shelf product, whereas SIF is a framework for simulation developers/experts being under development.

2.5 Architectures of Industrial Automation Systems

The architecture of industrial automation systems has already been introduced briefly in Sec. 1.2. In the following text, this architecture is described in more details in order to provide foundations for the contributions of this thesis.

Fig. 2.2 depicts a schematic layered architecture of automation systems considered in this thesis. On the lowest level of this automation pyramid, there are input and output process data, which are physically handled by I/O modules and fieldbuses. On the second level, there are control algorithms in programmable logic controllers (PLCs). PLCs are computers intended for industrial control and execution of control programs at real-time. The third level of the pyramid consists of a SCADA system [29], standing for "Supervisory Control and Data Acquisition". A SCADA system is a system that is intended to provide access to industrial plants, both for human operators and the upper software systems. In a wider sense, it can also include the hardware related to the supervisory control and data acquisition, but in this thesis, a SCADA is understood as a software system only. Therefore, it can be considered as a borderline between the hardware and software part of the automation system, because the aforementioned bottom levels of the automation pyramid are tightly connected to the hardware, whereas the higher pyramid levels are hardware-independent. SCADA includes human-machine interfaces (HMIs) [63] for interaction with human operators, historians for storing historical data, a subsystem for managing alarms and events and many other subsystems. Industrial experiences emphasize the need for supporting integration of simulation models on the SCADA level, which is the issue addressed in this thesis. The highest levels of the automation pyramid include the Manufacturing Execution System [102] (MES), providing planning and scheduling of the manufacturing production, and Enterprise Resource Planning (ERP), which is used for company management (e.g., SAP[4]).

The integration of simulations is required especially on the SCADA system level of the automation system. The design of supervisory control and data acquisition (SCADA) systems and the integration of automation tools into one consistent system are challenges, whose importance is growing fast. Current approaches to industrial integration and automation system design are based on repeating manual work. Even minor changes in a

[4]http://go.sap.com/index.html

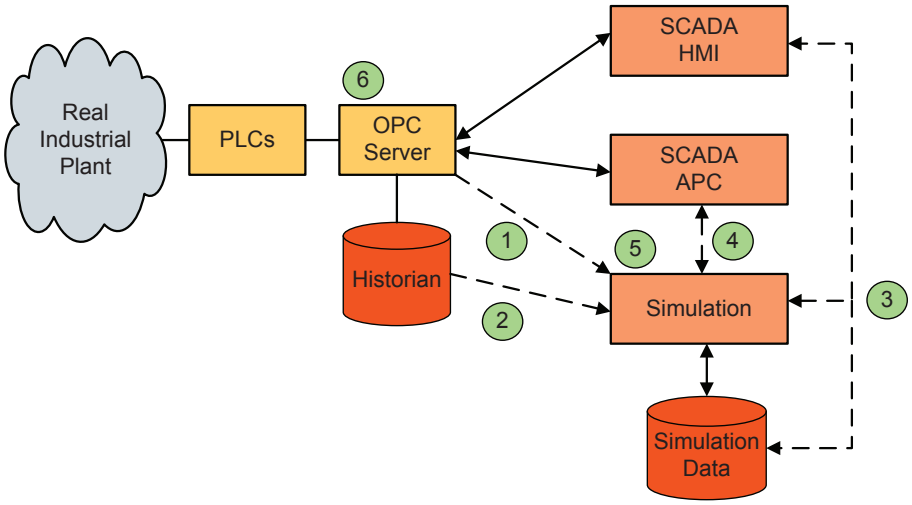

Figure 2.3: Current approach in industrial automation practice and its challenges.

real plant or in an industrial system imply time-consuming work of experts and due to the complexity, the results do not guarantee consistency and safety typically. The integration of simulations is required especially on the SCADA (i.e., Supervisory Control and Data Acquisition) system level of the automation system. These issues motivated the work in this thesis that is aimed at automating the integration process in the industrial automation area with a strong focus on process simulations.

Intelligent control techniques used on the SCADA level [80] can improve real system operation. Benefits of advanced process control have been clearly shown in the last decades [14], but its weak point is the necessity of simulation models, optimizers and other specific tools. The models and the whole simulations cannot be developed and operated without access to online and historical data to get appropriate results. SCADA systems, simulation tools together with data sources should be thus seamlessly integrated into control system architectures in order to create a powerful modern system.

2.6 Current Status Summary Motivating the Thesis

The shortcoming of the methods used in the industrial practice by now can be summarized as follows. The design phase requires repeating manual work based on copying large pieces of information. A lot of errors can occur during this process and it is not explicitly defined what actions should be done if a specific event occurs. In addition, the design and integration phases are rather independent, which causes problems with the consistency of interfaces and the interpretation of signals. Therefore, this research investigates methods, which capture relevant pieces of engineering knowledge in the knowledge base and utilize it for the configuration of the runtime integration level. The ontology-based representation of knowledge makes the representation flexible, and the steps in the simulation model design and integration are clearly stated.

The major challenges for the automation system integration at runtime are expressed in Fig. 2.3. It depicts the typical architecture used in the industrial practice now. The numbered circles refer to the following issues:

1. Import of runtime data into simulations.

2. Import of historical data into simulations.
3. Visualization of simulation data in standard HMIs and testing dispatchers' commands entered via HMI on simulation models.
4. Testing of SCADA advanced process control (APC) control actions on simulations.
5. Semi-automated design of simulation models.
6. Semi-automated configuration of OPC tags and other integration interfaces.

Chapter 3

Related Work

The topic addressed in this thesis covers a large variety of sub-problems related to simulation modeling, semantic and technical integration, and various issues dealing with mechatronic systems. This section summarizes outstanding related work in all of these areas systematically. To simplify understanding, the research scope and related work is summarized in Fig. 3.1. The works included in the figure are consequently discussed in the remainder of this section.

The extraction and reuse of knowledge for supporting engineering of industrial plants as well as their automation and control systems are crucial tasks in the area of knowledge-based engineering (KBE) [143, 161]. One of the first approaches is addressed in [37] and one of the data modeling approaches for supporting engineering plan reuse is discussed in [23].

3.1 Automated and Semantic Simulation Model Design

The process of simulation model design is frequently not formalized and the relevant pieces of information usually cannot be processed semi-automatically. It causes problems with maintenance and extensions of simulations as well as with their reuse. Although technical integration is partially possible, the integration task is time-consuming, costly and error-prone. These shortcomings lead to formalizing the simulation model structure in order to simplify and to semi-automate simulation model design. Most of the solutions are based on capturing knowledge in ontologies [55], which enable to represent knowledge flexibly and to process it efficiently.

The Ontology Driven Simulation design tool suite is presented in [152]. The described approach is based on two ontologies; the first one is called a domain ontology and it categorizes knowledge including a problem vocabulary in the domain scope. The second one is called a modeling ontology and it is used for the simulation model description. Such approach guarantees a high degree of reconfigurability of the solution and a separation of the whole problem knowledge into appertaining engineering scopes. The discrete event model is assumed and it is represented by the DeMO (i.e., Discrete Event Modeling Ontology, for further information see [151]). The presented approach is based on mapping concepts from domain ontologies to a modeling ontology, translating ontology instances to an intermediate XML markup language and generating an executable simulation [152]. The approach presented in this report is based on a similar idea, but it addresses other engineering tools as well as it reflects features of large-scale industrial systems.

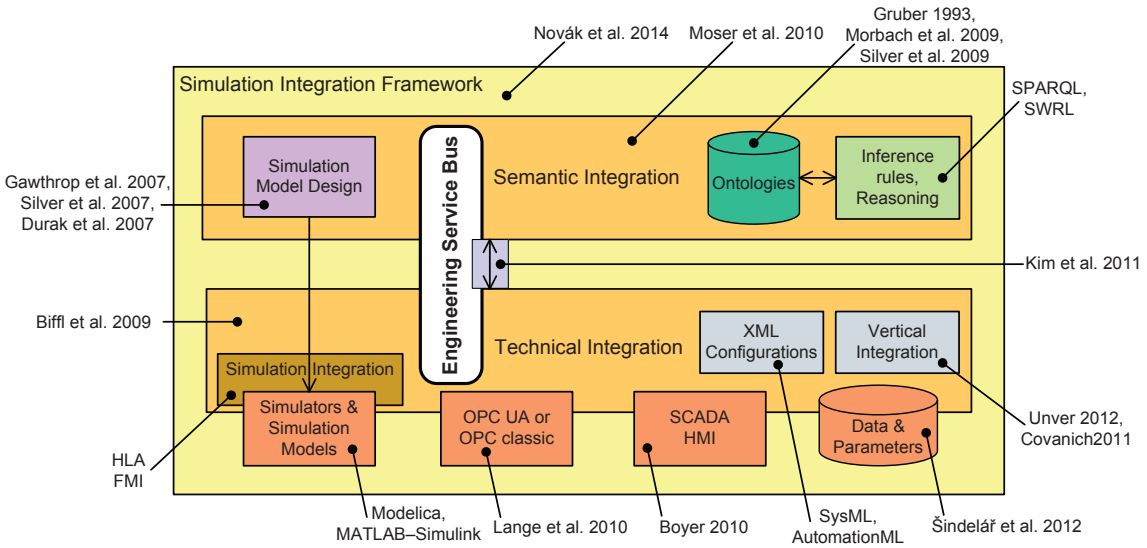

Figure 3.1: Simplified summary of related work.

The procedure to derive an ontology-based simulation interoperability from feasible sources without machine-readable semantics is described in [16]. The process starts with the extraction of relevant terms, followed by the tokenization of the source document, which retrieves sentences and collocations of the terms. The example in the paper uses WordNet ontology for retrieving further information. The paper also involves a description of ontology mapping related to different simulators, which constitutes an important tool for ontology-driven translation between independent vendor languages. Another approach to retrieve system design knowledge from data sources that are not in machine-understandable forms has been presented in [170].

The ontology-driven simulation model design is presented in the paper [45]. The paper is focused on generating MATLAB-Simulink blocks and defining them via DAVE-ML according to a domain ontology, which is the Trajectory Simulation Ontology in the presented case. Connections of these blocks are done manually.

Ontologies for discrete-event system description already exist, for example PIMODES or DeMO [153]. On the other hand, a small effort has been invested into continuous-time system modeling supported by ontologies.

Ontology-based support for integration and interoperability in the context of complex modeling and simulation environments is addressed in [162]. It formulates challenges in the aforementioned area and provides foundations for solving the modeling and simulation tool interoperability for cross-disciplinary systems by means of ontologies.

3.2 Bond Graphs for Simulation Model Design

A special type of simulation model design approaches is based on the bond-graph method. It is a research area where a large variety of publications has been published.

An introduction and motivation into bond graphs can be found in [24]. The whole description of this design method is given in [52], including various examples, system analogies, and practical issues. The author of this thesis learned about bond graphs from the text book [64]. A very good monograph on bond graphs for mechatronic systems is [38]. Last

but not least, a very complex monograph [28] was used to check bond graphs for hydraulic systems and to check definitions and glossary related to bond graphs.

An approach combining bond graphs and object-oriented modeling to build simulation models is discussed in [26]. The paper summarizes three model description languages: MAST, VHDL-AMS, and SIDOPS. An application of VHDL-AMS is discussed in more details for example in [138], where the chemical domain is considered as a use-case.

The main contribution of [26] is focused on the application of bond graphs for Modelica-based models. Bond graph ports are realized by the "connector" class in Modelica language to specify the interface, which can be consequently connected via the operator "connect". Compared to [26], this thesis is focused on signal-oriented simulators, whereas Modelica is equation-oriented. Furthermore, [26] does not discuss the problem of multiple implementations of specific components. The same author presented in [27] the whole bond graph method including specific steps to generate bond graphs for mechatronic systems, the sequential causality procedure introduced by Karnopp and Rosenberg, and finally several use-cases.

The generation of simulation models based on bond graphs is discussed in [15]. The outcome of this approach are generated simulation models in the Modelica language. The real plant description as the main input utilizes the CAEX, which is a vendor-independent format for process and instrumentation description. The use of Modelica implies that the causality assignment need not be resolved by the generation algorithm. Compared to this paper, the thesis is focused on signal-oriented simulators with explicit support for various component implementations respectively simulation blocks.

3.3 Integration of Simulation Models

To use simulations efficiently, they should be integrated with other automation tools at runtime and the structure and parameters of the simulation models have to be consistent with a real industrial plant and existing automation system tools.

The problem of the simulation model integration is classified in [16] into two categories: (i) A design-time interoperability refers to the simulation development when requirements for interoperability are known from scratch and a structure of simulations adopts them, whereas (ii) runtime interoperability is related to simulations and applications designed independently.

The main limitations of simulation models created via popular software packages are summarized in [152] as follows: (i) "*There is no formal way of specifying an agreed upon domain of discourse for the application domain of the process being modeled*, (ii) "*The modeling domain suffers from a similar problem*", and (iii) "*There is no commonly agreed upon format for representing and storing models*" [152]. Ontology-based integration of simulation models is discussed in [151].

On the technical level of integration, approaches using general-purpose distribution techniques can be found. They utilize technologies such as DCOM, CORBA, J233, etc. [66]. Several examples of frameworks including standard vocabularies such as DIS, SEDRIS, HLA are given in [91]. Simulation integration control is introduced in [86].

On the technical integration level, especially the High-Level Architecture (HLA) [62] is widely cited. The framework addresses the composition of simulations from sub-models,

but it does not address how to get input simulation data and how to store the results. In addition to such an absence of data source management, the shortcoming of this framework is the absence of semantics. The extension of HLA with semantics is proposed in [66], but any proposal for adding data source management into this framework is not known to the thesis author.

Functional Mockup Interface (FMI) for composition of simulation models from modules (i.e., functional mockup units – FMUs) has been already introduced in Sec. 2.4. It targets on similar types of problems that was addressed with the older HLA standard. However, FMI is more product-oriented in comparison to HLA. The usage of FMI is still in the pioneering stage, in the further text, we will focus on rather proprietary approaches based on other technologies.

System integration in electric power industry is discussed in [95]. The target systems for interoperability are for example SCADA, EMS (i.e., Energy Management System), DMS (Distribution Management System), AM (i.e., Automated Metering), MIS, GIS and others. The presented approach is based on the Utility Management System[1]. The proposed system is agent-based and the semantics of the APIs are described by Common Information Model (CIM)[2]. Communication between the systems is realized based on the Information Exchange Model (IEM), which is built on top of XML.

The use of generic frameworks for supporting integration of simulators and SCADA systems is discussed in [77]. It addresses the following five frameworks: Rational Rose, ObjecTime, SiMOO-RT, Elipse and Unisoft. The paper describes the architecture of the environment focused on modeling, simulation, and supervision.

A power system communication layer is discussed in [47]. Two viewpoints onto communication are defined: I-view (information view) and T-view (transport view). Both views are considered having three maturity levels - I-view levels differ in semantic capabilities and T-view levels differ in concurrent capabilities.

3.4 Integration of Industrial SCADA Systems

The integration of electrical network systems is discussed in [155]. The addressed systems are the SCADA system, Automated Mapping and Facilities Management (AM/FM) system and Outage Management system. The important contribution of the paper is the description of dataflows and their directions, classifying types of interchanged data, and evaluating bit rates of the transfers.

Integration of systems in the Garland Power & Light (GaPL) distribution network is discussed in [53]. The paper addresses data acquisition from the field devices into the SCADA/EMS system. The presented research is divided into three phases relevant for the GaPL enterprise: the first phase involves monitoring of analog and status points, the second phase is suggested to purchase circuit monitors in several distribution points and the third one addresses the application of the approach for some of the remaining substations of the GaPL grid. A format of transferred data is not solved in this article.

The integration of electrical network subsystems, which are used by the company Elektrosrbija, is discussed in [41, 42]. The presented approach is based on two databases, whose

[1] http://www.omg.org/technology/documents/formal/UMS_Data_Access_Facility.htm
[2] http://dmtf.org/standards/cim

data schemes are involved in the articles. The approaches suppose concentrating all data in a central database. The integration of a simulator is not addressed in the article.

A framework for integrated power system modeling, analysis and control called SIMIAN is discussed in [139]. Although SCADA systems are not supported in the version presented in the paper, the framework is expected to support integration of SCADA systems in the future. The SIMIAN architecture (abbrev. SIMulation Image and ANimation) supports the inter-object communication solved by sending messages by objects. The messages are routed through the Event Handler application. Inter-process communication is solved using CORBA.

The Henan Dispatcher Training Simulator of the local electrical network is discussed in [67]. This system consists of the following subsystems: (i) Control center model, (ii) Power system model, (iii) Instructor position. Off-line data are collected from the EMS database, whereas real-time data are coming from the SCADA database. The integrated system provides a unified HMI. It is realized on a database level by unifying two databases.

Distributed generation in electric networks, with the use of for example wind generators, is addressed in [98]. In the main part of this article, a simulation model is described. The model was implemented in MATLAB/Simulink environment and for the purpose of operator training simulator usage, it was converted into C code. The data exchange between dynamic models and SCADA systems is not specified in this article.

Contributions of the author of this thesis in the area of the integration of simulations and SCADA systems are summarized in the previous author's work in [127]. The article is focused on the architecture of the simulation integration framework as well as it introduces the usage of ontologies for simulation model design and integration, which are pioneering approaches in these areas.

3.5 Current Trends in System Integration

A lot of research effort has been invested into a wider use of abilities of the current Internet as well as its improvements for future use. Software as a service (SaaS) is an approach that "*focuses on separating the possession and ownership of software from its use*" [157]. In other words, SaaS assumes that the execution of software is provided by an external authority. This approach could be beneficial for simulations, especially due to dedicated hardware-intensive computations to an external provider. However, this option is not preferred by a lot of industrial partners due to claims about the risk of security threats.

The promising approach applicable in this area is Internet of Things (IoT) [87]. The IoT consists of "smart objects". They are embedded systems having static IPv6 addresses, which are accessible from any other node of the network, see [9] for further details. Such smart devices can be easily integrated into interoperable systems, whose architecture can be flexible due to uniformity of access interfaces from the technical point of view. Originally, the IoT address does not incorporate RFID tags, however, there are methods to integrate RFID technology within IoT seamlessly, see [87] for further details.

Advances in Semantic Web are summarized in [150]. Originally, the Semantic Web was considered as "*a Web of auctionable information–information derived from data through a semantic theory for interpreting the symbols*" [150]. The article shows the crucial role of ontologies for bridging various representations of data and integrating knowledge.

Another trend is the use of the service-oriented architecture (SOA) [48]. The governance of SOA for enterprise application integration is discussed in [19]. "*A major reason to have an SOA is to create business and IT agility*" [19]. "*Business agility is the ability to change or create a new business process in a readily adaptable manner*" [19]. This book also addresses so-called service-oriented enterprises, which is based not only on services, but also on connecting business processes in a much more horizontal fashion.

Service-oriented computing is discussed in [135]. It is based on SOA, however, it extends it with issues such as management, composition of services, service orchestration, and others, see [135] for further details.

Design of applications satisfying service-oriented architecture (SOA) as an architectural style that supports service orientation is summarized in [17]. Enterprise Service Bus is considered as a "*pattern that allows for the integration of new and existing systems using JMS, RMI, or of course, Web services, but also provides for heterogeneous integration with translation, mediation, and other capabilities that are required in complex environments*" [17]. Due to the importance of the enterprise service bus in the context of this thesis, it is described in more details in the following paragraph.

The use of Intelligent Services and an Intelligent Enterprise Service-based Bus is discussed in [96]. It discusses the problem of small-lot manufacturing, which was investigated within the European project called ARUM[3]. Other papers disseminated in the frame of the ARUM project that are related from the perspective of the scope of this thesis are for example [164], [165], or [59].

3.6 Enterprise Service Bus for System Integration

The enterprise service bus (ESB) [36] is a software-engineering concept for integration of heterogeneous tools and services. The ESB technology can be considered as a combination of the "*Service-Oriented Architecture (SOA), which is based on the request/response model and the Event-Driven Architecture, which is based on the publish/subscribe model*" [25].

The ESBs are frequently considered as abstract architectures, which offer developers a wide space for design and development. From the methodological point of view, they are general paradigms to be used for solving problems in various ways. The foundations of ESBs are motivated by electrical communication buses, where all stakeholders are connected to a shared medium. Although stakeholders can communicate peer-to-peer, the most frequent case is to let each message to be routed by a workflow engine to final data and event consumers.

The role of the workflow engine is very similar to the Directory Facilitator (DF) in multi-agent systems (MASs) [73]. The DF aggregates registered abilities of agents. When some request emerges, the DF is asked, which agent in the community can solve the request, and this agent is afterwards contracted. The agent abilities can be implemented by services in ESBs and they can thus address the very similar problems as well. However, MASs are better optimized for adding and removing particular agents and their skills, which are registered and unregistered on the DF. In case of ESBs, these operations are not expected to occur frequently, and if solved, such approaches are rather proprietary. In this thesis, we assume that tools can be added and removed arbitrarily, but the structure of tool domains

[3] http://arum-project.eu

is fixed during the run of the EngSB instance, on the contrary to MASs.

Although the architecture and development of ESBs is an active topic in the software engineering area, it is not necessary to have a detailed insight into the ESB architecture when developing and implementing applications utilizing ESBs. The ESB can be treated as a black-box [69], i.e., developers or users do not need an insight into the internal implementation of the service bus, but they can just use it.

The ESB utilized in this thesis is called the Engineering Service Bus (EngSB) [22]. The EngSB extends the concept of the ESB and of the SOA to offer a vendor-independent solution for the integration of engineering tools with heterogeneous data models of an engineering environment. It is thus not only a particular implementation of the ESB, but it provides features that are specific for the industrial automation domain. As it is explained in details in [68], the EngSB is an integration environment that combines the benefits of the "best-of-breed" and "one-tool-for-all" philosophies, while mitigating disadvantages of these approaches.

The EngSB is built on the top of the open-source project Open Engineering Service Bus[4] (OpenEngSB). As other ESBs, it provides an abstraction layer based on a specific implementation of an enterprise messaging system. The EngSB utilizes Apache ServiceMix[5] to solve messaging and communication issues, for more details see [127].

The EngSB-based infrastructure poses a solution mainly to the technical level of system integration. Nevertheless, it has to be configured to work properly, especially in terms of defining workflows. For this reason, ESBs have a workflow engine, which is responsible for controlling message workflows among stakeholders connected to this infrastructure. In particular, the data exchange within the EngSB is driven by pre-defined engineering workflows that are sets of configurable engineering process steps modeled in BPMN [3]. An essential benefit of the EngSB is thus a technological independent description of engineering processes and their automation [172]. The workflow represents the way of integration and it is defined by project requirements.

The EngSB uses the Engineering Knowledge Base (EKB) described in [108] as an approach for a semantic integration in heterogeneous engineering environments with a focus on providing links between data structures of engineering tools and systems to support the exchange of information between these tools. The usage of the EKB for the semantic integration of heterogeneous engineering environments is presented in [109]. The article explains that the EKB stores explicit engineering knowledge and it supports (i) data integration based on mappings between local and domain-level engineering concepts, (ii) transformations between local engineering concepts, and (iii) advanced applications built on these formalisms. The fundamental ideas of the EKB approach are adopted in this thesis.

On the top of the EngSB, the simulation integration framework has been developed [169]. It is planned as a generic environment for the seamless integration of simulation models within industrial automation systems. This framework is thus based on the EngSB and it supports integration of simulations and SCADA systems covering a simplified access to simulations from human machine interfaces (HMIs), version-control of data and models, and other issues typically required by industrial companies. The first version of the emerging simulation integration framework was proposed in [168]. It is mainly a technical integration framework, which is responsible for transferring runtime or batch data between stakeholders,

[4] http://openengsb.org/
[5] http://servicemix.apache.org/

such as simulations, HMI, OPC UA/classic connecting real plant devices and others. The outcomes of this thesis should be compatible with the simulation integration framework and furthermore, some of the thesis results are expected to be used as a native part of the simulation integration framework in the future.

The setting of the workflow engine of the EngSB in terms of the simulation integration framework as well as the setting of the entire infrastructure and all connectors is done by a set of XML files. The preparation of such configuration files can be solved either manually, bringing shortcomings in repeating error-prone manual work, or semi-automatically. In this thesis, the semi-automated approach is proposed, as it guarantees a consistent solution being flexible and capable to adopt changes without high human effort.

3.7 Semantic and Technical Levels of Integration

Ordinary integration approaches deal especially with the technical integration level, i.e., they implement the data transport between stakeholders. The rules for data transport are explicitly listed and the ability to transform data is limited only on hard-wired prescriptions. On the technical integration level, the following important issues are not specified: how to create routing-rule lists, whether interfaces are compatible, or whether the entire solution is consistent. The semantic integration level is not a substitution for the technical level, but it is an extension being on the top of the technical integration level. It is focused on the description of interfaces and on mapping them. Typically, the semantic integration level captures the representation of known entities, such as plant devices and their signals, and maps the adequate ones, such as really measured and simulated variables. In the following text, this distinction is compared more formally.

"*Technical system integration is the task to combine networked systems that use heterogeneous technologies to appear as one big system*" [110]. This definition implies that technical integration is related to bridge technical barriers to enable unified access to the whole system. This integration level is not in scope of this thesis, which assumes that the technical integration is solved via the simulation integration framework [168].

Semantic integration is defined as "*solving many semantic-heterogeneity problems, such as matching ontologies or schemes, detecting duplicate tuples, reconciling inconsistent data values, modeling complex relations between concepts in different sources, and reasoning with semantic mappings.*" [129]. In [128], three major dimensions of the semantic integration are summarized: (i) mapping discovery (i.e., finding similarities between ontologies), (ii) declarative formal representations of mappings (i.e., how to represent mappings between ontologies), and (iii) reasoning with mappings (i.e., the utilization of the mappings for reasoning). In case of this postgraduate research, the mapping is partially discovered by designed algorithms and partially explicitly entered by humans. The mappings are represented via specific object ontology properties and knowledge is explored with SPARQL queries.

3.8 Semantic Web

The concept of semantic integration is related to ideas behind the Semantic Web. The Semantic Web has been considered as an envisioned successor of the contemporary World

Wide Web, whose growth in terms of the amount of data included and the complexity causes difficulties in searching required information. Typical search engines are based on occurrence of keywords in Web documents. They take into account the presence of keywords, but not their semantic proximity (i.e., difference in their meaning). For example, the use-case included later on in Sec. 7.2 deals with a "tank model", which is a common term for a hydraulic system in an educational scale. However, searching for a "tank model" in a keyword-based search engine leads most likely to miniaturized models of tanks as armoured battle vehicles rather than tanks as kinds of hydraulic vessels. Moreover, the keyword-based search engines are not able to satisfactorily solve complicated queries such as "return pumps having the output pressure higher than 10 MPa and weighting less than 3 kg".

To tackle the searching within the growing text-natured World Wide Web, the idea of the Semantic Web emerged. The solution is based on "*augmenting Web information with a formal (i.e., machine-processable) representation of its meaning*" [146]. "*The Semantic Web is an extension of the current Web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation*" [18]. Such a well-defined meaning "*is provided by semantic descriptions, often referred to as metadata (i.e., data about data)*" [146]. The metamodeling of information within Semantic Web is done with ontologies. "*An ontology is a technical artifact that acts as a centerpiece of any Semantic Web based solution and allows the explicit and formal representation of the knowledge relevant for the application or use case at hand*" [146]. Due to the importance of ontologies not only in terms of the Semantic Web, but also in terms of this thesis, they are introduced in the following section in more details.

3.9 Ontologies for Knowledge Bases

Knowledge bases can be implemented in various ways, such as with databases, sets of xml files, sets of UML models, sets of first-order logic statements, or fuzzy systems. The typical way of implementing knowledge bases is the use of ontologies, which are core technologies of the Semantic Web. The knowledge base is frequently considered as an ontology together with a set of individual instances of classes.

The term "ontology" originates from philosophy, where it means a theory of existence [131]. In the area of the software and knowledge engineering, ontologies are considered as a formalism to represent knowledge in a machine-understandable (also called machine-interpretable) form. Many definitions of this term exist in the computer science area. One of the most cited definition is by T. Gruber: "*An ontology is an explicit specification of a conceptualization.*" [55]. Another definition comprises ontology fundamentals: "*Ontology is formal explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on slots (facets (sometimes called role restrictions)). An ontology together with a set of individual instances of classes constitues a knowledge base. In reality, there is a fine line where the ontology ends and the knowledge base begins.*" [130]. Further definitions and details can be found in numerous literature, such as in [131]. The design, populating the ontology with individuals, and maintenance phases are referred as ontological engineering, see [54] for more details.

Ontologies are able to provide a suitable paradigm for knowledge modeling and representation to improve simulation model design and integration. Nevertheless, working with

ontologies and understanding captured knowledge is specific compared to traditional techniques such as object models or relational databases. One of the main factors affecting their use is the "open world assumption". It means that the patterns captured in the ontology need not to be complete. This assumption definitely makes sense in the Semantic Web area as it can frequently happen that another ontology in this virtual space can change the meaning of the set of the original one. On the contrary, engineering projects are typically oriented on a closed-world manner, i.e., each engineering plan summarizes a specific snapshot of a system, yet it reflects only a particular engineering domain point of view or yet the plant itself can be evolving along the project steps or plant operation and maintenance.

In addition to the open world assumption, the Semantic Web relies on the "nonunique naming assumption". It means that the same entity in the Semantic Web can be referred with several different identifiers. This assumption causes that when we have two different identifiers, we will not know, whether they represent the same thing or not.

The open world assumption together with the nonunique naming assumption led to the formulation of an apposite statement called an "AAA slogan": "*Anyone is allowed to say Anything about Any topic*" [2]. As a corollary, a syntactical difference of two resource identifiers does not mean that two different entities are referred.

Since ontologies are frequently used for data modeling of knowledge bases implemented in object-oriented programming languages (such as Java, C#, or C++), ontology models are frequently confused with object models. However they are not the same [141]. The main difference is in the interpretation of attributes, i.e., object variables respectively ontology properties. In object models, attributes are local on the level of each class, whereas in ontology models, they are independent on classes. Ontology properties create separate hierarchies and each property can be reused in various classes, united with other properties, and reasoning can be performed across classes and instances. As subclasses of relations can be created, this is a next aspect specific for the ontology model. Another difference between the object model and the ontology model can be found on the level of classes. Classes are specifications of instance behavior in object models, whereas in the ontology model, they are specifications of the allocation of instances to classes. However, instances are assignable to more classes in the same time, which corresponds to the aforementioned open world assumption of ontologies. The next difference between object and ontology models on the class level is that a relationship can be described between two classes only in the object level. On the contrary, ontologies enable to describe relations between several different classes.

Design of ontologies (i.e., ontology engineering) is a complicated process that can be iterative and that can tackle with evolving nature of domain of interest. A detailed analysis of motivations and benefits of the use of ontologies is addressed in [130], which describes the workflow how to design a new ontology in terms of the specific process steps.

3.9.1 Ontologies and Description Logics

Ontologies and especially the languages how to represent them are based on description logics [12]. The description logics is a family of languages for formal knowledge representation. The representation of knowledge in description logics is based on TBoxes and ABoxes that can be used to characterize knowledge captured in ontologies and to support inferring new pieces of knowledge that are not explicitly included in ontologies.

TBox refers to terminological knowledge. It corresponds to ontology concepts and prop-

erties. It creates a model of the data and it is considered similarly as a database schema. On the contrary, ABox refers assertional knowledge. It corresponds to instances of concepts (i.e., individuals) defined in the TBox. In relational databases, ABox corresponds to the data, which are stored in tables, themselves.

ABoxes in terms of description logics correspond to ontology individuals directly. However, there is a slight difference between TBoxes and ontology classes. Ontology classes can be hierarchically structured and thus they create a glossary. Ontology properties can be specified and assigned with domains, hence relevance of properties to classes can be formulated. This is inline with TBoxes in the description logics, but the difference is that ontology classes cannot be associated with any specific values of ontology properties. In other words, one cannot explicitly distinguish different classes with property values (i.e., parameter values) on the ontology class level, whereas TBoxes enable to make such a distinction. Considering a passive house use-case discussed in Sec. 7.1 as an example, a "bungalow" is a specific type of house that has exactly one floor. However, one is not able to express this terminological fact with ontology, where the number of floors is entered as a defined parameter. This issue significantly affects describing industrial systems that frequently consist of devices being instances of device types.

In conjunction with the description logics, reasoning plays a crucial role. Reasoning is considered as "*deriving facts that are not expressed in ontology or in knowledge base explicitly*" [131]. A large variety of reasoners is available around the world. They can be used to infer knowledge based on transitional properties, such as each individual is the instance not only of its direct class, but also of all its upper classes. Beyond such a transitivity of properties, reasoners can use relatively complex rules to infer new pieces of knowledge, such as WSDL.

3.9.2 Ontology Languages

One of the most important ontology languages is the Resource Description Framework (RDF), which was originally intended to describe resources in the Semantic Web. To do this, RDF utilizes a concept of a Uniform Resource Identifier (URI), which is an identifier for each resource utilized to define namespaces in the similar way as in XMLs. Knowledge is represented in RDF in the form of triples *subject – predicate – object*. A set of such triples in RDF is called an RDF graph. RDF graphs can be serialized into several formats, including the Turtle family of RDF languages, JSON-LD, RDFa (i.e., Resource Description Framework in attributes), and RDF/XML [146].

Another ontology language is an RDF Schema (RDFS). It is built on the top of RDF and provides a basic vocabulary. The important property defined by RDF is `rdfs:subClassOf`, which describes the subsumption of classes. This predicate defines the hierarchy and thus it can be used for defining a glossary. In the original RDF, this issue was not possible in a standard way. Another important keyword specified by RDFS is `rdfs:class`, which is again related to the specification of a glossary.

A further ontology language is the Web Ontology Language (OWL). It is defined in three profiles[6]: OWL Lite, OWL DL (standing for "description logics"), and OWL Full. In principle, reasoning could be done in case of all of these profiles, but the most powerful yet

[6] http://www.w3.org/TR/owl-features/

efficient enough is in case of the OWL DL. Examples of frequent reasoners are Pellet[7] or RacerPro[8]. Hereinafter in this thesis, only the DL profile of OWL is considered.

The OWL is adding further expressive means on top of RDFS. Among others it better tackles classes and defines which of them are disjoint, which is quite important due to the open world assumption of the Semantic Web. The OWL also distinguishes between object properties and datatype properties, which is very important for mechatronic system modeling and supporting simulation model design and integration. The object properties are predicates interrelating either two individuals or one individual and one class. The datatype properties have literals as their values, i.e., they are suitable for parameterizing entities. Similar explicit distinction is frequent in object-oriented programming languages, thus it is easier to serialize and deserialize ontologies from/into such languages.

3.9.3 Querying of Ontologies

Knowledge can be easily retrieved from OWL ontologies with querying languages such as SPARQL[9] or SPARQL-DL[10]). The name "SPARQL" is a recursive acronym standing for "SPARQL Protocol and RDF Query Language". This query language enables to retrieve and to manipulate data stored in the RDF language. It supports triple patterns, conjunctions, disjunctions and other patterns. From the database point of view, the SPARQL usage is similar to retrieving data from NoSQL databases based on a key-value principle (i.e., values are assigned to identifiers called keys).

SPARQL supports four types of queries: SELECT, CONSTRUCT, ASK, and DESCRIBE queries. The most frequent type is the SELECT query which returns a set of n-tuples as its result. It is intended to retrieve knowledge from the ontology, which satisfies the conditions specified in the query. A simpler way of query is the ASK query, which returns whether the conditions specified in the query are satisfied within the ontology. The answer is thus either of type "yes" or empty result meaning that the specified pattern is not explicitly involved in the ontology. A more complicated is the CONSTRUCT query which is intended to create new triples in the ontology. This kind of query consists of two parts, the first one is equivalent to the SELECT query and the second one specifies how the returned results coming from the selective query should be captured in the ontology. Finally, the DESCRIBE queries are intended to get a graph as a result. It extracts from the entire RDF graph those triples that are not breaking conditions stated in the condition-part (i.e., WHERE section) of the query. This is useful for example for analyzing and debugging of the query being just formulated.

3.9.4 Tool Support for Ontologies

Ontologies are widely supported by tools for users as well as by frameworks for developers. As this thesis is focused on the use of OWL DL, we will focus only on this ontology language, respectively this profile of the language. For editing OWL DL ontologies can be used tools such as Protégé[11], which is a widely accepted ontology editor. It enables to create a new

[7] http://clarkparsia.com/pellet/
[8] http://www.racer-systems.com/products/racerpro/
[9] http://www.w3.org/TR/rdf-sparql-query/
[10] http://www.w3.org/2001/sw/wiki/SPARQL-DL
[11] http://protege.stanford.edu/

ontology in terms of classes, properties, and individuals, to check the consistency of the ontology, to visualize it in various ways, etc.

A lot frameworks supporting OWL DL have been implemented and tested around the world. In this thesis, Apache Jena[12] is used. It enables to load and save an ontology, create a new one, add or delete classes, properties, or individuals. It supports diverse storages and representations of ontologies, which is important for using within large-scale projects. Reasoners can be used easily, however, in this thesis, a reasoner is not used finally. A very important feature of Apache Jena is a support for SPARQL, which is utilized in the developed approach.

3.10 Existing Ontologies for Knowledge Representation

Ontologies are utilized in many areas of knowledge management and software engineering. They are typically used for the semantic integration to represent the knowledge about data models in a machine-understandable way as well as in many areas implementing a knowledge base.

Some of the existing ontologies in engineering and process automation domains are described in [104] and in [94]. Based on [104], existing ontologies in related domains can be summarized as follows: OntoCAPE, EngMath, YMIR, PhysSys, MDF, Plant Ontology and Functional Ontology, and ISO 15926.

The most relevant existing ontology is called OntoCAPE [105]. It is a set of ontologies for supporting computer-aided process engineering (abbreviated as "CAPE"), which was designed at the RWTH Aachen University. The OntoCAPE has a modular structure consisting of 60+ OWL-DL files (including the meta-model 80+ files), whose overall file size is 70+ MB. The lowest level is called the application-specific layer and it is intended to connect particular tool knowledge. The next level is the application-oriented layer, which describes the plant equipment and process control equipment. It also includes the view on the particular technology from the process point of view. The third level is the conceptual layer, which provides supporting concepts for modeling of processes, materials, and other representations and characteristics needed for modeling of processes. The very top level is called the upper layer and it provides expressive means for representing networks, coordinate systems and others. Tightly connected to the OntoCAPE is the aforementioned meta model, which is also denoted as meta layer of OntoCAPE. It is represented as a stand-alone ontology, which provides foundations for meta-modeling structures and other fundamental concepts.

A knowledge model for modular manufacturing systems is described in [4]. The basic idea is to describe the machinery and operations semantically in order to facilitate flexibility and agility of manufacturing. The entire ontology is not available online. The head part discussed in [4] includes 29 concepts and 39 properties. The main idea reflected in the ontology is separating and interrelating (i) required operations, (ii) physical machinery performing the required operations, and (iii) control of the machinery.

One of the applications of ontologies is related to multi-agent systems. In [167], the trend in multi-agent systems focused on incorporating semantics and sharing common knowledge captured in the ontology is described. Ontologies can also improve industrial fault diagnos-

[12] https://jena.apache.org

tics [101, 75] and defect detection (i.e., searching for inconsistencies) in process plant and automation system design [88].

In [133], one particular way to ontology transformation is discussed. Ontology translation is a promising approach as an existing ontology could be automatically translated, for example, for simulation design purposes, for multi-agent control system configuration, and for other applications. The utilization of ontologies in the software engineering area as a core of information systems is described in [90].

The shortcoming of the existing ontologies is that they frequently cope with one specific domain, and that they do not cover the whole automation system and simulation. The author of this thesis has not found any satisfactory existing ontology. The possibility of combining domain ontologies and upper ontologies (such as SUMO [112]) was rejected as it does not fit for industrial applications.

3.11 Process Data Representation and Big Data

Handling multi-dimensional data and storing its semantics is addressed in [74]. Two existing data formats are described: (i) COMTRADE is a protocol to store information in binary or text format, and (ii) HDF5 is intended to store data in binary representation and transmitting it to process. The main contribution of the paper is enhancing binary data with semantic information. The purpose is to integrate data from distributed units on the SCADA system level.

Industrial systems generate a large amount of data at runtime. In the last years, an overlap between industrial companies and Web-related companies emerges and the big data approach is coming into an important position.

Big Data are typically characterized by the following dimensions, which are frequently called "three Vs": (i) the "volume" corresponding to the large amount of data such as terabytes or larger, (ii) the "velocity" meaning that the data access is required under real-time constrained manner, and (iii) the "variety" symbolizing that the data can be heterogeneous and unstructured.

Processing of industrial data based on big data approaches is discussed in [132], which was co-authored by the author of this thesis. The idea behind this part of the research is the effort to create a data storage for simulated data and their integration with the real plant data. The big data approach can provide parallelism and efficiency for comparing and retrieving various simulated experiments.

3.12 Industrial Standards for Integration and Communication in Automation

Engineering data can originate from various sources, one of them can be an XML-based representation such as AutomationML or SysML, described in the two following paragraphs.

AutomationML [10] is an XML-based data format for representing engineering knowledge in the area of process automation and control. It is standardized as IEC 62714 [70] and topical issues about this format can be found at the Web page of the AutomationML

e.V.[13]. In the German language, detailed information about AutomationML can be found in [43].

The AutomationML can be considered as integrating format for the following standardized data representations: CAEX for plant topology information, COLLADA for geometry and kinematic information and PLCopen XML for logic information. AutomationML is a descriptive formalism, but it weakly copes with semantics and especially interdependencies between evolving knowledge. Since AutomationML is becoming an important standard in industrial plant description and automation in general, it is supported in this thesis. It could be used as an interface between the artificial intelligence area dealing with ontologies or reasoning, and the industrial practice.

In terms of the CAEX-part of AutomationML, the entire plant/system model is represented as an "instance hierarchy" in the AutomationML format. Devices of the plant are represented as instances, which are called "internal elements" of the aforementioned "instance hierarchy". The type of each device is represented as "system unit class", as each internal element is in this case an instance of a system unit class. The interconnections between devices are represented as "internal links". To support expressing of the meaning of captured information, AutomationML/CAEX defines "role class libraries", which should be shared among various projects. Interfaces of artifacts are modeled with "interface class libraries".

The view on the AutomationML data format from the perspective of data exchange with tool chains in engineering projects is addressed in [148]. The paper also includes a survey among experts on the preferred way of exchanging project data between tools. The majority of preferences was a standardized data exchange format, which was followed by the common project data base to integrate several tools. These preferences are in compliance with the approach proposed in this thesis, as the standardized data exchange format that is supported here is AutomationML and the common project data base is represented by the EKB/automation ontology.

OMG SysML is "*a general-purpose modeling language for systems engineering*" [134]. SysML is based on UML 2 and the specification defines the syntax (notation) for the complete language and specifies the extensions beyond UML 2. SysML is intended to the interaction of machines and humans during system engineering by providing graphical notions, but it is not focused on machine understandable processing of knowledge. SysML could be used to populate the automation ontology with plant data in the future.

The process of integration in the industrial automation area is standardized in ISA-95. In [159], the applications of the standard ISA-95 are described. ISA-95 is focused on vertical (or hierarchical) integration of automation tools and it can be regarded as an integration within the automation pyramid. ISA-95 does not provide a communication language or a standard but rather a methodology to design the system interoperable and it systemizes the integration approach. The standard is widely cited especially when integrating ERP systems and MES systems.

At the borderline between technical and semantic integration, there is a complex standard ISO 15926 [71]. Although it has been originally intended to oil industry, its ideas and approaches are general and usable also for industrial automation systems and integration in general. The crucial part of the standard is the part 2, dealing with the description of ob-

[13]https://www.automationml.org

jects and activities during various stages of the plant life-cycle. It includes diverse views on the process plant depending on the involved engineering disciplines. In the original version of the standard (parts 2–4), the EXPRESS language was used for data and process modeling. Due to a limited tool support for the EXPRESS language, implementation methods based on OWL were added as parts of the standard 7–10, see [84] for further details. The representation of the original part 2 in OWL language is accessible online in frame of the POSC Caesar Association[14].

OPC Unified Architecture (UA) is an industrial standard especially for the integration of field devices. It was developed on a basis of the OPC classic specification and it combines all of the following OPC classic standards: OPC Data Access, OPC Historical Data Access, and OPC Alarms and Events into one unifying specification [93]. The OPC classic is typically used for technical integration of the field level and the SCADA level of the automation pyramid. Instead of such a vertical integration, the OPC UA is able to integrate stakeholders on the same level as it introduces a configurable data model. Unlike OPC classic, OPC UA clients can request data from other clients, hence the UA clients can behave as OPC UA servers as well. OPC UA supports function calls from OPC UA servers and clients. The methodology for a reconstruction of a real plant structure in legacy automation systems utilizing the OPC UA tag list that is compliant with the labeling system IEC 81346, is presented in the co-authored paper [120]. The approach combining the service-oriented architecture with OPC UA is discussed in [100].

3.13 Multi-Agent and Holonic Systems

Distributed nature of many industrial systems and advances in distributed control came up with the concept of multi-agent and holonic systems. This area is addressed in this thesis as well, especially in terms of a connector to multi-agent systems described in Sec. 6.5.5.

Multi-agent systems are systems consisting of autonomous units called agents. Many different definitions of the term agent exist as this paradigm is used in various types of systems of different nature. One of the most famous definitions is by M. J. Wooldridge and N. R. Jennings: "*An agent is a self-contained problem-solving system capable of autonomous, reactive, proactive, and social behavior*" [174]. A multi-agent system term was defined by N. R. Jennings, K. Sycara, and M. Wooldridge as follows: "*An agent-based system is an environment where the agent abstraction is utilized*" [73].

From the industrial perspective, an important term is a holonic system, which means that a system is simultaneously a whole and a part of another system. Concepts for distributed control applications were investigated by Holonic Manufacturing Systems consortium as a part of the Intelligent Manufacturing Systems program. Several holonic manufacturing methodologies aiming at formalization of holon types, their behaviors, and interaction scenarios have been proposed. In addition, specific architectures of these types of systems have been proposed, such as PROSA (i.e., Product, Resource, Order, Staff Architecture), ADACOR (i.e., Adaptive Component Based Architecture), or HCBA (i.e., Holonic Component Based Architecture) [163].

Industrial process control is typically based on PLCs. Therefore, holonic and multi-agent systems have a layered architecture for control purposes, which consists of a low-level

[14] https://www.posccaesar.org/wiki/ISO15926inOWL

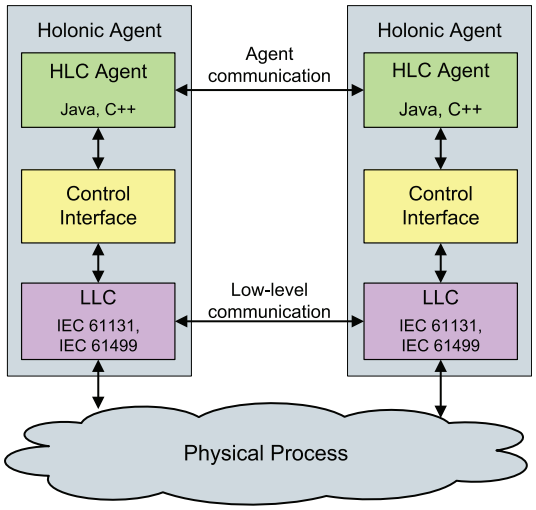

Figure 3.2: Holonic architecture, adapted from [156].

control (LLC) module and a high-level control (HLC) module, see Fig. 3.2. The LLC module is a control program running in a PLC in a classical scan-based manner, whereas the HLC is implemented by software agents, which communicate with LLC via a control interface.

In parallel to the standardization of holonic systems, an intensive effort has been done in the area of providing standards for the MAS domain as well. The Foundation for Intelligent Physical Agents (FIPA) organization produced sets of standards covering agent management, communication, and message transport. The FIPA standards introduce two services in MASs: (i) the Directory Facilitator (DF) is a list of agent capabilities, thus it is informally called yellow pages; and (ii) the agent management system (AMS) registers all agents existing in the MAS. Agent communication utilizes frequently a Contract-Net protocol (CNP), which is based on two stages for negotiation between agents [78]. Since this protocol does not lock resources, more sophisticated protocols have been proposed, such as Plan-Commit-Execute (PCE). Various agent platforms supporting running and development of agents such as JADE, ACS, FIPA-OS, AGLOBE, MAdkit, JACK or Zeus are available. Some of those platforms are evaluated in [58]. Design, analysis, and testing of multi-agent systems in general is discussed in [158].

Various applications of multi-agents systems can be found in [99]. Industrial applications of multi-agent technologies are summarized in the review [163]. Experiences with industrial adoption of holonic and agent-based systems for process control are described in [56]. The paper identifies four claiming areas: "*(i) Lack of skill in distributed thinking, (ii) Determining emergent behavior, (iii) Cost of adoption and implementation, and (iv) Design and maintainability of agent-based systems*" [56].

Multi-agent systems are by their nature close to IEC 61499 standard, which defines block-based PLC programming. It is considered as a promising successor of IEC 61131. Benefits of IEC 61499 for reusability of DCS applications are explained in [61]. Since only a few PLC vendors offer IEC 61499 controllers, the approach presented in [156] introducing the Agent Development Environment (ADE) supports the IEC 61131-3 standard of PLC programming languages for LLC implementation.

3.14 Semantic Technologies in Building Automation

Building automation is an emerging area whose importance is growing significantly. A use-case dealing with automation of passive houses is included in Sec. 7.1 to evaluate the proposed method for a non-traditional type of systems. Hence this section briefly introduces related work in the area of semantic technologies in building automation.

Four-layer system architecture bridging the gap between building devices (on the low control level) and semantic service-oriented technologies (on the high control level) is presented in [6]. These four layers are (1) device layer, including various devices connected via diverse buses, (2) connectivity layer, providing abstraction of the networking, (3) service layer, including safety and control loops such as authentication or temperature control, and (4) semantic agents facing high level goals.

The paper [142] deals with integration of heterogeneous building automation systems (BAS). Four protocols (BACnet, KNX, LonWorks, and ZigBee) are supported. The knowledge is captured in the BAS ontology in order to (1) configure the heterogeneous system centrally, (2) thanks to the machine-interpretable data representation an access point for other systems is realized, and finally (3) the ontology alleviates overhead that is encountered when heterogeneous systems shall be integrated. The proposed BAS ontology is hierarchically based on "Function_Block", "Datapoint", and "Address" concepts.

A flexible Service Oriented Architecture (SOA) platform for controlling and coordinating devices is proposed in [140]. Each controllable device from a house environment is modeled as a Web service, whereas functionalities such as pressing a button are modeled as Web service invocations of particular operations. A more generic approach for the building automation and control is presented in [39]. It proposes an efficient process for automating the design of building automation systems based on OWL ontologies.

In the use-case presented in Sec. 7.1, the utilized simulation library is called the Building Simulation Library. It was designed and implemented by the author of the thesis in his previous work [115], [117], and [125]. However, other simulation libraries for building simulation exist as well, for example the International Building Physics Toolbox (IBPT) [79]. It is an open-source library implemented in MATLAB-Simulink, which was originally developed for heat, air, and moisture system analysis. The toolbox defines a common modeling platform including unique communication signals, a material database and a documentation protocol.

Chapter 4

Knowledge Models for Improved Simulation Model Design and Integration

This chapter is focused on representing engineering knowledge to support the design and integration of simulation models for industrial systems. It proposes the automation ontology and its data model, which is the solution of the knowledge base. It is intended for capturing engineering data relevant for efficient simulation model design and integration. The chapter includes research contributions of the author into the area of engineering knowledge modeling. In particular, the chapter provides a solution for the research issue RI-1 and fulfills the goal of the thesis G-1. Last but not least, this chapter provides foundations for the two subsequent chapters.

4.1 Engineering Disciplines and Engineering Plans

Engineers of each engineering discipline utilize a set of software tools and types of engineering plans that can differ or overlap between various disciplines in mechatronic systems engineering. In many cases, even engineers of the very same discipline use different software tools because of the tool capabilities. It is necessary to bridge the gap between engineering tools and engineering plans but also between terminology of engineers of different disciplines.

Since complex mechatronic systems are very complicated frequently, engineers utilize domain-specific languages (DSLs) to describe relevant concepts and properties from particular discipline points of view. System modeling and designing simulation models is thus a cooperative work relying on engineers of various engineering disciplines. This is in compliance with the description in the monograph [8]: "*To capture phenomena of interest accurately and in tractable mathematical form is a demanding task, as can be imagined, and requires a thorough understanding of the physical process involved. For this reason, the mathematical description of complex electrical systems, such as power systems, is typically accomplished by electrical engineers, the equations of flight dynamics of an aircraft are derived by aeronautical engineers, the equations of chemical processes are arrived at by chemists and chemical engineers, and the equations that characterize the behavior of economic systems are provided by economists. In most nontrivial cases, this type of modeling process is close to an art form since a good mathematical description must be detailed enough*

to accurately describe the phenomena of interest and at the same time simple enough to be amenable to analysis" [8].

Prior describing how to represent engineering data and in particular how to represent them within the proposed data model of the automation ontology, the basic terminology from the area of industrial automation systems is summarized in this paragraph. The term real plant means a physical industrial system. An example of a real plant is a laboratory tank model, which is later used as a use-case in Sec. 7.2. The real devices in hydraulic systems are tanks (i.e., vessels), pipes, pumps, and valves. Examples of device parameters are lengths of pipes, volumes of tanks, or maximal flows through pumps. Parameters are properties of real devices and they are typically constants characterizing their size or shape. All values measured in the system by sensors are typically available on a software level on OPC[1] servers. Each variable on the OPC server is called an OPC tag and it is a triple (name of the variable, timestamp, and value). Values of tags are time-series. They are typically control actions for a real plant (i.e., inputs into the real plant), and measured variables (i.e., outputs of real sensors). Tags are also inputs and outputs of simulation models in the very same way as in the real plant case.

4.2 Design of the Knowledge Base

Knowledge about real systems involved in mechatronic system engineering has to be captured in the knowledge base. The data model of the knowledge base for simulation model design and integration is the automation ontology, which was designed by the author of this thesis (see for example [122] or [123]).

In this thesis, the concept of the engineering knowledge base is implemented by the automation ontology implemented in the OWL language and the ontology tool encapsulating the ontology. According to [85], the utilization of OWL ontologies provides the following benefits: reuse and interoperability, flexibility, consistency and quality checking, and reasoning. The application of ontologies in the semantic integration and design area is beneficial as it is related to the creation and evolution of the data model, which is not known from the beginning. That is the reason why relational database technologies do not frequently fit for this task. Several approaches utilize knowledge representation in XML formats, such as AutomationML or SysML. Their common shortcomings are difficult querying or inferring new pieces of knowledge as well as particular files cannot be combined easily. Still emerging direction trying to find a compromise between the high-performance relational databases with pre-defined data models and between the light-weight approaches with evolving data models (such as ontologies) are NoSQL databases. Basically, they are based on the key-value principle, where for each unique key is assigned a specific value. The evaluation of such technologies for automation system engineering projects is presented in [107]. The common point of all the four technologies is that if users are not skilled in these technologies, it will be difficult to work with them. Ontologies were selected as the most promising technology for the implementation of the knowledge base. Since the target users of the proposed approach are control engineers or experts in a simulation domain, experiences with ontologies are not expected. The solution thus proposes to encapsulate the automation ontology with the ontology tool implemented in Java. Although several names for this tool have been used, for simplicity reasons, it is called generally the "ontology tool" in this thesis.

[1] OPC stands for Object Linking and Embedding for Process Control

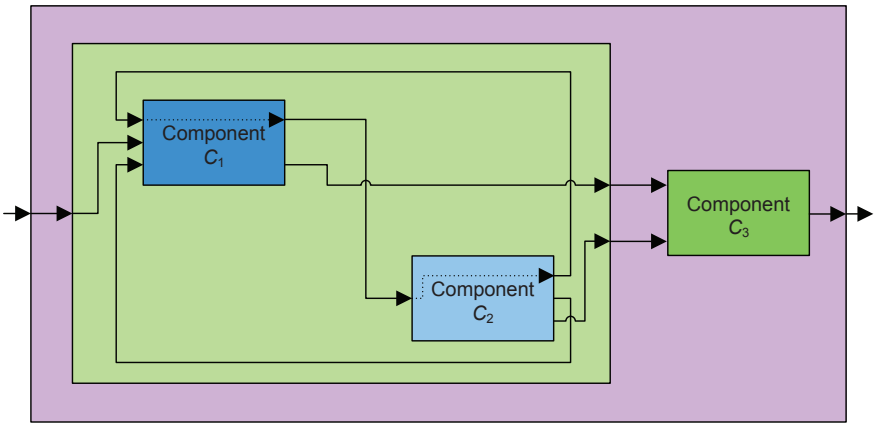

Figure 4.1: The patterns that are frequent in simulation and automation system engineering and that have to be supported by the automation ontology.

The common model has to combine information from various engineering domains creating a knowledge base that integrates information from available engineering sources such as electrical schemas, mechanical plans, or P&IDs (i.e., piping and instrumentation diagrams that are used for the technology description). The knowledge base supports efficient querying of the captured knowledge and inferring new pieces of knowledge, thus the information from the knowledge base can be easily used for supporting the design and integration of simulations. In future, such knowledge could be also used to support PLC programming or intelligent fault diagnostics.

4.3 Requirements on the Ontology Model

Based on the author's experiences, talks to industrial partners, and literature review, the required patterns to be supported by the designed ontology can be expressed by Fig. 4.1. It depicts the following frequent patterns that should be supported by the designed ontology:

1. Part-whole relationship;
2. Hierarchical (i.e., layered) structure of the system, including interfaces on the hierarchy levels;
3. Connections between components including port numbers (i.e., it is not enough to express two components are connected, but it is needed to express for example that the first output port of C_1 is connected with the first input port of C_2);
4. Coupling of several signals within input and output component interfaces (i.e., dotted lines in Fig. 4.1).

To represent mappings between simulation components and real devices needed for simulation model design, the proposed data model has to be able to represent the mapping pattern as it is symbolically depicted in Fig. 4.2. The idea behind this pattern is that each real device (in this case a resistor), can be modeled by one of n corresponding simulation blocks. In this case, two simulation blocks for a resistor can be used in the topology of the simulation model. This assumption is crucial for the proposed method for the simulation model generation and thus this pattern has to be addressed by the data model of the knowledge base.

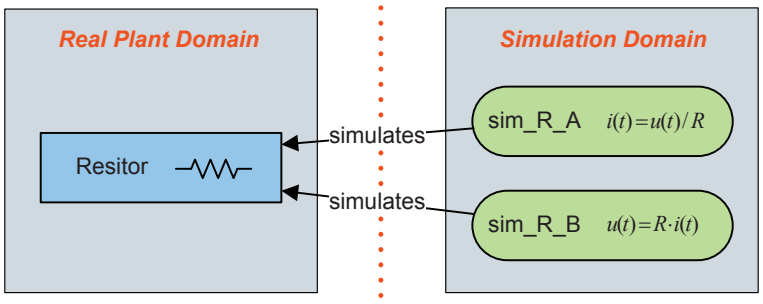

Figure 4.2: Example of mappings between real devices and simulation blocks in case of the electrical resistor. This mapping means that each resistor in the real system topology can be modeled by one of these two simulation blocks differing in input and output interfaces.

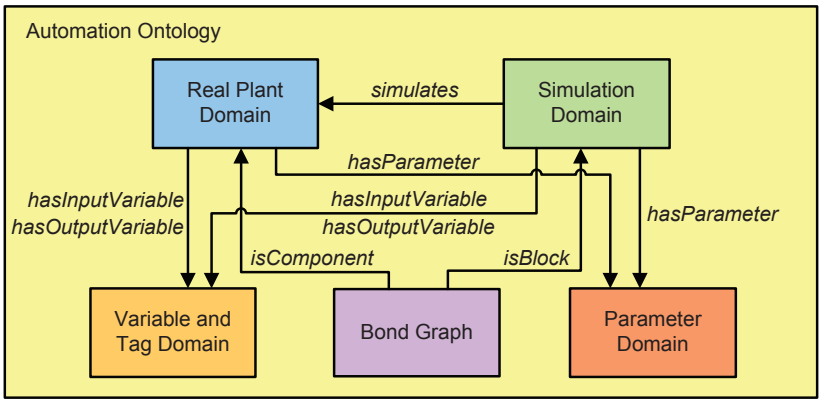

Figure 4.3: A simplified overview of automation system domains relevant for structuring of the proposed automation ontology.

4.4 Automation Ontology

The knowledge base formalizes relevant information for integrating and designing simulation models for industrial plants that can have a mechatronic nature. This section discusses the proposed data model of this knowledge base, which is suitable for the purposes of the simulation model design and integration.

4.4.1 Domains of the Automation Ontology

To introduce the proposed data model of the knowledge base, Fig. 4.3 depicts the most important domains represented in the designed automation ontology: real plant domain, simulation domain, variable and tag domain, parameter domain, and bond graph domain.

The real plant domain represents the structure of a real system, i.e., it includes physical devices and their connections. The simulation domain is focused on simulation models, which approximate the behavior of the real plant. This is the reason why the simulation domain and the real plant domain are connected with an arrow, expressing which parts of the real plant are simulated by simulation modules. Both real plant and simulation domains have variables/tags and parameters. Tags are variables with a unique name, which are inputs and outputs of devices or simulation models. On the contrary, parameters are properties of devices that characterize behavior and features of devices, models, etc. The difference between tags and parameters is that tags are time-series (i.e., sets of samples),

whereas parameters are constant values (i.e., single values). The variable and tag domain in Fig. 4.3 represents variables and tags that formalizes their names and types and interrelates diverse tag representations in different domains. The parameter domain formalizes parameters in the system and interrelates diverse parameter representations in different disciplines and domains. The bond graph domain represents elements of bond graphs including their extended version proposed in this thesis. It captures components, junctions, and connections, as well as causalities and power directions. This domain is crucial for generating simulation models in specific simulators as an intermediate simulation-independent step.

Following this brief introduction into relevant domains and required patterns, the further text explains the proposed data model in more details.

4.4.2 Real Plant Domain

The sub-ontology for modeling real industrial plants is depicted with blue blocks on the left-hand side of Fig. 4.4. A real system is denoted as an individual of the "real plant" concept. Each plant is structured into one or more "locations", which can typically correspond to geographical positions or functional behavior of system parts. Such plant locations consist of one or more "real devices" (such as pumps, pipes or tanks), which are key entities of the real system description. Furthermore, it is not enough just to list real devices, but it is very important to express how they are interconnected. The physical interconnections are formalized based on the predicate "hasPowerBond", whose meaning is adopted from the bond-graph theory. The second kind of a connection "hasSignalBond" represents the information flow between subsystems.

Each real device can have one or more "parameters" (such as diameter or length) and it can have input and output "variables" (e.g., flow or pressure), whose unique names are "tags" (e.g., "flow205" or "pressureTank101"). Parameters define conditions for the operations, whereas variables or tags are inputs and outputs. Since it is needed to intermap tags and parameters from various engineering disciplines and since such correspondences are crucial for simulation integration purposes, the proposed formalization considers them as stand-alone domains, which are described in the following subsections.

4.4.3 Variable and Tag Domain

Variables are time-series that are measured by sensors in the real plant, created by users, or exported by a connector in the case of simulations and other software systems. At runtime, values of each tag are represented as a set of triples (name, timestamp, value), which are called in the presented formalization "samples". In Fig. 4.4, variables are depicted in the middle part of the figure with the yellow color.

A tag is a concept representing unique names for variables. Tags are distinguished by a data source, for example, a "real tag" represents tags on the OPC server related to real variables, and a "simulation tag" is related to inputs and outputs of simulation modules. These tags must be inter-mapped in order to express that they have the same meaning, but their values can be different due to the various sources. Within the technical infrastructure presented later in Sec. 6.2, such tags must have unique names. The tag names are translated on tool connectors between the global names (i.e., tags shared within the EngSB-based infrastructure) and local names (defined by real, simulation and other types of tags).

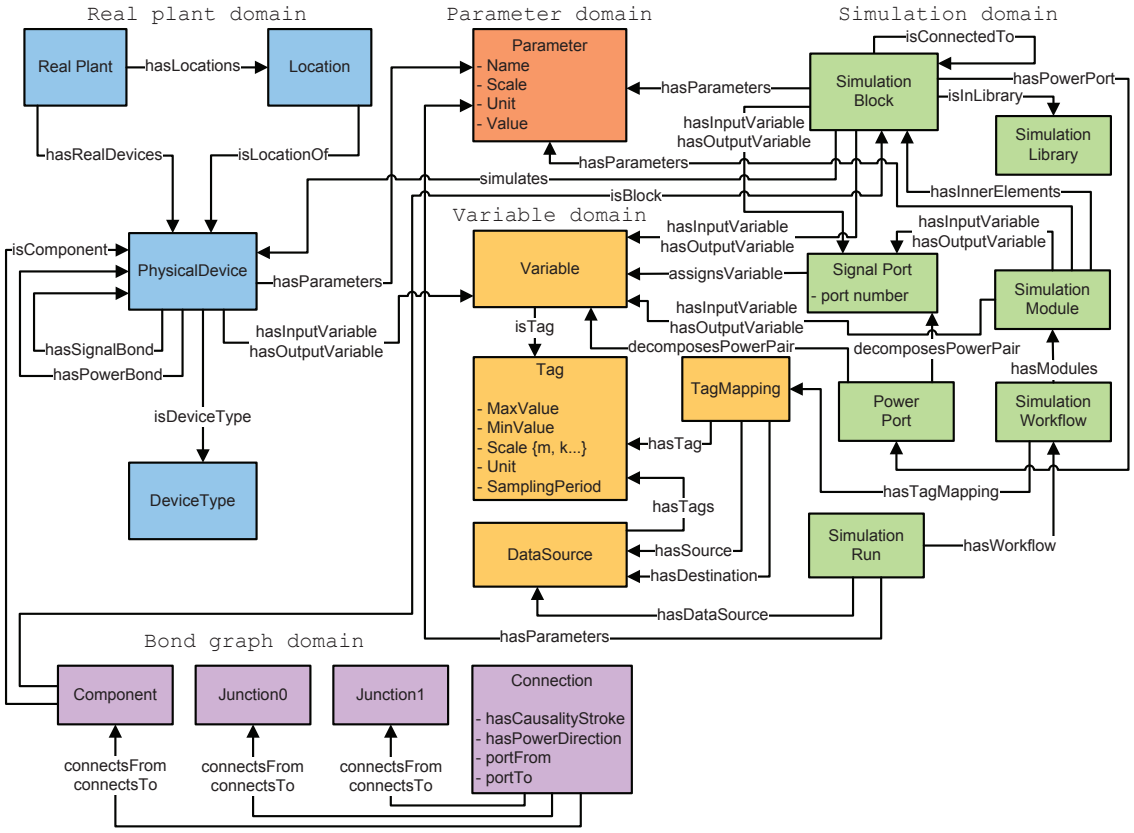

Figure 4.4: Simplified overview of the automation ontology showing the main classes and their relationships.

For integration purposes, "TagMapping" instances have a crucial role. This type of mapping specifies, between which stakeholders each specific tag values should be transferred, including the direction of the transfer. Tags are stored in various data sources, which are formalized by the concept "DataSource".

4.4.4 Parameter Domain

Parameters are properties of devices representing their features, shapes, and sizes (e.g., diameter of a pipe or nominal voltage of a voltage source). Parameters are constants, i.e., values independent on time.

The knowledge base distinguishes several kinds of parameters as each domain (i.e., real plant, simulation, or others) can be related to a dedicated set of parameters. From the user point of view, the most important set of parameters are parameters of a real system and its devices. These parameters are related to simulation modules and blocks. Control experts are familiar with various approximations of device or system behavior, such as transfer functions. For example, time constants and steady-state gains can be simulation parameters related to simulated devices. In general, simulation parameters can be based on real parameters, but due to simplifications of the simulation model or due to lack of information, approximations of real parameters can be used instead. The second important group of parameters is related to the run of the simulation model. They configure the simulation solver, initial and final time of the simulation, etc.

A special set of parameters are "simulation run parameters", which are related to one particular run, i.e., execution of the simulation solver on the simulation model under exactly defined conditions. For example, simulation run parameters can be the start time and the stop time, a type of a solver, or minimal and maximal time steps. Last but not least, simulation run parameters also cover initial conditions, which define the state of a system at the beginning of a simulation. This information depends on a specific scenario. For example, initial conditions are the liquid level or temperature in each tank at the simulation time $t = 0$.

Further tools can have their local parameters as well, such as HMIs, data sources or others. They can be based on real parameters, but can be also anyhow transformed. An example of this set of parameters is the setting of the connections of HMIs and OPC servers, refresh periods, etc.

4.4.5 Simulation Domain

Each simulation model can be composed of one or more "Simulation Modules", which are defined as abstractions for any executable simulation files. Examples of implementations of simulation modules can be MATLAB M-file, MATLAB-Simulink model, or a file of a fuzzy rule-based simulation. A simulation model of a real system can be decomposed to more than one simulation modules for example according to the spatial location of devices, functional behavior, etc. To execute the simulation, further modules can be needed, e.g., to prepare input data or to process output data, calculate statistics, etc. These modules do not implement the model of the real system itself. Simulation modules are executed in a specific order, which is formalized in the knowledge base with a class "Simulation Workflow". The process of decomposition of simulation problems into simulation modules and specifying

simulation workflows is strongly project-dependent. Experiences of simulation experts in these tasks are not substituted, but just supported both in the design and integration phases. On top of the simulation workflow, the class "Simulation Run" assigns a set of parameters to be used.

Simulation modules consist of simulation blocks. Simulation blocks can be either user-defined (used for specific and not repeating blocks) or included in a simulation library. The knowledge base model includes a class "simulation library" that represents available libraries and their simulation blocks. Typically, such blocks included in libraries have parameters only declared but not defined (they are called generic blocks), whereas their instantiations in simulation modules should be parameterized with specific values according to the required behavior.

The simulation domain is mapped to the real plant domain via the property "simulates" that expresses which real devices are simulated by specific simulation modules and blocks.

4.4.6 Bond Graph Domain

Bond graphs as well as their extended version require support in the automation ontology in terms of being able to model all needed concepts properly and efficiently. This sub-ontology is used to serialize the semi-automatically created bond graphs and thus it reflects the class model of the developed tool support in Java.

The "Node" concept should not be instantiated as it represents an abstraction for components/blocks and junctions for efficient implementation of the proposed method. In Java, the corresponding object is defined as an abstract object, this hierarchy was adopted to the automation ontology as well. Due to a low importance, it is not depicted in Fig. 4.4.

Instances of the class "Component" are components in the bond-graph sense, as it has been presented in Sec. 2.3.2. The instances of this class have links to specific simulation blocks, which are the appropriate approximations of these components. When the simulation model is created properly, each component links exactly one simulation block as its model. In addition, components have links to real devices, expressing which part of the plant they model.

The "Junction" concept should not be instantiated as it represents an abstraction for two types of junctions provided by the bond-graph theory, similarly as the Node concept on the upper level of the taxonomy. Instantiable objects are classes "Junction0", whose instances represent 0-junctions of bond graphs in the sense presented in Sec. 2.3.3, and "Junction1", whose instances represent 1-junctions of bond graphs in the sense presented in Sec. 2.3.3.

The class "Connection" and its instances represent signal connections in the sense of signal-oriented simulators. They are a union of the set of all signal bonds and the set of signal bonds created as decomposition of power bonds into signal bonds. As datatype properties, connections have assigned causalities and power directions in terms of the bond-graph theory. Moreover, they can capture information about port numbers of nodes on both sides of the connection, which is desirable especially in the case of signal-oriented simulations such as MATLAB-Simulink.

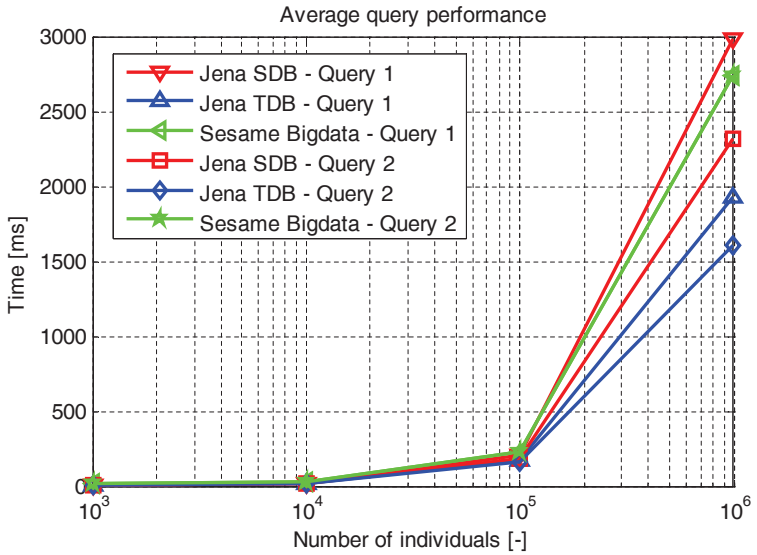

Figure 4.5: Analysis of the scalability of the ontology-based approach showing the increasing time per query depending on the number of individuals in the ontology. The figure was simplified from [149]; in this version, it has been published in the article co-authored by the author of this thesis [122].

4.4.7 Summary and Evaluation of the Automation Ontology

The automation ontology is implemented in OWL-DL, which was selected as an optimal compromise between expressive power and the possibilities to perform reasoning. Required pieces of knowledge are retrieved from the automation ontology by SPARQL queries. For the knowledge retrieval, the "SELECT" SPARQL queries are used, whereas for the knowledge transformation (which is used for example when transforming data from AutomationML as it is described later in Sec. 6.3.2), the "CONSTRUCT" SPARQL queries are used.

The main bottleneck, which was identified during analysis, testing, and profiling of the implemented software prototypes, was identified as the access to ontologies. However, the utilized Apache Jena framework reaches very good performance results, as it is demonstrated in Fig. 4.5. Based on this evaluation, the ontology-based approach was found as feasible and suitable for simulation model design purposes as well as for the configuration of a technical infrastructure providing runtime integration, because these tasks are not time-critical issues. The overhead of processing ontologies does not bring any significant limitations and thus it is a good choice due to flexibility of the data model and its easy adaptability.

The completeness of the ontology model was tested on the level of various use-cases dealing with systems of four engineering disciplines (i.e., hydraulic, electrical, chemical, and thermal systems) and it was found out as appropriate and complete.

The performance and efficiency of the ontology together with the aforementioned software prototype of the ontology tool was tested with various load tests. It was found out that this approach is suitable for 1 million of ontology individuals, which corresponds to engineering systems having approximately 30,000–50,000 real devices [122]. This is fully compliant with the scope of this thesis. If industrial applications require systems having higher scale, big data approaches based on map-reduce architectures pose a promising way.

The rate of improvement compared to the current status used in industrial practice is very high as knowledge bases are not used in majority of simulation engineering projects.

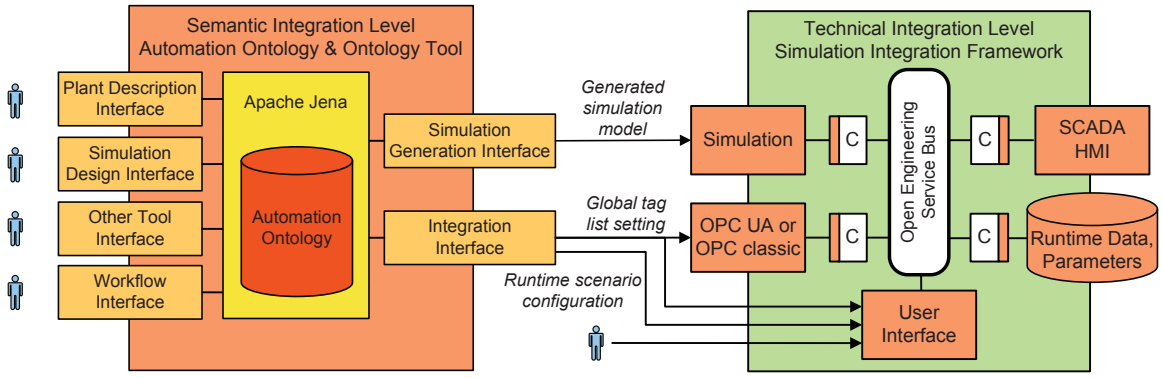

Figure 4.6: The overview of the ontology-based architecture in terms of its intended usage by experts of various engineering disciplines.

Since target users are not expected to access or to work with the ontology directly and it is encapsulated at least on three levels in the developed and tested cases, the user friendliness need not be taken into account as it is mainly affected by the specific wrappers.

Overall, the proposed representation of engineering knowledge was found out to be useful and efficient and it has significant potential to be applied in industrial practice satisfactorily.

4.5 Software Prototype of the Ontology Tool

The knowledge base should support domain engineers. For simulation or domain experts, it is not efficient to learn ontologies and to experience how to use them. Hence the utilization of ontologies should be transparent for final users. Therefore, they are encapsulated by the ontology tool. It was developed on the software prototype level in order to verify correctness, completeness, and efficiency of the proposed method and approaches.

The ontology tool utilizes Java ontology API called Apache Jena[2], which includes a SPARQL language support. Ontologies are stored as files on hard-drive and the performance was sufficient for all test cases that were used. If the performance was insufficient for a target user, the Jena framework supports the storage TDB, whose performance is significantly higher and it seems to be fully compliant with the simulation engineering requirements.

After collecting all relevant pieces of information in the automation ontology, they are processed and utilized for supporting the design of simulation models and the configuration of the technical infrastructure for runtime processes. Such an integration approach is a typical example of the model-driven system configuration [106].

The goal of the ontology tool is to provide interfaces (i) for populating the automation ontology with individuals representing a real plant structure, (ii) for retrieving knowledge out of the ontology and creating configuration files for the simulation integration framework and finally (iii) for generating simulation models semi-automatically. The structure of the ontology tool and its interfaces is depicted in Fig. 4.6. On the left-hand side, a support for engineering tools, which is oriented on populating the ontology with individuals, is demonstrated. The output interface for the generation of simulation models has been already implemented on a software prototype level. The output interface for the generation of configuration files has been implemented on the proof-of-concept level.

[2]https://jena.apache.org/

Chapter 5

Extended Bond Graphs for Object-Oriented Simulation Model Design

Design of simulation models is a complex task including various complicated process steps. This chapter proposes the author's method to improve this process by modifying it into such a form that can be semi-automatically solved by a machine. The improved method saves time and costs needed for the design and re-design of simulation models and mitigates design-time errors. The proposed method has been implemented and verified on several independent use-cases. This chapter addresses the research issue RI-2. It also provides theoretic foundations for the goals of the thesis G-2 and G-3, which however rely on technical issues discussed later on in Sec. 6.

5.1 Design of Simulation Models

Hereinafter, two basic scenarios for simulation model design are distinguished and improved:

1. Design and implementation of a simulation model without any prior artifacts (i.e., design from scratch)
2. Creating a simulation model with the use of simulation blocks from an existing simulation library

The process steps of these two scenarios are compared in Fig. 5.1. In both cases, general knowledge about the type of the real plant and its topology has to be captured in the automation ontology. For example, the general knowledge summarizes that water distribution networks can contain devices such as pumps, pipes, tanks, water wells, consumers or disturbances; pumps have variables flow and pressure as their inputs and flow and pressure as their outputs, real parameters of these devices and of simulation components can be length, diameter, or elevation. The general knowledge is a kind of knowledge skeleton, which can be filled up with real values when describing a specific plant. If needed, the general knowledge can be extended with other parameters which are device-specific. The process steps consequent to the specification of the general knowledge differ in case of available and unavailable

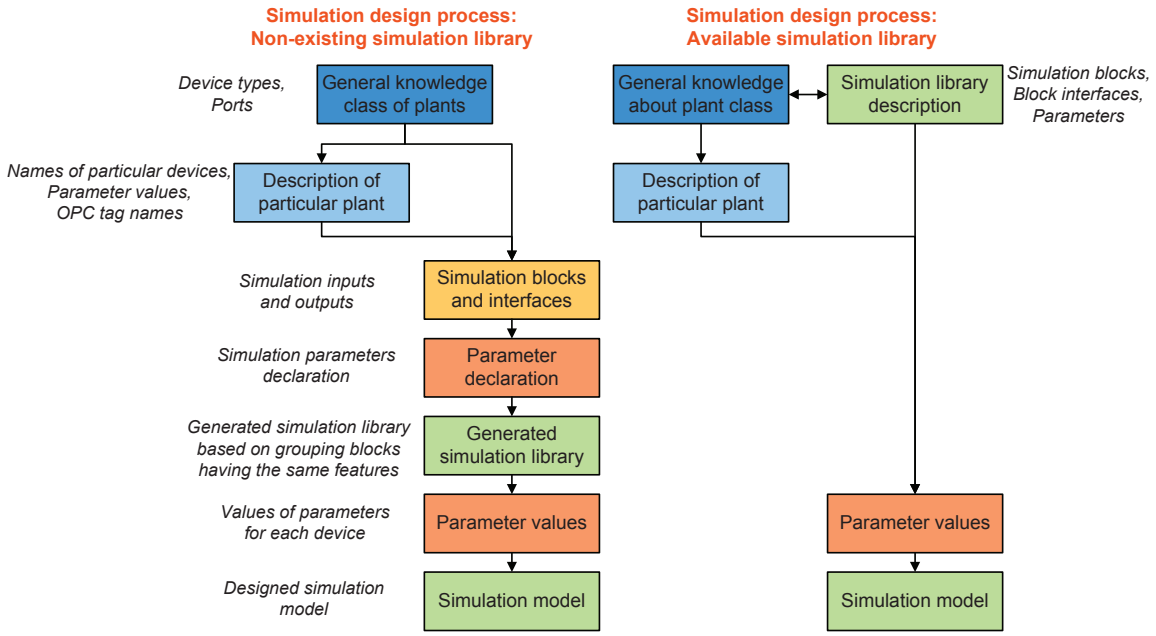

Figure 5.1: Simulation design workflow – Comparison of scenarios starting from scratch and utilizing the simulation library.

simulation libraries. Therefore, these workflows are described in the following subsections separately.

5.1.1 Simulation Design Scenario when a Simulation Library is not Available

The first scenario assumes that a simulation model is created without proper simulation artifacts (i.e., without simulation blocks or modules from previous projects). The method handles the process of gathering the following sequence of knowledge: decomposition of the real plant into simulation modules, selection of new simulation blocks on the device level and specification of their interfaces, declaration of simulation parameters, entering values of the parameters, and finally uploading the new simulation model into the simulation integration framework environment and registering it as one of the available simulations.

In more details, the general knowledge is extended with a particular plant description in a similar way, but afterwards, the simulation model structure cannot be obtained automatically. The considered simulation design process is depicted on the left-hand side of Fig. 5.1 and it is followed in the further text. The process of gathering expert knowledge starts up with specifying, which of the devices should be simulated. The step includes grouping devices into complex subsystems (such as the whole plant can be decomposed to several simulation modules or a group of devices can be modeled as one block) as well as a specification of block interfaces (such as the utilization of just one of the variables "flow" and "pressure" or both of them, an extension with further signals such as control values for devices). Afterwards, for each simulation block, simulation parameters have to be declared. For example, a pipe can have just one parameter R denoting the relationship between its flow and pressure. Another pipe, being very long, can be modeled in a different way, having two parameters τ and a denoting a time constant and a steady-state gain of the first-order dynamic system. The subsequent step of the workflow can seem surprising at first, because

the ontology tool can create the simulation library as a result of the knowledge capturing process. Since simulation libraries comprise generic blocks having defined input and output interfaces and simulation parameters, the tool can select the emerging simulation blocks, which have the same interfaces and parameters, to group them, and to extract the simulation library automatically. This issue is easy to solve using the ontology, and it saves development time and costs significantly compared to the manual expert work making the same substitution. Finally, the parameter values must be entered and the simulation model is ready to use.

5.1.2 Simulation Design Scenario Based on an Available Simulation Library with Simulation Blocks

If the simulation library is available (see the right-hand side of Fig. 5.1), the library blocks are annotated in the automation ontology incorporating their inputs, outputs, simulation parameters and initial conditions. Consequently, the plant description has to be formalized, which means that a real plant topology and real parameters are stored in the ontology. Technically, this step means populating the automation ontology with individuals denoting real devices, their interconnections, tags, and others. Based on these pieces of knowledge, the ontology tool is able to generate the simulation model structure automatically. Finally, the simulation expert is required to insert simulation parameter values for simulation blocks, such as diameters or lengths of pipes.

The components are considered as atomic objects, i.e., their internal representation is not considered, but only input and output interfaces are important. These components are hereinafter called atomic components. Although their internal representation is not taken into account for simulation design purposes, it has to be developed when the component is created or later until the component is simulated. The internal representation of these atomic components is based on mathematical-physical description. To support this task, a large variety of monographs can be used. Some of them are directly intended for the simulation engineering discipline (e.g., [113]), or general monographs on physics can be used, which are frequently structured according to system types (e.g., [57]). However, even if the simulation expert has such a simulation library with simulation blocks, their manual instantiation and inter-wiring are extremely time-consuming and error-prone. This issue was raised as a motivation for semi-automating this task, which is addressed in the following text.

5.2 Motivation for a New Method Supporting Multi-Level Object-Oriented Simulation Modeling

Researchers and practitioners emphasize the need for facilitating the design phase of simulation models. For example, the German standard VDI 3633 [160] claims that the design phase is the most time-consuming part of simulation modeling. Prior to explaining the simulation design method proposed by the author of this thesis, this section summarizes the requirements and challenges that should be satisfied by the proposed method and that drove the main part of the research presented in this thesis:

1. *Reuse of information from engineering plans to automate the simulation design*

The simulation model structure is based on the topology of the real industrial plant. Currently, engineering plans are used by a human expert creating a simulation, but the proposed method should reuse the original plan and eliminate repeating manual work.

2. *Support for a component-based approach*

   Simulation models of large-scale industrial systems typically consist of simulation components representing sub-parts of the plant. The goal is to handle each component as a whole and to work just with its interface, no matter how the block is internally implemented.

3. *Support for networks of components having an arbitrary topology*

   By definition, systems consist of networked components. Since the systems can have very complicated topologies in industrial practice, the goal is to support an arbitrary topology of the plant. For example, the method should support multiple parallel and serial coupled connections of components, or there should not be any maximal number of parallel branches for any junctions.

4. *Selection of appropriate simulation blocks*

   Even simple devices can have several implementations of their mathematical-physical description, which are called simulation blocks in the context of this thesis. Although they can be equivalent from the mathematical point of view, a simulation expert or a machine has to decide, which of those implementations to use for each instance of a component in the system topology. For example, a pipe in a hydraulic system can be modeled as (i) a transformer of the pressure loss to the liquid flow, or (ii) it can calculate pressure loss based on a given flow through the pipe. The method should address this problem and automatically select, which of the specific simulation blocks to use.

5. *Support for hierarchical topologies*

   A level of abstraction plays a crucial role in all engineering systems. The proposed method should support the hierarchical principles to enable the use of hierarchical structures of simulation models seamlessly.

6. *Specification of interfaces for integration of simulations*

   Simulation models can consist of several simulation modules. The proposed method should facilitate the specification of module interfaces as well as their inter-linking. In addition, a crucial issue in the use of simulations is the access to online and historical data. Therefore, the proposed method should also support specifying borderlines between simulations and SCADA systems.

5.3 Motivation for a Simulation Block Selection

The fourth challenge defined in the previous section (i.e., the selection of appropriate simulation blocks) played a crucial role in the investigation of a new simulation design method. Due to the importance of this topic, it is described in more details in this section.

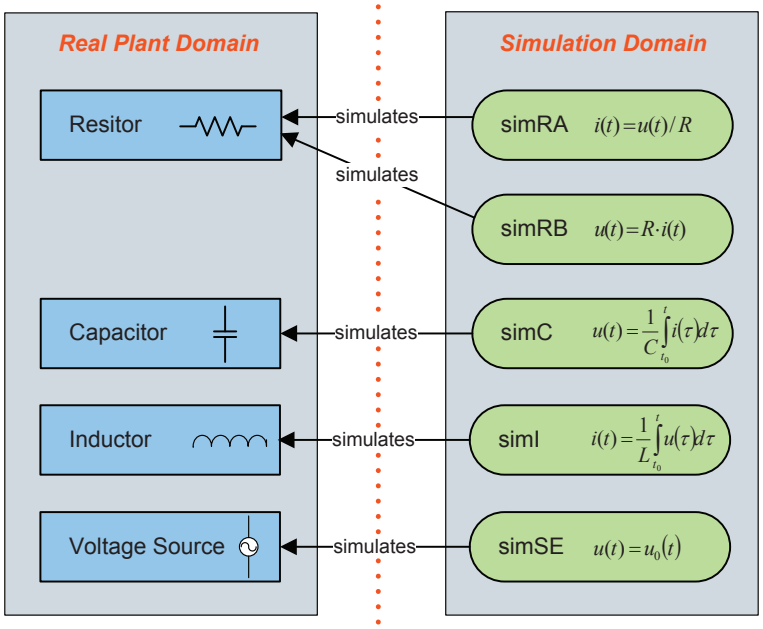

Figure 5.2: Mappings between real devices and simulation blocks needed for the design of a simulation model for the electrical circuit.

When the mapping between real devices and simulation blocks is $1:1$, the situation is quite simple. Such a case can be found in the passive house use-case discussed in Sec. 7.1. Each room, for example, is modeled by one simulation block representing the room. The algorithm selects the block according to the mapping between real device type and simulation block. Such a mapping is expressed by the object ontology property "simulates" in the automation ontology. But if the mapping is $1:n$, the selection is complicated. One real device can be simulated by one of n simulation blocks. The task for the design method is to decide for each device in the real plant topology, which of the simulation blocks should be used.

Two implementations for each one-port component can exist in signal-oriented simulations. The first simulation block has flow as input and effort as output, whereas the second one has effort as input and flow as output. This is the basic distinction, which is focused on input and output interfaces. In addition, the simulation blocks can differ in parameters or additional signals as well.

An example of an one-port component is a resistor in electrical systems. In electrical systems, the flow signal is electrical current $i(t)$, whereas effort is electrical voltage $u(t)$. For simplicity reasons, we can assume that the value of the block parameter resistance $R(t)$ is time independent and we denote it R. The component "resistor" can be modeled by implementing one of the equations:

$$u(t) = R \cdot i(t) \tag{5.1}$$

$$i(t) = \frac{u(t)}{R} \tag{5.2}$$

Fig. 5.4 depicts the mapping between two realizations of simulation blocks for a resistor implementing the aforementioned equations. Such an example can seem to be a simple

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX ontology: <http://cyber.felk.cvut.cz/simulation/automation_ontology#>

SELECT ?deviceIndividual ?deviceClass ?simulationClass
WHERE
{
    ?realPlant                 ontology:hasRealDevices        ?deviceIndividual.
    ?deviceIndividual          rdf:type                       ?deviceClass.
    ?simulationClass           ontology:simulates             ?deviceClass.
    ?deviceIndividual          ontology:hasPowerBond          ?previousDeviceIndividual.
    ?previousDeviceIndividual  rdf:type                       ?previousDeviceClass.
    ?previousSimulationClass   ontology:simulates             ?previousDeviceClass.
    OPTIONAL {
      ?simulationClass         ontology:hasInputVariable      ?inputVariable.
      ?previousSimulationClass ontology:hasOutputVariable     ?outputVariable.
      FILTER (?inputVariable != ?outputVariable)
    } FILTER (!bound(?inputVariable))
}
```

Figure 5.3: The SPARQL query to select appropriate generic simulation blocks from a simulation library for the case of SISO simulation blocks and no parallel connections.

mathematical anagram at first. However, in the case of complex and non-linear systems, the situation is much more difficult. In addition, such dualities do not exist in many cases (e.g., due to non-linearities such as dead-zones or due to risk of division by zero leading to unbounded results), or the mathematical description is as complex as it is not efficient to work with it directly. Simulation experts are expected to use the whole simulation components without knowing their specific implementation details in the proposed approach (i.e., simulation models are considered as gray-boxes with known meaning of the input and output variables). Since the resistor case can be considered as a very simple example, the more complex case from hydraulics is discussed in Sec. 7.2.

5.4 Simulation Block Selection for SISO Blocks and Serial Connections

The first version of the algorithm selecting an appropriate simulation block for each component was proposed in [123]. The entries for the selection algorithm are the plant sub-ontology of the automation ontology, which comprises the real plant structure, and the simulation ontology, which stores interfaces of available simulation blocks. The required entry for the selection algorithm is also the mapping between real devices and simulation components respectively simulation blocks. The goal of the ontology tool is to find for each device (i.e., for individuals of the plant ontology) an approximating generic simulation block with respect to compatibility of interfaces of blocks interconnected in series. The version discussed in this section is limited for SISO systems.

The basic rule *For each signal, an output interface of its producer must be equal to the input interface of its consumer* is satisfied by the SPARQL query depicted in Fig. 5.3. The query is motivated by the following steps:

1. Finding all ontology individuals that represent real plant devices. These individuals are recognized by the object ontology property “hasRealDevices”.

2. For each real plant device, finding its simulation equivalents, i.e., generic simulation blocks according to the object property "simulates".

3. For each device, finding a device producing the signal being consumed by the particular device.

4. For each simulation block, finding its input interface and an output interface of its input signal producer.

5. Selecting such simulation blocks, for which the interfaces are equal.

The results obtained by the selection can be classified as follows:

1. *For each plant device, exactly one simulation equivalent exists.*

   This solution is the most desirable; there exists exactly one simulation model of the industrial plant that is feasible and can be generated automatically.

2. *More than one solutions exist.*

   This class of solutions enables to create several different simulation models for the particular industrial plant. Although all of them can be generated automatically, the engineer has to decide which of them is the most suitable for the simulation.

3. *A solution satisfying all conditions does not exist.*

   This result proves that the universal simulation library has to be modified.

This selection algorithm can be used for lines of serial connections of SISO components. For other topologies of SISO components or for MIMO components, the algorithm had to be generalized and improved. The need for supporting arbitrary topologies and MIMO components required in Sec. 5.2 hence resulted in a more general solution discussed in the further explanation.

5.5 Motivation for the Use of the Bond-Graph Theory

When we are trying to assemble a simulation model from simulation blocks approximating plant components, the following obstacles complicate the design process:

1. The problem to select the appropriate simulation block for each component;

2. The problem of serial and parallel connections of components leading to two types of junctions;

3. The problem of causalities and algebraic loops.

Since all of the three problems are addressed by the bond-graph theory, the solution proposed in this thesis is based on bond graphs. However, the bond-graph theory is originally a human-based method intended to describe a physical system with mathematical equations. The proposed approach bridges the gap between the well-proven bond-graph theory and the contemporary computer-aided engineering. The proposed method combines

an object-oriented approach (where simulation components are atomic objects with possibly unknown inner implementation and behavior, but with semantically annotated input and output interfaces, parameters, and initial conditions) with the mathematical-physical foundations of the bond-graph theory.

The bond-graph theory is suitable for supporting the instantiation of components from simulation libraries according to the structure of a real plant. The important benefit of the bond-graph theory is that it is also suitable for machine-based processing. It offers simple lists of recommended steps and rules, which are easy to implement. For assembling simulation models from components, bond graphs were thus selected as a good starting point. However, their use in a standard way comes up with a way of thinking that does not push modularity and reuse satisfactorily. Hence bond graphs are extended in this thesis and the motivation for doing this is discussed in the following section.

5.6 Motivation for a New Causality Assignment Algorithm

When bond graphs are used in a conventional way (i.e., as it has been discussed in Sec. 2.3), the causality is assigned to the bond graph according to rules defined within the bond graph theory. The causality assignment algorithm used frequently is called sequential causality assignment procedure [144]. However, the standard sequential causality assignment procedure (SCAP) cannot be used in conjunction with the proposed explicit representation of simulation blocks (i.e., component implementations), because it does not support an existence of such explicitly specified available blocks. Other causality assignment procedures are discussed in [97], however, none of them fits to the proposed extended bond graph theory seamlessly.

As the standard SCAP algorithm cannot be used for the proposed method, a new causality assignment algorithm had to be developed. The algorithm proposed later in Sec. 5.7.3 is inherited from the depth-first search algorithm, which was selected as a well-proven graph search algorithm. Since the whole graph has to be gone through, the choice of the graph-search algorithm does not pose an important topic.

The depth-first search (DFS) algorithm is a well-known algorithm for searching within tree or graph data structures, widely used in artificial intelligence applications for solving problems by searching. It explores each branch of the graph as deeply as possible, after reaching the last node in a branch, backtracking is applied. One of the most frequent representations of the DFS algorithm is expressed in Alg. 1. Further information about the DFS algorithm can be found in numerous literature, for example in [145].

A bond graph is a special type of a graph, hence, this algorithm is applicable but it has to be extended in such a way that it reflects the bond graph specificities. The following aspects have to be supported with a new causality assignment algorithm:

1. The “goal” state is not a specific node but a state of the bond graph when all power bonds have assigned causality strokes.

2. Each exploration step has to be accompanied with the assignment of causality strokes to all visited bonds.

3. Backtracking is applied not only in the case of reaching the end of the graph branch, but also in the case when the causality cannot be assigned correctly. These two roots

Algorithm 1 Depth-first search (DFS) algorithm

```
procedure DEPTHFIRSTSEARCH(g, v)
    Stack stack = new Stack();
    stack.push(v);
    while stack.nonEmpty() do
        v = stack.pop();
        if v.notDiscovered() then
            v.setDiscovered();
            for each edge_v_w in g.getAdjacentEdges(v) do
                stack.push(edge_v_w.getW());
            end for
        end if
    end while
end procedure
```

of backtracking have to be explicitly distinguished.

4. During backtracking caused by violation of causality assignment rules, already assigned causality strokes have to be removed in all bonds being backtracked. During backtracking caused by reaching the end of the graph branch, already assigned causality strokes have to be kept as they are.

5. Entering a new component (i.e., node of the bond graph which is not a junction) has to be accompanied with selecting an available simulation block (i.e., a component implementation).

Analyzing this list of requirements for the causality assignment algorithm, a new causality assignment algorithm has been designed and implemented. In order to be contextualized with the proposed extended bond graph method, it is discussed later in Sec. 5.7.3.

5.7 Extended Bond Graphs Enhanced with Explicit Simulation Block Support

The proposed method assumes that systems consist of subsystems called "components" or "real devices". Each component (or real device) is modeled by a "simulation block", which was also called "simulation component implementation" in the previous author's work. For some components, the mapping between a component and a simulation block can be 1:1, but it can be also 1:n in general. It means that a component can be modeled by one of the n available simulation blocks. These simulation blocks can differ in (i) number of inputs and outputs, (ii) input and output interfaces, and (iii) required parameters. A very basic example of this situation has been depicted in Fig. 5.2. The component resistor can be modeled with either a simulation block having voltage difference as input or as a block having electrical current as input. The goal of the proposed method is thus not only to instantiate components, but also to select the appropriate simulation block for each node of the real system topology. In the terminology of bond graphs, the proposed approach extends the concept "simulation component" with a set of one or more "simulation blocks", which differ from each other in their interfaces and parameters.

5.7.1 Formal Specification of the Simulation Model Design Task

The proposed interpretation of the simulation model design task can be defined and formulated as follows:

- **Real plant, real devices, and their connections**

  The real plant $S = (D, C)$ consists of the set of real devices $D = \{d_1, ..., d_m\}$ and a set of physical connections $C = \{c_1, ..., c_q\}$. The connections $c_i \in C$ are power bonds in terms of the bond-graph theory and they define adjacency as well.

- **Device types of real devices**

  Each real device $d_i \in D$ is of a device type γ_i. All device types give a device type set $\Gamma = \{\gamma_1, ..., \gamma_r\}$. The set of device types is accompanied with a mapping between devices and device types: $M_{D\Gamma} : D \to \Gamma$.

- **Simulation blocks**

  Each device type $\gamma_i \in \Gamma$ can be modeled by up to n simulation blocks $\beta_{i,j}; i = 1, ..., m; j = 1, ..., n$. The simulation blocks are aggregated in a simulation library $\Lambda = \beta_{i,j}$ where $i = 1, ..., m$ and $j = 1, ..., n$. The mapping between simulation blocks and device types is defined as $M_{\Lambda\Gamma} : \Lambda \to \Gamma$.

- **Simulation model design task**

  The task of the simulation design for signal-oriented simulators is considered in this thesis as (i) selecting appropriate set of simulation block instances $I = \{\iota_1, ..., \iota_m\}$ that model the system S. In addition, (ii) the set of physical connections $C = \{c_1, ..., c_q\}$ (i.e., power bonds) has to be transformed into a set of signal bonds $\Sigma = \{\sigma_1, ..., \sigma_{2q}\}$.

In the further text, we will explain how the standard bond graph theory can be extended to support finding the aforementioned sets of simulation block instances $I = \{\iota_1, ..., \iota_m\}$ and signal bonds $\Sigma = \{\sigma_1, ..., \sigma_{2q}\}$.

5.7.2 Extended Bond Graph Method

To solve the simulation model design task as it was stated in the previous subsection, the bond graph theory provides good foundations. Therefore, the standard bond-graph method was extended by the author of this thesis with the two following aspects:

1. Explicit support for various simulation blocks for each component;
2. Improved causality assignment to support the previous point.

In more details, support for simulation blocks means that for each device in the real plant topology, i.e., for $D = \{d_1, ..., d_m\}$, a list of available simulation blocks $\beta_{i,j}; i = 1, ..., m; j = 1, ..., n$ is assigned. In the same way as it has been demonstrated in Fig. 5.2 for two different implementations of the device type resistor, each simulation block is mapped to its device type. If a simulation expert needs to avoid differential causality or problematic numerical representations, he or she simply does not implement that acausal/problematic implementation of the simulation component. For example, in case of electrical capacitors

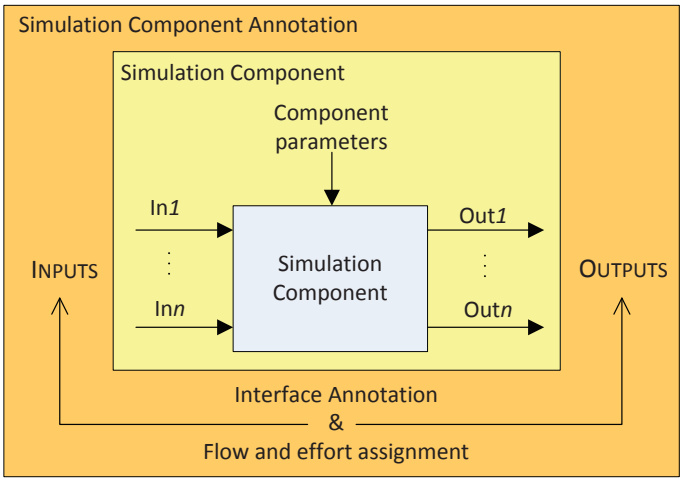

Figure 5.4: A simulation component and its interfaces. The interface annotation includes inputs and outputs; and it maps inputs and outputs to generic signals flow and effort.

and inductors, only the causal versions (i.e., the versions having integrals and no derivatives) are expected.

On the contrary to a top-down approach of a typical bond-graph theory, the idea behind the proposed approach is to focus on the components themselves, and to build simulations as a bottom-up approach with a complex behavior emerging from the components' behaviors. If some components have more than one implementations, the selection of the right implementation for each component is driven by the compatibility of interfaces. To recognize the compatibility, the causality algorithm known from the bond-graph theory is used, but it is extended for the support of various simulation blocks.

Simulation blocks and their interfaces have to be annotated in such a way that is computer-understandable. Fig. 5.4 depicts the basic features of a block assumed in this thesis. Each simulation component has input and output variables. These variables must be inter-mapped with bond graph variables flow and effort. In other words, there must be stated, which signals are related to the generalized bond graph variable flow and which ones to the effort.

For practical use, it would be necessary that the simulation engineer would be able to select the granularity of the simulation model, i.e., what will be atomic components. For example, atomic components in hydraulic systems can be either devices, or the whole pump stations. In case of electrical circuits, the situation is similar, but in order to avoid implementation details, the granularity level on the device level is assumed in further text. This assumption implies that each device in the source engineering plan (in the exemplary case of the electrical circuit) will be simulated by exactly one simulation block in the simulation model.

The second extension of standard bond graphs to explicitly support various simulation blocks for each component is an appropriate causality assignment enabling this task. When the mapping between real devices and simulation blocks is $1:1$, the situation is easy. Such a situation exists in the passive house use-case in Sec. 7.1. The algorithm selects the block according to the object ontology property "simulates". But if the mapping is $1:n$ (i.e., one real device can be simulated by more than one simulation blocks) as it was proposed in the previous text, the selection of the appropriate simulation block is not trivial. This case

is discussed in the water distribution network use-case in Sec. 7.2. The proposed solution adopts the requirement on compatibility of the input and output signals of the neighbor blocks in the simulation model topology.

5.7.3 Proposed Method in an Algorithmic Way

The proposed method can be summarized by the pseudo-code expressed in Alg. 2. First of all, simulation components are generated into a new extended bond graph. This issue is done in the same way as in the typical use of bond graphs. However, the explicitly specified available simulation blocks are added in the consequent step, which is specific for the extended bond graphs. In other words, each component is enhanced with 1 to n simulation blocks, being specific implementations of this component. From this set of blocks, the right one has to be selected for each individual component when creating the simulation model.

In the next step, 1-junctions are generated. For doing this, it has to be decided whether the system (or the relevant part of the mechatronic system) is mechanical or not, which is important for considering parallel and serial connections. This issue affects generation of 0-junctions as well. Since the structure of bond graphs (including their extended version as well) depends on a number of ports of the components respectively the blocks implementing them, the number of 1-junctions has to be selected. Considering m as a number of device connections, the component is regarded as an m-port if at least one simulation block implementing this component is an m-port in the bond graph sense and 1-junctions are thus generated m-times. Otherwise, the component is regarded as an 1-port component and thus exactly one 1-junction is created, respectively if neither the m-port block nor 1-port block is available, then the method is stopped with an error.

Consequently, power bonds are generated by adding edges into the graph representation. This step is followed by rather a technical issue of setting counts of connections for all components, 1-junctions, and 0-junctions, which is important for recognizing whether causality rules are satisfied when the causality is being assigned to the entire graph. The step is followed by the method denoted in Alg. 2 as `filterNPortBlocks()`, which goes through all components and simulation blocks implementing these components and it filters those simulation blocks that have different number of ports than is the type of the m-port component in the created bond graph. To simplify the searching through the graph, graph components in terms of the graph theory are found with the method `generateGraphComponents()`, which results into a set of entry elements to each graph components. Finally, the power directions are selected, a reference junction is excluded from the bond graph and the causality is assigned to the entire graph.

Since the assignment of the causality is a complicated issue that is one of the important contributions of this thesis, the proposed causality assignment algorithm is expressed separately as Alg. 3. The idea behind the proposed extended causality assignment algorithm is that it selects the first implementation of each component and tries to assign causality strokes to the power bonds. In case of junctions, it is necessary to iterate and to try several possibilities in order to satisfy all causality requirements. When selecting an assignment, each choice is pushed into an enhanced stack that is able to capture information about those causality strokes that have been selected randomly and could be changed. The goal state of the causality assignment is such an assignment ensuring that all causality requirements

Algorithm 2 Generation of the extended bond graph supporting simulation blocks

```
procedure GENERATEBONDGRAPH
    BondGraph bondGraph = new BondGraph();
    List <Component> components = bondGraph.generateComponents();
    bondGraph.addBlocks(components);
    List<Junction1> junctions1 = bondGraph.generateJunctions1(components);
    bondGraph.generateBonds(junctions1, components);
    List<Connection> connections = bondGraph.queryConnections(components);
    List<Junction0> junctions0 = bondGraph.generateJunctions0(connections);
    bondGraph.generateBonds(junctions0, junctions1);
    bondGraph.setConnectionCount(components, junctions1, junctions0);
    bondGraph.filterNPortBlocks(components);
    List<Node> graphComponents = bondGraph.generateGraphComponents();
    bondGraph.assignPowerDirection(graphComponents);
    bondGraph.excludeReferenceJunction(referenceJunction);
    bondGraph.assignCausality(graphComponents);
end procedure
```

are satisfied for all components, junctions, and power bonds. When the goal is reached, the situation is solved and it is found out that this particular set of simulation blocks implements the given system. When causality requirements are not satisfied at a particular power bond, backtracking is done. Exactly one simulation block is switched to the other available implementation of this simulation component and the causality of the respective power bond is switched. The process of causality assignment then continues. At the end, the right combination of simulation blocks is found, if it exists. Otherwise it is proven that the given plant topology cannot be modeled with the given set of simulation blocks and the algorithm throws an error. As both causality assignment corrections and selection of the appropriate simulation blocks are based on stacks and systematic graph search preferring the depth branch, the proposed method can be considered as a two-level depth first search algorithm.

One of the core methods of the proposed causality assignment algorithm (see Alg. 3) is the method `pushConnectionsToExplore(node node)`. This method corresponds to the method `push(vertice v)` of the standard DFS algorithm with the difference that the parameter of the method is not a vertice, but a node of the bond graph (i.e., a component, 0-junction, or 1-junction). Moreover, the stacked elements are not only the outcoming power bonds of a specific node (i.e., vertices) but both nodes as well. All power bonds added to the stack are marked as not visited. The dual method is the method `popConnectionsToExplore()`. It corresponds to the `pop()` method in the standard DFS algorithm. But a fundamental difference is that power bonds are not removed from the stack, they are only marked as visited. Therefore the flag visited is manipulated by the aforementioned extended versions of the push and pop methods systematically. The removal from the stack is done only when causality cannot be assigned and backtracking is performed. During this action, all removed power bonds and nodes are reset to the default values that mainly means that the position of the causality strokes is unset. This set of actions can be performed during calling of the method `popConnectionSubstituable()`. This method pops (and really removes) bond graph objects from the top of the stack until an object with a flag `isSubstituable` is found. In this case it switches the causality (i.e., changes

Algorithm 3 Causality assignment algorithm for extended bond graphs

```
procedure assignCausality(ListNode graphComponents)
    for each graphComponent in graphComponents do
        stack.push(graphComponent);
        stack.pushConnectionsToExplore(graphComponent);
        Connection connection = stack.popConnectionToExplore();
        while connection != null do                          ▷ Causality step
            int causality = 0;
            while causality == 0 do
                causality = resolveConnectionCausality(connection);
                if causality == 0 then
                    connection = stack.popConnectionSubstituable();
                end if
            end while
        end while
        connection.setCausality(causality);                  ▷ Exploration step
        stack.pushConnectionsToExplore(connection.getConnectionTo());
        connection = stack.popConnectionToExplore();
    end for
end procedure
```

the site of the causality stroke) and the exploration of the bond graph continues. The flag `isSubstituable` is set during calling of the method `resolveConnectionCausality()` in those cases the causality is assigned arbitrarily or when a next simulation block is available. Each component has a list of simulation blocks that can model its behavior hence it is easy to recognize whether a further simulation block is available. If a specific power bond leads to a component that has not been explored yet (i.e., it is on top of the stack), the first simulation block is selected first. The method tries to assign the causality to the current power bond. If it is not possible, it cyclically tries to select a next simulation block if applicable. If the causality cannot be assigned (because it violates causality assignment rules specified by the bond graph theory), it returns the value "0". If it can be assigned, the possible causality stroke position is returned. If the position is arbitrary, one of the possibilities is selected and the flag `isSubstituable` is set up, which means that this particular power bond is a candidate to be switched during backtracking, that can occur later. Finally, the method `setCausality(int causality)` simply assigns the causality stroke that was worked out by the method `resolveConnectionCausality()`.

5.7.4 Output of the Extended Bond Graph Method

The results of the proposed method can be classified as follows:

1. *One or more simulation models exist.*

   This group of results means that either one simulation model approximates the given system or more than one simulation models can model the given system. The current implementation of the method returns one created simulation and it does not automatically recognize whether more of them can be created.

2. *A solution satisfying all conditions does not exist.*

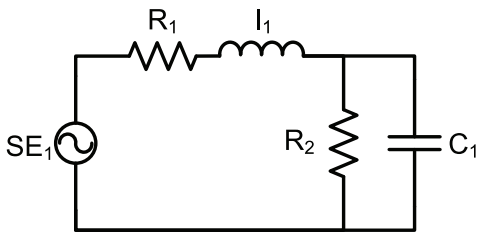

Figure 5.5: Exemplary electrical circuit including a voltage source, resistors R_1 and R_2, and both accumulators of energy – a capacitor C and an inductor I.

> If a solution does not exist, the simulation library has to be enhanced with further simulation blocks or the contemporary ones have to be modified in terms of block interfaces to be able to model the specific industrial system/plant.

In an exemplary run that is discussed in the next section and that is depicted in Fig. 5.6, the first case of this result classification was reached. With the given simulation library blocks, exactly one simulation model can be created. Later on, it is shown how to generate this simulation model in the signal-oriented simulator MATLAB-Simulink.

5.8 Electrical Circuit Example

To illustrate the work of the proposed causality assignment algorithm in practice, we selected the exemplary electrical circuit that is depicted in Fig. 5.5. It is the same as the circuit utilized for the demonstration of the standard bond graph usage presented in Fig. A.1, where its simulation models are created manually. The aforementioned run of the proposed causality assignment algorithm for the exemplary electrical circuit is depicted in Fig. 5.6, where we can see how the proposed causality assignment algorithm explores the bond graph and assigns the causality strokes. The impact of backtracking when the assignment under construction is violating restrictions related to a junction can be also seen twice in this figure.

In more details, the example in Fig. 5.6 shows several key points in the causality assignment algorithm run. The causality assignment is applied to the plain bond graph depicted under the gray numbered circle 1. The red numbered circles (i.e., bond graphs marked as 7 and 17) mean the violation of causality assignment rules that cannot be solved by switching a simulation block, because none other block is available. Backtracking is needed as a continuation part of this step. The blue numbered circles (i.e., 8 and 18) correspond to the state of the causality assignment algorithm when those bonds whose causality can be switched are reached. Thus the backtracking is stopped in these blue points. The green numbered circles (i.e., 9 and 19) are related to the already switched bonds, after whose the exploration and causality assignment can continue. The purple numbered circle (i.e., 22) denotes the final stage, when causality is assigned to all bonds and all simulation blocks are selected. Based on this result, the simulation model can be generated.

Although this small-size system cannot prove the usefulness of the method for large-scale systems from industrial perspective, it was selected due to the simplicity to follow the process steps and the workflow of the proposed extended bond graph method in details. The more complex example is discussed later in Sec. 7.2. The Alg. 2 is followed in the further

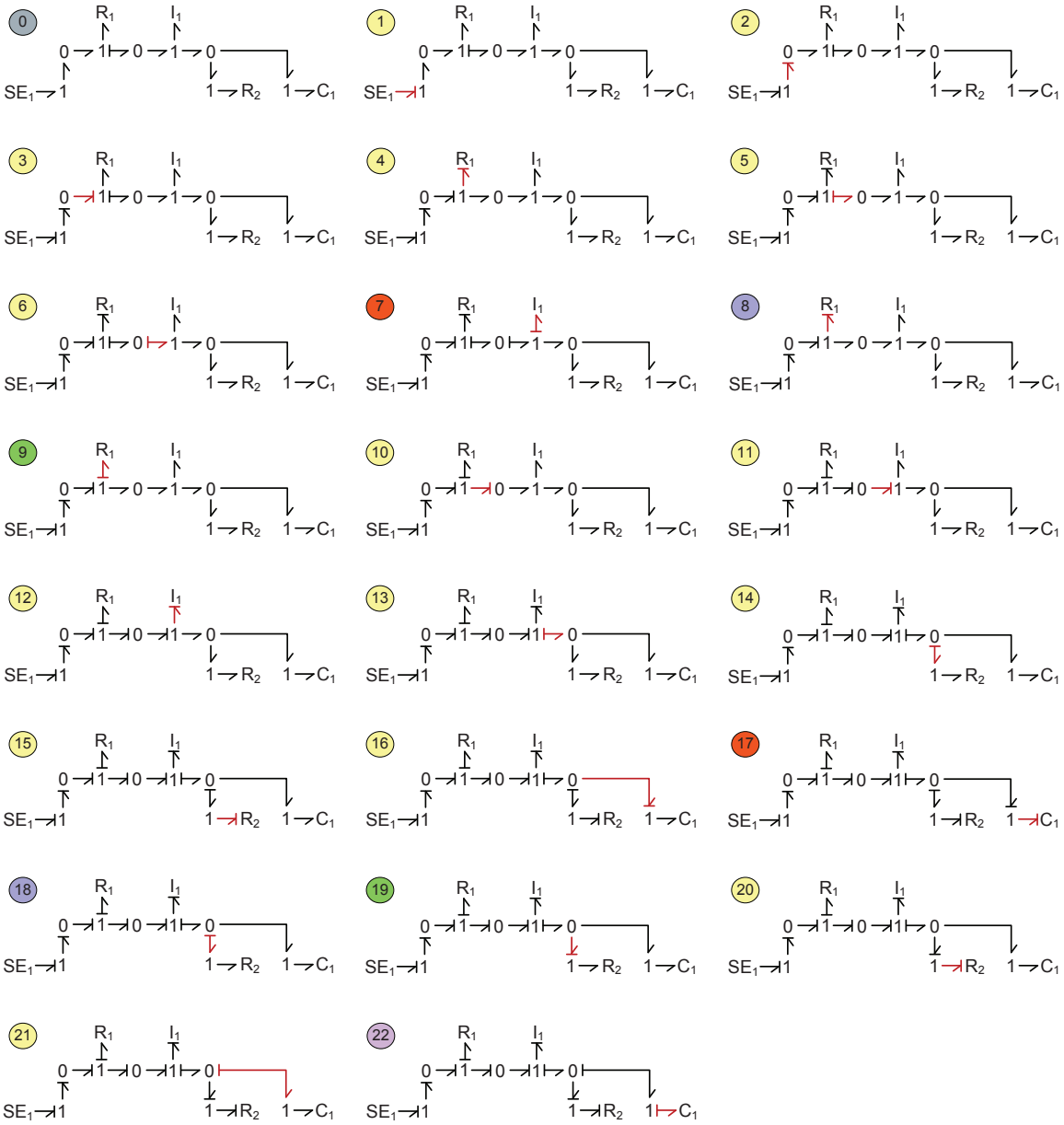

Figure 5.6: An example of the proposed causality assignment algorithm run in case of the exemplary electrical circuit. The gray numbered circle denotes the initial bond graph without causality assigned. Yellow circles symbolize normal steps, when causality is assigned to the next power bond properly. Red circles mean that the causality assigned violates causality assignment rules and backtracking has to be applied as a part of this step. Blue circles correspond to stopping of the backtracking at those bonds, whose causality can be switched. Green circles are related to switched bonds, after whose the exploration and causality assignment can continue. The purple circle denotes reaching the goal, when causality is assigned to all bonds and all simulation blocks are properly selected.

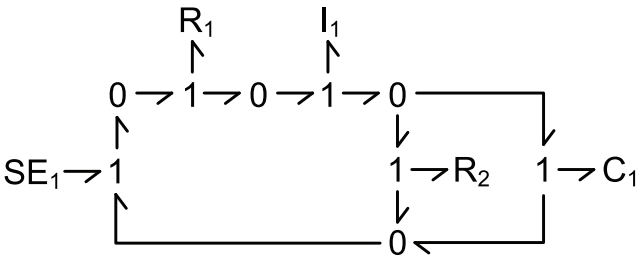

Figure 5.7: Structure of the extended bond graph for the electrical circuit.

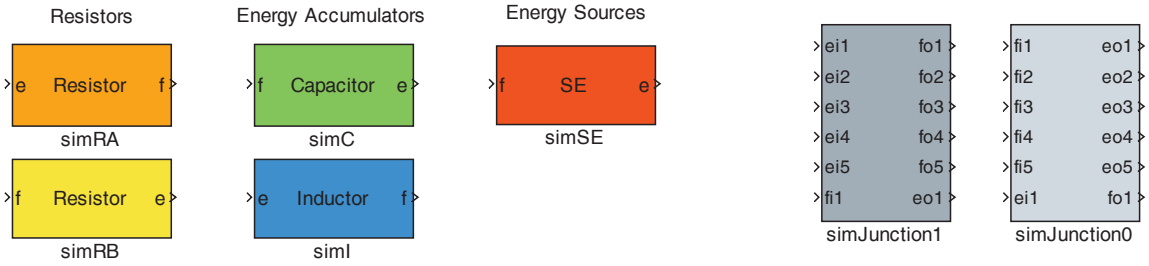

Figure 5.8: Simulation library with electric simulation blocks implemented in MATLAB-Simulink.

text and the simulation model for the electrical system already introduced in Fig. 5.5 is generated based on extended bond graphs.

The first step is the generation of components, 0-junctions, 1-junctions, and power bonds. This issue is similar to the standard version of bond graphs. The generated graph is depicted in Fig. 5.7 and it is graphically the same as in the manually created version presented in Fig. A.3.

The next step differs methodologically, however, it leads to the same results as in the case of the standard bond graph theory. Assigning causality strokes is not driven by causality assignment rules, but by combining available simulation blocks, which are combined based on the depth-first search. Therefore, it is necessary to load available implementations for each component at first. In this step, just the annotations of interfaces (i.e., descriptions of inputs and outputs) are required. The internal implementation of simulation block behavior is necessary later for the simulation model execution.

The available simulation components respect the mapping presented in Fig. 5.2. These simulation blocks are aggregated in the available simulation library in MATLAB-Simulink, depicted in Fig. 5.8. This simulation library was included into the mechatronic library, which is discussed in details in Appendix B. The following simulation blocks from the electrical discipline are included in the library:

1. Resistor $simR_A = (\text{Effort}_{\text{in}}; \text{Flow}_{\text{out}})$ having the detailed description in Fig. B.2;
2. Resistor $simR_B = (\text{Flow}_{\text{in}}; \text{Effort}_{\text{out}})$ having the detailed description in Fig. B.3;
3. Capacitor $simC = (\text{Flow}_{\text{in}}; \text{Effort}_{\text{out}})$ having the detailed description in Fig. B.4;
4. Inductor $simI = (\text{Effort}_{\text{in}}; \text{Flow}_{\text{out}})$ having the detailed description in Fig. B.5;
5. Source of effort $simSE = (\text{Flow}_{\text{in}}; \text{Effort}_{\text{out}})$ having the description in Fig. B.6.

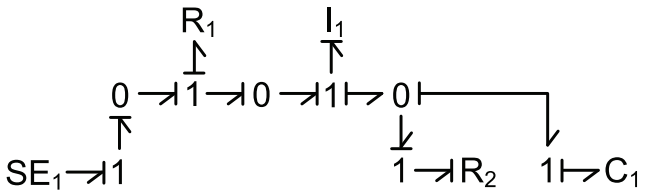

Figure 5.9: Bond graph of the electrical circuit, including components, junctions, and direction of power.

In addition to the above mentioned discipline-specific simulation blocks, the simulation library has to be able to approximate the behavior of bond graph junctions of types 0 and 1. Therefore, the library was also equipped with cross-discipline junction implementations, whose detailed descriptions are depicted in Fig. B.7 and B.8.

The assignment of causality strokes is based on the interface description of available components in the presented method. The extended bond graph for the electrical system is depicted in Fig. 5.9, which includes components, junctions, bonds, power directions, and assigned causality strokes. Although each component has mapped simulation blocks behind it, this issue is not reflected in the diagram graphically. In this case, both resistors R_1 and R_2 map their possible implementations R_A and R_B. The result of the extended causality assignment algorithm is depicted in Fig. 5.9. Since this assignment has been found, the simulation model can be generated with the simulation blocks from the given library. The resistor R_1 is modeled with the simulation block $simR_B$ and the second resistor R_2 is modeled with the simulation block $simR_A$.

The last step in the presented method is the generation of a simulation model for a signal-oriented simulator. Since we have selected the appropriate simulation blocks in the previous process steps, the main task is thus to instantiate all required simulation blocks, 0-junctions, and 1-junctions. This issue can be done with the implemented MATLAB-Simulink connector, which is presented later in Sec. 6.4.1. The generated model in MATLAB-Simulink is depicted in Fig. 5.10. It is a complete simulation schema, but to execute this schema, the required parameters have to be defined prior the simulation is started in the simulation tool suite.

5.9 Verification of the Generated Simulation Model for the Electrical Circuit

To verify the correctness of the generated simulation model, its simulated results were compared with outputs of circuit models from other simulation tools. Prior doing that, specific values of simulation parameters had to be selected. The utilized parameter values are summarized in Tab. 5.11. To distinguish between names of components and parameters, the parameter naming notation respects the object-oriented nature of simulation blocks.

Using the bond graph depicted in Fig. 5.9, the structure of the simulation model for the electrical circuit was generated in MATLAB-Simulink. After configuring simulation parameters with the values presented in Tab. 5.11, the simulation was started for the simulation time 0.1 seconds. The obtained simulation results are visualized in Fig. 5.12. The figure depicts the simulation results of the simulation model generated by the method proposed in this thesis. We can see the transient response on non-zero initial conditions followed by

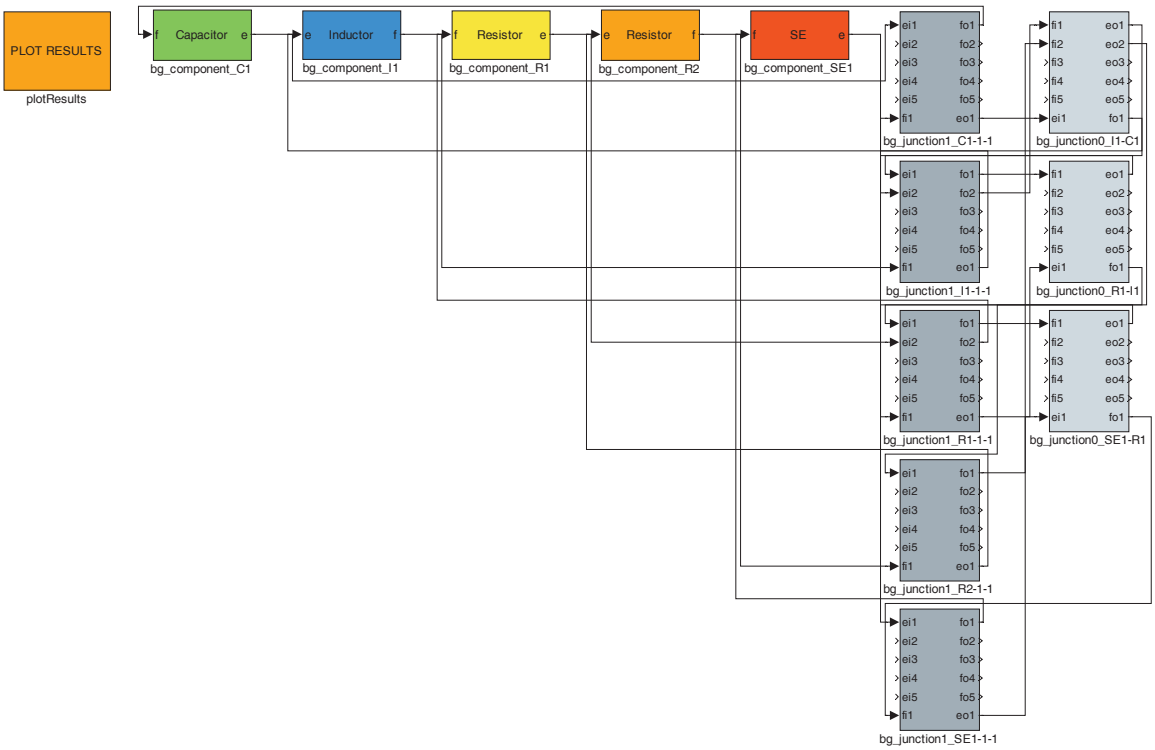

Figure 5.10: Generated simulation model for the electrical circuit in MATLAB-Simulink. This schema represents the structure of the model; for its execution a configuration of simulation parameter values is required.

Component	*Parameter name*	*Parameter value*
Resistor R_1	Resistance	$R(R_1) = 100\ \Omega$
Resistor R_2	Resistance	$R(R_2) = 200\ \Omega$
Capacitor C_1	Capacitance	$C(C_1) = 0.1$ mF
	Initial voltage	$v_0(C_1) = 10$ V
Inductor I_1	Inductance	$L(I_1) = 0.5$ H
	Initial current	$i_0(I_1) = 0$ A
Voltage Source SE_1	Voltage magnitude	$V(SE_1) = 10$ V
	Frequency	$f(SE_1) = 50$ Hz

Figure 5.11: Selected values of parameters for the exemplary electrical circuit.

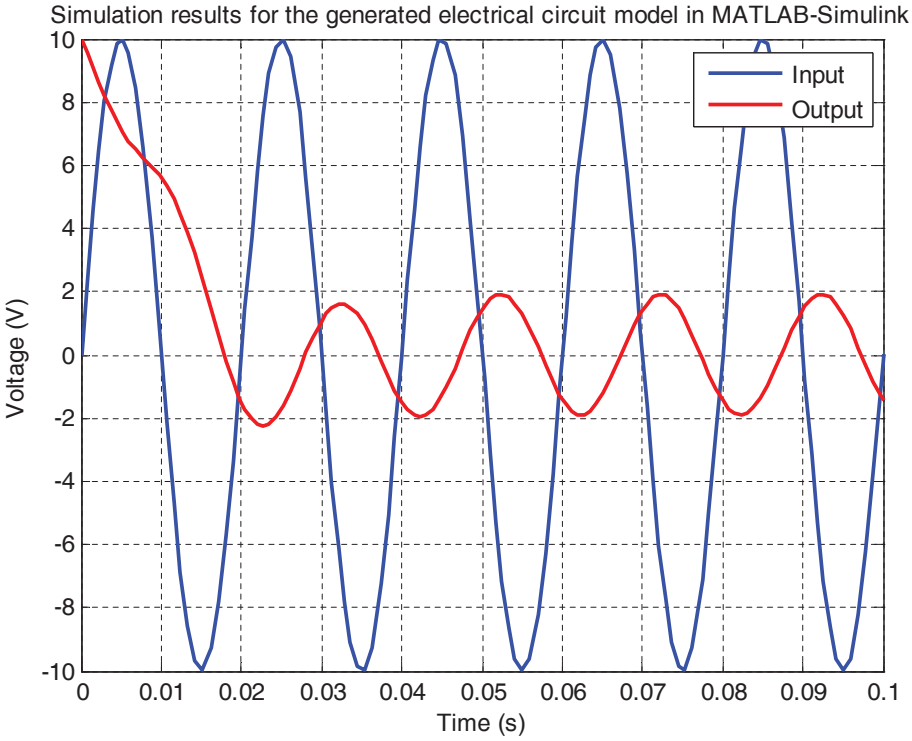

Figure 5.12: Simulated response of the electrical circuit obtained using the proposed method.

harmonic response on input excitation.

Two other tools working on different foundations and technologies were utilized for verification of the correctness of simulation results. The first tool is called QUCS[1] and its name is the abbreviation standing for "Quite Universal Circuit Simulator". The main benefits of this tool are that it is a free software, it offers switching between time and frequency analysis easily, and last but not least the overall software is user-friendly. On the other hand, there is no warranty as the software is free. The electrical plan of the circuit in QUCS is depicted in Fig. 5.13. This circuit was simulated in the time domain and the results of the simulation are shown in Fig. 5.14. The simulation in the QUCS utilizes a different simulation solver than MATLAB-Simulink and the used blocks are not implemented by the author of this thesis, but the built-in components are used. Therefore, this simulation of the same circuit is independent on the model generated by the proposed method and it thus proves the correctness of the simulation model designed by the proposed approach.

Due to the free nature of the QUCS environment corresponding to no warranties, the commercial tool National Instruments Multisim[2] was used in the second step to verify the simulated results. The version 14.0.1 of this tool was utilized in its Education Edition. The electrical plan in this tool is depicted in Fig. 5.15, whereas the simulated results are shown in Fig. 5.16. Since the output courses are the same as the results obtained by the method proposed in this thesis as well as the results simulated by QUCS, the proposed method has been verified with two independent software environments specialized for electrical circuit simulation and the generated simulation model was found correct.

Last but not least, the simulation model generated with the proposed method was

[1] http://qucs.sourceforge.net
[2] http://www.ni.com/multisim/

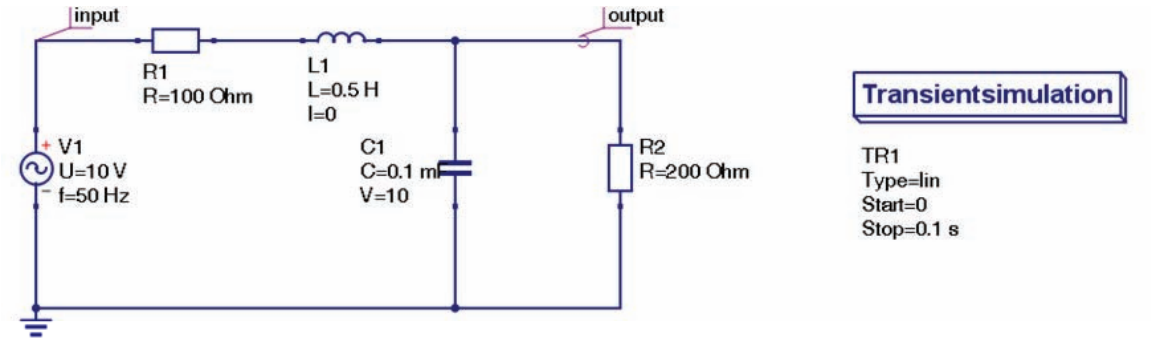

Figure 5.13: Electrical plan of the electrical circuit in the tool QUCS.

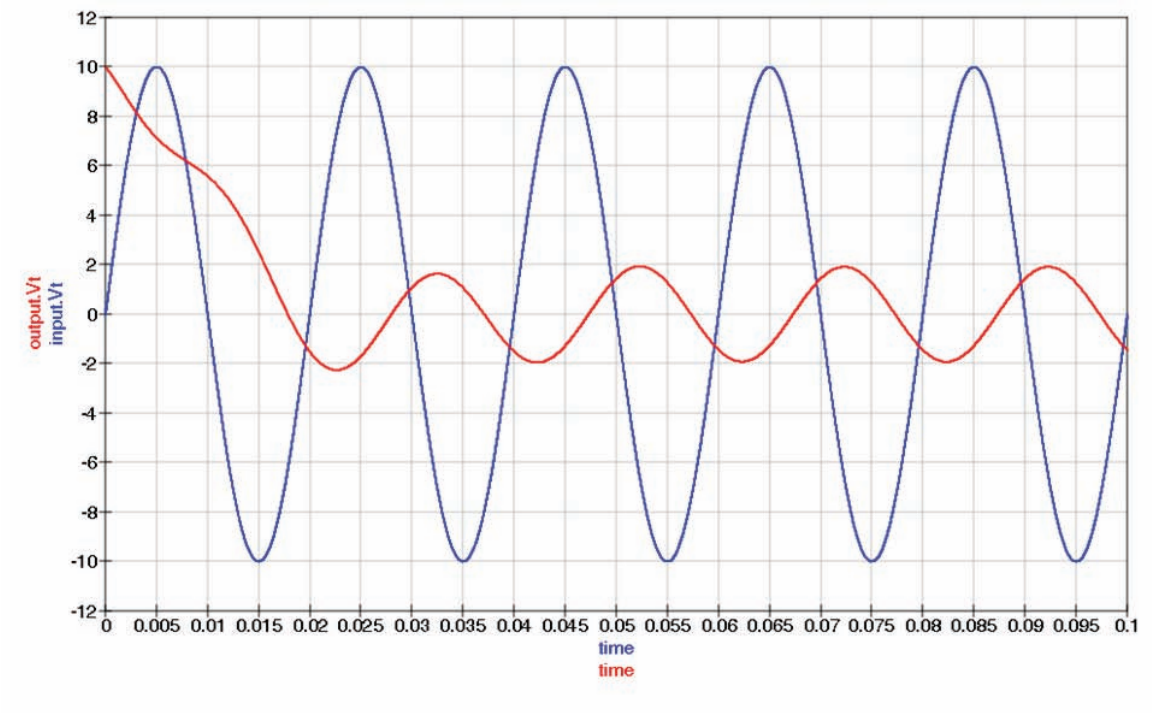

Figure 5.14: Simulated results of the electrical circuit in the tool QUCS.

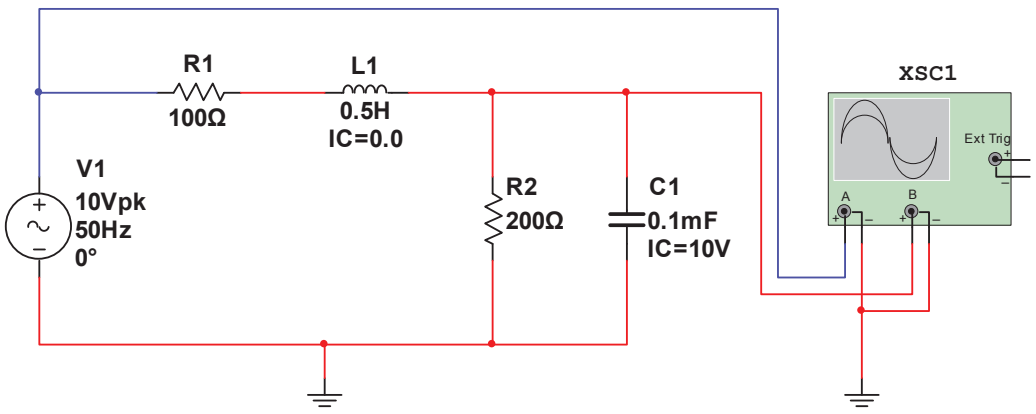

Figure 5.15: Electrical plan of the electrical circuit in the tool NI Multisim 14.0 Education Edition.

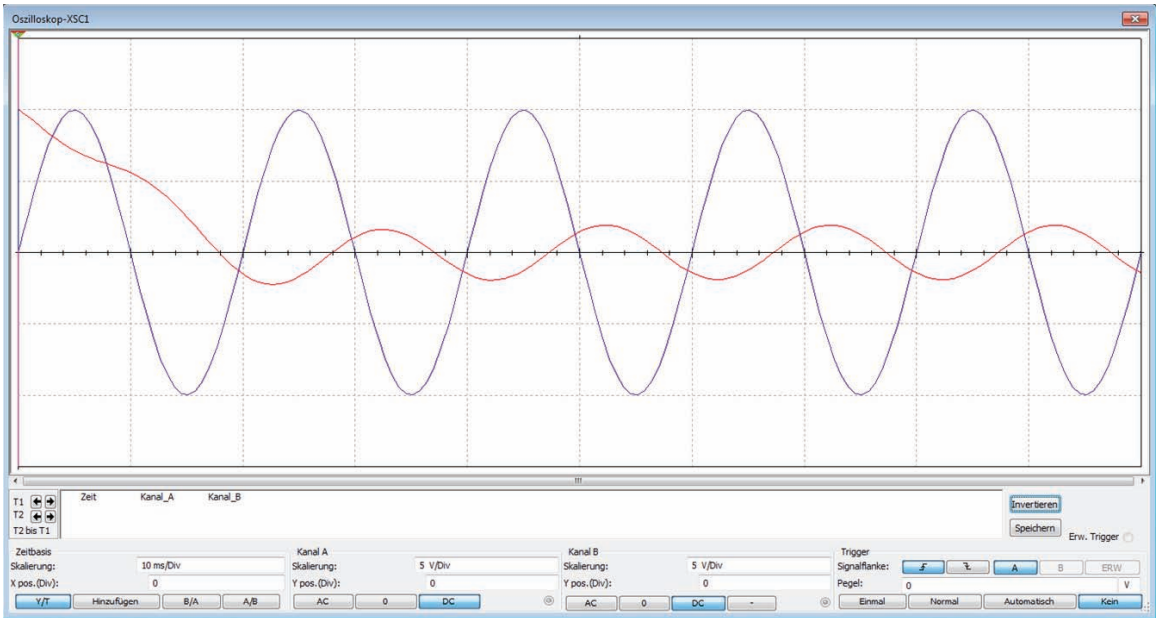

Figure 5.16: Simulated results of the electrical circuit in the tool NI Multisim 14.0 Education Edition.

compared to the three simulation models created manually in Appendix A. For doing this, the matrices A.11 were enumerated with the aforementioned values of parameters. The state-space representation matrices for the selected parameter value set have the following form:

$$A' = \begin{pmatrix} -50 & 10000 \\ -2 & -200 \end{pmatrix} \; B' = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \; C' = \begin{pmatrix} 1 & 0 \end{pmatrix} \; D' = 0 \tag{5.3}$$

The computation of the transfer function based on these matrices of the state-space representation is straightforward. We get the following expression of the transfer function:

$$\begin{aligned} G(s) &= C'(sI - A')^{-1}B' + D' = \\ &= \frac{20000}{s^2 + 250s + 30000} \end{aligned} \tag{5.4}$$

Fig. 5.17 depicts the three time-series of simulated results obtained with the manually created simulation models discussed in Appendix A. The simulation starts with non-zero initial conditions, which disqualifies the solution utilizing the transfer function. The transfer function cannot consider non-zero conditions and thus the simulation starting with zero initial conditions reaches unsatisfactory results. The other two realizations of simulation models according to Fig. A.7 and Fig. A.9 result into the same simulated response as in the case of the simulation model created with the proposed method. These simulation results also correspond to responses calculated by the tools QUCS and NI Multisim. Since various parameter settings and simulation structures have been tested and reached the same results, the proposed method had been validated.

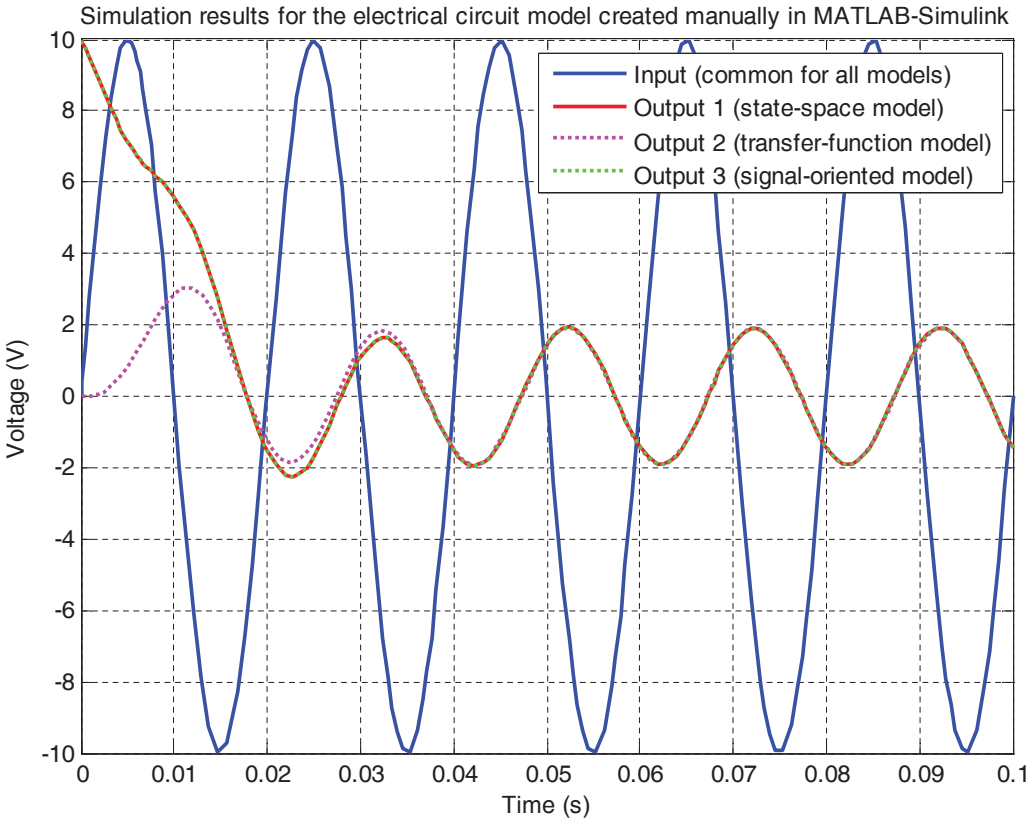

Figure 5.17: Simulated response of the electrical circuit obtained by the manually created simulation model presented in Appendix A. The realization utilizing the transfer-function block is not suitable as it does not support setting non-zero initial conditions.

5.10 Evaluation of the Proposed Method: Benefits and Weak Points

Although the proposed approach offers many benefits for designing both linear and non-linear simulations for large-scale industrial systems, it has several disadvantages that should be taken into account as well.

The first weak point of the proposed method is related to the fact that the simulation design is frequently referred as a kind of "art", relying on deep insight and experiences of skilled simulation experts. Unfortunately, the proposed computer-centric method cannot tackle minor improvements, which get better the simulation execution or precision in such a way the simulation expert does. On the other hand, such modifications are not fully compliant with the physical world frequently. For example, a stability of a hydraulic system simulation can be improved by adding a tank with a very low area of the tank bottom, thus it does not influence simulations results. Nevertheless, such an artificial element stabilizing the simulation does not have any physical equivalent, which can cause troubles when the overall simulation model is redesigned or reused.

In addition, the absence of mathematical equations in the design-time process makes the design phase easier for engineers and technicians. On the other hand, such equations cannot consequently be reused for inspecting features of the systems, such as controllability or observability [8]. In the approach presented in Appendix A, these system features can be inspected in the following way.

The linear time-independent system is controllable if and only if:

$$rank\begin{pmatrix} B & AB & A^2B & \dots & A^{(n-1)}B \end{pmatrix} = n \tag{5.5}$$

where the matrix $\begin{pmatrix} B & AB & A^2B & \dots & A^{(n-1)}B \end{pmatrix}$ is called controllability matrix and it is typically denoted as $\mathcal{C}$. In the case of the electrical circuit, the controllability matrix has the following form for the selected set of parameter values:

$$\mathcal{C} = \begin{pmatrix} 0 & 20000 \\ 2 & -400 \end{pmatrix} \tag{5.6}$$

More generally, the controllability matrix for the exemplary electrical circuit has the following form:

$$\mathcal{C} = \begin{pmatrix} 0 & \frac{1}{C_1 L_1} \\ \frac{1}{L_1} & -\frac{R_1}{L_1^2} \end{pmatrix} \tag{5.7}$$

Based on these forms of the controllability matrix, we can see that the electrical circuit is controllable for all possible settings of parameter values. It means that for all settings of parameter values (i.e., resistances, capacitance, and inductance), any required value of the output voltage can be reached by the appropriate input voltage.

The linear time-independent system is observable if and only if:

$$rank \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{(n-1)} \end{pmatrix} = n \tag{5.8}$$

where the matrix $\begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{(n-1)} \end{pmatrix}$ is called observability matrix and it is denoted $\mathcal{O}$.

For the particular parameter value set, the observability matrix has the form:

$$\mathcal{O} = \begin{pmatrix} 1 & 0 \\ -50 & 10000 \end{pmatrix} \tag{5.9}$$

More generally, the observability matrix has the form:

$$\mathcal{O} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{C_1 R_2} & \frac{1}{C_1} \end{pmatrix} \tag{5.10}$$

Based on these forms of the observability matrix, we can see that the electrical circuit is observable for all possible settings of parameter values. It means that for all settings of parameter values (i.e., resistances, capacitance, and inductance), one can estimate values of system states, which are in this case voltage of the capacitor C_1 and electrical current through the inductor I_1.

In addition, the state-space representation relying on matrices A, B, C, and D can be used for supporting the design of a model-based predictive control[3] (MPC).

Unfortunately, these system indices are analyzable in such an easy form only in case of linear systems. When we need to work with non-linear system models, we loose this compact apparatus to inspect important features of the system. For this reason, the absence of this system analysis on the proposed methodology does not pose any significant restriction and can be neglected.

On the other hand, we obtain many benefits when the proposed method is used. The most important benefit is the suitability also for very complex and large-scale systems, when the manual effort cannot be scaled-up enough. The next benefit is that even non-experts can design (i.e., generate) a simulation model and work with it.

The whole design process is significantly accelerated, thus it is faster from the required development time point of view and it mitigates design-time errors. These improvements can be evaluated as the reduction up to 40% of time needed for the design phase of simulation models, and the reduction up to 50% of design-time errors. This issue also improves the overall safety and reliability.

[3]Frequently called shortly "model predictive control"

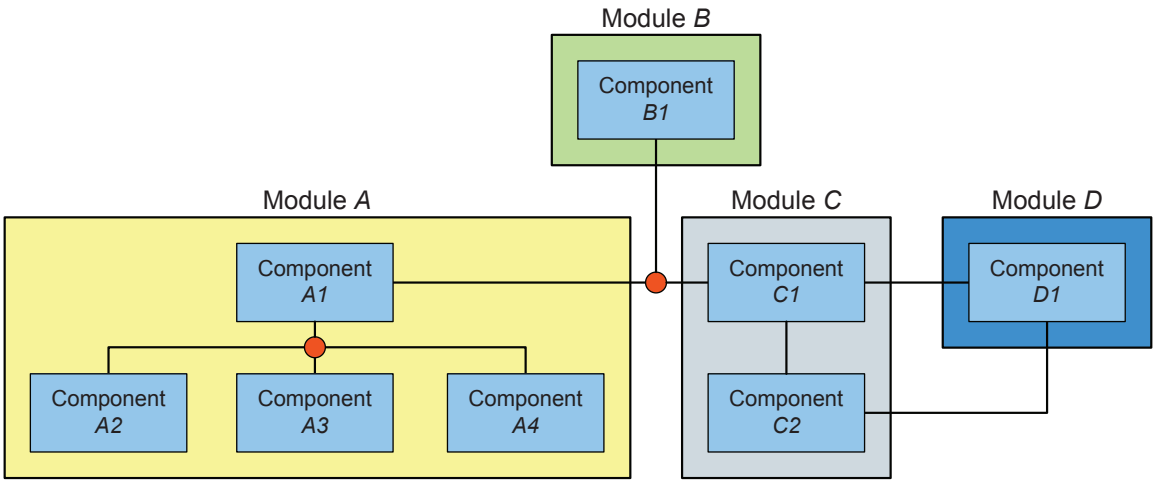

Figure 5.18: Large-scale system and its separation into a set of simulation modules.

Last but not least, the proposed method leads to more effective and efficient engineering workflows that enable re-design and reuse easily, which are important aspects of the emerging Industry 4.0 applications. These benefits are caused by the fact that either the entire simulation model or its part can be re-generated easily based on a new real plant topology or parameter setting. Moreover, the compatibility of simulation interfaces is improved and its feasibility can be inspected automatically.

Evaluating the pros and cons of the proposed method, the two areas for further investigation were identified: (i) providing methods for system analysis in the similar directions as inspection of controllability and observability in case of linear systems, and (ii) to take into account a possible presence of algebraic loops in the simulation model structure in terms of building an algebraic-loop avoidance algorithm in the proposed method or in terms of mitigating the impact of algebraic loops on generated simulations.

5.11 Semi-Automated Generation of Simulation Module Interfaces Using Extended Bond Graphs

The extended bond graphs can be used for supporting the division of a monolithic simulation model into a set of independent simulation modules.

In order to increase the modularity, computational performance, and maintainability of simulations, the simulation models are often required to be split into several simulation modules. These modules are relatively independent, but can be dynamically coupled. To illustrate this situation, an exemplary system of systems [72] is symbolically depicted in Fig. 5.18. The figure shows the components and requirements on the decomposition of the plant into four simulation modules (depicted by rectangles representing Modules *A*–*D* in Fig. 5.18). The problems to be solved are the definition of interfaces of these modules and the connection of these modules in order to get a simulation module topology.

Each power bond (i.e., a connection in a physical system) is represented by a pair of signals flow and effort in signal-oriented simulators. Integration of simulation modules means to transfer values of these two variables between the modules. Hence each power bond corresponds to two signals to deliver flow and effort between these modules.

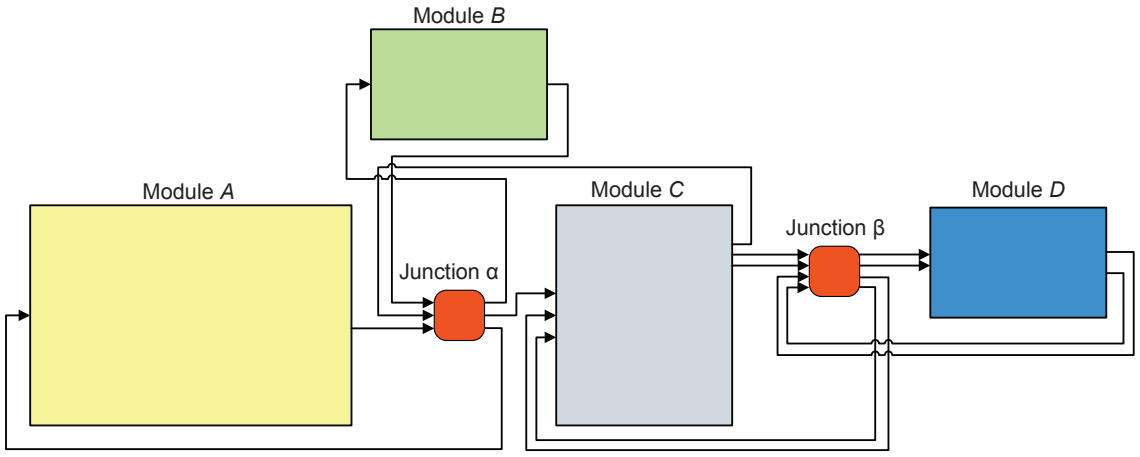

Figure 5.19: A set of simulation modules interconnected with signals.

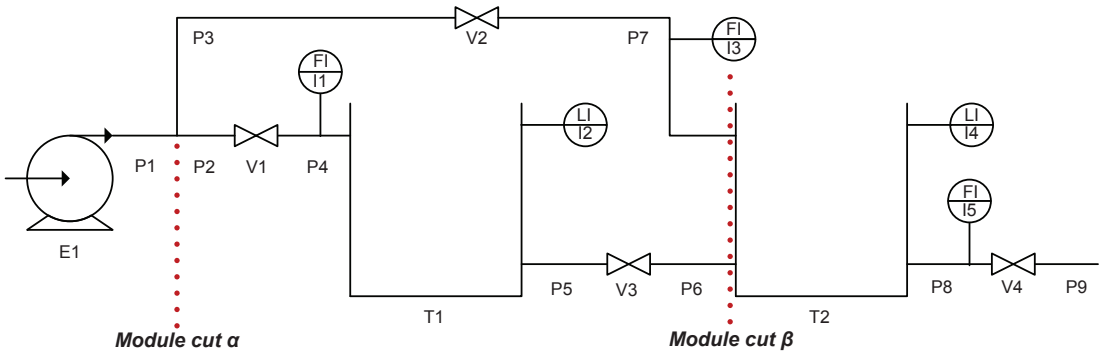

Figure 5.20: P&ID for the selected two tank system.

The splitting of the system into required simulation modules defined in Fig. 5.18 is depicted Fig. 5.19. The key elements are the junctions α and β. The figure depicts their topological position, but it does not include the internal representation of the junctions, i.e., how input and output signals are interconnected and mathematically subtracted or added. This is specified with the use of the extended bond graph theory as it discussed later.

The splitting simulations into several modules is driven by a human simulation expert, while the proposed method supports the expert with structural and technical tasks. The method supports two basic types of cuts of a plant or its parts into modules:

1. Cuts on the junction level
2. Cuts on the bond level

To illustrate the proposed method in practice, a hydraulic tank system depicted in Fig. 5.20 is used. A practical example of these two kinds of simulation cuts into separated modules is depicted in Fig. 5.21 for the case of the tank system bond graph. When doing such cutting, the system has to be simulated with three simulation modules, which are connected with two "glue" modules implementing junctions to connect these modules. The junctions are thus considered as an integration glue between simulation modules. The extended bond graph method facilitates the implementation of the internal structure of these "glue modules" as well as it improves the definition of the interfaces of the simulation modules.

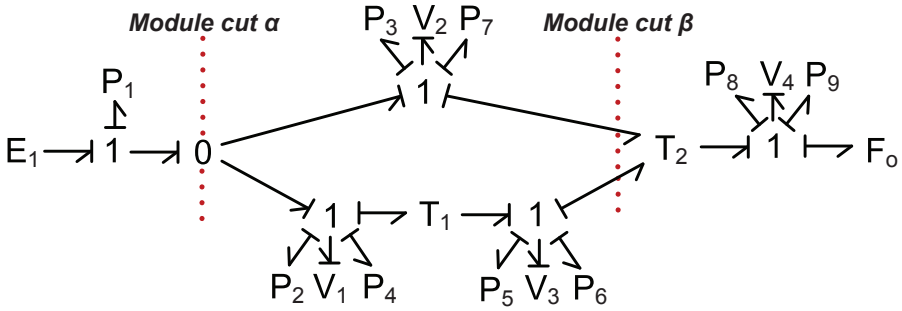

Figure 5.21: Bond graph for the two tank system.

Extended bond graphs can be utilized for designing glue modules as follows. The positions of given plant cuts are inserted into the bond graph as it is shown in Fig. 5.21 for the case of the two cuts of the tank system. The splitting process significantly depends on the type of each cut, i.e., whether it is a cut on the junction level, or whether it is a cut on the bond level.

5.11.1 Prerequisites of the Simulation Splitting Support

The three predicates are defined and implemented:

- $hasStrokeProximity(b, N)$
- $hasPowerIn(b, N)$
- $hasPowerOut(b, N)$

The first variable b represents the power bond, to which this property is associated. The second variable N represents one of the two bond graph nodes that are connected together by this specific power bond b.

In case of the first predicate (i.e., $hasStrokeProximity(b, N)$), the well-constructed bond graph implies that this predicate holds for exactly one power bond in case of N is a 0-junction or it holds for exactly n-1 power bonds in case of N is a 1-junction, where n is number of power bonds $b_i, i = 1, ..., n$ connected to this specific node N.

In case of the predicates $hasPowerIn(b, N)$ and $hasPowerOut(b, N)$, no restrictions are required. However, the bond-graph theory assumes that especially in case of 0-junctions at least one is outgoing and in particular, it is the strong bond (i.e., for the 0-junctions the strong bond is such a bond that has a causality stroke on the side nearby to the 0-junction). When this rule is broken, the simulation model can diverge.

5.11.2 Cuts on the Junction Level

A more complex situation occurs in case of cuts on the junction level, where signals should be added or subtracted. The final mathematical expression is obtained based on surrounding bonds and their causality assignments as well as the utilized power directions. The mathematical description of the cut on junction level depends on the type of the junction. We can start with 0-junctions. Considering that 0-junctions add inflows and set the same effort to the connected power bonds according to Eq. 2.5 and Eq. 2.6, we get the following equations characterizing the cut on the 0-junction level:

The bond graph node N is the selected 0-junction. To this 0-junction, n power bonds are connected; $b_j(n)$ is a j-th power bond connected to the 0-junction N. The integrating junction on the module level can be characterized as follows:

$$\exists! j(hasStrokeProximity(b_j, N))$$

$$\begin{pmatrix} eo_1 \\ \vdots \\ eo_{j-1} \\ eo_{j+1} \\ \vdots \\ eo_{n-1} \\ fo_j \end{pmatrix} = \begin{pmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ (-1)^{\sigma_1} & \cdots & (-1)^{\sigma_{j-1}} & (-1)^{\sigma_{j+1}} & \cdots & (-1)^{\sigma_{n-1}} & 0 \end{pmatrix} \begin{pmatrix} fi_1 \\ \vdots \\ fi_{j-1} \\ fi_{j+1} \\ \vdots \\ fi_{n-1} \\ ei_j \end{pmatrix}$$

where $\sigma_k = hasPowerOut(b_k, N)$, $k = 1, ..., n-1$.

In case of 1-junctions as integrating junctions on the module level, the situation is dual. Considering that 1-junctions add efforts and set the same flow to the connected power bonds according to Eq. 2.7 and Eq. 2.8, we get the following equations characterizing the cut on the 1-junction level:

The bond graph node N is the selected 1-junction. To this 1-junction, n power bonds is connected; $b_j(n)$ is a j-th power bond connected to the 1-junction N. The integrating junction on the module level can be characterized as follows:

$$\exists! j(\neg hasStrokeProximity(b_j, N))$$

$$\begin{pmatrix} fo_1 \\ \vdots \\ fo_{j-1} \\ fo_{j+1} \\ \vdots \\ fo_{n-1} \\ eo_j \end{pmatrix} = \begin{pmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ (-1)^{\sigma_1} & \cdots & (-1)^{\sigma_{j-1}} & (-1)^{\sigma_{j+1}} & \cdots & (-1)^{\sigma_{n-1}} & 0 \end{pmatrix} \begin{pmatrix} ei_1 \\ \vdots \\ ei_{j-1} \\ ei_{j+1} \\ \vdots \\ ei_{n-1} \\ fi_j \end{pmatrix}$$

where $\sigma_k = hasPowerOut(b_k, N)$, $k = 1, ..., n-1$.

5.11.3 Cuts on the Power Bond Level

For the cuts on the bond level, the effort/flow assignment depends on the relative position of the causality stroke only. The direction of the power is not considered as it has been already taken into account when constructing the entire bond graph and specifying sign conventions of signals at junctions lying nearby the module cut.

Whereas in the case of cuts on the junction level the utilized predicates are useful or even needed for implementation of the extended bond graph method, in case of cuts on the bond level, two further predicates have to be added. The two predicates are defined for this case:

- $strokeModuleA(b)$

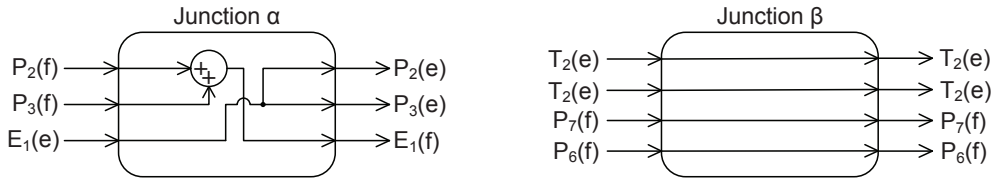

Figure 5.22: Junctions of the simulation model for the two tank system.

- $strokeModuleB(b)$

Each cut separates the simulation model into two modules, denoted here as modules A and B. Since the position of the causality stroke is crucial for cuts on the power bond level, these predicate express the position of such a stroke in the relationship to the designated simulation modules for each power bond b. The former predicate $strokeModuleA(b)$ holds if and only if the stroke belongs to module A, whereas the predicate $strokeModuleB(b)$ holds for those bonds b that have causality stroke as a part of the second module. It is straightforward that for the both predicates holds the following statement:

$$\forall j \Big(\big(strokeModuleA(b_j) \wedge \neg strokeModuleB(b_j) \big) \vee \big(\neg strokeModuleA(b_j) \wedge strokeModuleB(b_j) \big) \Big)$$

To specify the interfaces of the integration modules, the positions of causality strokes is aggregated into vectors according to the following equations:

$$\vec{\alpha} = \big\{ j : strokeModuleA(b_j) \big\}$$

$$\vec{\beta} = \big\{ j : strokeModuleB(b_j) \big\}$$

The structure of the integration modules on the power bond level holds the following equations, which could be also rewritten in a dual form with the parameter β:

$$\begin{pmatrix} \vec{eo}_\alpha \\ \vec{fo}_\alpha \end{pmatrix} = \begin{pmatrix} \vec{ei}_\alpha \\ \vec{fi}_\alpha \end{pmatrix}$$

Considering the example depicted in Fig. 5.20, we can see the case of the bond level as the junction β. The power flows into the 0-junction via a bond from the left 1-junction. Effort is an input of the 0-junction and the junction calculates output flows as the sum of the two flows to the rest of the system on the right-hand side. We can see that the inner implementation of the glue module depends on neighboring bonds only. However, it is necessary to create the bond graph for the whole system in order to be able to assign causality and power flows correctly.

5.11.4 Example of Integrating Junctions and Evaluation

For the two tank system, the internal representation of glue modules is depicted in Fig. 5.22. The entire simulation workflow including these glue modules is then depicted in Fig. 5.23. In both cut types, the glue modules are important for (i) timing aspects, where the module can provide aggregation of data, re-sampling and synchronization. In addition, (ii) the glue modules are useful for the performance analysis as a probe into the system.

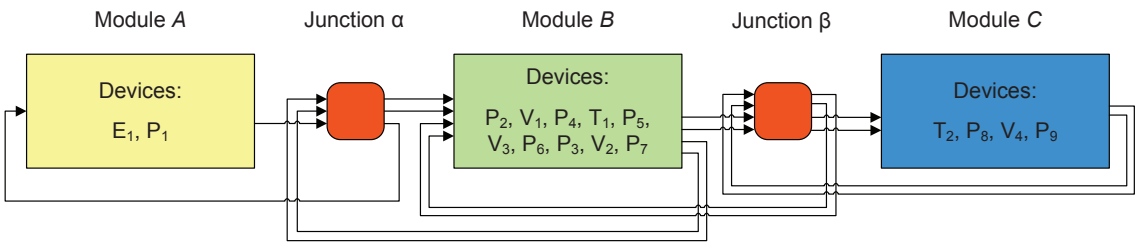

Figure 5.23: Modules of the simulation model for the two tank system.

Such a separation of complex simulations into a set of coupled simulation modules brings the following benefits:

1. Parallelization of simulation execution
2. Easier maintenance and (re-)design of simulations
3. Significantly faster initialization of simulation environments
4. Simplified testing of simulation modules and their fine-tuning

The proposed method solves the simulation module integration problem from the structural point of view. Timing and synchronization issues are not in scope of this thesis. It is assumed here that they are solved by simulation solvers or optimized by simulation experts. The following section describes how to execute such simulation workflows consisting of several simulation modules.

5.12 Execution of Complex Coupled Simulations at Runtime

As each simulation model can consist of one or more simulation modules, which can be designed for example according to the methodology proposed in the previous sub-section, the overall methodology proposed in this thesis has to deal with the problem of integration of simulation modules into complex workflows and their execution at runtime. Each simulation module approximates a specific part of the real plant or provides data transformations needed for simulation, such as pre-processing of initial conditions or calculation of boundary conditions.

The main rule for the execution of the entire simulation workflow is the satisfaction of the data-driven architecture, i.e., data are transferred within simulation modules or among simulation modules and other tools such as HMIs when tag values are available or when a batch task should start by a user command. The timing issues are in charge of simulation engineers, and the proposed approach does not pose any additional constraints for this task. In the basic case, the execution of simulations is stopped when waiting for required data. The exchange of tag values between simulation modules and the rest of the integrated environment is the task provided partly by the Java-implementation of the connector and partly by data exchange blocks in the language of the simulation. To execute a complex simulation composed of several simulation modules, the proposed infrastructure utilizes the EngSB workflow engine, i.e., the rules for the execution of simulation modules being parts of the complex simulation workflows have the same nature as the rules for the integration

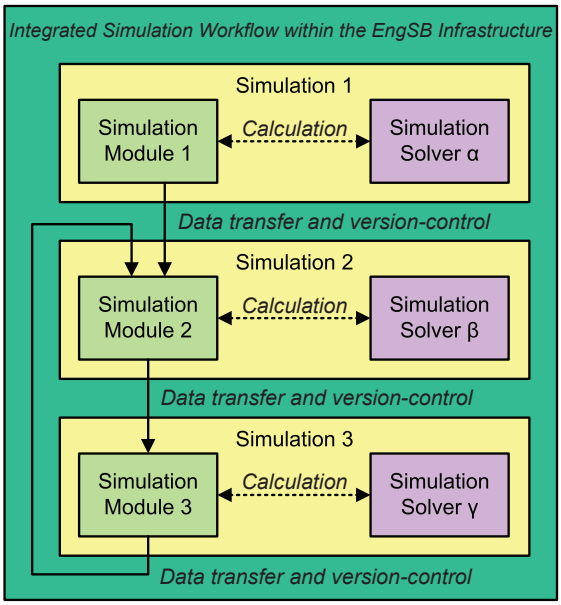

Figure 5.24: An exemplary use of three simulation modules within the EngSB environment.

of simulations with the rest of the automation system. Both sets of rules are based on the engineering knowledge, from which they are automatically generated.

To illustrate the rules for execution of complex simulation workflows, we can use an exemplary scenario depicted in Fig. 5.24. The solid arrows illustrate the transfer of tag values between simulation module interfaces, whereas the dashed lines express that the simulation modules are executed by a simulation solver. Simulation solvers are typically included in simulation environments, such as MATLAB. Frequently, the simulation modules utilize the same simulation solver, however, each module can use a different solver in general. The role of the simulation solver can be compared to an interpreter of programming languages where the code corresponds to the simulation module and the interpreter to the simulation solver executing the simulation module. The core of the simulation solver is an implementation of a numerical method that is able to solve equations defined by the simulation module directly (in case of equation-based simulations) or equations inferred from simulation models by the simulation environment (in case of signal-oriented simulations). In compliance with the mathematical description of dynamic systems introduced in Sec. 2.1, the main problem to be supported by the numerical method is the calculation of time-series related to the differential equations 2.1. In current solvers, widely used numerical methods are Runge-Kutta methods. An overview of these numerical methods can be found in [32]. Since several simulation solvers are included in simulation environments (such as MATLAB), the simulation or integration engineers do not need to have a detailed insight into the numerical method or the entire solver.

From the simulation design and integration points of view, important aspects are configurable parameters that simulation solvers require. The parameters are directly passed as a configuration to the numerical method algorithm, such as absolute and relative precision of the calculation, or start and end of the simulation time. Some of the parameters configure a version of the numerical method to be used, for example the number of steps in multi-step methods. The parameters, which are expected to change during the simulation model engineering or execution, are captured in the automation ontology, whereas those

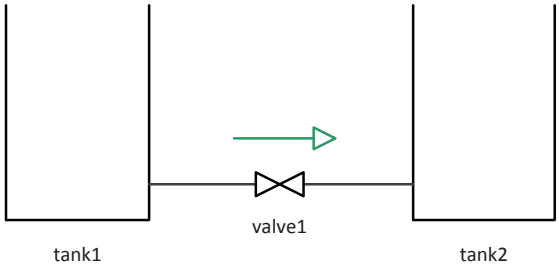

Figure 5.25: The two-tank system – The case having the same elevation of tank bottoms.

that are fixed during the whole simulation model life-cycle can be directly stored in the simulation modules. In case of batch processes, input data have to be available already at the beginning of the execution of the simulation workflow. In case of synchronized tasks, several stakeholders run in parallel and exchange data.

5.13 Optimization of Complex Simulation Model Execution

The performance of simulation model execution is of course significantly affected by computational resources. However, numerical stability of the simulation plays a crucial role as well, in many cases even more crucial. To get better the performance of generated simulations, one or both aforementioned factors have to be improved. Buying a new piece of hardware is not feasible as outcome of this thesis, therefore, we will focus on the latter aspect.

The bottleneck of simulation model runtime, which occurs in industrial projects frequently, is handling of physical constraints. The author of this thesis investigated various options how to mitigate the impact of the constraints to the performance of the created simulations systematically. The proposed solution is based on the utilization of signal bonds of bond graphs. Prior describing the author's proposal, the problem of constraints is described on the hydraulic system use-case.

Keeping the example as simple as possible, we can assume that we have a two-tank system, where the two tanks are connected via two pipes with a valve. From the perspective of water level time-courses, the positions of tank bottoms as well as connection points of pipes have crucial impacts on the system behavior.

The simplest case is depicted in Fig. 5.25, where both tank bottoms have the same elevations and the connection points of the pipes are at the levels of the tank bottoms. For both hydrostatic pressures on the connection points holds:

$$p_i = h_i \rho g \tag{5.11}$$

where p_i is hydrostatic pressure at the i-th tank or pipe connector, h_i is the height of the liquid above the i-th tank or pipe connector, and g is gravitational acceleration.

The flow through the serial combination of two pipes and the valve is given by the pressure difference:

$$Q \propto h_1 \rho g - h_2 \rho g \tag{5.12}$$

where the indices of heights correspond to numbers of tanks according to their names in Fig. 5.25 and the positive direction of liquid flow corresponds to the arrow involved in the figure above the valve.

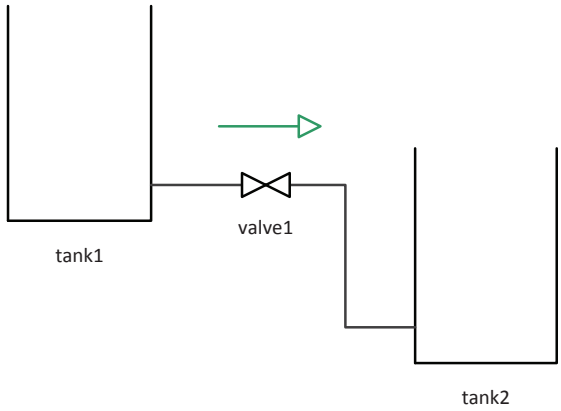

Figure 5.26: The two-tank system – The second version of the system having different elevations of tank bottoms.

The liquid level in the tanks is given by the equation:

$$h = \frac{1}{S} \int_{t_0}^{t} Q(\tau) \mathrm{d}\tau \tag{5.13}$$

where the flow Q is considered not in the absolute value but with the sign convention denoting the direction of the flow. For completeness, an initial condition (i.e., an initial liquid level) has to be pre-set for each tank before starting the simulation. This value is denoted as h_0 for each tank.

If the physical positions and mountings are in the configuration depicted in Fig. 5.25, liquid levels in tanks cannot reach negative values in any tank. Therefore, we are on the safe side when designing a simulation model and physical constraints need not be taken into account. When simulation accuracies are set roughly, negative peaks of liquid level can emerge, but they can be neglected frequently.

On the other hand, if elevations of tanks bottoms are different or pipe connection points are not at tank bottoms, the situation is becoming significantly more difficult. The different elevations of tank bottoms are depicted in Fig. 5.26. In this case, $h_1 > h_2$. To make the design of simulation models reasonably modular, the liquid level has to be treated in comparison to a reference level:

$$p_i = (h_i + h_{e,i})\rho g \tag{5.14}$$

where p_i is hydrostatic pressure at the i-th tank or pipe connector related to a common base elevation, h_e, i is the elevation of the tank bottom of the i-th tank or pipe connector. The meaning of other symbols remains the same as in Eq. 5.11. Considering hydrostatic pressures in absolute dimensions leads to another expression for the flow between the two tanks:

$$Q \propto (h_1 + h_{e,1} - h_2 - h_{e,2})\rho g \tag{5.15}$$

Considering the aforementioned equations, we can see that negative liquid levels can be reached during the system simulation for the configuration depicted in Fig. 5.26.

To guarantee that the simulation model state values are within the feasible sub-space, the original linear representation has to be changed to a non-linear model. In principle, two ways of guaranteeing such limitations can be applied. The first one is based on changing state values artificially, in this case changing output effort of the tank if it is reaching

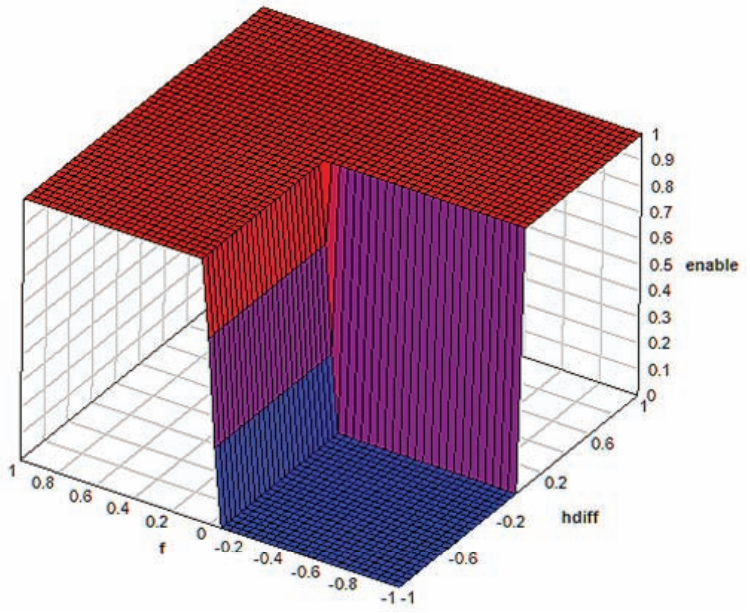

Figure 5.27: Visualization of fuzzy rules.

negative value of amount of liquid in the tank. The second possibility is not to change the state variables, but to block energy transfer paths that lead to unfeasible states. In the following text, these alternatives are discussed in more details and compared.

The modification of state variables to guarantee realistic states can be done in various ways, one example are the following rules:

- `IF` $((h_i - h_x) < 0) \wedge (Q < 0)$ `THEN` *multiplier* `IS 0`
- `IF` $((h_i - h_x) < 0) \wedge (Q \geq 0)$ `THEN` *multiplier* `IS 1`
- `IF` $((h_i - h_x) \geq 0) \wedge (Q < 0)$ `THEN` *multiplier* `IS 1`
- `IF` $((h_i - h_x) \geq 0) \wedge (Q \geq 0)$ `THEN` *multiplier* `IS 1`

The variable h_i means the height of the i-th pipe connector into the specific tank, h_x is the level of liquid in the tank, Q is the flow through the pipe connector where positive values mean inflow whereas negative values mean outflow, and *multiplier* is a correction factor that multiplies the calculated output effort of the tank. This version of constraint support is hereinafter called a basic constraint handling method. The aforementioned rules can be interpreted in a fuzzy way in order to make the switching between values smooth and the simulation faster and more stable. The graphical interpretation of these IF-THEN rules interpreted in the fuzzy sense is depicted in Fig. 5.27.

Important obstacles of this method based on changing state variables are (i) the problematic interpretation of state variables by humans and (ii) the numerical destabilization of the simulation and destroying impacts on simulation precision. Last but not least, the latter case has impact on the simulation performance, which can be significantly lower. In large-scale industrial system practice, another method has to be used due to these reasons.

The second way how to realize the constraints is blocking flows between the state devices/components. In this particular case it means "closing" the valve or "switching off" pumps artificially. It is difficult to manage the searching and solving these issues manually

hence it is beneficial to provide foundations for this issue that enable to automate or at least semi-automate this process. The method proposed in the following explanation is hereinafter called optimized constraint handling.

The state components, which are the tanks in this case, produce externally visible output signals denoted "*il*". The meaning of this abbreviation is inter-locking and it is inspired by the process control approaches. Positive values of this signal indicate that a physical constraint is reached, but it does not solve the inter-locking of flow paths itself:

- `IF` $(h_i - h_x) < 0$ `THEN` il_i `IS 1`
- `IF` $(h_i - h_x) \geq 0$ `THEN` il_i `IS 0`

To handle the inter-locking mechanism itself, the simulation blocks responsible for flow transfer should consider the desired direction of the flow and stop it when inter-locking is required due to reaching the physical constraint. The needed functionality can be expressed as follows:

- `IF` $(Q \geq 0) \wedge (il_f \leq 0)$ `THEN` *multiplier* `IS 1`
- `IF` $(Q \geq 0) \wedge (il_f > 0)$ `THEN` *multiplier* `IS 0`
- `IF` $(Q < 0) \wedge (il_b \leq 0)$ `THEN` *multiplier* `IS 1`
- `IF` $(Q < 0) \wedge (il_b > 0)$ `THEN` *multiplier* `IS 0`

The variable Q denotes directed flow and *multiplier* remains the correction factor that is in this case applied to the calculated flow variable. The variables il_f and il_b are the pair of *il* signals from the tanks, between which the simulation block implementing the inter-locking lies. The positive direction of flow is denoted as forward direction (i.e., respective inter-lock signal is coming from the source tank and it is expressed as il_f), whereas the negative flow is denoted as backward direction (i.e., respective inter-lock signal is coming from the destination tank and it is expressed as il_b).

To automate the process of working with constraints and to improve the performance of simulation models on the numerical stability level, it is needed to address them in compliance with the bond-graph theory. In the first case, when the values of the state variables are modified, no additional support is needed as the logic is solved within individual simulation blocks.

In the case of the latter case introducing the inter-locks on the simulation level, the propagation of inter-lock signal can be done with the support of signal bonds introduced by the bond-graph theory. Reaching constraints of state elements is propagated by dedicated signal bonds to all branches transferring energy until another state element is reached. The example of this proposed approach is depicted in Fig. 5.28 for the case of the two-tank system. Due to simplicity reasons, the pipes surrounding the valves are merged with this valve to avoid algebraic loops in the model.

The final implementations do not utilize the aforementioned fuzzy rule bases, but the logistic function is used:

$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}} \tag{5.16}$$

The parameter setting utilized in Sec. 7.2 is $L = 1$ and $x_0 = 0$. For the state signals (both their artificial change and calculation of the inter-lock signal), the constant k is set $k = 5 \cdot 10^3$ and for the flow variables, the constant k is set $k = 5 \cdot 10^5$.

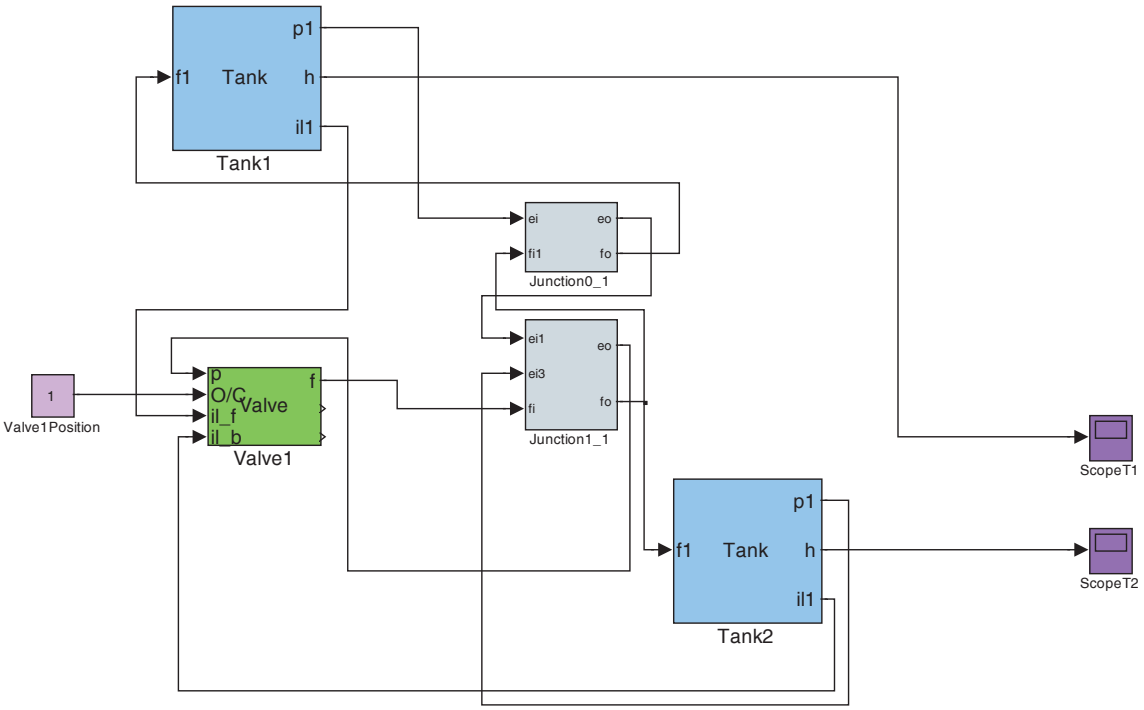

Figure 5.28: Simulation model for the two-tank system with the optimized structure satisfying physical constraints of feasible values of liquid levels.

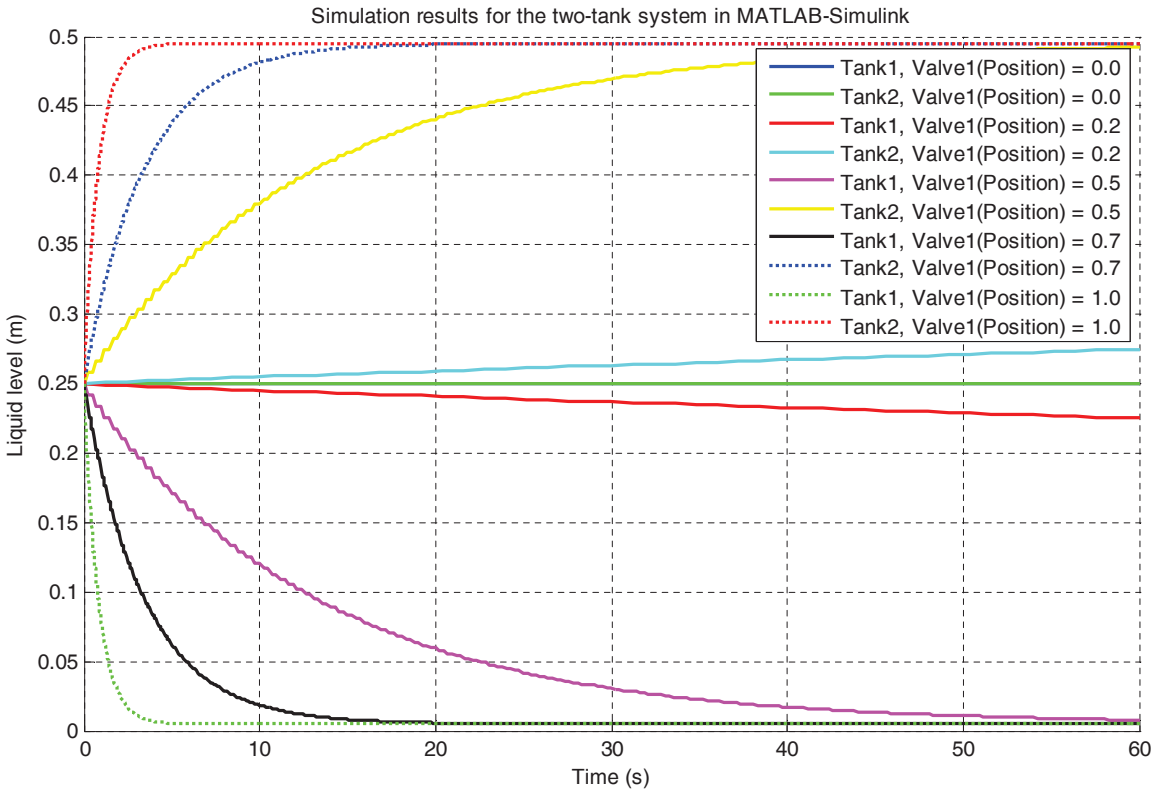

Figure 5.29: Simulated results of the two-tank system with the optimized structure satisfying physical constraints of feasible values of liquid levels.

An example of simulated results is depicted in Fig. 5.29, which depicts the positive impact of this solution on the simulated results. No instabilities or unwanted resets of the simulation solver are apparent in the figure. Although in the case of the two-tank system any of the methods mentioned above can be used, the large-scaled simulations that are targeted by this thesis need fast and computationally stable solution, which holds for the proposed approach perfectly.

5.14 Developed Tool Support for the Simulation Model Generation Based on Extended Bond Graphs

Within this thesis, several software prototypes for the simulation model generation were implemented. Their main purpose was to provide a proof-of-concept solution in order to verify the feasibility, correctness, and efficiency of the proposed methods and algorithms.

The graphical user interface (GUI) of the main software prototype implemented in Java is depicted in Fig. 5.30. It is intended to generate a simulation model in MATLAB-Simulink based on a given AutomationML plant model and a simulation library. This prototype enables to select the AutomationML file that includes the plant model, which is done by a button on the right-hand side at the bottom. The file selected in Fig. 5.30 is called "myElectricalCircuit.aml". To generate a simulation model, a simulation library including generic simulation blocks has to be selected. In Fig. 5.30, the selected simulation library is called "MechatronicLibrary" and it was developed by the author of this thesis. It includes basic simulation blocks for electric and hydraulic systems as well as necessary blocks such as both types of junctions or a block for visualization of simulation results. The details about this simulation library are discussed in Appendix B and its high-level layer is depicted in Fig. B.1. To generate the simulation model, the software prototype is equipped with the button "Generate Simulation Model", which is depicted in the central part of Fig. 5.30. This button starts the entire method described in this thesis, which results into a executable simulation model generated in MATLAB-Simulink. To run the final simulation, the user can change the simulation time and click the button "Simulate" in the MATLAB-Simulink environment. The obtained simulation results are automatically depicted in a 2-D graph when each simulation model run is finished.

A more vivid example of the operation of this software prototype can be seen in the screencast that is available online[4]. The screencast includes two use-cases: (i) the electrical circuit and (ii) the two-tank system discussed in Sec. 5.13.

To represent the extended bond graph approach in Java, the inheritance-based class model was found to be effective and efficient. A simplified UML representation is depicted in Fig. 5.31. The abstract classes are depicted in the gray color, whereas classes that can be instantiated are depicted in the green color. Together with this architecture snippet, other software parts discussed in this thesis are used. The access to AutomationML is later on described in Sec. 6.3.2 and the control of MATLAB is explained in Sec. 6.4.1.

[4]Screencast is available online: https://youtu.be/vn890gGndeM

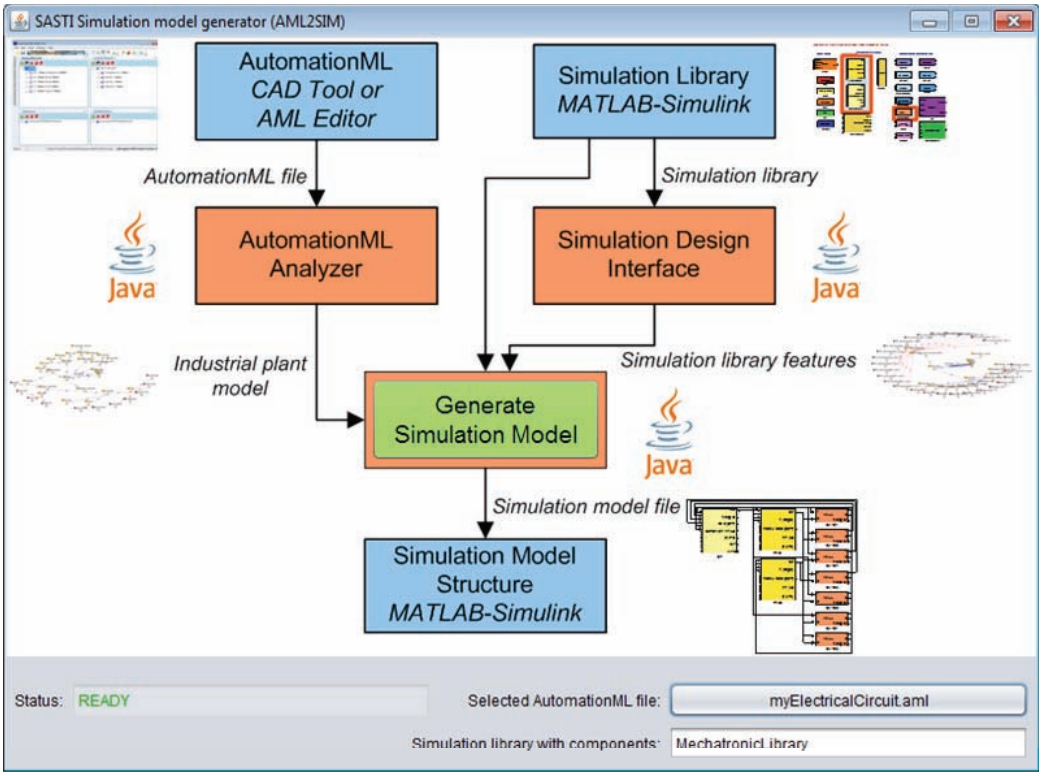

Figure 5.30: Graphical user interface of the implemented software prototype for generating simulation models from simulation library blocks according to the system model represented in the AutomationML format.

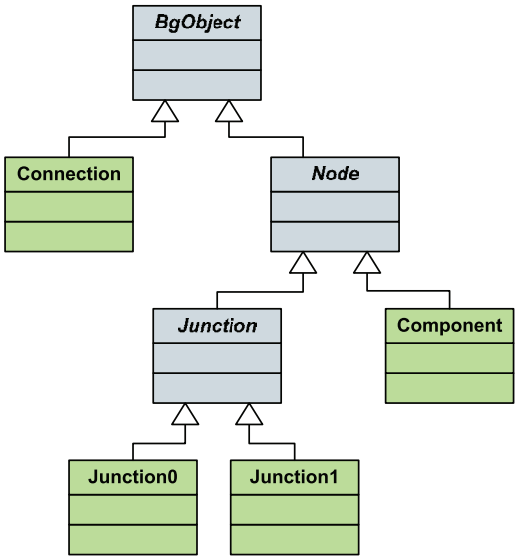

Figure 5.31: The UML class diagram expressing the representation of bond graph structures in the ontology tool in Java.

Chapter 6

Improved Integration of Simulation Models

While the previous section addressed the design of simulation models and their splitting into a set of simulation modules, this section is focused on their integration. It describes the integration both in the runtime phase, when tag values are transported between stakeholders, and in the design-time phase, when engineering tools have to share knowledge and this knowledge is captured in and retrieved from the knowledge base implemented by the automation ontology.

This chapter addresses the research issue RI-3 and together with the technical background provided here, it also solves the goal of the thesis G-4. In addition, this chapter provides a technical background for the goals of the thesis G-2 and G-3 presented in Sec. 5.

The problem of the integration of simulation models can be divided into two parts: (i) a physical (i.e., technical) integration using service-oriented interfaces [135], and (ii) a configuration of the technical level, which is frequently referred as the use of a semantic integration [128] on the tool level. The former part is related to finding a suitable technical infrastructure to transfer data, whereas the latter part covers finding mappings between adequate entities. For example, semantic mappings interrelate a really measured variable and its simulated approximation, real devices and their equivalents in a simulation model or local names of tags used in a particular tool with the global representation of the tag name [127].

6.1 Requirements and Challenges on Integrated Automation Systems

The basic challenges of simulation model design and integration for dynamic industrial systems are depicted in Fig. 6.1. The numbered circles mean the following challenges that are addressed later in details in this thesis:

1. The overall architecture of the integrated system that supports efficient simulation model integration;
2. Bond graph modeling for the improved design of simulations;
3. Representation of engineering knowledge relevant for simulation design and integration

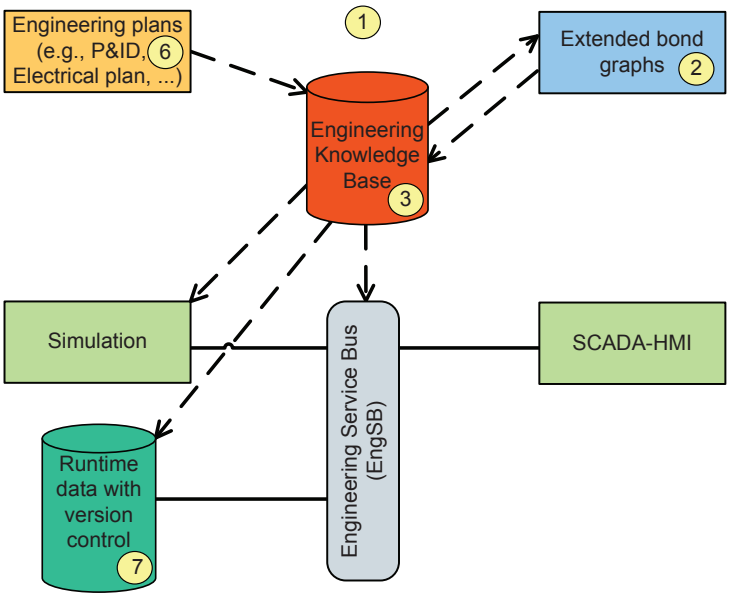

Figure 6.1: Challenges in integration and design of simulation models.

and access to it;

4. Specification of simulation scenarios and their execution;
5. Integration of simulations with SCADA systems;
6. Support for knowledge originating in various engineering plans as a knowledge background for simulation integration;
7. Management and access to runtime data;

The architecture and processes satisfying the aforementioned challenges are strongly affected by the process of industrial plant engineering and automation system engineering. That is the reason why the description in the following sections starts up with the engineering plans as entry points to simulation model design and integration.

6.2 Proposed Architecture of the Integrated SCADA Level of Automation Systems

After analyzing challenges in the simulation model design and integration that were formulated in Sec. 6.1, requirements of industrial partners, and state-of-the-art presented in Sec. 3, a new architecture for integrated industrial automation systems including process simulations has been proposed.

The approach proposed and utilized in this thesis relies on the system architecture depicted in Fig. 6.2. The central point of the infrastructure is the Engineering Service Bus (EngSB), which is a middle-ware responsible for transferring data and their proper and safe routing among stakeholders. All stakeholders (i.e., tools) are connected to the EngSB via connectors. Each connector has a domain specific part and a tool specific part. The data are transferred to the EngSB according to the pre-defined workflows that are executed by the workflow engine. The development of the EngSB itself was not a part of this thesis, but

the author's contribution is the application of the EngSB for the simulation model design and integration.

The main benefit of using an ESB is that they have proved to be a highly flexible, comprehensible, and maintainable infrastructure for data and tool integration. Although ESBs are de-facto standards for financial and business integration for several years as well as for enterprise application integration in general, the utilization in the industrial automation area is still rare and this approach is pioneering and promising [121]. The utilized platform EngSB is not only a particular implementation of the ESB concept, but it provides features specific for the industrial automation area. The key enhancements of the ESB concept, which characterize the EngSB, can be summarized as follows:

1. *Tool domains*

   The tool domain [21] is a tool independent interface and it may be interpreted as a standardization of connectors and engineering tool types to facilitate the easy exchange of tools without affecting data exchange with other tools. In industrial automation, such domains are for example a simulation domain including several process simulators, a tag domain providing data-exchange, and others.

2. *Engineering objects*

   Engineering objects [172] are entities in the EngSB that represent common concepts, i.e., the artifacts accessed from various tool domains. Examples of engineering objects are tags (signals) whose names and values can differ in various domains such as in simulations, OPC or OPC UA, or historians.

3. *Engineering Knowledge Base (EKB)*

   The EKB [109] is an approach for a semantic integration in heterogeneous engineering environments. It stores explicit engineering knowledge and supports data integration based on mappings between local and domain-level engineering concepts, as well as other more complex transformations. For the simulation model design and integration purposes, it utilizes the proposed automation ontology as its data model.

From the workflow perspective, two basic types of processes are distinguished in this thesis: (i) design-time processes, and (ii) runtime processes. This differentiation is considered from the operation of simulations point of view. Therefore, design-time processes are related to concentrating engineering knowledge from engineering plans and tools in the knowledge base and to engineering simulation models based on the aggregated knowledge. The runtime processes are focused on the runtime operation of simulation models, which means that simulation models are given with input data and parameters, simulations are executed, and output data are delivered where required. Runtime processes do not cover batch processes only, but also synchronized tasks where simulation inputs and outputs are integrated simultaneously.

From the technical perspective, the configuration of the simulation integration framework, which is facilitated by this thesis, is based on the following XML files: (i) global tag list, (ii) tag translation tables, and (iii) tag routing tables. The global tag list file is aimed at setting all existing tags in the EngSB environment and their properties (such as type, minimal and maximal values or others, which are useful for a control system). The tag translation tables are related to EngSB connectors, which translate tag names between

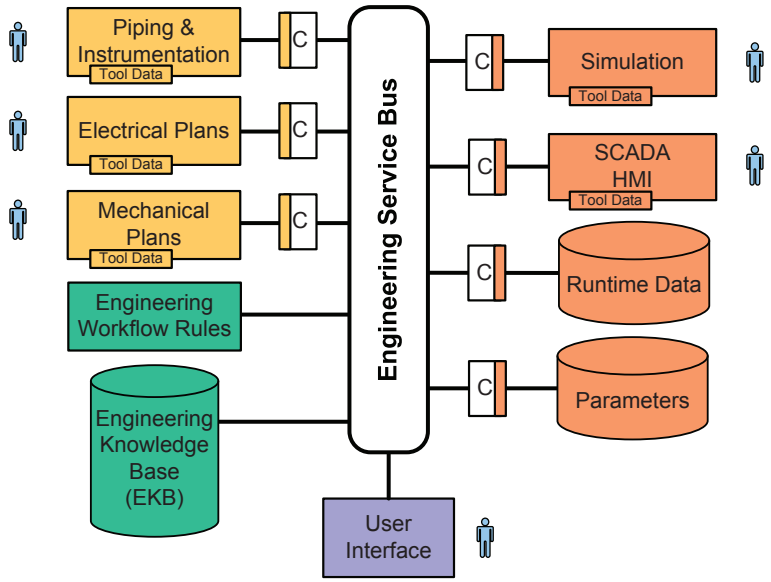

Figure 6.2: Runtime integration based on the Engineering Service Bus (EngSB) from the technical point of view.

local names (occurring in the particular tool data model scope) and global tag representations, being uniquely available in the EngSB. The tag routing tables define how tag values should be distributed between the tools (such as simulation results must be transmitted to HMI, simulation inputs have to be entered with an operator-training data-set). Note that the global tag list and tag translation tables are project-specific, whereas routing tables are scenario-specific. In other words, every project can have several scenarios, such as simulation can be used as a soft-sensor to estimate unmeasured states, or as a test-bed to analyze control system behavior and to train human operators.

The overall architecture is depicted in Fig. 6.2. The most important tool domains and implemented tool connectors are addressed in more details in further sections. Moreover, industrial projects can require further tools or domains such as various optimizers, or operation planners. The EngSB does not restrict their inclusion among the tools or domains discussed in the following text.

6.3 Engineering Tool Domain

The EngSB-based infrastructure involves the engineering tool support as its native part, for which this infrastructure was intended to. In a general case, each engineering discipline utilizes its own tool domain, such as there are a piping and instrumentation domain, electrical domain, mechanical domain, etc. If the project enables, it is also possible to merge these domain into one engineering domain. Although an EngSB-based support for a large variety of tools has been already implemented by the EngSB community, in the further text, just two connectors for the engineering domain are discussed, because these two have been developed by the author of this thesis. First, the connector to Microsoft Visio is addressed. Next, the connector to AutomationML data format is discussed and a transformation of data from this format is explained.

6.3.1 Connector to Microsoft Visio

At the beginning of the simulation and automation engineering processes, a description of the entire real plant is needed. This description is captured in the automation ontology, however, an important issue is how is this ontology populated. In this section, the developed connector into Microsoft Visio is motivated and described.

Motivation for the MS Visio Plug-in Development

The basic motivation for this approach is summarized by the following requirements:

1. *Reuse of engineering knowledge to support simulation model design and integration*

   The simulation model structure is based on the topology of the real industrial plant. Tags of HMIs as well as simulation models reflect tags of the real plant, which are included in the process description and plans, such as P&IDs. When original engineering plans are not used automatically, traceability of changes and determining consistency of a specific simulation model with the real plant realization are hard tasks. The goal is to have a simple user interface that can be used for entering the real plant structure, tags and parameters in a visual way. The integration of existing engineering tools (such as EPLAN) is very complicated and suffers from licensing limitations. For that reason, a prototype of a plug-in into Microsoft Visio has been implemented.

2. *Support of a component-based approach*

   Simulation models of industrial systems typically consist of simulation components representing sub-parts of the plant. The goal is to handle each component as a whole and to work just with its interface, no matter how the block is internally implemented.

3. *Export simulation model interfaces for integration*

   A crucial issue of simulations is the access to on-line and historical data as well as the integration with supervisory control and data acquisition (SCADA) systems. This task is tightly coupled to expressing input and output tags and their mappings in specific engineering tasks.

Microsoft Visio as a General-Purpose Design Tool

Microsoft (MS) Visio is a multi-domain general-purpose drawing tool. MS Visio is becoming widely used for drawing graphical schemas and it can be considered as a simplified version of industrial CAD/CAE tools. The drawing in MS Visio is created mainly with so-called master shapes, which are located in a stencil toolbox in MS Visio. From the user perspective, it has the same design and interfaces as Microsoft Office tools, but it is not a standard part of MS Office. Similarly as MS Office tools, MS Visio supports macros implemented in Microsoft Visual Basic for Applications (VBA) and diverse plug-ins implemented usually as ActiveX components. Such a combination of MS VBA and ActiveX plug-ins makes an opportunity for a simple access to drawing data.

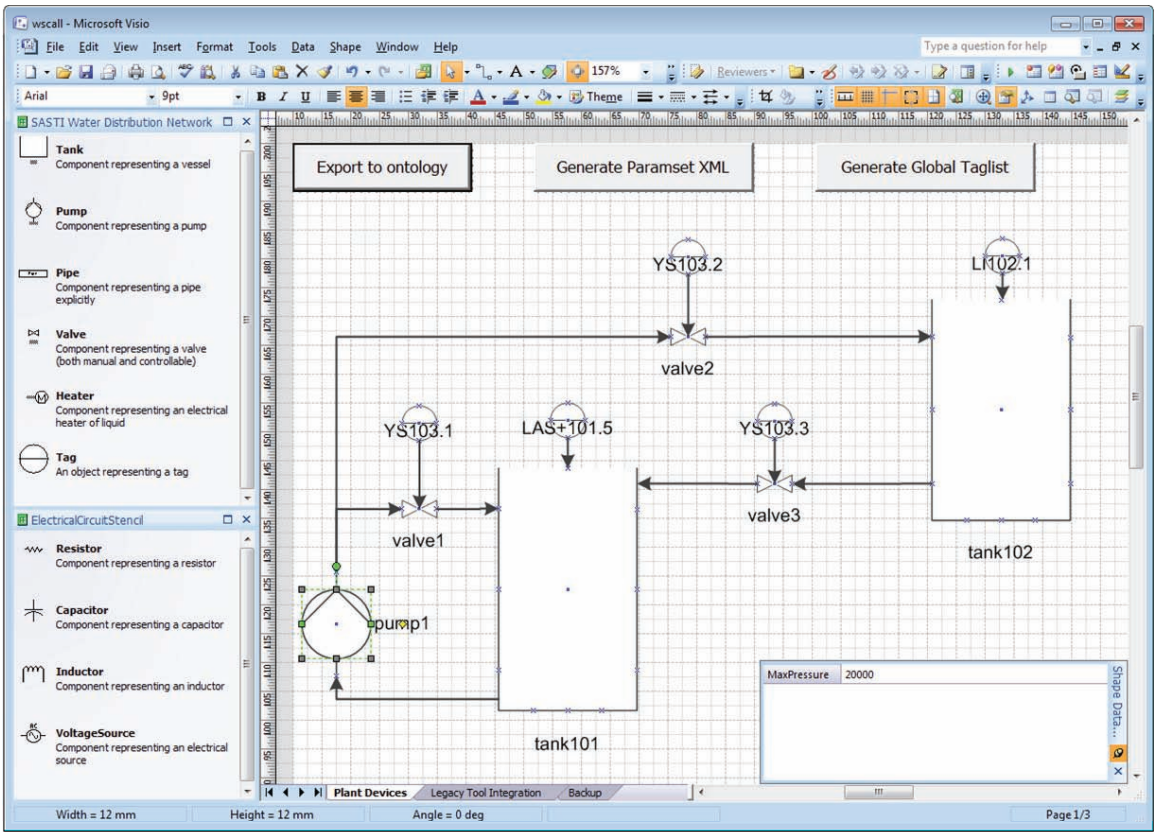

Figure 6.3: Screenshot of a drawing in Microsoft Visio including a button for the generation of individuals in the automation ontology via ontology tool interfaces. This drawing is later on used as an entry point for the hydraulic tank model use-case in Sec. 7.2.

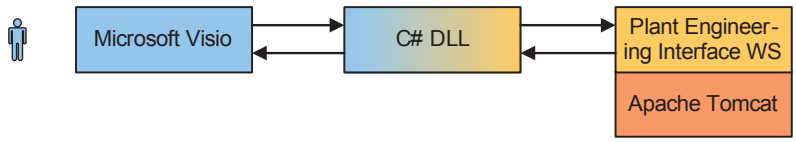

Figure 6.4: Bridging various programming languages to access the ontology tool in Java via Web Services.

Implemented Microsoft Visio Connector

To test the feasibility and efficiency of the proposed approach, a connector to Microsoft Visio was developed on the software prototype level. It provides the following features:

1. Utilization of domain-specific symbols for devices;
2. Expressing real plant structures;
3. Declaration and definition of parameters for each device;
4. Assignment of input and output tags to the devices;

Since the screenshots of the developed tool support pose rather technical issues that are intended for evaluation of the proposed method, the figures with these screenshots are included in Appendix C (i.e., Fig. C.1, Fig. C.2, and Fig. C.3). The basic macro, which ensures the export to the ontology, is implemented as a method handling a button click located in the upper-left-hand side of the MS Visio drawing, see Fig. 6.3. After clicking their button, a dialog with several user settings is shown – including the path to the ontology, the path to save the populated ontology, the logical name of the real system, and the logical name of the plant location, see Fig. C.1. Consequently, the MS Visio drawing is processed and ontology individuals and properties are instantiated via the real plant interface of the ontology tool. The prerequisite of this process is a synchronized structure of the Visio stencil and ontology model, in other words, each master shape (e.g., a pump) must have an equivalent class in the ontology. The algorithm in the macro iteratively accesses each shape in Visio and creates an ontology individual having the same name (i.e., a local URI) and the type of the individual is a name of the master shape (i.e., a local URI of the ontology class).

Since the ontology tool is implemented in Java, it is not possible to create an interface or a whole assembly as an ActiveX component directly. Java is an interpreted language and it runs via Java Runtime Edition tools. Therefore, it was necessary to bridge the programming languages MS Visual Basic for Applications and Java. The final implementation utilizes Web services, which are considered as the most promising solution for future maintenance, see Fig. 6.4 for more details. The ontology tool interfaces are compiled as Web services and performed on the Apache Tomcat container, therefore the tool methods can be invoked remotely from any programming language supporting Web service access. The implementation of the connector in MS Visio utilizes a very simple DLL implemented in C# which has the same methods as an interface of the ontology tool and which calls such methods via Web services. Such a solution was selected because of its compatibility with diverse versions of MS Visio. The technical limitation of the current approach is the maximum quota limit related to each Web service message, which must be properly set in all tools under integration. Since in large-scale projects this limit can be reached, in future work, it would be beneficial

to face the problem of large Web service message in a systematic way, such as defining and implementing rules for splitting messages into several sub-messages, which are consequently joined. Technically, the MS Visio plug-in calls methods of the "Plant Description Interface" of the ontology tool, which is depicted in Fig. 4.6. The current implementation is stable and the performance is satisfactory on performed test-cases.

6.3.2 AutomationML Connector

AutomationML is a neutral data format that is becoming an important and standardized way for capturing various knowledge dealing with structuring of real plants and many other issues related to automation systems. Although it has been intended for a point-to-point integration first, it perfectly fits for data exchange within complex tool chains as well [20]. Nowadays, only a minority of engineering tools is already supporting the AutomationML format, but a large variety of tools supporting AutomationML are expected in near future. Therefore, the author of this thesis decided to design and to implement a connector to AutomationML as well.

To justify the difference between the use of the AutomationML data format and the automation ontology proposed in this thesis, this paragraph summarizes the similarities and assumptions behind these approaches. In the AutomationML data format, the devices should be expressed with the "System Unit Class" libraries containing expected system units, from which the real plant should be composed. Moreover, AutomationML utilizes "Role Class" libraries, where semantics of system units as well as of their instances should be annotated. Both types of libraries should be specified prior to modeling system hierarchy and topology, and they should be unchanged during the whole automation project if possible. The attitude to the general knowledge in the automation ontology is very similar to the approach used in AutomationML (especially the CAEX part of AutomationML), but the main difference is that the automation ontology assumes that devices and other artifacts are unambiguously specified in the ontology by means adequate to system unit classes in the AutomationML data format on the project level. On the contrary, system unit classes as well as instances can be enriched by mappings to roles in AutomationML, which should specify the exact meaning of the utilized artifacts. Hence the concept of roles does not have an equivalent formalism in the automation ontology. On the contrary, both the automation ontology and AutomationML utilize formalisms for expressing interfaces and signals. In the automation ontology, signals are represented by the variable and tag sub-ontology, whereas in AutomationML, they are represented by "Interface Class" libraries.

The approach proposed in this thesis assumes to grab data about a real plant topology only. This amount of data is captured in the CAEX-part of AutomationML, thus neither PLCopen nor COLLADA have been supported by the implemented connector yet. The supported elements from AutomationML are depicted in Fig. 6.5. This figure also illustrates the mapping between the AutomationML elements and the respective elements of the automation ontology. Such a mapping was used to design a connector between the AutomationML data format and the automation ontology. In other words, the connector to AutomationML is basically a data transformer from AutomationML to automation ontology triples. To work properly, the correspondences between system unit classes and simulation components (respectively simulation blocks) have to be defined explicitly.

Since the implementation of the AutomationML support poses a complex technical task,

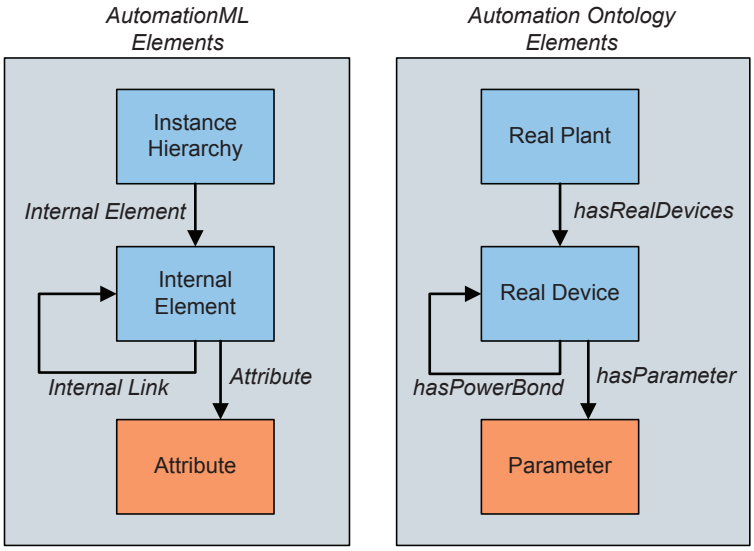

Figure 6.5: Mapping between the AutomationML data format and the class model of the automation ontology, which is used for transforming plant models from the AutomationML to the ontology-based representation.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX caex: <http://data.ifs.tuwien.ac.at/aml/ontology#>
PREFIX : <http://cyber.felk.cvut.cz/simulation/automation_ontology#>

CONSTRUCT {
  ?real_plant_n a :RealPlant .
  ?phy_device_n a ?device_type_n .
  ?real_plant_n :hasRealDevices ?phy_device_n .
  ?phy_device_n :hasPowerBond ?sideB_n .
  ?attrName_n a :Variable .
  ?attrValue_n a :Tag .
  ?phy_device_n :hasOutputVariable ?attrName_n .
  ?attrValue_n :isVariable ?attrName_n .
  ?attrName_n :hasLabel ?attrName .
  ?attrValue_n :hasName ?attrValue .
} WHERE {
  ?real_plant a caex:InstanceHierarchy .
  ?real_plant caex:internalElement ?phy_device .
  ?phy_device caex:refBaseSystemUnitPath ?systemUnit .
  OPTIONAL {
    ?phy_device caex:internalLink ?iLink .
    ?iLink caex:refPartnerSideA ?sideA .
    ?iLink caex:refPartnerSideB ?sideB .
    ?sideB caex:name ?sideB_ln .
    BIND (URI(concat("http://cyber.felk.cvut.cz/simulation/automation_ontology#", ?sideB_ln )) as ?sideB_n ) .
  }
  OPTIONAL {
     ?phy_device caex:attribute ?attr .
     ?attr caex:name ?attrName .
     ?attr caex:value ?attrValue .
  }
  ?real_plant caex:name ?real_plant_ln .
  BIND (URI(concat("http://cyber.felk.cvut.cz/simulation/automation_ontology#", ?real_plant_ln )) as ?real_plant_n ) .
  ?phy_device caex:name ?phy_device_ln .
  BIND (URI(concat("http://cyber.felk.cvut.cz/simulation/automation_ontology#", ?phy_device_ln )) as ?phy_device_n ) .
  ?systemUnit caex:name ?device_type_ln .
  BIND (URI(concat("http://cyber.felk.cvut.cz/simulation/automation_ontology#", ?device_type_ln )) as ?device_type_n ) .
  BIND (URI(concat("http://cyber.felk.cvut.cz/simulation/automation_ontology#Variable", ?phy_device_ln , ?attrName)) as ?attrName_n ) .
  BIND (URI(concat("http://cyber.felk.cvut.cz/simulation/automation_ontology#Tag", ?phy_device_ln , ?attrValue)) as ?attrValue_n ) .
}
```

Figure 6.6: The formulated SPARQL query to transform data from the AutomationML Analyzer Prototype ontology to the automation ontology.

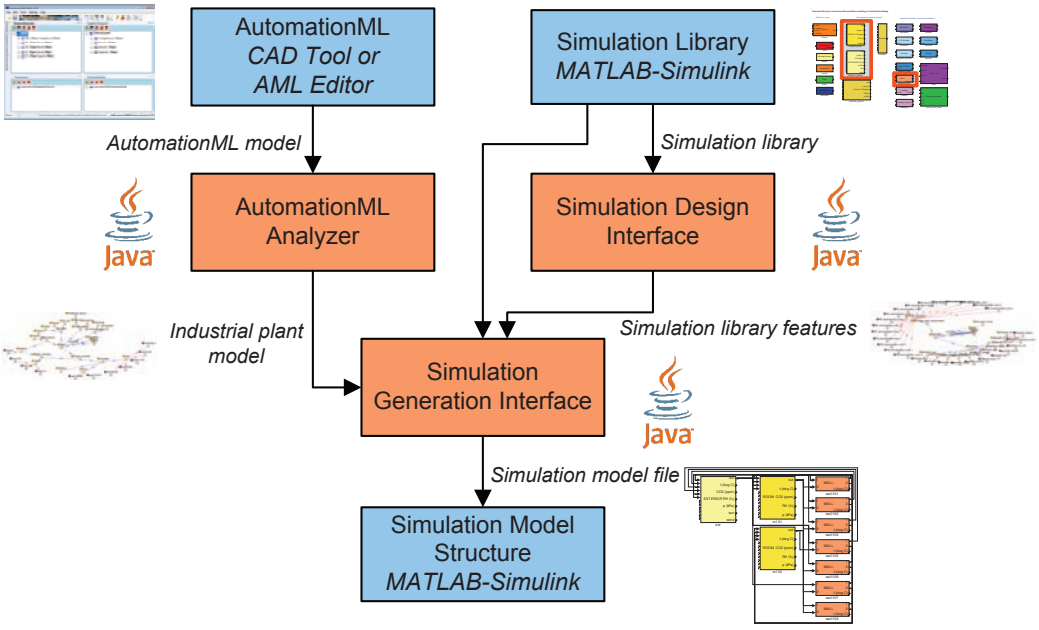

Figure 6.7: The proposed workflow for the transformation process starting at the AutomationML plant description and finishing at a generated simulation model in MATLAB-Simulink.

the existing AutomationML Analyzer[1] [147] was used to parse AutomationML data and to create their object model. This analyzer was developed by F. J. Ekaputra at the research laboratory CDL-Flex[2] at the Vienna University of Technology. The AutomationML file is loaded by the AutomationML Analyzer, it is transformed into an ontology that adopts the data model of the CAEX-part of the AutomationML. The AutomationML connector is consequently querying the AutomationML Analyzer with a SPARQL query depicted in Fig. 6.6. This CONSTRUCT SPARQL query transforms a set of triples from the AutomationML Analyzer to the automation ontology, which is done according to the mappings expressed in Fig. 6.5. Consequently, the knowledge from the automation ontology is used to design the simulation model in the very same way as in case of the MS Visio connector. The entire workflow is summarized in Fig. 6.7, which shows the process steps from the neutral process structure to a generated simulation model in MATLAB-Simulink.

To illustrate the AutomationML integration in practice, the electrical circuit that has been depicted in Fig. 5.5 is used. An AutomationML file for the electrical circuit was created in the standard AutomationML Editor[3]. It includes the devices of the electrical circuit, their types and interconnections. The entire electrical circuit is represented as an "instance hierarchy" in the AutomationML format. The devices are represented as instances, which are "internal elements" of the aforementioned "instance hierarchy". The type of each device is represented as "system unit class", as each internal element is in this case an instance of a system unit class. The interconnections between devices are represented as "internal links". The created AutomationML file is graphically expressed in the screenshot in Fig. 6.8, which is more comprehensible than the representation of the source XML. The screenshot depicts the internal elements of the instance hierarchy (i.e., the devices of the electrical circuit), and types of internal elements (i.e., system unit classes). The internal elements are not visible

[1] http://data.ifs.tuwien.ac.at/aml/analyzer
[2] http://cdl.ifs.tuwien.ac.at
[3] https://www.automationml.org/o.red.c/dateien.html?cat=1

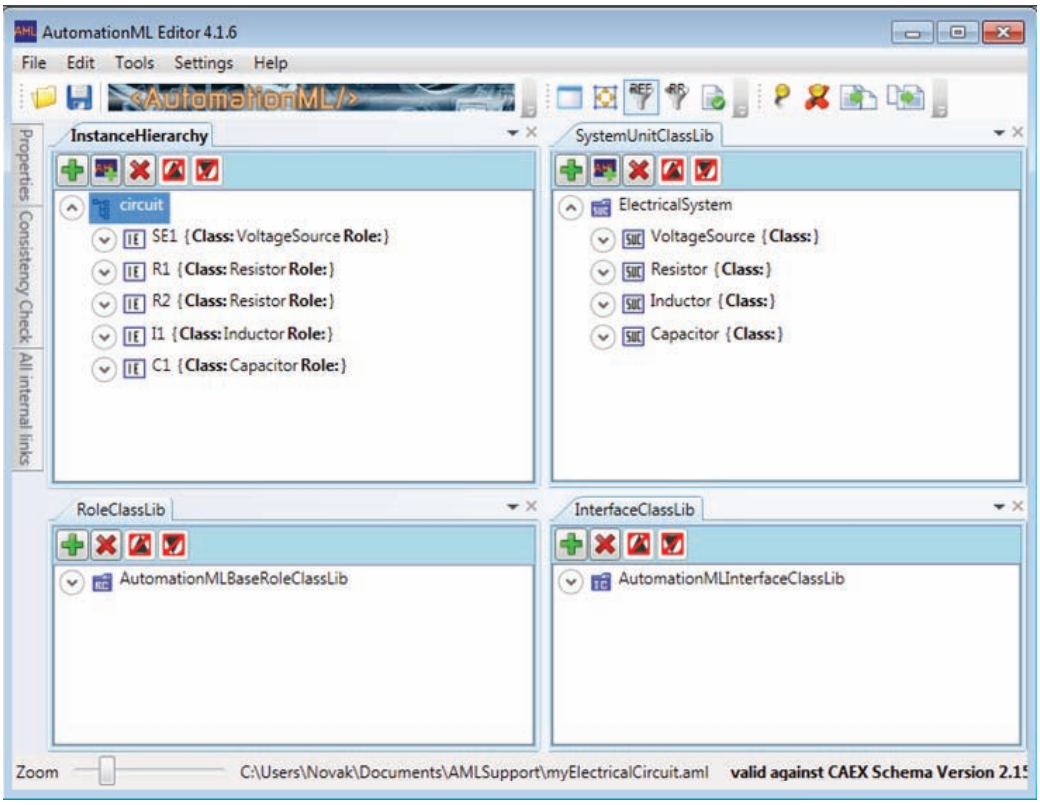

Figure 6.8: Screenshot of the AutomationML Editor with the system unit classes representing basic electrical system components and with the instance hierarchy describing the exemplary electrical circuit. Both hierarchies were created by the author of this thesis.

in this screenshot, as they have to be listed separately.

The created AutomationML file was read and parsed by the AutomationML Analyzer, whose screenshot is depicted in Fig. 6.9. We can see there the AutomationML file transferred into the ontology representation, which is visualized by the Pubby framework[4]. After applying the aforementioned SPARQL query, the representation of the electrical circuit is transformed to the automation ontology. The results are equivalent to the representation obtained with the Microsoft Visio connector, which has been already presented in Sec. 6.3.1.

6.4 Simulation Domain

At runtime, the interaction of simulations with the rest of the integrated system is provided by the engineering objects representing tags (i.e., samples assigned to tag names). Due to the complicated engineering phase of simulation models, their design process includes engineering objects covering devices and their representation in the simulation model, parameters, etc. In the knowledge base, the simulation domain includes annotations of simulation interfaces, structures of simulation models and annotations of parameters configuring simulation solvers.

[4]http://wifo5-03.informatik.uni-mannheim.de/pubby/

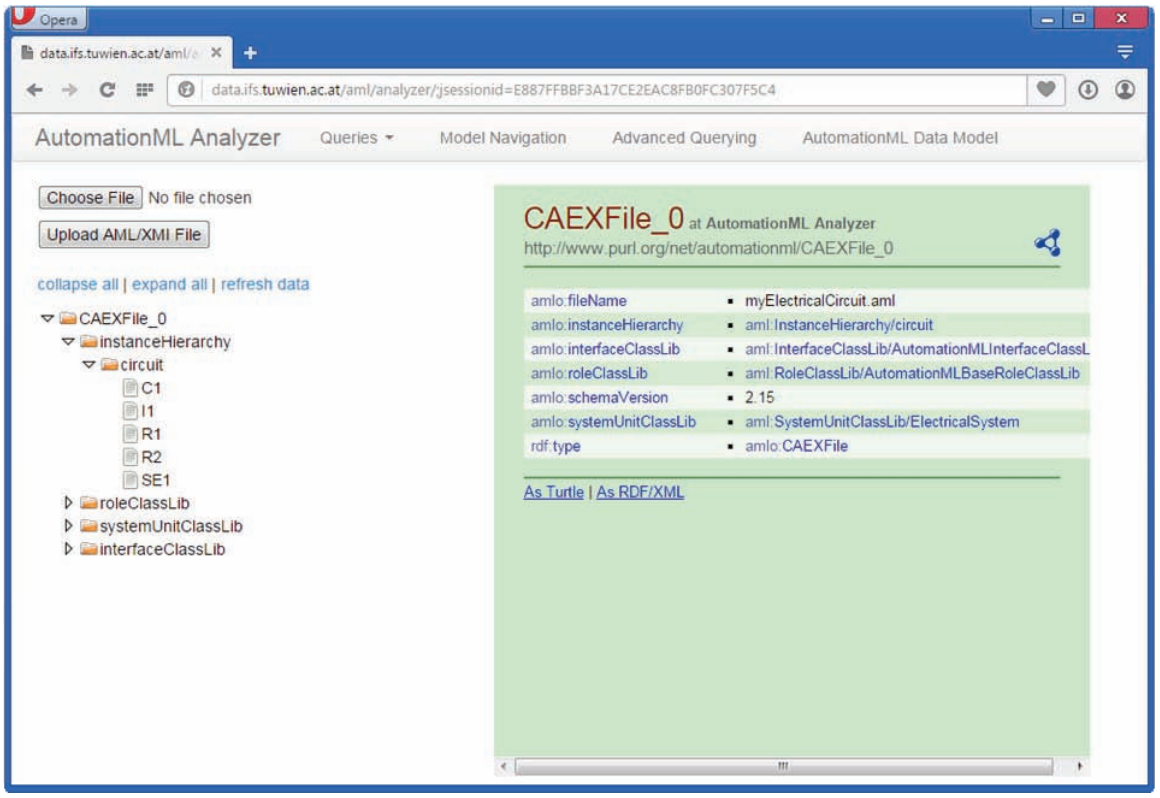

Figure 6.9: Screenshot of the AutomationML Analyzer, which was developed by Fajar J. Ekaputra [147].

6.4.1 MATLAB-Simulink Connector

One of the widespread simulation tools is MATLAB-Simulink[5]. It is a graphical signal-oriented simulation environment with various simulation solvers. It is popular due to a wide range of functionalities, rapid prototyping, and extension possibilities. The integration of MATLAB within the EngSB is basically based on the MATLAB C external API. This API is a part of MATLAB and includes methods for opening and closing the MATLAB engine, getting and setting variables and their values in MATLAB workspace and finally a method for a remote execution of commands. The used integration methods include loading of a simulation engine, loading of a simulation module, passing on all kinds of parameters and settings of a simulation solver, loading input tags, or exporting output tags to the EngSB.

During the first experiments, the simulation model creation was based on the coupled workflow of the ontology tool (as a neutral tool-independent platform) and a code in the simulation software (as a tool-specific code), which is MATLAB in this case. The workflow of the first implemented approach is depicted in Fig. 6.10. The MATLAB code queries the ontology tool for blocks as well as interconnections and the MATLAB-Simulink file is created via MATLAB API. Since the generator does not modify the source code of the simulation model directly, the solution is immune to changes of MATLAB-Simulink versions. However, the usage from the target user point of view is complicated and the code is difficult to be debugged and configured. Therefore, the second approach was used in the presented research as well.

[5]http://www.mathworks.com/products/simulink/

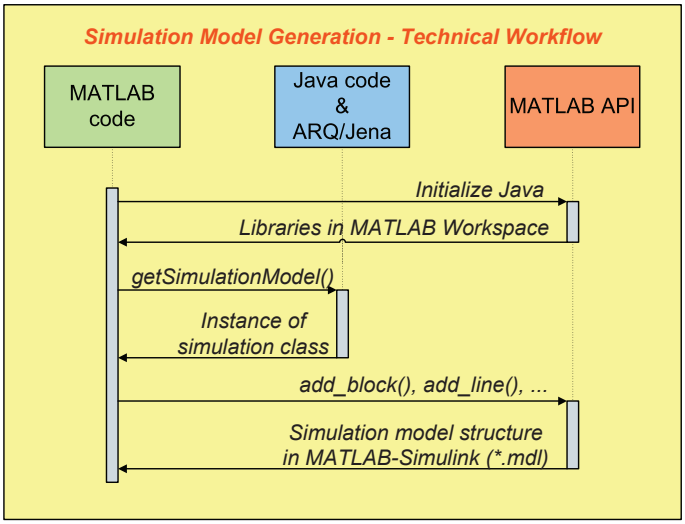

Figure 6.10: The initial version of the generation of simulation models in MATLAB-Simulink via MATLAB API.

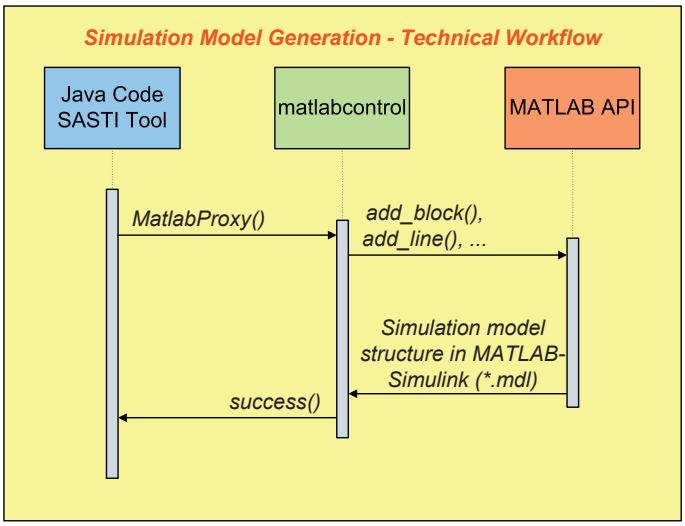

Figure 6.11: The latest version of the generation of simulation models in MATLAB-Simulink via MATLAB API.

In the second version of the MATLAB connector, the API called matlabcontrol[6] was used. It is user-friendly and efficient. The solution is depicted in Fig. 6.11 However, it is still considered as a prototype by the author of this thesis, as it is not guaranteed that this API will work in future versions of MATLAB. From this point of view, the use of APIs included and supported by MATLAB[7] is promising.

6.4.2 Other Simulation Tool Connectors

Another group of simulation tools is a set of equation-based simulators that are frequently based on the language Modelica[8]. It is a language for describing problems, which can be simulated by an external simulation solver, delivered typically in software packages such as OpenModelica[9] or Dymola[10]. The set of methods for integration of service-oriented simulations is the same as for MATLAB-Simulink, therefore the tools can be in one tool domain. This offers to switch between Modelica and MATLAB-Simulink models without changing the rest of the integrated automation system at all. However, the connector itself has not been implemented yet and it poses a future work topic.

6.5 SCADA System Domain

The SCADA systems are important parts of automation systems. They are not monolithic systems, but consist of various sub-tools, which are frequently relatively independent. This is the reason, why SCADA systems are abstracted with two tool domains in the proposed approach. The HMI part and the data acquisition part of SCADA systems are tackled separately in the proposed infrastructure and data models. In addition, a connection of multi-agent systems on the SCADA level of the automation pyramid is included as a part of this tool domain as well. The SCADA domain itself is considered as an abstract domain, i.e., it cannot be instantiated directly.

6.5.1 SCADA Systems – HMI Domain

The most visible part of SCADA systems are human machine interfaces (HMIs). They are intended to access runtime data by human operators, and to set actions and set-points by them. They typically visualize trends, current values and their limits; they can violate alarms when any value exceeds its required limits.

The core methods of the SCADA HMI tool domain are reading and writing data. By data, tags with one sample or a series of samples including time and value are understood. The accessed engineering objects for this tool domain are tags. The knowledge base is required to annotate the interfaces of the HMI, i.e., measured tags in the real plant and set-points to be configured from the HMI. As the internal structure of HMI screens can be very complex, we do not capture these internal elements in the knowledge base. On the contrary to the simulations, we do not expect to support their internal design. The specific

[6] https://code.google.com/p/matlabcontrol/
[7] http://www.mathworks.com/help/matlab/programming-interfaces-for-c-c-fortran-com.html
[8] https://www.modelica.org/
[9] https://www.openmodelica.org/
[10] http://www.3ds.com/products-services/catia/capabilities/systems-engineering/modelica-systems-simulation/dymola

SCADA HMIs, for which prototypal tool connectors were implemented in order to check their feasibility and efficiency, are ScadaBR[11] and Promotic[12]. The Promotic screenshot has been used in Fig. 1.2 and both tool connectors are addressed in details later in Sec. 6.5.2 respectively in Sec. 6.5.3.

The basic ideas of the SCADA integration effort were to evaluate the designed and implemented models and algorithms in terms of the relevant automation ontology concepts as well as interfaces of the ontology tool relevant for the SCADA integration. For the runtime integration of simulations and SCADA systems, an Apache-Tomcat-based light-weight tag integrator was developed, which was used as a mockup for the envisioned simulation integration framework.

6.5.2 ScadaBR Tool Connector

One of the main technical outcomes in this area is the connector for the ScadaBR environment. ScadaBR is an open-source SCADA–HMI system originating from Brazil, which inspired the acronym "BR". In Layman's terms, it is a graphical extension of the SCADA engine called Mango. ScadaBR is written in Java and it runs on Apache Tomcat. The HMI screens themselves are available via standard Web browsers. This SCADA system has been selected due to the following reasons: (i) it is an open-source software, hence it can be not only tested for free, but it can be also modified and integrated in desired ways, (ii) as it runs on the same technology as the ontology tool, it simplifies deployment and advertising (Apache Tomcat needs not be pre-installed, but the whole bundle of tools can be just copied), and finally (iii) ScadaBR proved that it can be used in industrial practice efficiently.

The ScadaBR connector is written in Java and it is prepared to be integrated within the simulation integration framework, namely into the EngSB as a connector implementing the SCADA-HMI domain connector for this particular tool. As ScadaBR offers a data exchange based on HTTP send and HTTP request methods, this type of data exchange was used due to its simplicity and transparency.

6.5.3 Promotic Tool Connector

The second outcome is a connector for Promotic system. Promotic is a SCADA HMI system available also in a freeware profile. The exemplary screenshot of this tool has been already depicted on the right-hand side of Fig. 1.2 in Sec. 1. Originally, Promotic is a commercial tool, but when a limited set of tags as well as a restricted set of plug-ins are used, it can be used for free. The vendor of this SCADA system, which is the Czech company Microsys, cooperates with the PLC producer Teco. A PLC Tecomat Foxtrot by Teco was available for the development and testing of the designed approaches and software for passive houses as part of use-case presented in Sec. 7.1. Summarizing the decision for the Promotic SCADA system, (i) it is a SCADA system enabling also a free use, which moreover belongs to the family of standard desktop-application SCADA systems, (ii) it has native plug-ins to read and write tag values from/to files, which can be used in the very same way as already implemented file connectors for the EngSB, and finally (iii) it has a native support for PLCs Tecomat, which is for free under specific conditions. Promotic is able to read and

[11] http://www.scadabr.com.br/
[12] www.promotic.eu/

write tag values from/to local files. The EngSB is equipped with a native file connector, which is suitable for this kind of integration.

6.5.4 SCADA Systems – Data Acquisition Domain

Data acquisition is the second part of the responsibility of a SCADA system. The goal of data acquisition is to read and to write process data, make them available on a server or to deliver them to HMIs. The runtime data are accessed via tags, which have a name, a timestamp, and a value. Engineering objects for data acquisition are thus tags and their values can differ according to tag sources. Tag values can be measured, simulated, or retrieved from a database in basic cases. The requirement on the knowledge base is to capture a mapping of data sources in order to simplify switching between working with real data and simulated data. The specific tools for data acquisition are especially implementations of OPC and OPC UA.

OPC UA is a universal multi-platform SOA-based standardized platform for process data sharing. Compared to the OPC classic, it does not require Microsoft Windows technologies for its running and it provides a security model. The author of this thesis co-operated on a prototypal implementation of an OPC UA connector, which is built on the top of the "freeopcua" stack[13]. This open-source stack is implemented in C++ and Python. The C++ version was used hence a bridging of Java and C++ languages is a part of the solution.

Since OPC UA has not been widespread in industry yet, the classic OPC technology has been considered as well. Its usage has been found out as feasible, but a connector for the OPC classic has not been implemented by the author of this thesis yet.

6.5.5 SCADA Systems – Multi-Agent System Domain

A large variety of current systems consist of relatively autonomous units. Such kinds of systems are frequently called systems of systems [72]. The problem of integrating autonomous units into one virtual system emerges in many areas such as smart grids, water distribution networks, or logistics. An important formal approach how to tackle these types of systems is a concept of multi-agent systems [171].

Although the multi-agent community has invested a lot of effort into a standardization of various properties and methods regarding software aspects of distributed and multi-agent systems, the multi-agent or holonic systems still have not been widely spread in industrial applications. One of the open problems is the integration of multi-agent systems with really used industrial automation systems. The goal of this section is to improve this situation and to bridge the gap between multi-agent systems responsible for coordination of autonomous system units and the SCADA level of automation systems especially in terms of simulations.

Due to the scope of this thesis, only the agents on the supervisory level of the industrial automation system are addressed. Goals of this kind of multi-agent systems are for example satisfying global requirements on the agent community including a maximization of a production in the given time frame, minimization of production costs, or minimization of waste or heat consumption. These goals should be reached under diverse constraints such as environmental limits, maximal operation time without technological downtimes, minimal/maximal number of entities running, maximal total power, etc. This topic was

[13] https://freeopcua.github.io

Table 6.1: Evaluation of the agent deployment alternatives.

Agents running on a server Centralization	**Agents on industrial PCs** Decentralization	**Agents running on PLCs** Decentralization
− High communication density	+ Hierarchical communication	+ Hierarchical communication
− Less safety	+ Safety	+ Safety
+ Vendor freedom	+ Vendor freedom	− PLC vendor dependent
+ Minimal HW requirements	− Required industrial PC on-site	+ No other HW
− Limited scalability	+ Good scalability	+ Good scalability

discussed in [119] and this section summarizes the achievements of this paper.

Software agents representing nodes of the distributed control systems of industrial plants can be deployed in the following three basic ways:

1. Agents running on a central server
2. Agents running on industrial PCs on-site
3. Agents running on PLCs on-site

These three approaches for the physical deployment of agents for industrial process control are compared and evaluated in Tab. 6.1. The term "communication density" denotes the communication traffic between the server and the DCS nodes in the aforementioned table. The term "safety" is related to the hierarchy of data exchange and threats within public networks. The term "vendor freedom" means that the solution is not restricted to specific hardware vendors. Finally, the "scalability" is considered from the number of DCS nodes point of view, i.e., it expresses how many nodes can be added into the integrated system. Summarizing the pros and cons of each architecture, the most promising way is the alternative utilizing industrial PCs on the agent level, although an additional hardware is required.

The entire proposed architecture is depicted in Fig. 6.12. The HMI can be accessed via various protocols based on the secured HTTP protocol and Web services. The central servers that are needed for connecting the multi-agent system with SCADA systems as well as for running the directory facilitator (i.e., yellow pages) and agent management system (i.e., white pages) are duplicated in order to support back-up. When the main server is not available, the backup server overtakes all tasks, which increases safety and mitigates risks of the single point of failure. The communication between the servers and the distributed agents deployed on industrial PCs together with OPC servers is solved by two independent protocols within the same communication channel. Either a classic OPC or OPC UA are used for SCADA-relevant process data monitoring and reporting, whereas agent communication is solved by means of remote method invocation provided by an agent platform. Due to interoperability and cost reasons, various protocols for the communication between PLCs and agents can be supported as it is depicted in the figure, nevertheless, OPC DA or OPC UA are the preferred ways.

6.6 Processes for Simulation Design and Integration

Whereas the previous sections addressed the integration from the infrastructure perspective, this section is focused on processes how to utilize and to operate this infrastructure to save

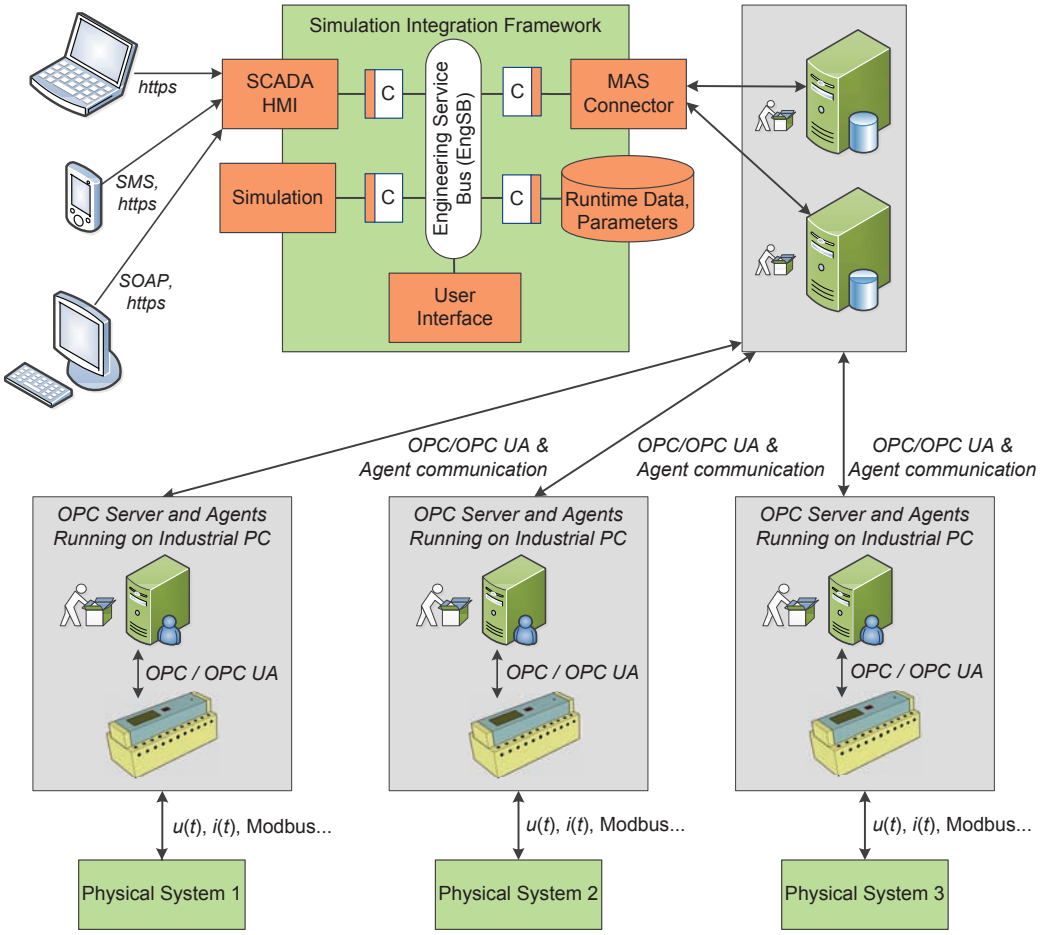

Figure 6.12: Proposed architecture of the supervisory multi-agent control system for smart distributed environments.

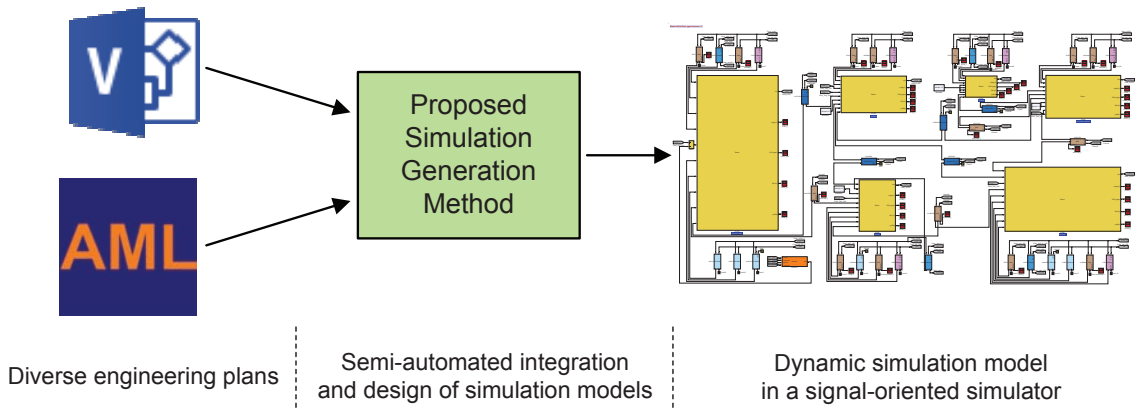

Figure 6.13: Improved design of simulations based on integrated engineering knowledge.

time and human effort for simulation model design and integration.

From the simulation modeling point of view, the approach proposed in this thesis is shown in Fig. 6.13. The figure depicts that industrial plant models represented either in the Microsoft Visio tool or in the AutomationML data format are processed and transformed by the methods proposed in this thesis and a simulation model in a signal-oriented simulator is generated as the outcome of these methods. From the perspective of the processes within the integration infrastructure, which implements the aforementioned design-phase vision technically, the simulation model design is focused on aggregating tool domain knowledge into the knowledge base. It means that these processes are based on instantiating ontology classes in the knowledge base (respectively in the automation ontology that implements the data model of the knowledge base) to capture available engineering knowledge. Such design-time processes incorporated into the EngSB-based infrastructure are schematically expressed in Fig. 6.14. The figure depicts that engineering data are aggregated in the knowledge base and such data is consequently used to support the design of simulation models as well as the configuration of the infrastructure built over the EngSB for operation at simulations at runtime.

The simulation model should be consequently integrated with other software parts of the automation system as it has been introduced in Fig. 1.2. On the contrary to the aforementioned design-time processes related to simulation models, the runtime processes are based on transferring values of tags between stakeholders within the EngSB. The aforementioned benefit of the EngSB is that both types of processes can utilize the same technical infrastructure seamlessly. While the design-time processes based on the extended bond graphs, automation ontology, and AutomationML and Microsoft Visio connectors have been already addressed in the previous sections, the runtime integration processes are related to the EngSB only and these processes are described in the following section in details.

6.7 Integration of Simulations and SCADA Systems from the Process Perspective

The runtime integration covers two sets of tasks: (i) routing between stakeholders based on simulation workflows and data source specifications, and (ii) translation between variable

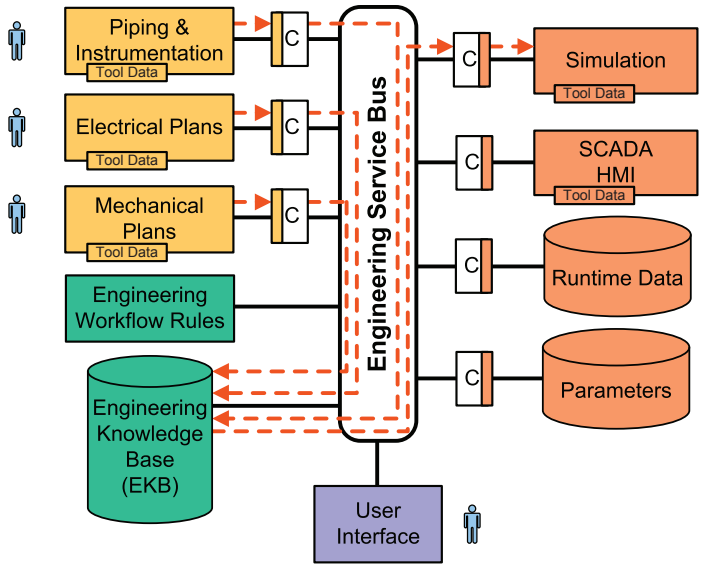

Figure 6.14: Design-time processes of the simulation model life-cycle. The dash-dot arrows mean the data exchange between stakeholders.

names and tag names, which is done on the tool connector level. The core issue is the knowledge support for the improved runtime integration that can be used for model-driven configuration of the integration infrastructure.

The basic workflow for the runtime integration of simulation models is depicted in Fig. 6.15. The integration approach adopts principles of distributed systems and distinguishes local tool data models and global data models. For the integration, relationships between tags are crucial. The workflow assumes to define local tag names in each specific tool or technology, such as in OPC UA or OPC classic, in a simulation model, in an HMI screen or in other industrial automation tools. In the second step, there should be selected a representation of each tag name that will be used in the EngSB environment and which is considered as a global tag name representation. Such an attitude to tags enables to support legacy artifacts as well. The process of mapping tags can be supported by algorithms preferring the real OPC UA or OPC classic tag names, simulation inputs/outputs, but a user has to be allowed to modify the names manually. The annotation of the tag names is stored in the automation ontology as well as the mappings between global tag names and local tag names. In simple cases, when local tag names are derived from real tags by a pre-defined prefix or suffix, a merging algorithm can be able to find mappings between local and global tag representations automatically. In some cases, it can work semi-automatically and for example to group the tags according to devices, locations or other pieces of information. This can be done easily in cases when tag names satisfy a standardized naming convention including these pieces of information, such as according to the IEC 81346 standard [120].

A particular example of the tag routing is depicted in Fig. 6.16. It schematically shows how tag values are transferred between stakeholders within the EngSB-based infrastructure. Runtime processes utilize a version-control system to improve traceability of experiments in terms of utilized settings and data as it has been introduced in the case of the simulation workflow depicted in Fig. 5.24.

The automation ontology formalizes the mappings between local and global tags as well as between tags and variables. In the local context, a lot of variables frequently exist,

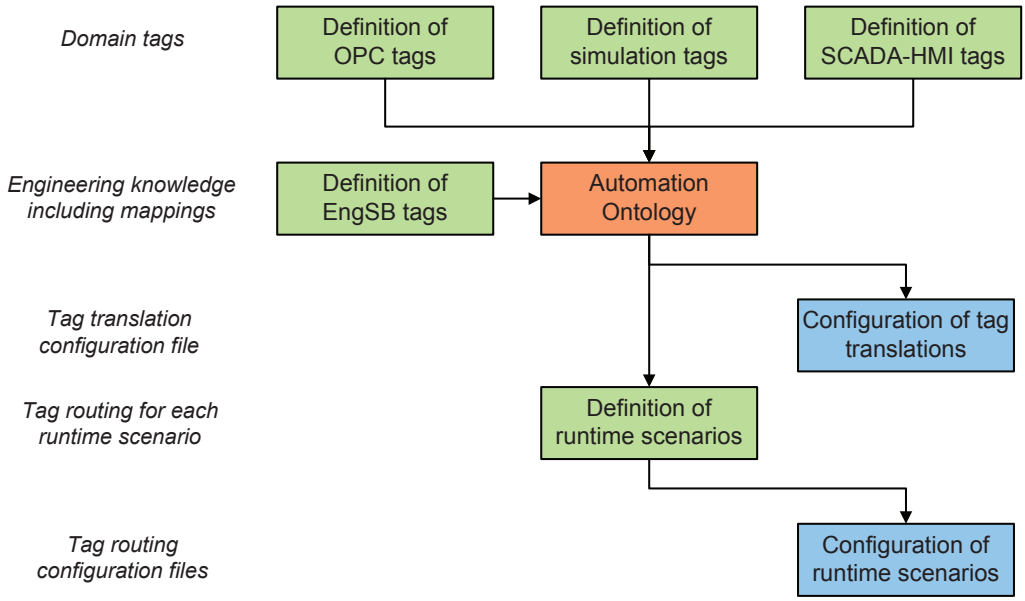

Figure 6.15: Conceptual workflow for integration of simulation models at automation system runtime.

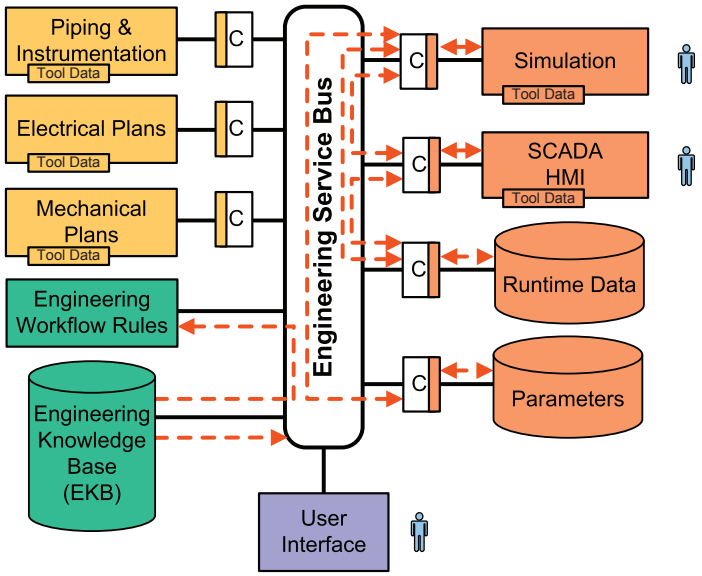

Figure 6.16: Runtime processes for integration of simulations and SCADA level of industrial automation and control systems.

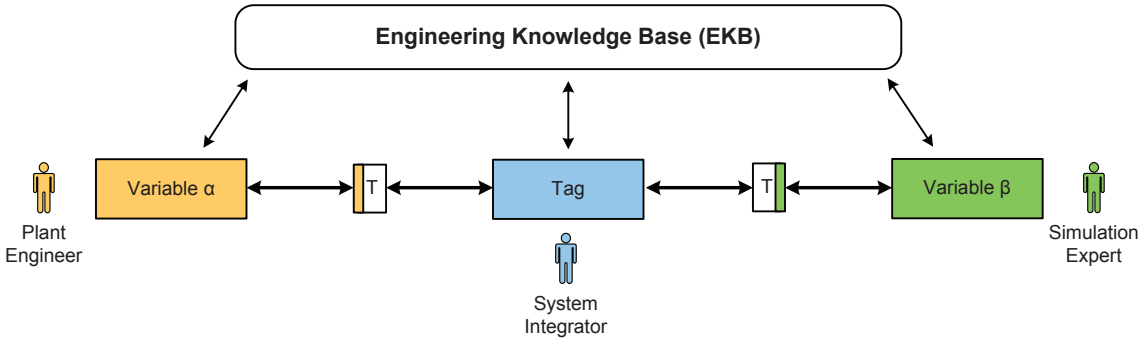

Figure 6.17: Translation between variables and tags.

but only a subset of these variables is labeled with local tag names. From the integration perspective, local tags are externally visible variables. The translation between variables and tags is realized on the connector level of the integrating infrastructure. To illustrate this translation on a practical example, variables are stored in MATLAB workspace in case of a single simulation module, whereas when such variables are either visualized in a plotted graph or exported/imported to/from the EngSB, the name of the variable/tag is changed. This approach guarantees a very high degree of the reuse even for legacy models, sets of models, and tools, because local representations are overshadowed by common definitions on the top of them. The tag routing is defined by simulation workflows, which are expressed in the automation ontology as well. Although the name of the simulation workflow can be confusing as it also covers routing for example among HMIs and OPC stacks, it was defined during the cooperation with industrial partners consensually. The translation between variables and tags is demonstrated in Fig. 6.17.

The aforementioned connector to Microsoft Visio supports not only capturing engineering data into the ontology, but also visualizing the data stored in the ontology in a form of XML hierarchies intended for the configuration of the simulation integration framework. Since the screenshots showing this are rather technical issues intended for evaluation of the proposed method only, they are included in Appendix C (i.e., Fig C.2 and Fig. C.3).

An important issue is the problem of timing and synchronization of the tag exchange. Since simulators are strongly influenced by numerical stability of the model itself as well as required relative and absolute precision, the simulation time passes in different time steps during continuous-time simulations frequently. The whole model is performed either synchronously with the real time, faster, or exceptionally slower. Such timing issues are strongly project-dependent, hence they are not addressed in this thesis.

Chapter 7

Use-Cases and Experiments

This section summarizes two use-cases from different engineering disciplines to illustrate the proposed approach in practice and to emphasize the mechatronic abilities of the proposed methods. These use-cases are also utilized as basic test-beds to estimate the efficiency and effectiveness of the proposed methodology.

7.1 Passive House Simulation Use-Case

Simulation models for passive houses are useful test-beds for testing and tuning building automation and control systems. The goal of this use-case is to evaluate the proposed simulation model design method in the case of creating a simulation model for a particular passive house semi-automatically. In addition, the goal is to verify the simulation library and created models with the use of measured data and to fine-tune model parameters. This use-case also addresses the data acquisition in the real testing house built near Prague, which is necessary for comparison of simulation results and measured courses of variables form the real passive house.

7.1.1 Motivation for the Passive House Simulation Use-Case

The building sector offers the highest potential for energy savings in Europe, see for example Directive 2010/31/EU [40] for more details. Although in the past, buildings have been in scope of civil engineering only, the passive or especially zero-energy houses require proper control of building systems and devices that belong to the control engineering discipline. This kind of buildings is thus a good example of mechatronic systems. The design and fine-tuning of building control systems is a complex task, which has to cover a large variety of heterogeneous hardware platforms and software tools. The optimization of control algorithms requires series of experiments, hence the simulation model is a useful test-bed, performing experiments much faster and guaranteeing initial conditions to be repeatable.

By now, passive house control has reflected inhabitants' requirements only, but the emerging area of smart grids [166], referred as electrical distribution network of the future, brings further challenges. Houses will no longer be autonomous entities optimizing their own goals only, but they will participate in achieving common goals of the grid as well. This promising area requires simulation models, which would be capable to simulate the behavior and to test the functionality of the whole smart grid as well as the particular houses.

7.1.2 Passive House Standard

Although the proposed approach is general and can be used for any kind of a house, this use-case is focused on passive houses. The reasons are (i) the author of the thesis implemented in the previous work a simulation library suited for air-heated houses with a low energy consumption, and (ii) having measured data from a real passive house.

Several civil-engineering standards for passive houses exist world-wide, however, their common points are the following requirements [137]:

- Annual required energy for heating not exceeding 15 $\mathrm{kWh \cdot m^{-2}}$
- Annual combined primary energy consumption not exceeding 120 $\mathrm{kWh \cdot m^{-2}}$
- Building envelope air change rate under pressure difference 50 Pa (n_{50}) not exceeding 0.6 $\mathrm{h^{-1}}$

To meet these requirements, there are many widely accepted rules, which are not explicitly involved in the definition of a passive house. Passive houses are continuously supplied with fresh air via a ventilating system with high-efficient heat recovery. To minimize the air change rate through the building envelope (i.e., air flows without heat recovery), it has to be sealed well.

7.1.3 Measuring and Control in Passive Houses

To evaluate the presented approach and to estimate the scalability of investigated methods for future use on real-world industrial problems, a use-case following results from former projects was selected. This use-case, which was presented in [125], deals with measuring physical behavior of passive houses and fine-tuning a dynamic simulation model approximating their operation. Thus the overall goal of this use-case is to improve simulation and control of environmental parameters of passive houses by means of big data.

Operation of residential buildings is fundamentally characterized by indoor air quality and energy consumption. Indoor air quality can be represented by environmental parameters, which are in this case the following variables:

- Temperature;
- Carbon dioxide (CO_2) concentration;
- Relative humidity (RH);
- Air pressure.

7.1.4 Simulation Modeling of Houses

Although many building simulation models have been designed over the world, they do not support the usage of advanced control techniques (such as fuzzy control or model predictive control) in the most cases. Compatibility and support for such modern control techniques led to implementation of a simulation library in MATLAB-Simulink, which perfectly fits for these and other kinds of techniques for control and data processing.

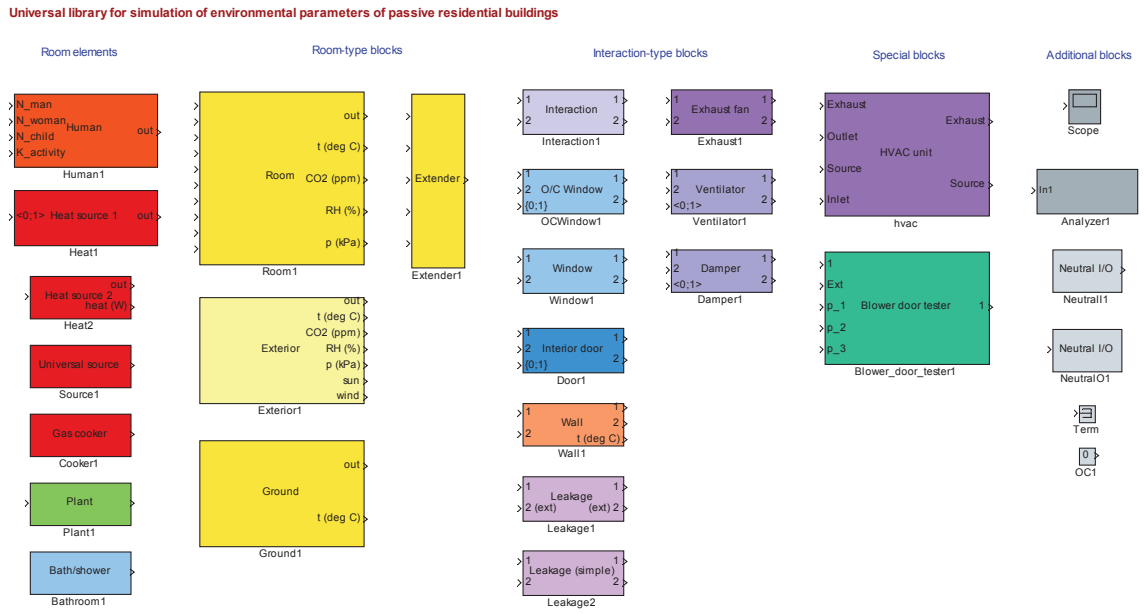

Figure 7.1: Building Simulation Library (bldsimlib) for simulation modeling of passive houses, which was designed by the author of this thesis.

The utilized simulation library has been designed and implemented by the author of the thesis already in his previous work [114]. The early version of this "Building Simulation Library" ("bldsimlib") was disseminated in [115], an improved version was presented in [117], which includes considerations about HVAC (i.e., heating, ventilation, and air conditioning) control based on fuzzy theory. The latest version of the library was presented in [125].

The library simulates all of the four physical quantities called environmental parameters stated in Sec. 7.1.3. The model is continuous-time and dynamic. The simulation time can be arbitrary, nevertheless, most of the experiments simulate a one-year-period. Since the library is intended to support testing and fine-tuning of control systems, it adopts a so-called multi-zone model, i.e., a building consisting of multiple zones and each zone represents a real room. This assumption implies that the model uses only one representing value of the environmental parameters for each room.

The two types of interfaces of simulation blocks are used in this simulation library. In compliance with the (extended) bond graph theory, both types of signals representing (i) a flow of the environmental variables and (ii) a state of environmental parameters. In the former case, the signal type corresponds to the flow in terms of bond graphs and this *flow* is a vector having elements: heat flow and molar flows of air, CO_2, and H_2O. In the latter case, the signal type corresponds to to the effort and to the integrated flow in terms of bond graphs and this *state* is a vector of temperature, amounts of substance of air, CO_2, and H_2O, and air pressure. The simulation library involves five types of simulation blocks: so-called "room element" blocks, "room-type" blocks, "interaction-type" blocks, "special" blocks, and supporting "additional" blocks, as it is depicted in Fig. 7.1. These sets of blocks differ not only in the functionality, but especially in input and output interfaces.

The *room-type blocks* are the simulation blocks representing a room, an exterior, and a ground. They have thermal flow and molar flows as inputs; and room temperature, CO_2 and H_2O amounts of substance, and air pressure as outputs. These input and output signal vectors are implemented as multiplexed signals, hence they seem just as one signal

in Fig. 7.1. In addition, the library blocks include also variables in human-comprehensible units and scales as outputs, such as temperature scaled in [^{o}C], RH scaled in [%], or air pressure scaled in [kPa]. The exterior and ground blocks have the similar interfaces as room blocks, but they specify boundary conditions of the simulation.

Fundamental physical laws describing room-type blocks can be summarized as follows:

$$T = \frac{1}{C_t} \int_{t_0}^{t} Q_t(\tau)\, \mathrm{d}\tau \tag{7.1}$$

$$n = \int_{t_0}^{t} Q_n(\tau)\, \mathrm{d}\tau \tag{7.2}$$

where T denotes the temperature, C_t denotes the thermal capacity, Q_t is the heat flow, n is the amount of substance, Q_n denotes molar flows, and τ represents time. The output vector of the room block also includes air pressure p. The reason is that the volumetric flows between the rooms are given by their pressure difference, hence the pressure output is crucial for defining the direction of the flow and its value. The pressure is calculated with the ideal gas equation, where R denotes the gas constant and V is the volume of the room:

$$p = \frac{n_{air} R T}{V} \tag{7.3}$$

The *interaction-type blocks* basically connect two rooms. Since ventilation ducts are considered as a special kind of rooms having a very low volume, the interaction-type blocks connect air ducts and a room as well. The simulation library includes the following interaction-type blocks: window, door, wall, leakage, ventilator, damper, and exhaust fan. They have states of the two connected rooms as inputs; and flows into the two rooms as outputs. Some of the blocks also include input signal ranged between 0 and 1 representing openness. The majority of the interaction-type blocks is based on the pressure difference between zones:

$$Q_V = q_0 S (\Delta p)^{0.67} \tag{7.4}$$

where Q_V is the volumetric flow, q_0 is the specific flow constant, S is the area of air permeability, and Δp is the pressure difference. The volumetric flow given by Eq. 7.4 is re-calculated to the molar flow Q_n with the use of Eq. 7.3:

$$Q_n = \frac{p Q_V}{RT} \tag{7.5}$$

The heat flow Q_t between the rooms is given by the following equation:

$$Q_t = Q_n c_m \Delta T \tag{7.6}$$

where c_m is a molar thermal capacity of transferred air. Conductive transfer of heat, which occurs, e.g., in the case of closed windows is defined by Eq. 7.7, where U is thermal transmittance, S is area, and ΔT is a difference of temperatures:

$$Q_t = U S \Delta T \tag{7.7}$$

Finally, the simulation library includes *room elements* affecting one room only (including the human, heat source, universal source, gas cooker, plant and bath/shower) and *special*

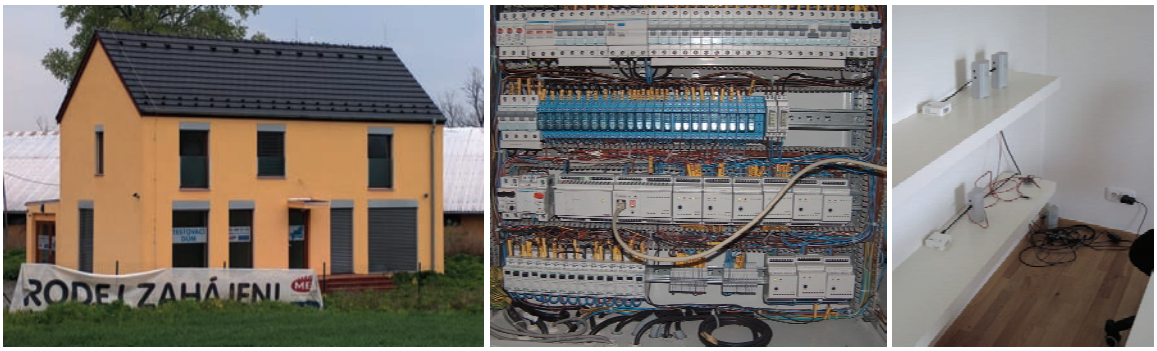

Figure 7.2: Illustrative views on the experimental passive house.

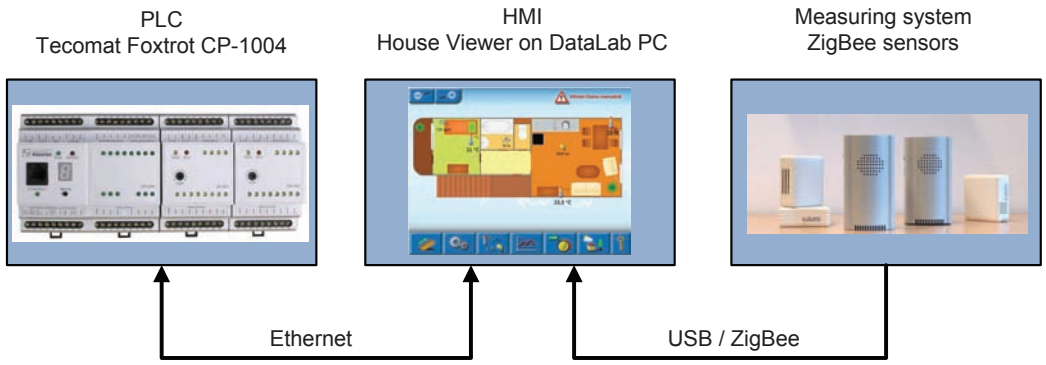

Figure 7.3: Deployment of the real hardware for measuring, control, and data logging in the experimental passive house.

blocks that are the HVAC unit, approximates a heat exchanger including ventilators, and a blower door tester. *Additional blocks* pose rather a technical issue only.

7.1.5 Experimental Passive House

To acquire real data and experiences as well as to verify a simulation model and advanced process control algorithms, an experimental passive house was built near Prague, in Úvaly–Hostín. The exterior of this two-floor wooden passive family house is depicted in Fig. 7.2 on the left-hand side.

The author's work did not cover modeling and simulation tasks only, but also deployment of the measurement and control equipment to the house as well as technical and development issues, such as hardware configuration or programming of the installed PLC according to IEC 61131 standard, programming of the industrial PC in C# to visualize and to save runtime data of the building, deployment, maintenance, and repairing of sensors, or supervision of the instrumentation and wiring of the electrical cabinet. The final solution of the electrical cabinet of the house is depicted in the middle part of Fig. 7.2.

The architecture of the used physical deployment is depicted in Fig. 7.3. The house was controlled by the PLC Tecomat Foxtrot CP-1004[1], whose data were read via an ethernet cable by an industrial PC with a touch screen. The industrial PC hosted the software called House Viewer, which was used as the SCADA HMI for this house. The communication between the PLC and the industrial PC was based on the protocol EPSNET[2]. The author

[1] http://www.tecomat.com/kategorie-308-tecomat-foxtrot.html
[2] http://www.tecomat.com/wpimages/other/DOCS/eng/TXV00403_02_Comm_Serial32_en.pdf

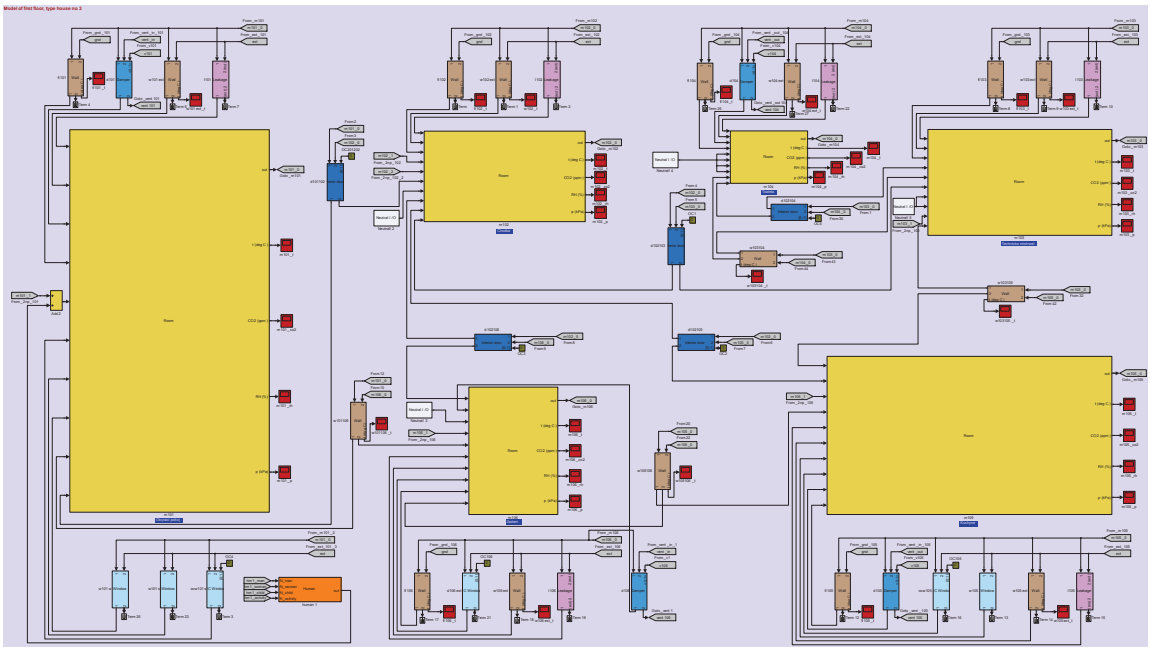

Figure 7.4: Simulation model for the second floor of the experimental passive house to illustrate the complexity of the simulation model.

of this thesis implemented a communication module into the House Viewer software, which realizes the data synchronization. The same industrial PC was also used to download and to save interior data, which were measured by a set of wireless sensors. As an example, several sensors of this equipment are depicted on the right-hand side of Fig. 7.2. These sensors communicated according to the ZigBee standard with the central element of the measurement equipment, which was connected to the industrial PC via a standard USB connection.

The simulation model for the experimental passive house was developed by the author of this thesis mainly in his prior work [116]. It is a multi-layer simulation model in MATLAB-Simulink. To illustrate its complexity in terms of number of elements and signal connections, the model of the first floor is shown in Fig. 7.4. It depicts simulation blocks representing rooms, walls, windows, as well as their signal-oriented interconnections. This simulation model was tested and verified in the real passive house and its results were found out as satisfactory for intended purposes of supporting the development and testing of building automation and control systems for passive houses.

A long-term measurement of the environmental parameters as well as the total and heating electricity consumptions and boundary conditions of the passive house was done by the author in the frame of this thesis. Measuring of environmental parameters and energy consumption on real buildings is a long-term process as time-constants of residential buildings are very long and experiments are not repeatable due to changing weather as well as initial state conditions. The experimental passive house was equipped with a lot of sensors in all rooms (excluding a toilet and an entry foyer), the HVAC unit was equipped with power meters and there were installed outdoor wind velocity and direction, sunlight intensity, and temperature sensors. In total, the experimental passive house was equipped with 36 sensors of the following types:

1. Room temperature and relative humidity – coupled sensor (13 sensors)

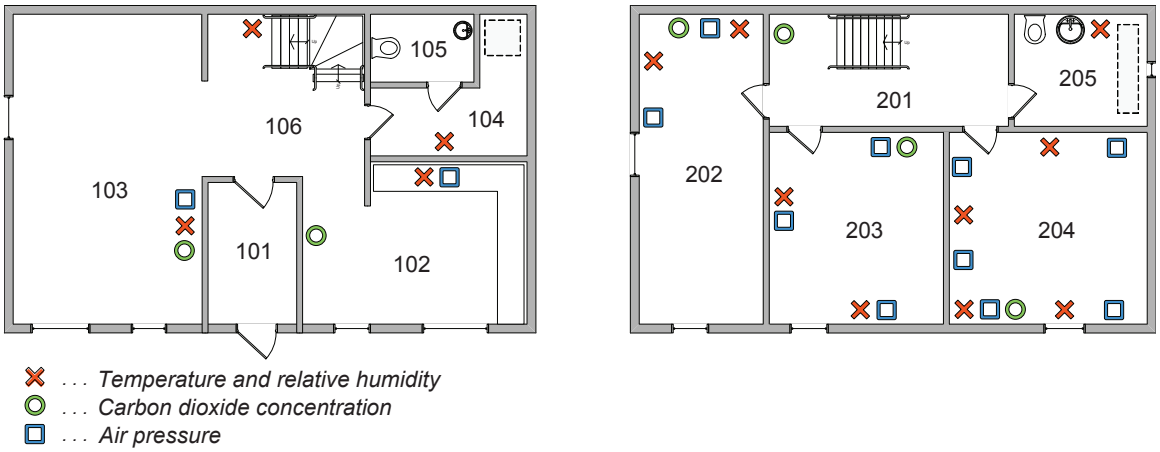

Figure 7.5: A simplified floor-plan of the experimental passive house depicts the positions of the interior sensors and their types.

2. Room carbon dioxide concentration (16 sensors)
3. Room pressure (12 sensors)
4. Heating energy (1 sensor)
5. Ventilation energy (1 sensor)
6. Outdoor temperature, relative humidity and air pressure – coupled sensor (1 sensor)
7. Sunlight intensity (1 sensor)
8. Wind speed and direction (1 sensor)

The final positions of the installed interior sensors are depicted in Fig. 7.5, where the ground floor is located on the left-hand side of the figure, whereas the upper floor is on the right-hand side. The sensor positioning was selected after testing diverse arrangements of sensors and finding appropriate settings. Sensors for carbon dioxide concentration are located in all rooms with expected long-term presence of humans, such as a living room 103, a kitchen 102, and bed rooms 202, 203, and 204. Coupled temperature and relative humidity sensors are located in all rooms excepting a toilet 105 and an entry foyer, where it was not technically possible. Air pressure sensors are in most of the rooms. They were intended to monitor air pressure, which corresponds to air flows between rooms, however, due to the limited resolution of the detector, this task was satisfied only partially. In the further text, we will focus on courses of temperature and carbon dioxide concentration, which are the most important variables for data analysis.

An example of real measured time-series and their comparison with dynamically simulated courses is depicted in Fig. 7.6, showing the experiment of ventilating polluted indoor air. It shows the impact of machinery ventilation on the indoor temperature and air quality, which is indicated by the CO_2 concentration. Fig. 7.6 depicts a situation in a bed room 204, having one window, sunblind up, and ventilation on. The room has been just left by a group of people.

Analysis of sensor time-series needed pre-processing and filtering of outliers mainly in case of carbon dioxide concentration. The results of the preliminary data processing were

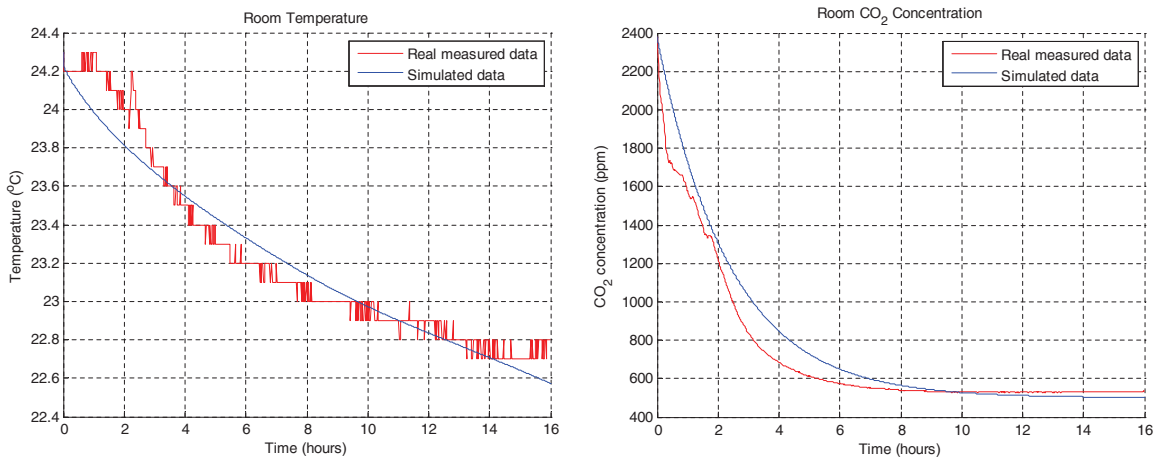

Figure 7.6: Comparison of measured and simulated temperature and carbon dioxide concentration time-series.

discussed in [76]. Among others, it proved that the simulation model of the ventilation system having an impact on the CO_2 concentration is in compliance with the measured data from the experimental passive house.

7.1.6 Semi-Automated Design of Simulation Models for a Passive House

Lessons learned from practice showed that it is very time-consuming and error-prone to create the whole simulation model manually. When a house is being built, there exist a lot of CAD drawings, depicting the floor plans, views, etc. By now, it has been necessary to take these plans and to copy simulation blocks from the library to the simulation model file and to interconnect the simulation blocks according to specified rules manually. It is very beneficial to automate this repeating work and to create the simulation model structure semi-automatically.

This use-case poses the simulation model design scenario when the simulation library is available. To create a simulation model automatically based on the available simulation library blocks and existing engineering plans (e.g., AutoCAD, MS Visio, or other electronic drawing), the design workflow is depicted in Fig. 7.7 and it was disseminated in [125] after initial considerations presented in [124]. Both papers described that assembling a simulation model structure can be done automatically. However, values of simulation parameters have to be entered manually by a user thus the entire simulation generation process is referred as semi-automatical. In future work, parameter values could be extracted from a widely used tool PHPP [136] and from material databases.

Following the workflow depicted in Fig. 7.7 in details, the simulation generation method starts with the output file of the House Builder software [65]. The House Builder tool was originally intended to prepare the configuration for the HMI tool House Viewer in a user-friendly way. Both House Builder and House Viewer software were implemented at Dept. of Cybernetics, FEE, CTU in Prague. The XML file being the output of House Builder software is interpreted as a kind of a CAD drawing. The parser, which was implemented in Java, populates the ontology with individuals representing real rooms, walls, and the exterior. As the automation ontology involves annotations of the simulation library and relationships of the real devices and simulation blocks defined by the object property

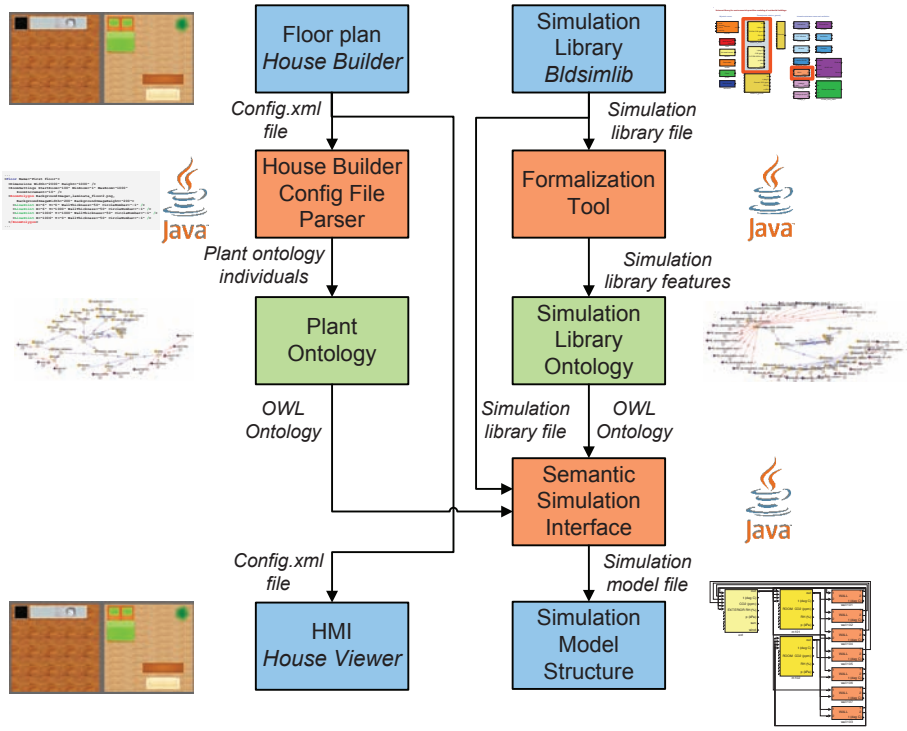

Figure 7.7: The scenario of the semi-automated design of the simulation model for a particular passive house.

"simulates", the ontology tool creates the simulation structure.

The results of this approach are demonstrated in Fig. 7.8. The left-hand side depicts the floor plan of a simplified passive house created in the House Builder tool. On the right-hand side, the generated simulation model for this house is shown. The layout of the signal-wire routing was changed manually in order to make all signals visible. The extension of the algorithm to improve the layout is a future work topic, as well as merging the blocks simulating the same thing (such as walls between rooms and the exterior, which are now implemented as three independent blocks, but which could be merged into just one simulation block).

When the floor plan is changed (i.e., it is re-drawn in the House Builder software), the simulation model can be easily re-generated. Hence users can create several versions of the house in the project phase easily and to compare their behaviors in the simulated world. An important aspect of this result is that dynamic simulation models can be created by users that are not skilled in dynamic simulation engineering, such as architects or civil engineers designing residential buildings. In the implementation done within this thesis, the tool support includes House Builder only, but for example Microsoft Visio could be added with the developed connector easily.

7.1.7 Lessons Learned and Evaluation of the Passive House Use-Case

The passive house use-case was the author's first complex use-case he worked on. The use-case proves that simulation models can be designed even by non-experts in process simulation as well. Although the passive house use-case has brought a lot of pieces of inspiration and ideas, its complexity from the modeling and simulation perspectives is rather simple. The first reason is that passive houses include a relatively small number of elements

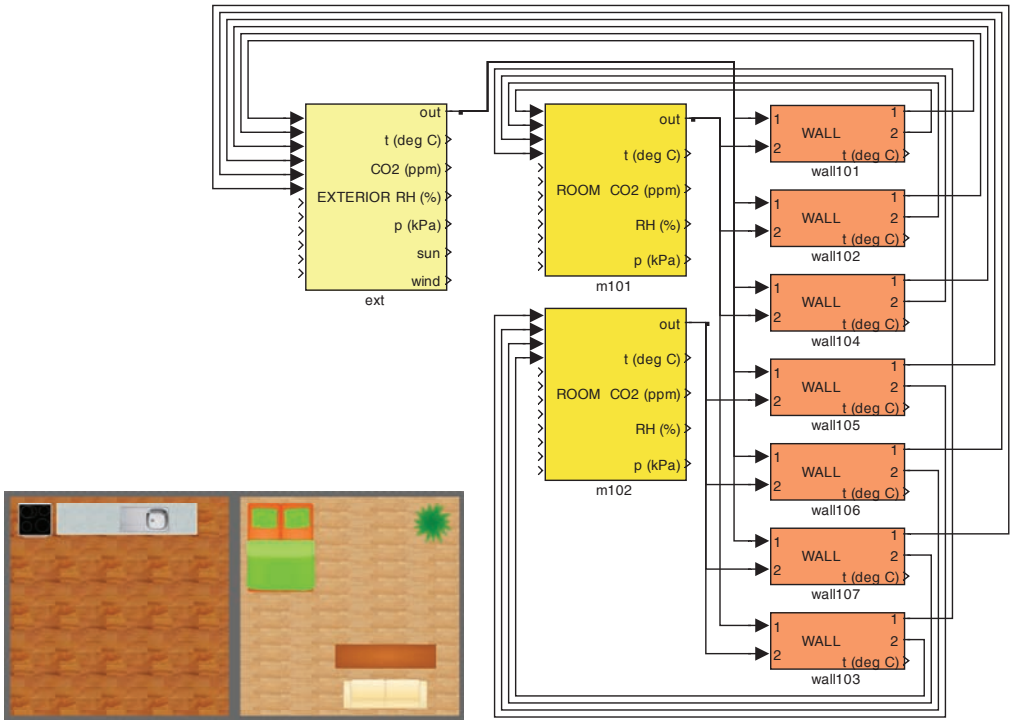

Figure 7.8: An exemplary two-room passive house floor plan in the House Builder tool on the bottom-left side and the automatically generated simulation model structure in MATLAB-Simulink on the right-hand side.

compared to large-scale systems from industrial practice. The second reason is that the structuring of the system is significantly simplified by the fact that there is no inductance in thermal models used in building modeling and simulation, see Tab. 2.1.

The lack of inductance is the root for much easier structuring of the simulation model than is typical in other engineering disciplines in case of models for systems of similar sizes (in terms of the number of devices, interconnections, etc.). Junctions can be represented in other ways as they need not be modeled explicitly, which leads to an easier interpretation of the simulation model structure by humans. On the other hand, parallel connections realized on the room block level required a more complicated representation of power bonds in the automation ontology than in the case of explicit representations of all junctions.

Although the scenario proposed in Fig. 7.7 reached good results and proved the effectiveness and efficiency of the proposed simulation model generation method, the lack of inductance elements was found as an important factor that affects the complexity of the simulation model engineering. In other words, the passive house use-case belongs to the family of simulation models whose semi-automated generation is not as difficult as in general. In addition, the mapping between physical objects and simulation blocks is 1:1, which poses the simple case. Therefore, the hydraulic use-case was added and addressed as well in order to verify and to evaluate the proposed method on a system type including both types of energy accumulators and various mappings between devices and simulation blocks. Nevertheless, the passive house use-case shows that the proposed methods can be used on various types of mechatronic systems, including not only the industrial ones.

7.2 Hydraulic Network Use-Case

The main goal of this use-case is to demonstrate the improved simulation model design on a more complex system than are the passive houses. In this use-case, the mapping between real devices and adequate simulation blocks is $1 : n$, hence it shows how to select the appropriate blocks for each component. Moreover, the nature of hydraulic systems does not enable to merge junctions with other artifacts, hence the explicitly modeled junctions are shown in this use-case as well.

Hydraulic systems are widespread in many types of industrial systems. They are used as parts of water supply, power plants, chemical and petroleum industries, liquefied gas service, and other areas of applications [82]. In this section, an educational hydraulic model, which is located in the Odo Struger Laboratory at the Automation and Control Institute[3] (ACIN) of the Vienna University of Technology[4], is addressed. The laboratory tank model is depicted in Fig. 7.9, showing the piping and instrumentation diagram (P&ID) of the most important subsystem of this laboratory test bench. This use-case can be considered as a laboratory-scaled educational example of water supply systems. Mathematical-physical description of hydraulic systems is introduced and summarized in numerous literature, for example in [1] or [103]. Due to low pressures and flows in the system, the flows can be considered as laminar for simplicity reasons. The proposed method addresses the following challenges related to the simulation model design and integration tasks:

1. Design of a simulation model with extended bond graphs

   The support for the simulation model engineering is the crucial issue for virtual commissioning of the real plant, its control systems, and for operator training. The use of the proposed extended bond-graph theory is shown in this use-case. The simplicity and fastness of the usage proves the efficiency of the proposed method.

2. List of simulation parameters

   To configure the behavior of simulation blocks, their parameters have to be set up. The implemented connectors for MS Visio and AutomationML are used to grab the parameter values and to capture them in the automation ontology. The parameter values are consequently queried from the ontology and used as a main part of the simulation model configuration.

3. List of OPC tags

   The OPC server is an application acquiring data from plant devices and sending control actions to the devices. The configuration of this basic interface between the plant and the supervisory automation layers is one of the key issues for semi-automated SCADA system integration.

7.2.1 Simulation Library for Hydraulic Systems

In this use-case, a simulation library was first designed by the author of this thesis and afterwards it was considered as an available simulation library. Later on, it was included into the mechatronic library, which is discussed in details in Appendix B.

[3] http://www.acin.tuwien.ac.at
[4] http://www.tuwien.ac.at

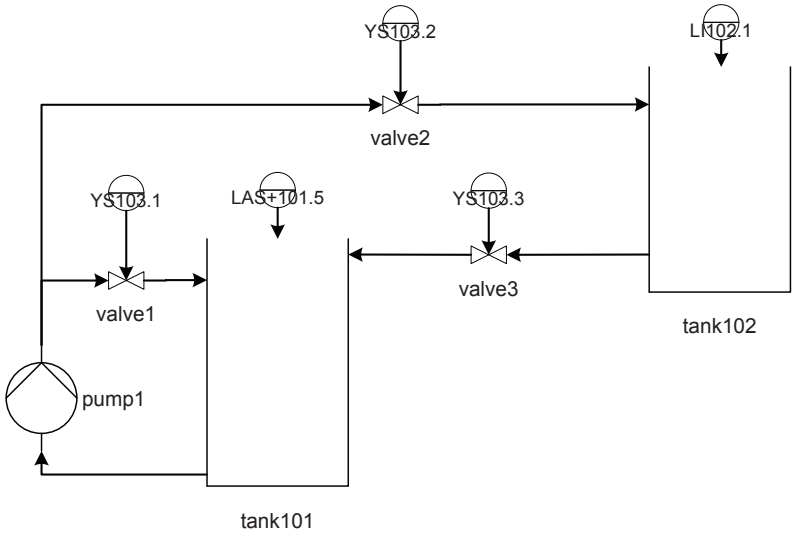

Figure 7.9: The basic test-bed: Laboratory tank model in the Odo Struger Laboratory at the Vienna University of Technology; adapted from previous author's work [120].

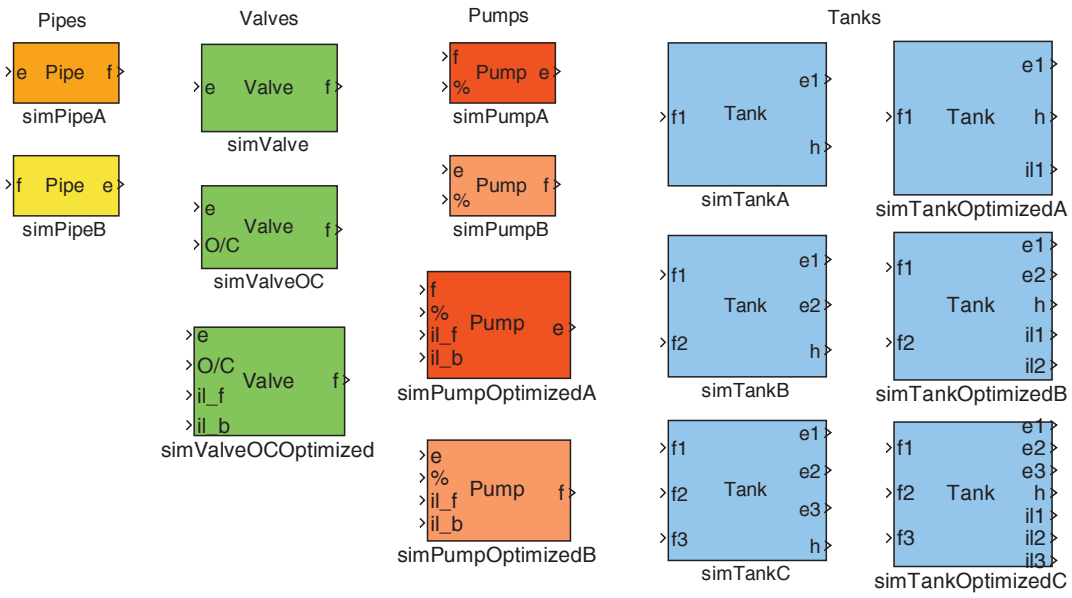

Figure 7.10: The hydraulic system part of the mechatronic simulation library depicted in Fig. B.1, which includes generic simulation blocks for the simulation of hydraulic systems.

The top layer of the simulation library (i.e., of the hydraulic system part of the mechatronic library) is depicted in Fig. 7.10 in MATLAB-Simulink. It shows that the library includes simulation blocks for approximating pipes, valves, pumps, and hydraulic tanks. This section describes the ideas behind the structure of the simulation library, whereas the details about internal structure, interfaces, and block parameters are addressed in Appendix B.

In case of the simulation blocks approximating pipes and pumps, the simulation library includes versions of the blocks having effort as input and flow as output, as well as blocks having interfaces vice-versa. In the case of pumps, the library includes two pairs of pumps. The first one assumes the basic constraint handling, whereas the second one supports the optimized constraint handling as it was proposed in Sec. 5.13.

The simulation library includes three types of valves, in particular the two valves differing in how openness is input into the simulation block (i.e., whether it is a block parameter or whether it is a block input) and one valve supporting the optimized constrained handling and having the openness signal as an input.

Tanks have the same interfaces in terms of positions of effort and flow, but they differ in number of power ports. When constructing a simulation model, the number of connection points has to be taken into account to select the appropriate simulation block. Similarly to the case of pumps, the versions implementing the basic constraint handling are included in the simulation library, as well as the blocks implementing the optimized constraint handling. Since even the basic versions of the simulation blocks implement switching off the output pressure when the height of the liquid in the tank is in the unfeasible region in relationship to the specific connection point of the tank, the basic versions are useful as well and applicable for paths between tanks without pressure or flow sources (i.e., pumps in the current version of the simulation library).

To create simulation models for hydraulic systems, simulation blocks realizing 0-junctions and 1-junctions in the sense of the bond-graph theory are needed, too. Since their design and implementation are straightforward, they are not discussed here and they can be seen in Appendix B.

7.2.2 Simulation Models for the Tank Model

When the structure of the laboratory tank model is captured in the automation ontology and when the simulation library blocks are annotated in the ontology as well, the simulation model for this plant can be generated utilizing the extended bond-graph theory. The structure of the plant was captured in the automation ontology with the Microsoft Visio connector presented in Sec. 6.3.1 (for further details see Fig. 6.3 and Fig. C.1) and the mechatronic simulation library was annotated manually in the automation ontology.

The first process step of the proposed simulation model design is the creation of the acausal extended bond graph. This issue is similar to the standard bond-graph theory case with the difference of considering available simulation blocks from the mechatronic library. The resulting bond graph is depicted in Fig. 7.11, which characterizes how components are interconnected and what are the junctions in the simulation model that is being designed.

In the next process step, the causality is assigned and thus specific simulation blocks are selected. After the causality assignment algorithm (i.e., Alg. 3) is performed, the causal bond graph is obtained. The outcome of this process step is depicted in Fig. 7.12. The

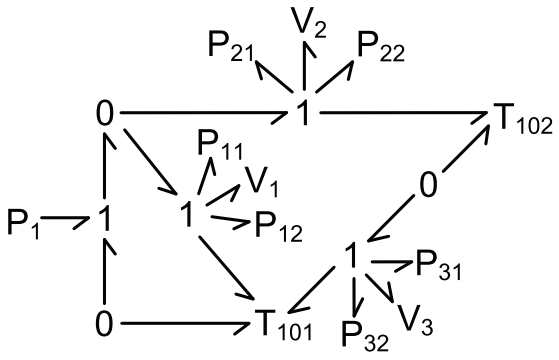

Figure 7.11: Structure of the bond graph for the laboratory tank model. In this process step, neither the causality nor selection of simulation blocks is not known.

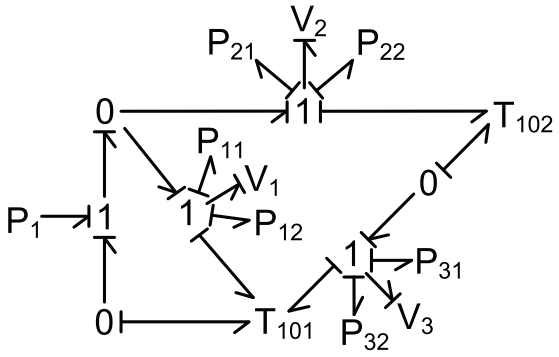

Figure 7.12: The bond graph for the laboratory tank model having assigned the causality. Specific simulation blocks and realization of junctions have been solved.

generated bond graph is serialized in the automation ontology within the bond graph domain in such a form that is convenient for generation of simulation models.

Based on the causal bond graph, the signal-oriented simulation model can be generated in a specific simulation environment. The generation of simulations in MATLAB-Simulink has been discussed in Sec. 6.4.1. Since the two versions of constraint handling were discussed in Sec. 5.13, the following two different versions of the simulation model are shown for the entire system. The first one is depicted in Fig. 7.13 and it includes the basic constraint handling, which means that output efforts of tanks are artificially lowered to the reference level when a constraint is reached. The second version is depicted in Fig. 7.14, which shows the optimized version of constraint handling that switches off such flows that lead to violating defined constraints.

The necessary condition for the generation of simulation models is that the library includes simulation blocks with such interfaces that are compatible each other to create the model, as it was described in details in Sec. 5. Nevertheless, this condition is not the only one to obtain usable and useful simulation models. The second obstacle in the simulation design is the need for the lowest possible number of algebraic loops. This is the reason, why simulation model structures are optimized frequently and why some of the components/blocks are merged with other ones. In this use-case, algebraic loops are caused by pipes. To increase the stability and performance of the simulation, the pipes can be merged with valves. This refinement of the simulation model structure has been done as well and it is discussed in the following explanation.

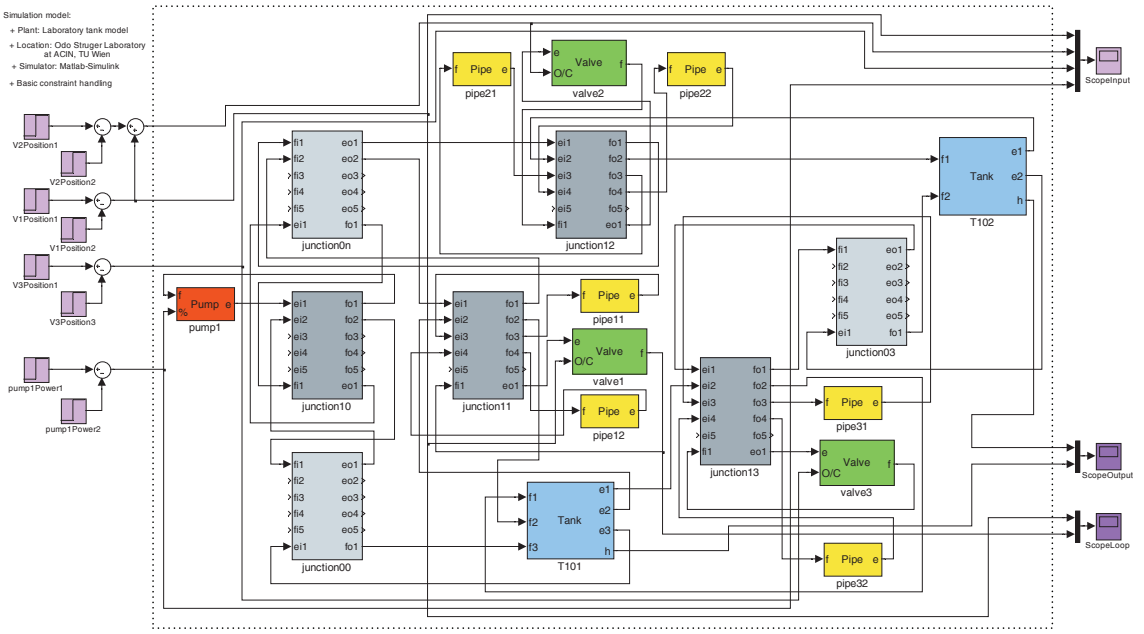

Figure 7.13: The simulation model for the entire version of the laboratory tank model utilizing the basic constraint handling.

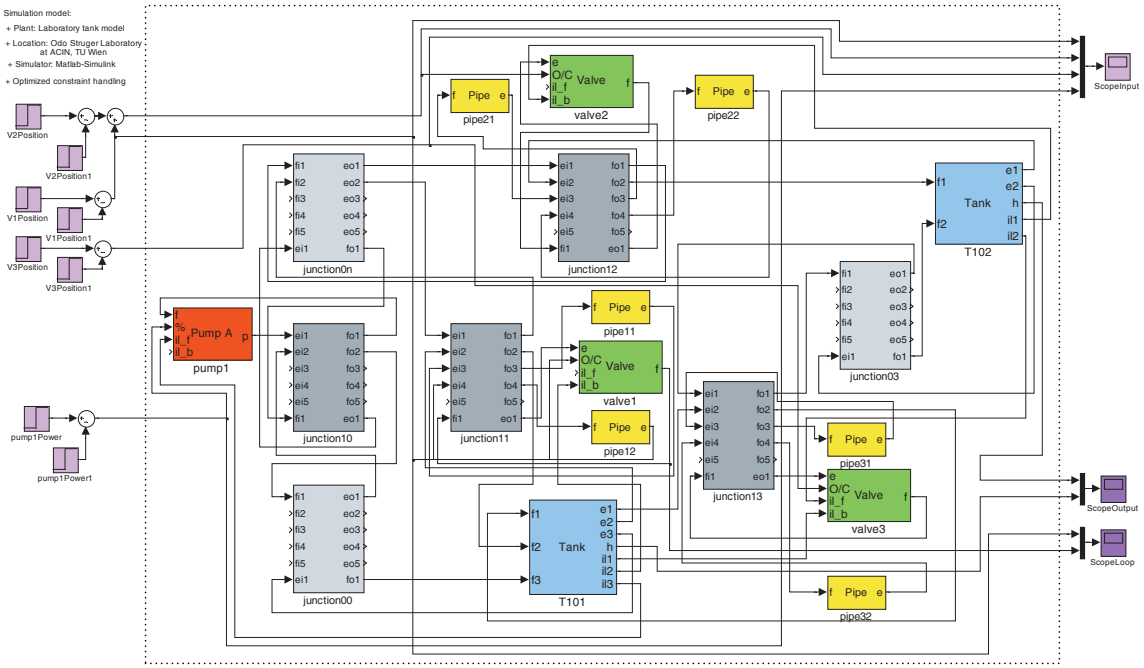

Figure 7.14: A simulation model for the entire version of the laboratory tank model utilizing the optimized constraint handling.

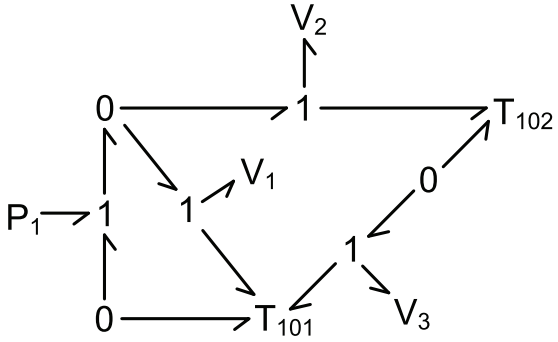

Figure 7.15: Structure of the bond graph for the refined version of the laboratory tank model with merged pipes and valves. In this process step, neither the causality nor selection of simulation blocks is not known.

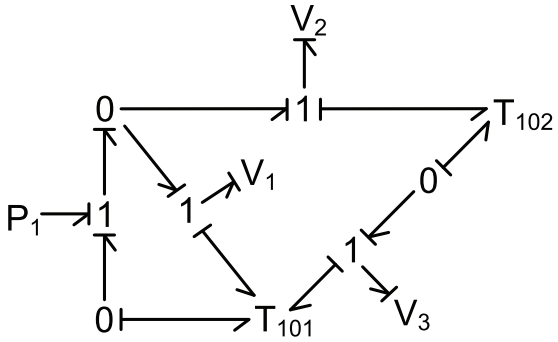

Figure 7.16: The bond graph for the refined version of the laboratory tank model having assigned the causality. Specific simulation blocks and realization of junctions have been solved.

The original system and its extended bond graph were simplified by merging the pipes and valves where possible. The extended bond graph of the simplified system is depicted in Fig. 7.15. Compared to the previous version, there are no explicit pipes in the graph.

Similarly as in the previous version of the simulation model, the causality has to be solved and specific simulation blocks selected with the use of the proposed causality assignment algorithm. The resulting extended bond graph is depicted in Fig. 7.16. For the refined system, where both two pipes surrounding each valve are merged with this valve, the two versions of simulation model were created based on this extended bond graph. The version implementing the basic constraint handling is depicted in Fig. 7.17, and the version implementing the optimized constraint handling is depicted in Fig. 7.18. The set of parameters is slightly changed for the refined versions of simulation model. Since the diameters of pipes and valves are the same, the value of diameter remained unchanged, but their length is the sum of lengths of the original valve and pipes.

In overall the four instances of simulation models were thus created for this use-case. They compare the models for the entire system and for the refined system, as well as for the basic and optimized constraint handling. The simulated experiments showed that the existence of algebraic loops does not pose a significant problem for the simulation of this scale[5]. The significant difference was found out in constraint handling, as the basic version based on signal pairs effort and flow signals need not work satisfactorily, which happens especially in the case of pumping the liquid to the upper tank. The optimized

[5]The simulation solver "ode23tb (stiff/TR-BDF2)" was used for the experiments in this use-case.

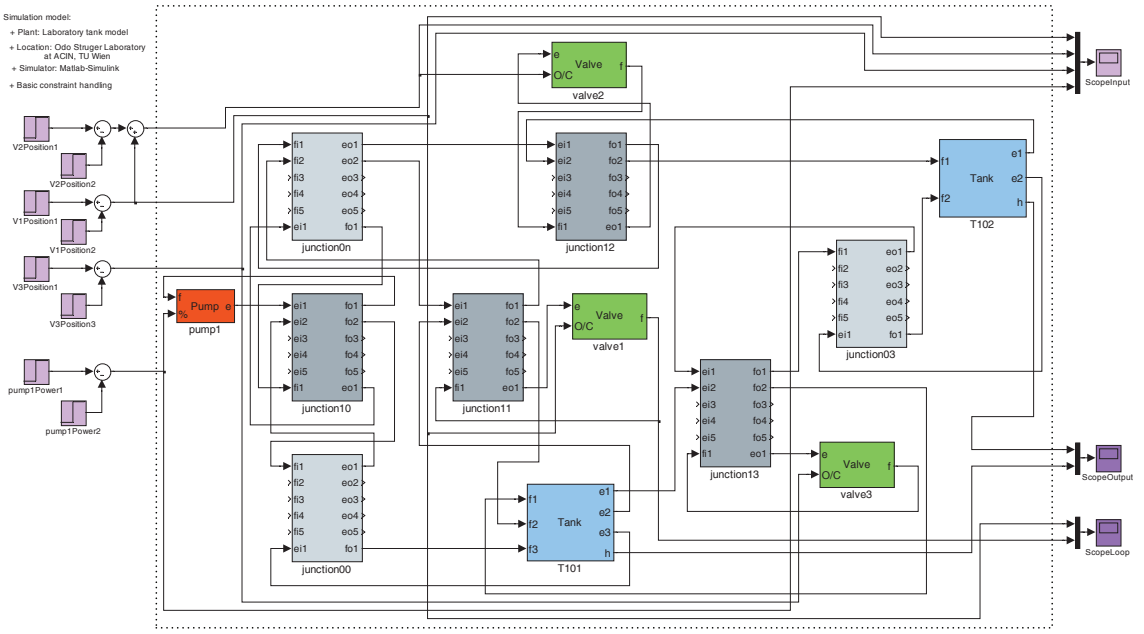

Figure 7.17: The simulation model for the refined version of the laboratory tank model utilizing the basic constraint handling.

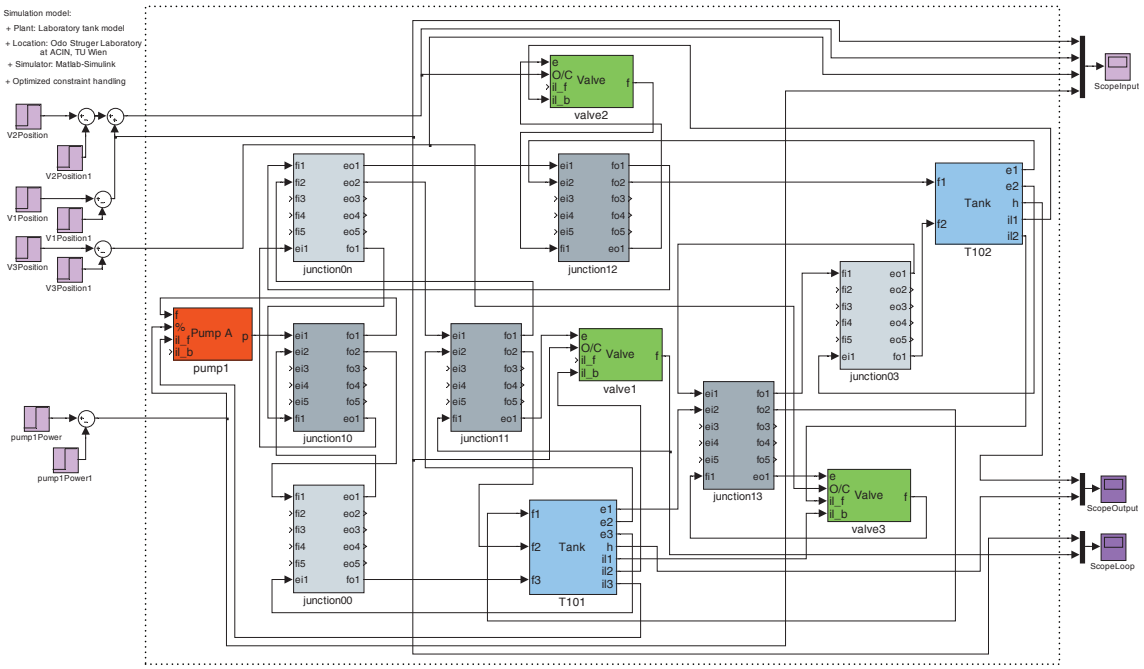

Figure 7.18: A simulation model for the refined version of the laboratory tank model utilizing the optimized constraint handling.

version of constraint handling simulates correctly all considered situations dealing with constraints on the level of this use-case. On the other hand, the optimized constraint handling approach requires to use additional signal bonds between components, thus the structure of the simulation model is not as straightforward as in the basic case. Nevertheless, the alternative having the refined system structure with eliminated algebraic loops and implementing the optimized constraint handling proved that the simulation expert is on the safe side and the simulation model works effectively and efficiently.

7.2.3 Generation of the Lists of Simulation Parameters and Tags

Prior executing a simulation model, simulation parameter values have to be set up. Furthermore, input tags have to be configured to define boundary conditions of the simulation model. If simulation experiments are used in a batch mode, time-series of values of input tags have to be available in advance. On the contrary in a synchronized mode of simulation execution, input tag samples have to be available at runtime and delivered in time by the integrating infrastructure. To visualize simulation results or to capture them in a database for runtime data, output tags have to be configured and integrated. In the following text, parameters and tags are discussed for the case of the hydraulic tank system.

Simulation parameters configure the behavior of simulation blocks for each particular position in the system topology. They are captured in the automation ontology and they are supported by the MS Visio connector. For the laboratory tank model, the simulation parameters queried from the ontology are summarized in Tab. 7.1.

To connect the simulation model of the laboratory tank model to the OPC server, the simulation model has 5 tags. These tags are summarized in Tab. 7.2 including whether each tag is an input or output tag. This configuration of tags is expected to be read by the simulation integration framework as part of the simulation model integration configuration.

7.2.4 Simulation Model Testing and Comparison of Measured and Simulated Experiments

The created simulation model for the hydraulic tank model was internally verified as the first step of the simulation model testing. The internal verification is focused on checking whether the created computer simulation is compliant with the mathematical-physical behavior of the system. Since the mathematical behavior of the modeled system is distributed into specific simulation blocks, the entire model behavior had to be checked in terms of convergence, behavior under limit circumstances and pressure–flow analysis in the three basic hydraulic paths.

As the second step of the simulation model testing was done the calibration of simulation model parameters. Especially the output pressure of the pump was set up based on measured characteristics as well as internal diameters of valves were fine-tuned based on measured data because several roots of pressure loss are neglected. The results of the simulation model fine-tuning are depicted in Fig. 7.19. It shows the comparison of the measured and simulated responses of the hydraulic tank model system and both simulation models for this system. A fundamental part of this testing phase is thus an external validation of the simulation model, which is focused on comparing the simulation results of the created simulation model with really measured responses under the intended operation scenarios.

Table 7.1: List of simulation parameters and their values for the laboratory tank model.

Simulation parameter	Parameter value	Parameter scale
$T_{101}.a$	0.0324	m
$T_{101}.h_0$	0.1000	m
$T_{101}.h_1$	0.1600	m
$T_{101}.h_2$	0.3300	m
$T_{101}.h_3$	0.0000	m
$T_{101}.h_i$	{0.1860; 0.0070}	m
$T_{102}.a$	0.0324	m
$T_{102}.h_0$	0.5900	m
$T_{102}.h_1$	0.0550	m
$T_{102}.h_2$	0.0080	m
$T_{102}.h_i$	{0.0070; 0.1860}	m
$valve1.l$	0.0500	m
$pipe11.l$	0.0500	m
$pipe12.l$	0.0500	m
$valve1.d$	0.0050	m
$pipe11.d$	0.0050	m
$pipe12.d$	0.0050	m
$valve2.l$	0.0500	m
$pipe21.l$	0.3700	m
$pipe22.l$	0.3000	m
$valve2.d$	0.0050	m
$pipe21.d$	0.0050	m
$pipe22.d$	0.0050	m
$valve3.l$	0.0500	m
$pipe31.l$	0.3000	m
$pipe32.l$	0.3800	m
$valve3.d$	0.0050	m
$pipe31.d$	0.0050	m
$pipe32.d$	0.0050	m
$pump1.e_{max}$	$2 \cdot 10^4$	Pa

Table 7.2: List of the tags for the laboratory tank model.

Tag type	Tag name	Tag scale
Input_1	$V_1Position$	0–1
Input_2	$V_2Position$	0–1
Input_3	$V_3Position$	0–1
Output_1	T_1Level	m
Output_2	T_2Level	m

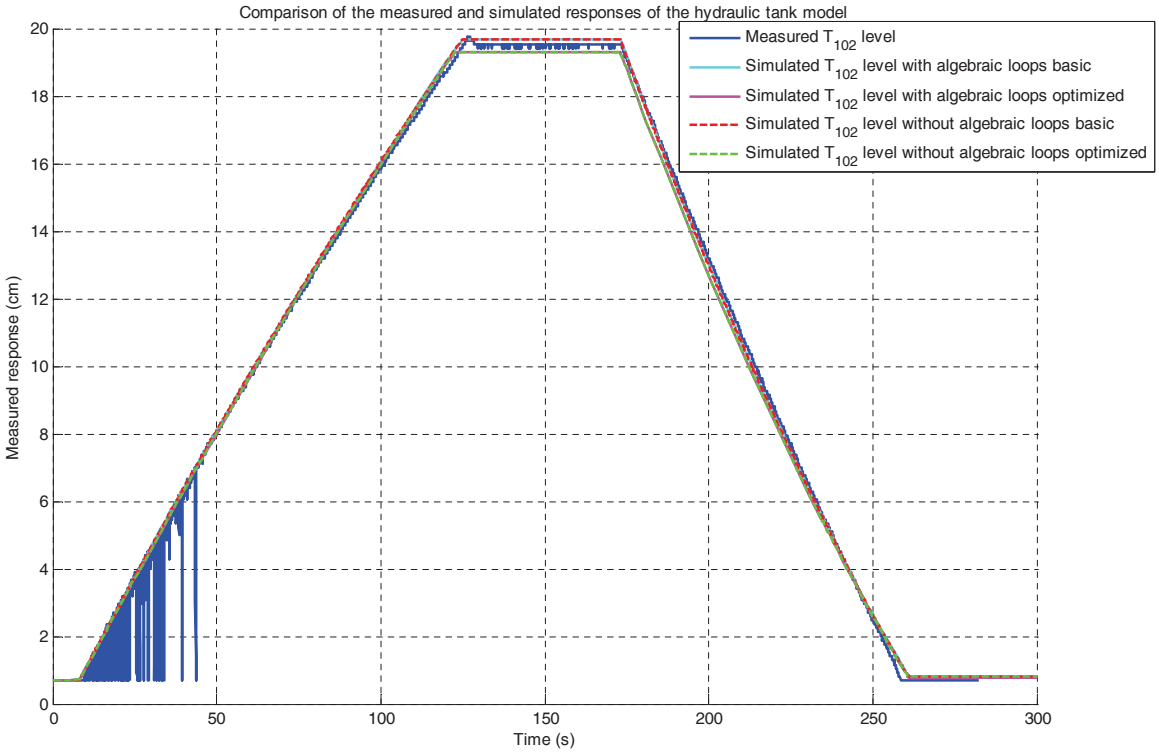

Figure 7.19: A simulation model for the laboratory tank model.

As the last step of the simulation model testing, a set of experiments on artificial input data in terms of pre-defined time-courses of the openness of the three valves and of the power of the pump has been done. The obtained responses of all simulation models for one of the experiments are compared in Fig. 7.20. We can see that all models tackle physical constraints for the case of valves connecting tanks with different water and bottom levels, however, the simulation model responses differ in facing constraints related to the pump. In this figure, we can see that the optimized constraint handling leads to realistic simulation results, whereas the basic constraint limit handling reaches its limits and cannot be satisfactorily used in this abnormal region.

Normal and abnormal operation conditions of the simulation model for the hydraulic tank model are testable mainly when the system is connected to a control system or algorithm, otherwise the simulation model can get into the operation region, where its behavior is not tested enough and the model can reach high error. In principle, any simulation model cannot get results under arbitrary mode of operation especially in terms of combination of high frequency inputs.

7.2.5 Lessons Learned and Evaluation of the Reached Results

Summarizing the entire hydraulic use-case, it was shown how to design the simulation model for the laboratory tank model with the extended bond graph method, and how to configure the integration of the model within the integrated simulation environment. These tasks are required by industrial practitioners frequently and the proposed method contributes to the significant reduction of manual effort, resulting into mitigation of human errors and saving time and costs for the development and testing of simulation models.

Compared to the previous passive house use-case, the hydraulic tank model system poses

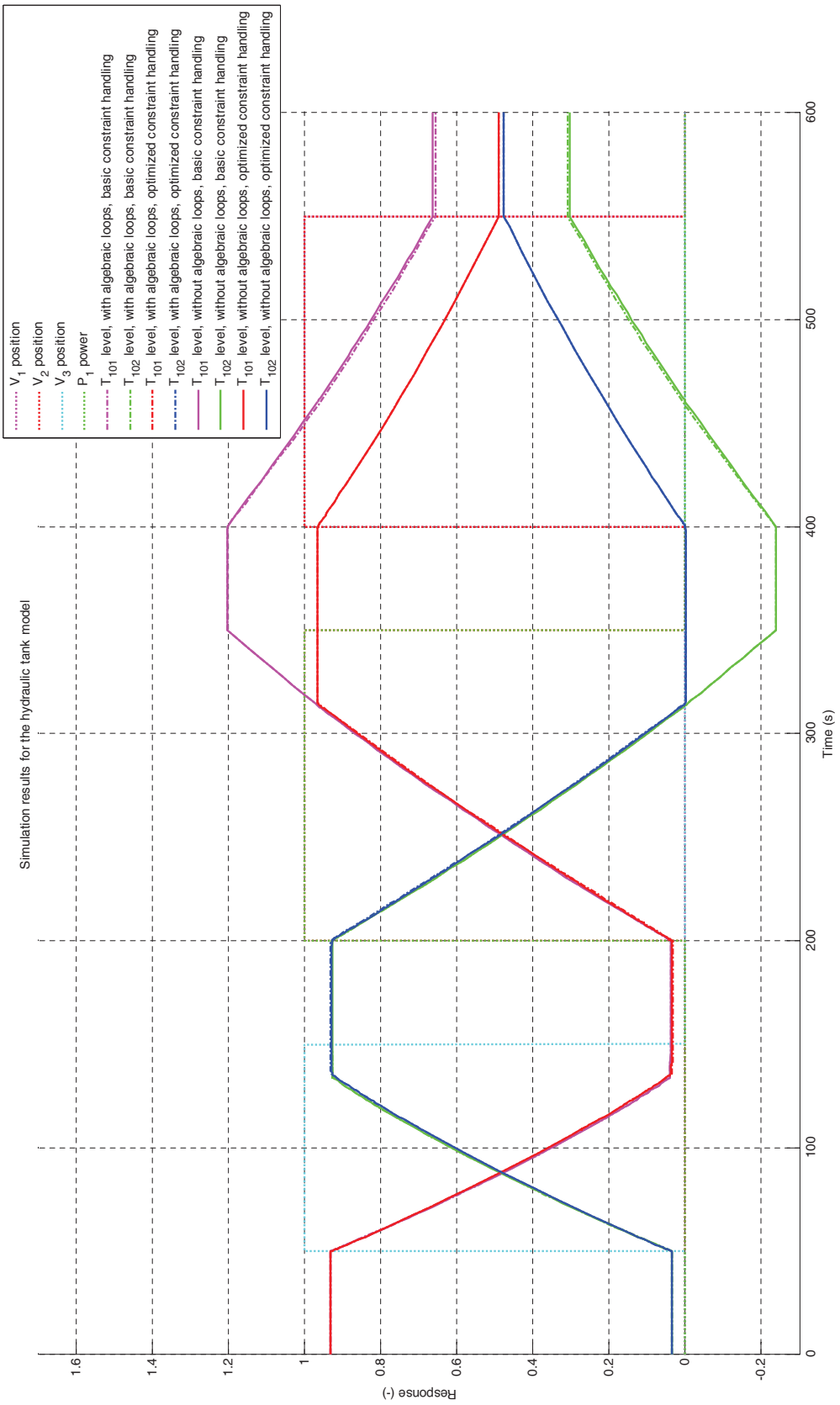

Figure 7.20: A simulation model for the laboratory tank model.

a more complicated case in terms of the needed selection of simulation blocks, which had to take into account (i) the number of inputs and outputs, as well as (ii) the assignment of the effort–flow signals to the power bonds. Since this laboratory system can be scaled-up easily, it also proved that the proposed method is suitable for simulation design problems in the industrial scale. In compliance with the results mentioned in Sec. 7.2.4, this use-case stressed the importance of constraint handling that plays a crucial role for testing control systems especially under abnormal conditions.

Within this use-case, the performed refinement of the simulation model structure to avoid algebraic loops (i.e., merging pipes with valves) had positive impact on simulation stability and performance. However, it required human effort as neither the detection nor solving of algebraic loops has been supported by the proposed semi-automated method. It also proved that the simulation model is applicable without this refinement (i.e., in the version corresponding to the entire plant model), which is a good news for the automated simulation model design.

Chapter 8

Conclusions and Future Work

This thesis is aimed at improving the design and integration of simulation models for industrial systems. The presented research is focused on simulations on the supervisory level of automation systems. The main goals of this thesis were to design and to implement methods to support the creation of simulation models for large-scale systems and to integrate simulation models within the remainder of the automation system. The proposed approach is based on the reuse of engineering knowledge and it has been developed in such a way that is flexible and applicable in industrial practice easily.

In more details, the research presented in this thesis addresses the research issue RI-1, which is presented in Sec. 4. It is focused on the formalization of engineering knowledge in the automation ontology. From the scientific perspective, this issue is considered rather as a prerequisite for further two research issues. Such a prerequisite is needed for solving the RI-2, which poses a core research contribution of this thesis presented in Sec. 5. The traditional and well-proven bond graph approach for simulation model engineering was extended and adapted for the needs of the current computer-centric simulation design way of thinking. Therefore, simulations for signal-based simulators can be designed from existing simulation blocks with the proposed method easily and seamlessly. The main criterion for the selection of appropriate simulation blocks is the need for compatibility of input and output interfaces of simulation blocks in simulation model topologies. Decomposition of complex simulation models into a set of simulation modules has been investigated as a part of this research issue as well. Since simulation models should be integrated within industrial automation systems and within data of the automation system runtime, these problems were addressed in the frame of RI-3 in Sec. 6, dealing with the integration of simulations on the SCADA level of industrial automation systems. To illustrate the proposed methods in practice, the two use-cases are presented in Sec. 7. The examples from the passive house area and hydraulic system engineering prove the efficiency of the proposed approach and its applicability even for non-experts in simulation modeling.

8.1 Fulfillment of the Thesis Goals

Following the thesis goals presented in Sec. 1.3, this section summarizes the research results of this thesis from the high-level perspective.

- G-1: Representation of engineering knowledge for simulation design and integration

The automation ontology for representing engineering knowledge needed for simulation model design and integration has been designed and disseminated for example in [122] or in [124]. Since a lot of ontologies have been designed all over the world, the author of this thesis considered the applicability in practice as a very important feature. Therefore, the knowledge transfers from plant models represented in Microsoft Visio as well as AutomationML data format have been addressed in this thesis.

- G-2: Object-oriented design of simulation modules

  Simulation models are considered as one of the corner-stones for the Industry 4.0 movement and for virtual commissioning in the frame of factories of the future. In this thesis, a component-based method for simulation model design was motivated [123], formulated, and supported by the proposed extended bond graph method, disseminated in [126]. It is accompanied by the proposal of a new causality assignment algorithm supporting the enhanced aspects of extended bond graphs. The software prototype implementing the extended bond graph theory has been developed and presented at the flag-ship industrial tools fair SPS IPC Drives 2015 in Nuremberg, where it was positively rated by visitors from industrial practice. The implementation on the software prototype level has proved that time and effort for simulation model design can be significantly saved and design-time errors can be eliminated.

- G-3: Design of simulation workflows consisting of simulation modules

  The proposed extended bond graphs can be also used for facilitating the design of complex simulation workflows utilizing a set of interlinked simulation modules. This approach can be used for example in the case of large-scale simulation projects, whose solving takes more than one or two years. With this method, one can design interfaces of simulation modules easily. The proposed paradigm distinguishes cuts of large-scale simulation models on a junction level and on a bond level. The simulation modules themselves can be designed manually or automatically, depending on the decision of the simulation project lead. The benefit of the proposed method is that interfaces of simulation modules and their inter-connections can be set first, thus misunderstandings of independent teams or engineers on the interface level during simulation projects are mitigated. This goal was disseminated for example in [127] or [118]. The motivation was discussed in the co-authored paper [168].

- G-4: Integration of simulations within SCADA systems

  Access of simulations to process data and their versioning pose important issues. The thesis provides foundations for addressing these issues by supporting the architecture of the overall system and prototypal implementation of HMI connectors and partially also the data acquisition approach based on OPC UA. The results related to this goal were disseminated for example in [127] or in [121].

8.2 Scientific Contributions Reached in the Thesis

The research presented in this thesis resulted into the following scientific contributions:

1. Utilization of ontologies for supporting simulation model design and integration as well as the design and implementation of the structure of the automation ontology.

2. Adaptation of the bond graph theory for non-traditional applications and identification of bond graph benefits in modern computer-based simulation design.

3. Abstracting and separating specifications of interfaces and internal representations of simulation components to support explicit specification of simulation blocks.

4. Selection of alternative simulation blocks for each component and design of a new causality assignment algorithm supporting this.

5. Design of junctions for integrating simulation modules within complex simulation workflows based on the bond-graph theory.

6. Improved engineering process for simulation model design in the area of component-based and module-based dynamic simulations for industrial systems.

7. Improved tool support for capturing engineering knowledge and integration of simulations within industrial SCADA systems.

Combining the aforementioned points, the outcomes of this thesis lead to significant improvements of the design and integration of simulation models for industrial systems, which are important enablers for the emerging Industry 4.0 applications. The overall contribution of this thesis is thus improving the simulation model life-cycle to make it more flexible and compliant within the engineering and runtime tools utilized for automation system engineering and operation. Since generated simulation models can be used for finding bottlenecks of automation and control systems efficiently, the thesis results can be utilized for analyzing and improving safety and security of critical infrastructures.

8.3 Future Work

Although the proposed approach has been designed as generally as possible in terms of the time frame of the doctoral studies, a tool connector for an equation-based simulator (such as Dymola utilizing the Modelica language) has not been implemented yet. The development and evaluation of the utilization of the proposed approach for this type of simulations is a promising topic, which can be beneficial from the commercial point of view. In addition, future work could be focused on wider tool support and engineering tool integration in conjunction with the presented methods and approaches.

Bond graphs as well as their proposed extended version address structural aspects of simulation model composition, but do not face timing and synchronization issues. The future work can be focused on the investigation of numerical methods to verify and to assure the synchronization of simulation modules in complex simulation workflows.

Bibliography

[1] Akers, A., Gassman, M., and Smith, R. *Hydraulic Power System Analysis*. CRC Press Taylor & Francis Group, 2006.

[2] Allemang, D., and Hendler, J. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[3] Allweyer, T. *BPMN 2.0*. BoD, 2010.

[4] Alsafi, Y., and Vyatkin, V. Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. *Robotics and Computer-Integrated Manufacturing 26*, 4 (2010), 381 – 391.

[5] Amadori, K., Tarkian, M., Ölvander, J., and Krus, P. Flexible and robust CAD models for design automation. *Advanced Engineering Informatics 26*, 2 (2012), 180–195. Special Section on Knowledge based engineering to support complex product design.

[6] Andrushevich, A., Staub, M., Kistler, R., and Klapproth, A. Towards semantic buildings: Goal-driven approach for building automation service allocation and control. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2010)* (Sept. 2010), pp. 1–6.

[7] Anglani, A., Grieco, A., Pacella, M., and Tolio, T. Object-oriented modeling and simulation of flexible manufacturing systems: a rule-based procedure. *Simulation Modelling Practice and Theory 10*, 3-4 (2002), 209–234.

[8] Antsaklis, P. J., and Michel, A. N. *Linear Systems*, 2nd ed. Birkhäuser, Boston, 2006.

[9] Atzori, L., Iera, A., and Morabito, G. The internet of things: A survey. *Computer Networks 54*, 15 (Oct 2010), 2787–2805.

[10] <AutomationML/> The Glue for Seamless Automation Engineering. AutomationML consortium, version April 2016. Cited on 2016-07-18. Available at: https://www.automationml.org/.

[11] Awais, M., Palensky, P., Elsheikh, A., Widl, E., and Matthias, S. The high level architecture RTI as a master to the functional mock-up interface components. In *International Conference on Computing, Networking and Communications (ICNC 2013)* (Jan 2013), pp. 315–320.

[12] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, USA, 2003.

[13] Bagdasaryan, A. Discrete dynamic simulation models and technique for complex control systems. *Simulation Modelling Practice and Theory 19*, 4 (2011), 1061 – 1087. Special Issue on Sustainable Energy and Environmental Protection "SEEP2009".

[14] Bauer, M., and Craig, I. K. Economic assessment of advanced process control – a survey and framework. *Journal of Process Control 18* (2008), 2–18.

[15] Beez, S., Fay, A., and Thornhill, N. Automatic generation of bond graph models of process plants. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2008)* (2008), pp. 1294–1301.

[16] Benjamin, P., and Akella, K. Towards ontology-driven interoperability for simulation-based applications. In *Proceedings of the 2009 Winter Simulation Conference (WSC)* (Dec. 2009), pp. 1375–1386.

[17] Bernal, J. *Application Architecture for WebSphere: A Practical Approach to Building WebSphere Applications*. IBM Press, 2009.

[18] Berners-Lee, T., Hendler, J., and Lassila, O. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American* (May 2001).

[19] Bieberstein, N., Laird, R. G., Jones, D. K., and Mitra, T. *A Practical Guide for the Service-Oriented Architect*. IBM Press, 2008.

[20] Biffl, S., Mätzler, E., Wimmer, M., Lüder, A., and Schmidt, N. Linking and versioning support for AutomationML: A model-driven engineering perspective. In *13th IEEE International Conference on Industrial Informatics (INDIN 2015)* (July 2015), pp. 499–506.

[21] Biffl, S., and Schatten, A. A platform for service-oriented integration of software engineering environments. In *Proceedings of the Eighth Conference on New Trends in Software Methodologies, Tools and Techniques (SoMeT 2009)* (Amsterdam, 2009), IOS Press, pp. 75–92.

[22] Biffl, S., Schatten, A., and Zoitl, A. Integration of heterogeneous engineering environments for the automation systems lifecycle. In *Proc. of the 7th IEEE International Conference on Industrial Informatics (INDIN 2009)* (June 2009), pp. 576–581.

[23] Biliris, A. A data model for engineering design objects. In *Second International Conference on Data and Knowledge Systems for Manufacturing and Engineering* (1989), pp. 49–58.

[24] Blundell, A. J. *Bond Graphs for Modelling Engineering Systems*. Ellis Horwood Limited, Chichester, England, 1982.

[25] Bo, D., Kun, D., and Xiaoyi, Z. A high performance enterprise service bus platform for complex event processing. In *Seventh International Conference on Grid and Cooperative Computing (GCC 2008)* (Oct. 2008), pp. 577–582.

[26] Borutzky, W. Bond graph modeling from an object oriented modeling point of view. *Simulation Practice and Theory 7* (1999), 439 – 461.

[27] Borutzky, W. Bond graph modelling and simulation of multidisciplinary systems – an introduction. *Simulation Modelling Practice and Theory 17*, 1 (2009), 3 – 21.

[28] Borutzky, W. *Bond Graph Methodology – Development and Analysis of Multidisciplinary Dynamic System Models*. Springer-Verlag London, 2010.

[29] Boyer, S. A. *SCADA Supervisory Control and Data Acquisition*. ISA – International Society of Automation, USA, 2010.

[30] Broenink, J. F. 20-sim software for hierarchical bond-graph/block-diagram models. *Simulation Practice and Theory 7*, 5 - 6 (1999), 481 – 492.

[31] Broman, D., Brooks, C., Greenberg, L., Lee, E., Masin, M., Tripakis, S., and Wetter, M. Determinate composition of FMUs for co-simulation. In *Proceedings of the International Conference on Embedded Software (EMSOFT 2013)* (Sept 2013), pp. 1–12.

[32] Butcher, J. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics 125*, 1-2 (2000), 1 – 29. Special Issue on Numerical Analysis 2000. Vol. VI: Ordinary Differential Equations and Integral Equations.

[33] Byrne, J., Heavey, C., and Byrne, P. A review of web-based simulation and supporting tools. *Simulation Modelling Practice and Theory 18*, 3 (2010), 253–276.

[34] CDL-Flex Whitepaper: Making Collaborative Engineering more efficient [online]. Cited on 2012-07-26. Available: http://cdl.ifs.tuwien.ac.at/files/CDL-Flex_ASB_UC_overview_en.pdf.

[35] Cellier, F., and McBride, R. Object-oriented modeling of complex physical systems using the dymola bond-graph library. In *Proceedings of the 6th SCS International Conference on*

Bond Graph Modeling and Simulation (ICBGM'03), Orlando, Florida (2003), pp. 157–162.

[36] Chappell, D. A. *Enterprise Service Bus*. O'Reilly Media Inc., 2004.

[37] Cornelio, A., and Navathe, S. Database support for engineering CAD and simulation. In *Second International Conference on Data and Knowledge Systems for Manufacturing and Engineering* (1989), pp. 38–48.

[38] Das, S. *Mechatronic Modeling and Simulation Using Bond Graphs*. CRC Press Taylor & Francis Group, 2009.

[39] Dibowski, H., Ploennigs, J., and Kabitzsch, K. Automated design of building automation systems. *IEEE Transactions on Industrial Electronics 57*, 11 (November 2010), 3606–3613.

[40] Directive 2010/31/EU of the European Parliament and of the Council on the Energy Performance of Buildings [online]. Cited on 2012-04-22. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010: 153:0013:0035:EN:PDF.

[41] Dragojlovic, S., and Milenkovic, O. Integration of SCADA system in technical information system. In *18th International Conference and Exhibition on Electricity Distribution (CIRED 2005)* (2005), pp. 1–3.

[42] Dragojlovic, S. M., and Milenkovic, O. T. Integration of SCADA in the information system. In *International Symposium CIGRE/IEEE PES* (2005), pp. 254–258.

[43] Drath, R. *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA*. Springer-Verlag Berlin Heidelberg, 2010.

[44] Drath, R., Weber, P., and Mauser, N. An evolutionary approach for the industrial introduction of virtual commissioning. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2008)* (Sept 2008), pp. 5–8.

[45] Durak, U., Güler, S., Oğuztüzün, H., and İder, S. K. An exercise in ontology driven trajectory simulation with MATLAB SIMULINK(R). In *Proceedings of the 21th European Conference on Modelling and Simulation (ECMS)* (2007).

[46] Elsheikh, A., Awais, M., Widl, E., and Palensky, P. Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface. In *Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES 2013)* (May 2013), pp. 1–6.

[47] Ericsson, G. Modeling of power system communications – recognition of technology maturity levels and case study of a migration scenario. *IEEE Transactions on Power Delivery 19*, 1 (2004), 105–110.

[48] Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[49] Factories of the future: Multi-annual roadmap for the contractual PPP under Horizon 2020. European Factories of the Future Research Association (EFRA), European Union, 2013.

[50] Fay, A., Biffl, S., Winkler, D., Drath, R., and Barth, M. A method to evaluate the openness of automation tools for increased interoperability. In *Proc. of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)* (Vienna, Austria, 2013), pp. 6842–6847.

[51] Foeken, M., Cabrera, A. A., Voskuijl, M., and van Tooren, M. Enabling control software generation by using mechatronics modeling primitives. *Advanced Engineering Informatics 26*, 2 (2012), 196–206. Knowledge based engineering to support complex product design.

[52] Gawthrop, P., and Bevan, G. Bond-graph modeling. *IEEE Control Systems Magazine 27*, 2 (April 2007), 24–45.

[53] Ghoshal, K. Distribution automation: SCADA integration is key. *IEEE Computer Applications in Power 10*, 1 (1997), 31–35.

[54] Gómez-Pérez, A., Fernández-López, M., and Corcho, O. *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*, 2nd ed. Springer-Verlag, London, 2004.

[55] Gruber, T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition 5(2)* (1993).

[56] Hall, K. H., Staron, R. J., and Vrba, P. Experience with holonic and agent-based control systems and their adoption by industry. In *Holonic and Multi-Agent Systems for Manufacturing*, V. Mařík, R. Brennan, and M. Pěchouček, Eds., vol. 3593 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 1–10.

[57] Halliday, D., Resnick, R., and Walker, J. *Fundamentals of Physics*, 5th extended ed. John Wiley & Sons, Inc., 1997.

[58] Hao, Q., Shen, W., and Zhang, Z. An autonomous agent development environment for engineering applications. *Advanced Engineering Informatics 19*, 2 (2005), 123 – 134. Special issue on Collaborative Environment for Design and Manufacturing.

[59] Harcuba, O., and Vrba, P. Ontologies for flexible production systems. In *Proceedings of the 20th IEEE International Conference on Emerging Technologies Factory Automation (ETFA 2015)* (September 2015), pp. 1–8.

[60] Harjunkoski, I., Nyström, R., and Horch, A. Integration of scheduling and control – theory or practice? *Computers & Chemical Engineering 33*, 12 (December 2009), 1909 – 1918.

[61] Hegny, I., Strasser, T., Melik-Merkumians, M., Wenger, M., and Zoitl, A. Towards an increased reusability of distributed control applications modeled in IEC 61499. In *Proc. of 17th IEEE Conference on Emerging Technologies and Factory Automation* (2012).

[62] HLA - High Level Architecture. U.S. Defense Modeling and Simulation Office, 2001.

[63] Hollifield, B. R., Oliver, D., Nimmo, I., and Habibi, E. *The High Performance HMI Handbook*. PAS, Houston, 2008.

[64] Horáček, P. *Systémy a modely [In the Czech language]*. Czech Technical University in Prague, 1999.

[65] Horáček, P., Kříž, J., Langer, M., Riedel, M., and Fetterik, J. House builder v1.0. Tech. rep., Czech Technical University in Prague, Faculty of Electrical Engineering, Dept. of Cybernetics, 2010.

[66] Hu, J., and Zhang, H. Ontology based collaborative simulation framework using HLA and Web services. In *World Congress on Computer Science and Information Engineering, 2009 WRI* (April 2009), vol. 5, pp. 702–706.

[67] Hua, B., Zhou, J., and Yu, J. Integration of exist SCADA/EMS with dispatcher training simulator system. In *Power Systems Conference and Exposition, IEEE PES, 2004* (2004), vol. 2, pp. 829–838.

[68] Hundt, L., and Luder, A. Development of a method for the implementation of interoperable tool chains applying mechatronical thinking – Use case engineering of logic control. In *Proc. of the 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2012)* (2012), pp. 1–8.

[69] Hurwitz, J., Bloor, R., Baroudi, C., and Kaufman, M. *Service Oriented Architectures for Dummies*. Wiley Publishing, Indianapolis, 2007.

[70] IEC 62714: Engineering data exchange format for use in industrial automation systems engineering – Automation Markup Language, 2014.

[71] Iso 15926: Industrial automation systems and integration – integration of life-cycle data for process plants including oil and gas production facilities. international organization for standardization, 2009.

[72] Jamshidi, M. *Systems of Systems Engineering – Principles and Applications*. CRC Press Taylor & Francis Group, 2008.

[73] Jennings, N. R., Sycara, K., and Wooldridge, M. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems 1*, 1 (1998), 7–38.

[74] Jimeno, J., Laresgoiti, I., Oyarzabal, J., Stene, B., and Bacher, R. Architectural framework for the integration of distributed resources. In *Proc. of IEEE Power Tech Conference* (2003), vol. 2, p. 5.

[75] Jirkovský, V., Kadera, P., Obitko, M., and Vrba, P. Diagnostics of distributed intelligent control systems: Reasoning using ontologies and hidden Markov models. In *Proc. of the 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)* (May 2012).

[76] Jirkovský, V., Obitko, M., Novák, P., and Kadera, P. Big data analysis for sensor time-series in automation. In *Proc. of the 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (Barcelona, 2014).

[77] Junior, W., and Pereira, C. A supervisory tool for real-time industrial automation systems. In *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing* (2003), pp. 230–237.

[78] Kadera, P., and Tichý, P. Plan, commit, execute protocol in multi-agent systems. In *Proc. Holonic and Multi-Agent Systems for Manufacturing* (2009), V. Mařík, T. Strasser, and A. Zoitl, Eds., vol. 5696 of *LNCS*, Springer-Verlag Berlin Heidelberg, pp. 155–164.

[79] Kalagasidis, A. S., Weitzmann, P., Nielsen, T. R., Peuhkuri, R., Hagentoft, C.-E., and Rode, C. The international building physics toolbox in simulink. *Energy and Buildings 39*, 6 (2007), 665 – 674.

[80] Kane, L. A. *Handbook of advanced process control systems and instrumentation.* Gulf Pub. Co., 1987.

[81] Kaplinsky, R., and Morris, M. *A handbook for value chain research*, vol. 113. IDRC Ottawa, 2001.

[82] Karassik, I. J., Messina, J. P., Cooper, P., and Heald, C. C. *Pump Handbook*, 4th ed. The McGraw-Hill Companies, Inc., 2008.

[83] Karnopp, D., and Rosenberg, R. C. *Analysis and simulation of multiport systems: The Bond Graph Approach to Physical System Dynamics.* The Massachusetts Institute of Technology Press, 1968.

[84] Kim, B. C., Teijgeler, H., Munc, D., and Han, S. Integration of distributed plant lifecycle data using ISO 15926 and Web services. *Annals of Nuclear Energy 38* (2011), 2309–2318.

[85] Knublauch, H., Tetlow, P., Wallace, E., and Oberle, D. A Semantic Web primer for object-oriented software developers [online]. W3C Note. Cited on 2012/07/27. Available at: http://www.w3.org/TR/2006/NOTE-sw-oosd-primer-20060309, 2006.

[86] Köksal, S., Aldoğan, D., Akdemir, C., Taşdelen, I., and Dikenelli, O. A control architecture for integration of different simulation systems. In *Second International Conference on Advances in System Simulation (SIMUL 2010)* (August 2010), pp. 134–139.

[87] Kopetz, H. Internet of things. In *Real-Time Systems*, Real-Time Systems Series. Springer US, 2011, pp. 307–323.

[88] Kovalenko, O. Using explicit and machine-understandable engineering knowledge for defect detection in automation systems engineering. In *Proceedings of International Doctoral Symposium on Software Engineering and Advanced Applications (IDoSEAA)* (Oulu, Finland, 2011), pp. 1–5.

[89] Krammer, M., Martin, H., Radmilovic, Z., Erker, S., and Karner, M. Standard compliant co-simulation models for verification of automotive embedded systems. In *Forum on Specification and Design Languages (FDL 2015)* (September 2015), pp. 1–8.

[90] Křemen, P. *Building Ontology-Based Information Systems.* PhD thesis, Czech Technical

University in Prague, Faculty of Electrical Engineering, Dept. of Cybernetics, 2012.

[91] Lacy, L., and Gerber, W. Potential modeling and simulation applications of the Web ontology language - OWL. In *Proc. of the Winter Simulation Conference* (2004), vol. 1.

[92] Lamit, L. G. *Piping Systems, Drafting and Design.* Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1981.

[93] Lange, J., Iwanitz, F., and Burke, T. J. *OPC - From Data Access to Unified Architecture.* VDE Verlag, 2010.

[94] Lepuschitz, W., Lobato-Jimenez, A., Axinia, E., and Merdan, M. A survey on standards and ontologies for process automation. In *Proceedings of the 7th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS 2015), Valencia, Spain, September 2-3, 2015*, V. Mařík, A. Schirrmann, D. Trentesaux, and P. Vrba, Eds. Springer International Publishing, 2015, pp. 22–32.

[95] Li, X., Gao, M., Liu, J., Ding, Z., and Duan, X. A software architecture for integrative utility management system. In *IEEE Power Engineering Society Winter Meeting, 2001* (2001), vol. 2, pp. 476–480.

[96] Marin, C., Monch, L., Leitao, P., Vrba, P., Kazanskaia, D., Chepegin, V., Liu, L., and Mehandjiev, N. A conceptual architecture based on intelligent services for manufacturing support systems. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC 2013)* (Oct 2013), pp. 4749–4754.

[97] Marquis-Favre, W., and Scavarda, S. Alternative causality assignment procedures in bond graph for mechanical systems. *Journal of Dynamic Systems, Measurement, and Control 124 (3)* (2002), 457–463.

[98] Matics, J., Krost, G., Roggatz, C., and Spanel, U. Dispersed generation modeling in scada time scale. In *IEEE Russia Power Tech 2005* (2005), pp. 1–7.

[99] Mařík, V., Vrba, P., and Fletcher, M. Agent-based simulation: MAST case study. In *Emerging Solutions for Future Manufacturing Systems*, L. Camarinha-Matos, Ed., vol. 159 of *IFIP International Federation for Information Processing*. Springer US, 2005, pp. 61–72.

[100] Melik-Merkumians, M., Baier, T., Steinegger, M., Lepuschitz, W., Hegny, I., and Zoitl, A. Towards OPC UA as portable SOA middleware between control software and external added value applications. In *Proc. of 17th IEEE Conference on Emerging Technologies and Factory Automation* (2012).

[101] Melik-Merkumians, M., Zoitl, A., and Moser, T. Ontology-based fault diagnosis for industrial control applications. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)* (Sept. 2010), pp. 1 –4.

[102] Meyer, H., Fuchs, F., and Thiel, K. *Manufacturing Execution Systems: Optimal Design, Planning, and Deployment.* McGraw Hill, New York, 2009.

[103] Mohitpour, M., Golshan, H., and Murray, A. *Pipeline Design & Construction: A Practical Approach*, 3rd ed. American Society of Mechanical Engineers (ASME Press), 2007.

[104] Morbach, J. *A Reusable Ontology for Computer-Aided Process Engineering.* PhD thesis, RWTH Aachen University, 2009.

[105] Morbach, J., Wiesner, A., and Marquardt, W. OntoCAPE – A (re)usable ontology for computer-aided process engineering. *Computers and Chemical Engineering 33* (2009), 1546–1556.

[106] Mordinyi, R., Moser, T., Kühn, E., Biffl, S., and Mikula, A. Foundations for a model-driven integration of business services in a safety-critical application domain. In *35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2009)* (August 2009), pp. 267–274.

[107] Mordinyi, R., Schindler, P., and Biffl, S. Evaluation of NoSQL graph databases for querying and versioning of engineering data in multi-disciplinary engineering environments.

In *20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2015)* (September 2015), pp. 1–8.

[108] Moser, T., and Biffl, S. Semantic tool interoperability for engineering manufacturing systems. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2010* (September 2010), pp. 1–8.

[109] Moser, T., and Biffl, S. Semantic integration of software and systems engineering environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42*, 1 (January 2012), 38–50.

[110] Moser, T., Mordinyi, R., Sunindyo, W., and Biffl, S. Semantic service matchmaking in the atm domain considering infrastructure capability constraints. In *Canadian Semantic Web: Technologies and Applications*, W. Du and F. Ensan, Eds. Springer US, 2010, pp. 133–157.

[111] Mosterman, P. J. Hybrsim – a modeling and simulation environment for hybrid bond graphs. *Journal of Systems and Control Engineering - Part I 216* (2002), 35–46.

[112] Niles, I., and Pease, A. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems* (USA, 2001), ACM, pp. 2–9.

[113] Noskievič, P. *Modelování a identifikace systémů*. Montanex, 1999.

[114] Novák, P. Modelování environmentálních veličin pasivních domů pro podporu návrhu řídicích systémů, 2009. Available online: http://cvutmedialab.cz/fileadmin/user_upload/HVAC_zprava.pdf.

[115] Novák, P. Modelování environmentálních veličin pasivních domů [technical report in the Czech language]. In *Pasivní domy 2009* (Brno, 2009), Centrum pasivního domu.

[116] Novák, P. Control algorithms for environmental parameters of passive houses. Master's thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, 2010.

[117] Novák, P. Modelling and control of environmental parameters of passive houses. In *Proc. of the 14th International Student Conference on Electrical Engineering POSTER* (2010).

[118] Novák, P., Kadera, P., Jirkovský, V., Vrba, P., and Biffl, S. Engineering of coupled simulation models for mechatronic systems. In *Service Orientation in Holonic and Multi-agent Manufacturing*, T. Borangiu, A. Thomas, and D. Trentesaux, Eds. Springer International Publishing, 2015, pp. 3–11.

[119] Novák, P., Kadera, P., Vrba, P., and Šindelář, R. Architecture of a multi-agent system for SCADA level in smart distributed environments. In *18th IEEE Conference on Emerging Technologies Factory Automation (ETFA 2013)* (Sept 2013), pp. 1–8.

[120] Novák, P., Melik-Merkumians, M., Steinegger, M., Moser, T., Šindelář, R., and Zoitl, A. Semantic runtime interface description based on engineering knowledge. In *Proc. of the 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)* (May 2012).

[121] Novák, P., and Mordinyi, R. Runtime integration of industrial automation system tools based on engineering service bus. In *Proc. of the 2015 IEEE International Conference on Industrial Technology (ICIT)* (Seville, 2015).

[122] Novák, P., Serral, E., Mordinyi, R., and Šindelář, R. Integrating heterogeneous engineering knowledge and tools for efficient industrial simulation model support. *Advanced Engineering Informatics* (2015).

[123] Novák, P., and Šindelář, R. Applications of ontologies for assembling simulation models of industrial systems. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops* (Hersonissos, 2011), Springer, Dordrecht, pp. 148–157.

[124] Novák, P., and Šindelář, R. Integrated design of simulation models for passive houses. In *CEUR workshop proceedings* (2011), vol. 821, pp. 13–18.

[125] Novák, P., and Šindelář, R. Design and Verification of Simulation Models of Passive Houses. In *IEEE International Conference on Emerging Technologies and Factory Automation*

(ETFA 2012) (2012).

[126] Novák, P., and Šindelář, R. Component-based design of simulation models utilizing bond-graph theory. In *Proceedings of the 19th IFAC World Congress (IFAC 2014)* (Cape Town, 2014), pp. 1–6.

[127] Novák, P., Šindelář, R., and Mordinyi, R. Integration framework for simulations and SCADA systems. *Simulation Modelling Practice and Theory 47* (September 2014), 121–140.

[128] Noy, N. F. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record 33 No. 4* (Dec. 2004).

[129] Noy, N. F., Doan, A., and Halevy, A. Y. Semantic integration. *AI Magazine 26*, 1 (2005), 7–9.

[130] Noy, N. F., and McGuinness, D. L. Ontology development 101: A guide to creating your first ontology [online]. Cited on 2012-07-23. Available: http://protege.stanford.edu/publications/ontology_development/ontology101.pdf.

[131] Obitko, M. *Translations between Ontologies in Multi-Agent Systems*. PhD thesis, Czech Technical University in Prague, 2007.

[132] Obitko, M., Jirkovský, V., and Bezdíček, J. Big data challenges in industrial automation. In *Industrial Applications of Holonic and Multi-Agent Systems*, V. Mařík, J. L. Martinez Lastra, and P. Skobelev, Eds., vol. 8062 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 305–316.

[133] Obitko, M., and Mařík, V. Ontologies for multi-agent systems in manufacturing domain. In *Proc. of the 13th International Workshop on Database and Expert Systems Applications* (Sept. 2002), pp. 597– 602.

[134] OMG Systems Modeling Language (OMG SysML^{TM}) [online]. Version 1.3, June 2012. Available: http://www.sysml.org/docs/specs/OMGSysML-v1.3-12-06-02.pdf.

[135] Papazoglou, M. P. Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003)* (Dec 2003), pp. 3–12.

[136] Passivhaus Institut: Passive House Planning Package (PHPP) [online]. Cited on 2016-07-18. Available online: http://passivehouse.com/04_phpp/04_phpp.htm.

[137] Passivhaus Institut: Passive House Requirements [online]. Cited on 2016-07-18. Available online: http://passivehouse.com/02_informations/02_passive-house-requirements/02_passive-house-requirements.htm.

[138] Pêcheux, F., Allard, B., Lallement, C., Vachoux, A., and Morel, H. Modeling and simulation of multi-discipline systems using bond graphs and VHDL-AMS. In *Proceedings of the International Conference on Bond Graph Modeling and Simulation (ICBGM)* (2005).

[139] Phillips, N. B. P., Gann, J. O., and Irving, M. The SIMIAN architecture – An object-orientated framework for integrated power system modelling, analysis and control. In *Fourth International Conference on Power System Control and Management* (1996), pp. 148–153.

[140] Popovici, M., Muraru, M., Agache, A., Negreanu, L., Giumale, C., and Dobre, C. An ontology-based dynamic service composition framework for intelligent houses. In *10th International Symposium on Autonomous Decentralized Systems (ISADS 2011)* (March 2011), pp. 177–184.

[141] Rebstock, M., Fengel, J., and Paulheim, H. *Ontologies-Based Business Integration*. Springer-Verlag Berlin Heidelberg, 2008.

[142] Reinisch, C., Granzer, W., Praus, F., and Kastner, W. Integration of heterogeneous building automation systems using ontologies. In *34th Annual Conference of IEEE Industrial Electronics (IECON 2008)* (November 2008), pp. 2736–2741.

[143] Rocca, G. L. Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. *Advanced Engineering Informatics 26*, 2

(2012), 159–179. Special issue on Knowledge based engineering to support complex product design.

[144] Rosenberg, R. C., and Karnopp, D. *Introduction to Physical System Dynamics*. McGraw-Hill Publishing Company, New York, 1983.

[145] Russell, S. J., and Norvig, P. *Artificial intelligence: a modern approach*, 3rd ed. Upper Saddle River: Prentice Hall, 2013. Prentice Hall series in artificial intelligence.

[146] Sabou, M. An introduction to semantic web technologies (accepted). In *Semantic Web Technologies in Intelligent Engineering Applications*, S. Biffl and M. Sabou, Eds. Springer-Verlag Berlin Heidelberg, 2016.

[147] Sabou, M., Ekaputra, F. J., Kovalenko, O., and Biffl, S. Supporting the engineering of cyber-physical production systems with the AutomationML Analyzer. In *Proc. of the Cyber Physical Production System Workshop (CPPS)* (Vienna, Austria, 2016).

[148] Schmidt, N., Lüder, A., Steininger, H., and Biffl, S. Analyzing requirements on software tools according to the functional engineering phase in the technical systems engineering process. In *Proc. of the 19th IEEE International Conference on Emerging Technology and Factory Automation (ETFA 2014)* (2014), pp. 1–8.

[149] Serral, E., Mordinyi, R., Kovalenko, O., Winkler, D., and Biffl, S. Evaluation of semantic data storages for integrating heterogenous disciplines in automation systems engineering. In *39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)* (Vienna, Austria, 2013), IEEE, pp. 6858–6865.

[150] Shadbolt, N., Hall, W., and Berners-Lee, T. The semantic web revisited. *Intelligent Systems, IEEE 21*, 3 (Jan 2006), 96–101.

[151] Silver, G. A., Bellipady, K. R., Miller, J. A., Kochut, K. J., and York, W. Supporting interoperability using the Discrete-event Modeling Ontology (DeMO). In *Proceedings of the 2009 Winter Simulation Conference (WSC)* (Dec. 2009), pp. 1399 –1410.

[152] Silver, G. A., Hassan, O.-H., and Miller, J. A. From domain ontologies to modeling ontologies to executable simulation models. In *Proceedings of the 2007 Winter Simulation Conference (WSC)* (Dec. 2007), pp. 1108–1117.

[153] Silver, G. A., Lacy, L. W., and Miller, J. A. Ontology based representations of simulation models following the process interaction world view. In *Proceedings of the 2006 Winter Simulation Conference (WSC)* (Dec. 2006), pp. 1168 –1176.

[154] Stoermer, C., and Tibba, G. Powertrain co-simulation using AUTOSAR and the functional mockup interface standard. In *Proc. of the 51st ACM/EDAC/IEEE Design Automation Conference (DAC 2014)* (June 2014).

[155] Su, C.-L., Yin, S.-A., and Chang, Y.-C. Computer simulations of an integrated distribution information system. In *IEEE Power Engineering Society Winter Meeting 2002* (2002), vol. 1, pp. 690–694.

[156] Tichý, P., Kadera, P., Staron, R. J., Vrba, P., and Maříк, V. Multi-agent system design and integration via agent development environment. *Engineering Applications of Artificial Intelligence 25*, 4 (2012), 846 – 852. Special Section: Dependable System Modelling and Analysis.

[157] Turner, M., Budgen, D., and Brereton, P. Turning software into a service. *Computer 36*, 10 (Oct. 2003), 38–44.

[158] Uhrmacher, A. M., and Weyns, D., Eds. *Multi-Agent Systems: Simulation and Applications*. Computational Analysis, Synthesis, and Design of Dynamic Models Series. CRC Press, Taylor and Francis Group, 2009.

[159] Unver, H. O. An isa-95-based manufacturing intelligence system in support of lean initiatives. *The International Journal of Advanced Manufacturing Technology* (2012), 1–14.

[160] VDI 3633: Simulation von logistik-, materialfluss- und produktionssystemen. verein deutscher

ingenieure. Available online: https://www.vdi.de/3633.

[161] Verhagen, W. J. C., Bermell-Garcia, P., van Dijk, R. E. C., and Curran, R. A critical review of knowledge-based engineering: An identification of research challenges. *Advanced Engineering Informatics 26*, 1 (2012), 5–15. Network and Supply Chain System Integration for Mass Customization and Sustainable Behavior.

[162] Villberg, A. Design challenges of an ontology-based modelling and simulation environment. Master's thesis, Helsinki University of Technology, 2007.

[163] Vrba, P. Review of industrial applications of multi-agent technologies. In *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*, T. Borangiu, A. Thomas, and D. Trentesaux, Eds., vol. 472 of *Studies in Computational Intelligence*. Springer-Verlag Berlin Heidelberg, 2013, pp. 327–338.

[164] Vrba, P., Harcuba, O., Klíma, M., and Mařík, V. Agent-based production scheduling for aircraft manufacturing ramp-up. In *Proceedings of the 7th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS 2015), Valencia, Spain*, V. Mařík, A. Schirrmann, D. Trentesaux, and P. Vrba, Eds. Springer International Publishing, 2015, pp. 145–156.

[165] Vrba, P., Kadera, P., Myslík, M., and Klíma, M. JBoss ESB Sniffer. In *IEEE 23rd International Symposium on Industrial Electronics (ISIE 2014)* (June 2014), pp. 1724–1729.

[166] Vrba, P., Mařík, V., Siano, P., Leitao, P., Zhabelova, G., Vyatkin, V., and Strasser, T. A review of agent and service-oriented concepts applied to intelligent energy systems. *IEEE Transactions on Industrial Informatics 10*, 3 (Aug 2014), 1890–1903.

[167] Vrba, P., Tichý, P., Mařík, V., Hall, K., Staron, R., Maturana, F., and Kadera, P. Rockwell Automation's Holonic and Multiagent Control Systems Compendium. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 41*, 1 (Jan. 2011), 14 –30.

[168] Šindelář, R., and Novák, P. Framework for simulation integration. In *Proceedings of the 18th IFAC World Congress* (Bologna, 2011), vol. 18, IFAC, pp. 3569–3574.

[169] Šindelář, R., and Novák, P. Simulation integration framework. In *10th IEEE International Conference on Industrial Informatics (INDIN 2012)* (2012), pp. 80–85.

[170] Wang, H., Johnson, A. L., and Bracewell, R. H. The retrieval of structured design rationale for the re-use of design knowledge with an integrated representation. *Advanced Engineering Informatics 26*, 2 (2012), 251–266. Special issue on Knowledge based engineering to support complex product design.

[171] Weiss, G., Ed. *Multiagent Systems*, 2nd ed. Massachusetts Institute of Technology, 2013.

[172] Winkler, D., Moser, T., Mordinyi, R., Sunindyo, W. D., and Biffl, S. Engineering object change management process observation in distributed automation systems projects. In *Proceedings of 18th European System & Software Process Improvement and Innovation (EuroSPI 2011)* (2011), pp. 1–12.

[173] Woods, R. L., and Lawrence, K. L. *Modeling and Simulation of Dynamic Systems*. Prentice-Hall, New Jersey, 1997.

[174] Wooldridge, M., and Jennings, N. Software engineering with agents: pitfalls and pratfalls. *IEEE Internet Computing 3*, 3 (May 1999), 20–27.

Appendix A

Application Example of the Traditional Bond Graph Method for Simulation Design

This section illustrates a design of a bond graph for a very simple electrical system step-by-step as well as its manual transformation to a simulation model in MATLAB-Simulink. The main purpose of this detailed workflow is to simplify the understanding of the extended bond graph method, which is described in Sec. 5.

The exemplary electrical circuit is depicted in Fig. A.1. This electrical schema shows that the circuit consists of a serial connection of a voltage source, a resistor, and an inductor, which are connected to a parallel combination of a capacitor and the second resistor. The example in such a form was selected due to the combination of both types of energy stores (i.e., inductance and capacitance), energy dissipation (i.e., two resistors), and an energy source (i.e., the voltage source).

The first and second steps of creating a simulation model with bond graphs (see the list of process steps presented in Sec. 2.3.4) represent the creation of the structure of the bond graph. This issue is based on the generation of components adequate to sub-systems or devices. According to the system type, 0-junctions and 1-junctions have to be created to model parallel and serial connections, as it has been already discussed in Sec. 2.3.3. The created structure of the bond graph for the exemplary electrical circuit from Fig. A.1 is depicted in Fig. A.2. Since it is an electrical circuit, parallel connections are modeled by 0-junctions, whereas serial connections are modeled by 1-junctions.

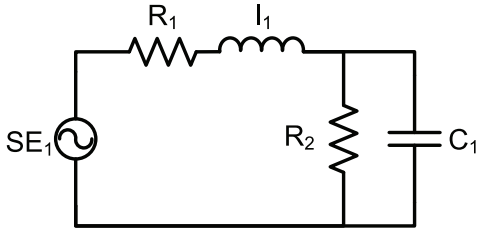

Figure A.1: Exemplary electrical circuit including a voltage source, resistors R_1 and R_2, and both accumulators of energy – a capacitor C and an inductor I.

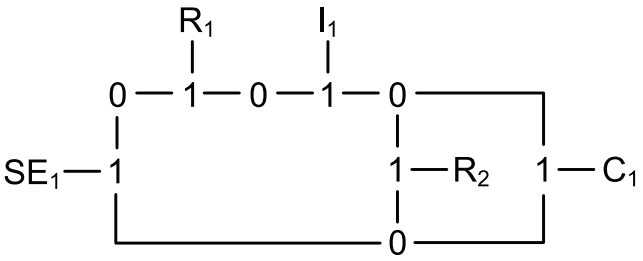

Figure A.2: Structure of the bond graph for the electrical circuit.

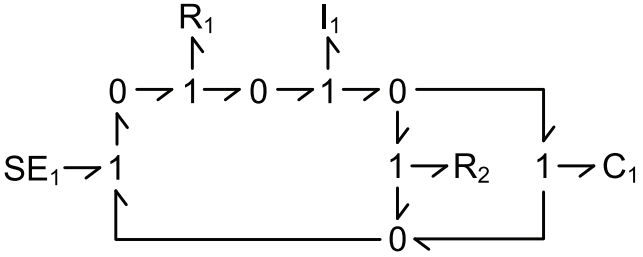

Figure A.3: Structure of the bond graph for the electrical circuit with the assigned power direction.

The third step is the assignment of the positive power direction. Applying the rules summarized in Sec. 2.3.4, the result of this step is depicted in Fig. A.3. The causality assignment is crucial for the interpretation of simulation results by humans, because it defines the sign convention of the final simulation model of this electrical circuit.

The fourth step is the exclusion of the reference junction. In case of electrical systems, the reference junction is frequently the ground voltage. Therefore, the ground was selected and excluded from the bond graph. The obtained bond graph is depicted in Fig. A.4. This graph could be used as a basis for engineering simulation models for Modelica or other equation-based languages. However, a signal-oriented simulation model in MATLAB-Simulink is required and thus causality has to be assigned to the bond graph in order to be able to recognize input and output signals for all component blocks used for simulation.

The fifth step deals with the reduction of the bond graph in order to make the transformation from the representation by means of bond graphs to the mathematical description easier. Applying the rules for reduction of the graph that are defined by the bond-graph theory, we obtain the bond graph depicted in Fig. A.5, where we can see the significantly reduced number of junctions and power bonds. However, this reduction does not influence computational complexity significantly, hence it is not considered as mandatory.

The sixth step is focused on the assignment of causality. Using the rules from Sec. 2.3.4, causality was assigned to the source of effort (i.e., to the voltage source) first. Subsequently, integral causalities were assigned to the capacitor C_1 and inductor I_1. Then the causality

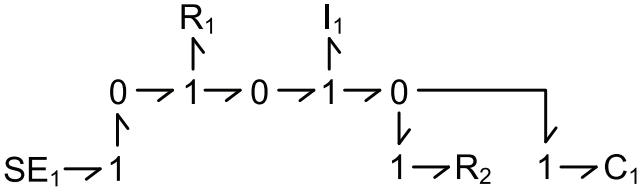

Figure A.4: Structure of the bond graph for the electrical circuit with excluded reference junction.

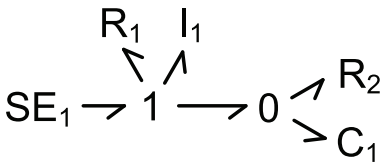

Figure A.5: The simplified bond graph for the electrical circuit prepared for assigning of the causality.

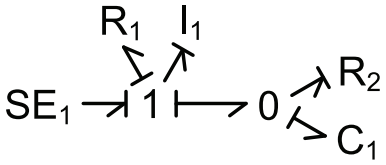

Figure A.6: The complete bond graph for the electrical circuit with the assigned causality.

was assigned to the power bond connecting the 1-junction and the 0-junction. Finally, resistors R_1 and R_2 were assigned with the remaining causality possibilities. The outcome of this causality assignment step is depicted in Fig. A.6.

Based on the completed bond graph depicted in Fig. A.6, the mathematical description of the system can be obtained as the seventh step of the simulation model design process. We start at the accumulators of the energy, which are in this particular case the capacitor and the inductor. For each of them, we can express the equation representing the adding and subtracting signals according to the assigned causality and power direction.

$$u_{C_1}(t) = \frac{1}{C_1} \int_0^t i_{I_1}(\tau) - i_{R_2}(\tau) d\tau \tag{A.1}$$

$$i_{I_1}(t) = \frac{1}{L_1} \int_0^t u_{SE_1}(\tau) - u_{R_1}(\tau) - u_{C_1}(\tau) d\tau \tag{A.2}$$

Considering voltage on the resistor R_1 and electrical current through the resistor R_2, we can get the integral description of this electrical circuit:

$$u_{R_1}(t) = R_1 \cdot i_{I_1}(t) \tag{A.3}$$

$$i_{R_2}(t) = \frac{u_{C_1}(t)}{R_2} \tag{A.4}$$

The integral description of the electrical circuit can be summarized as follows:

$$u_{C_1}(t) = \frac{1}{C_1} \int_0^t i_{L_1}(\tau) - \frac{u_{C_1}(\tau)}{R_2} d\tau \tag{A.5}$$

$$i_{I_1}(t) = \frac{1}{L_1} \int_0^t u_{SE_1}(\tau) - R_1 \cdot i_{I_1}(\tau) - u_{C_1}(\tau) d\tau \tag{A.6}$$

To get the state-space representation but also to get the simulation model in the equation-oriented simulator, it is beneficial to transform these equations to the differential form. The

whole circuit behavior can be expressed by the following equations:

$$\dot{u}_{C_1}(t) = \frac{1}{C_1}(i_{L_1}(t) - \frac{u_{C_1}(t)}{R_2}) \tag{A.7}$$

$$\dot{i}_{I_1}(t) = \frac{1}{L_1}(u_{SE_1}(t) - R_1 \cdot i_{I_1}(t) - u_{C_1}(t)) \tag{A.8}$$

This description can be expressed in the matrix form, respecting the state-space system model according to Eq. 2.1:

$$\begin{aligned} \begin{pmatrix} \dot{u}_{C_1}(t) \\ \dot{i}_{I_1}(t) \end{pmatrix} &= A \begin{pmatrix} u_{C_1}(t) \\ i_{I_1}(t) \end{pmatrix} + B \cdot u_{SE_1}(t) \\ &= \begin{pmatrix} -\frac{1}{C_1 R_2} & \frac{1}{C_1} \\ -\frac{1}{L_1} & -\frac{R_1}{L_1} \end{pmatrix} \begin{pmatrix} u_{C_1}(t) \\ i_{I_1}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{L_1} \end{pmatrix} u_{SE_1}(t) \end{aligned} \tag{A.9}$$

$$\begin{aligned} y(t) &= C \begin{pmatrix} u_{C_1}(t) \\ i_{L_1}(t) \end{pmatrix} + D \cdot u_{SE_1}(t) \\ &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} u_{C_1}(t) \\ i_{I_1}(t) \end{pmatrix} + 0 \cdot u_{SE_1}(t) \end{aligned} \tag{A.10}$$

The matrices A, B, C, and D have the following form, which will be later parameterized with specific values for comparing responses of the obtaining mathematical model manually and automatically with the proposed method:

$$A = \begin{pmatrix} -\frac{1}{C_1 R_2} & \frac{1}{C_1} \\ -\frac{1}{L_1} & -\frac{R_1}{L_1} \end{pmatrix} B = \begin{pmatrix} 0 \\ \frac{1}{L_1} \end{pmatrix} C = \begin{pmatrix} 1 & 0 \end{pmatrix} D = 0 \tag{A.11}$$

Since this specific circuit is a continuous time-invariant linear system, its state-space description can be transformed to the equivalent description by the transfer function according to Eq. 2.2.

$$\begin{aligned} G(s) &= C(sI - A)^{-1}B + D \\ &= \begin{pmatrix} 1 & 0 \end{pmatrix} \left(s \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} -\frac{1}{C(C_1)R(R_2)} & \frac{1}{C(C_1)} \\ -\frac{1}{L_1} & -\frac{R(R_1)}{L(I_1)} \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ \frac{1}{L(I_1)} \end{pmatrix} + 0 \\ &= \frac{R_2}{s^2 + \frac{R_1 R_2 C_1 + L_1}{R_2 L_1 C_1} s + \frac{R_1 + R_2}{R_2 L_1 C_1}} \end{aligned} \tag{A.12}$$

When a signal-oriented simulator is used, the mathematical equations should be transformed into such a form that is compliant for the simulator as the eighth process step. Since this particular electrical circuit is linear and time-invariant, we can use simulation blocks in the standard Simulink library implementing the state-space model or the transfer-function. Both implementations of simulation models are depicted in Fig. A.7 and A.8. The strong point of such an implementation is its simplicity, because mathematical description of the physical system is directly passed to the simulation without any complicated or manual transformations. However, the weak point is the restriction on linear time-invariant (LTI)

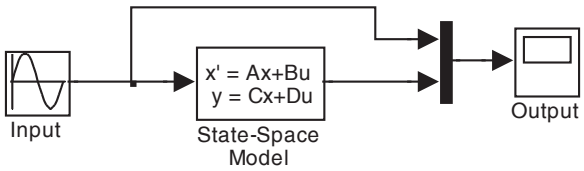

Figure A.7: Simulation model for the electrical circuit created in MATLAB-Simulink manually. The model behavior is implemented as a state-space model relying on matrices A.11.

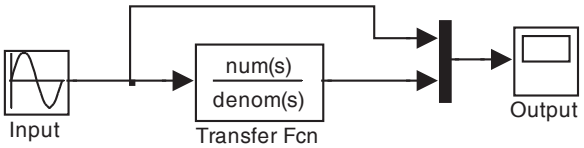

Figure A.8: Simulation model for the electrical circuit created in MATLAB-Simulink manually. The model behavior is implemented as a transfer function relying on the expression A.12. Non-zero initial conditions cannot be set in the transfer function block.

systems. However, large-scale industrial systems in practice are not LTI systems. Therefore, another approach has to be used. In addition, this representation is hard to be modified (e.g., when a new device is added, the entire system description has to be redefined) as well as split and re-connected into co-simulation or hardware-in-the-loop schemas. An important disadvantage of the representation utilizing transfer functions is that simulation blocks implementing the transfer functions do not support setting of non-zero initial conditions frequently.

A more complicated realization of the simulation model is depicted in Fig. A.9. This simulation schema poses a typical approach, how mathematical equations are represented in signal-oriented simulators. Its benefit is a visible routing of signals leading to solve the set of differential equations. Furthermore, this simulation schema can be used for co-simulation consisting of several simulation modules as well as for a hardware-in-the-loop operation modus. On the contrary to the schema based on the transfer function (see Fig. A.8), non-zero initial conditions can be set easily. However, it is very complicated to create such a schema, to test it and to debug it. It would be beneficial to have a tool support for generating such a schema, in order to reduce human effort. Moreover, it is difficult to re-design or reuse patterns from this simulation schema, for example when a specific part of the real system is changed. Therefore, the aforementioned tool support should face these aspects, too.

The further section is focused on the tool support for bond-graph modeling, which is a good candidate for addressing the aforementioned problems and needs. However, we will see that this tool support is not satisfactory enough to tackle the above stated requirements dealing with the simulation artifacts reuse and re-design support.

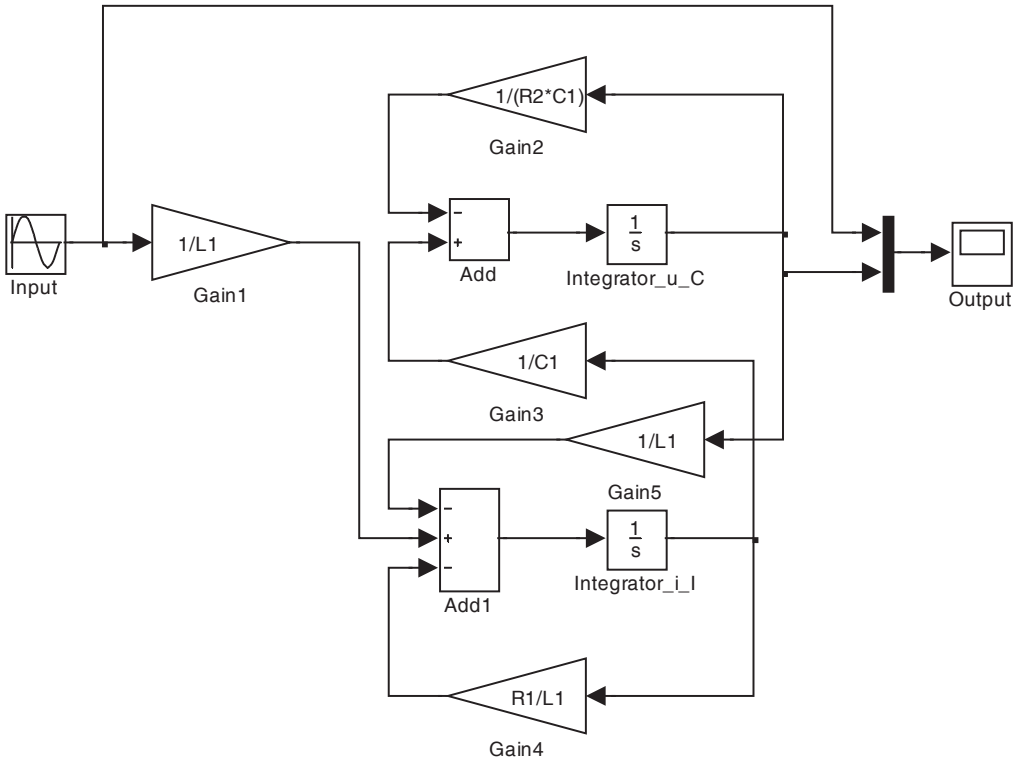

Figure A.9: Simulation model for the electrical circuit created in Matlab-Simulink manually. The model behavior is wired in a fully signal-oriented form, which is typical for Matlab-Simulink.

Appendix B

Simulation Blocks of the Mechatronic Library

The entire simulation library implemented in MATLAB-Simulink is depicted in Fig. B.1. It includes simulation blocks for modeling 0-junctions and 1-junctions as well as for plotting simulation results (see the common blocks on the upper-right part of Fig. B.1). The library also includes two sets of simulation blocks of the electrical system engineering discipline and the hydraulic system engineering discipline. The simulation blocks included in the mechatronic library are discussed in details in the following sections.

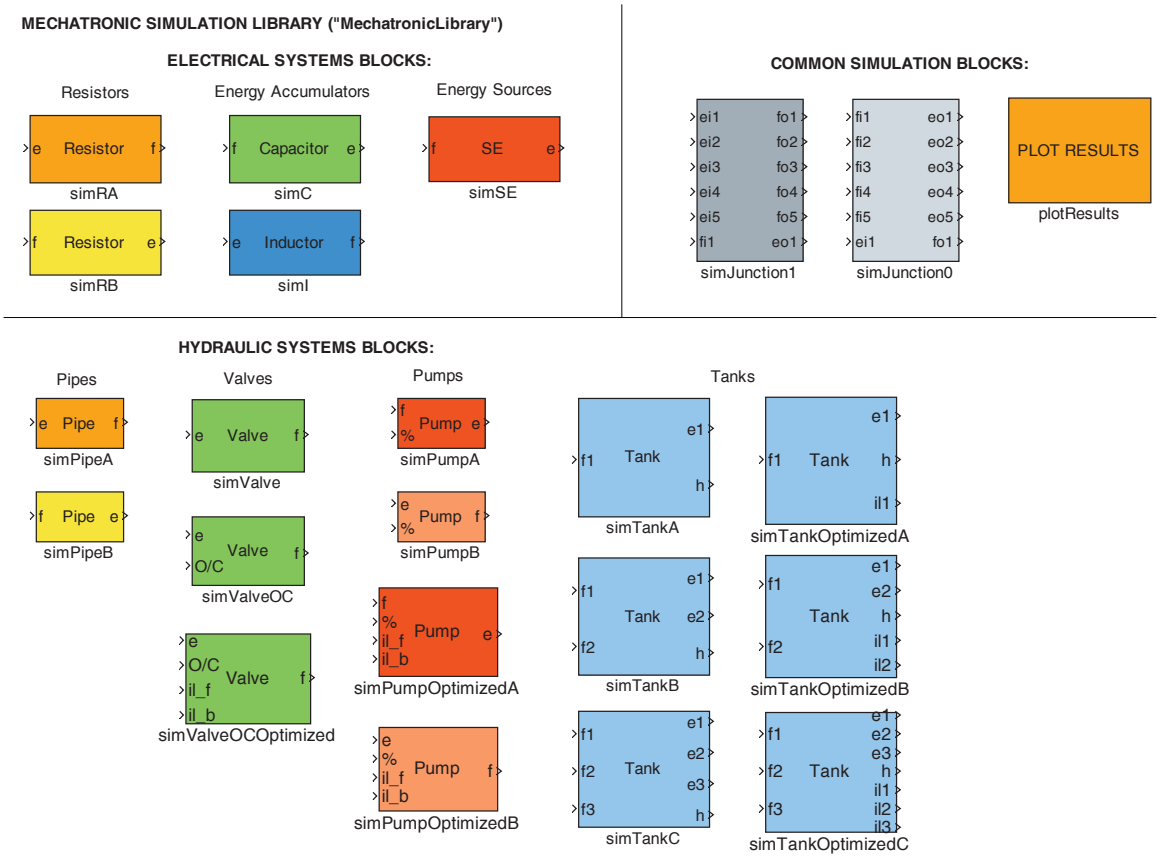

Figure B.1: Library with mechatronic components for the simulation model generation in MATLAB-Simulink.

Simulation Block "simRA" Approximating a Resistor

Resistors realize resistance in electrical systems. The interface of this simulation block is summarized in Tab. B.1, the parameters of this simulation block are explained in Tab. B.2, and the internal representation of the simulation block is depicted in Fig. B.2.

Table B.1: Simulation block interface

Interface	Signal
Input	Effort
Output	Flow

Table B.2: Simulation parameters of the block

Parameter name	Parameter meaning
R	Resistance (Ω)

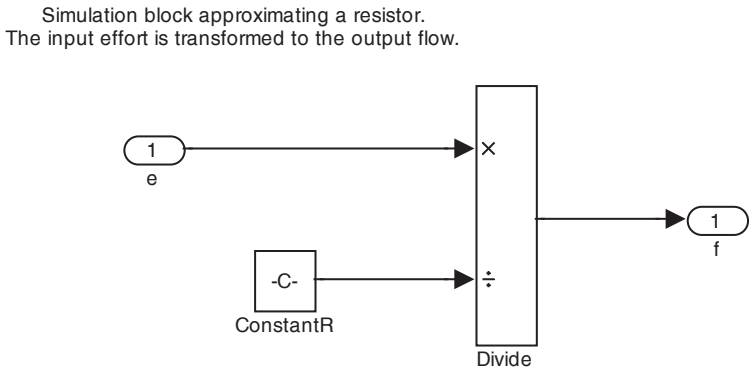

Figure B.2: Internal representation of the simulation component Resistor "simRA".

Simulation Block "simRB" Approximating a Resistor

The simulation block "simRB" is the second block expressing the resistance in electrical systems. The interface of this simulation block is summarized in Tab. B.3, the parameters of this simulation block are explained in Tab. B.4, and the internal representation of the simulation block is depicted in Fig. B.3.

Table B.3: Simulation block interface

Interface	Signal
Input	Flow
Output	Effort

Table B.4: Simulation parameters of the block

Parameter name	Parameter meaning
R	Resistance (Ω)

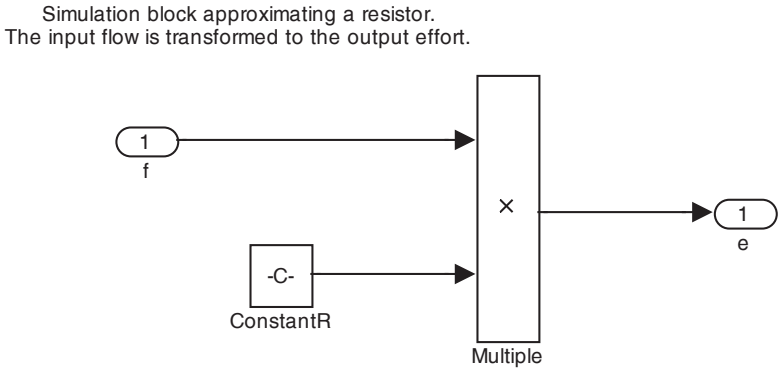

Figure B.3: Internal representation of the simulation component Resistor "simRB".

Simulation Block "simC" Approximating a Capacitor

The simulation block "simC" models capacitance in electrical systems. The interface of this simulation block is summarized in Tab. B.5, the parameters of this simulation block are explained in Tab. B.6, and the internal representation of the simulation block is depicted in Fig. B.4.

Table B.5: Simulation block interface

Interface	Signal
Input	Flow
Output	Effort

Table B.6: Simulation parameters of the block

Parameter name	Parameter meaning
C	Capacitance (F)

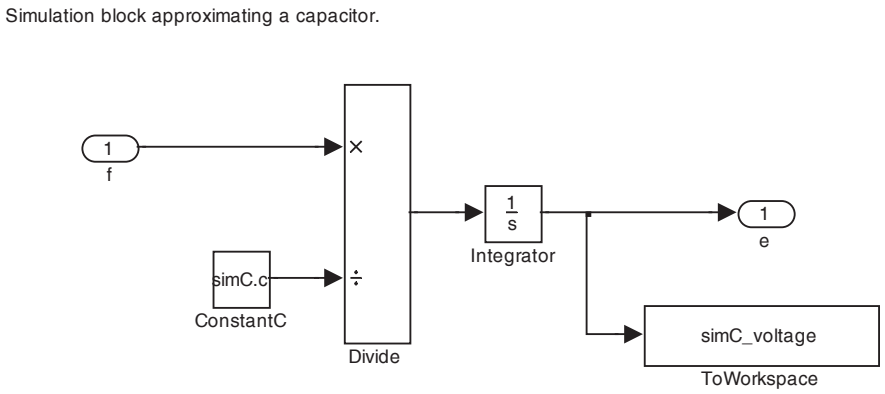

Figure B.4: Internal representation of the simulation component Capacitor "simC".

Simulation Block "simI" Approximating an Inductor

The simulation block "simI" models inductance in electrical systems. The interface of this simulation block is summarized in Tab. B.7, the parameters of this simulation block are explained in Tab. B.8, and the internal representation of the simulation block is depicted in Fig. B.5.

Table B.7: Simulation block interface

Interface	Signal
Input	Effort
Output	Flow

Table B.8: Simulation parameters of the block

Parameter name	Parameter meaning
I	Inductance (H)

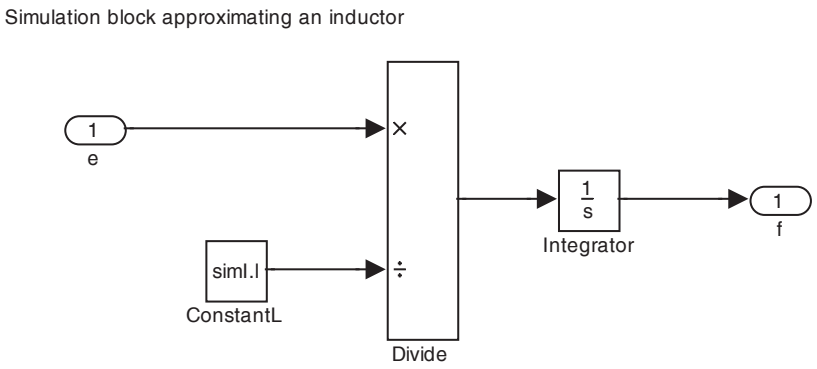

Figure B.5: Internal representation of the simulation component Inductor "simI".

Simulation Block "simSE" Approximating a Voltage Source

The simulation block "simSE" models a source of alternating voltage in electrical systems. The interface of this simulation block is summarized in Tab. B.9, the parameters of this simulation block are explained in Tab. B.10, and the internal representation of the simulation block is depicted in Fig. B.6.

Table B.9: Simulation block interface

Interface	Signal
Input	Flow
Output	Effort

Table B.10: Simulation parameters of the block

Parameter name	Parameter meaning
V	Voltage magnitude (V)
f	Frequency (Hz)

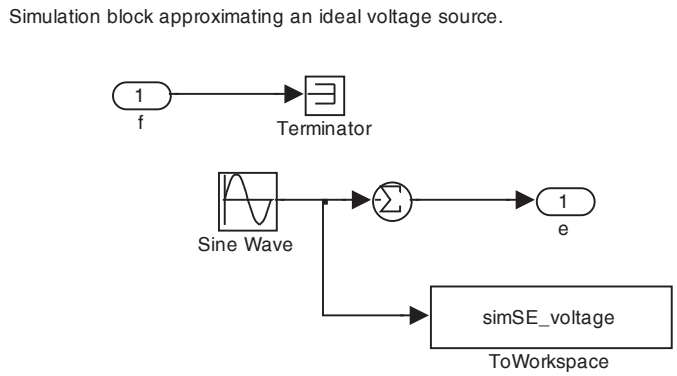

Figure B.6: Internal representation of the simulation component source of effort "simSE".

Simulation Block "simJ0" Approximating a 0-Junction

The simulation block "simJ0" models a 0-junction. The interface of this simulation block is summarized in Tab. B.11 and the parameters of this simulation block are explained in Tab. B.12. The sign corrections being parameters of this block are constants having one of the values $\{-1; +1\}$ to implement sign convention for the connected power bonds. The internal representation of the simulation block is depicted in Fig. B.7.

Table B.11: Simulation block interface

Interface	Signal
Input_1	Flow
Input_2	Flow
Input_3	Flow
Input_4	Flow
Input_5	Flow
Input_6	Effort
Output_1	Effort
Output_2	Effort
Output_3	Effort
Output_4	Effort
Output_5	Effort
Output_6	Flow

Table B.12: Simulation parameters of the block

Parameter name	Parameter meaning
s_1	Sign correction $(-)$
s_2	Sign correction $(-)$
s_3	Sign correction $(-)$
s_4	Sign correction $(-)$
s_5	Sign correction $(-)$

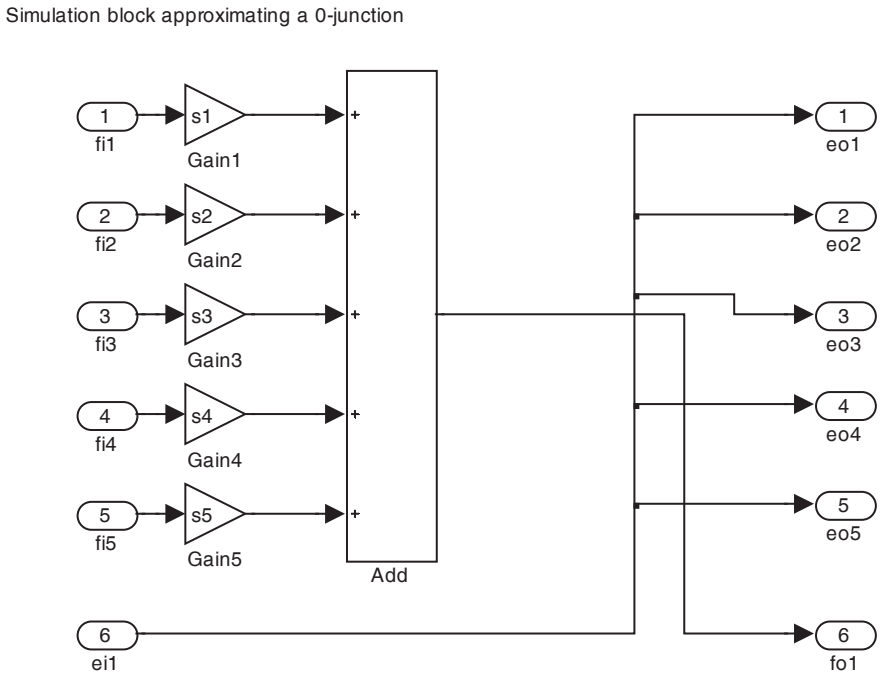

Figure B.7: Internal representation of the simulation component 0-junction "simJ0".

Simulation Block "simJ1" Approximating an 1-Junction

The simulation block "simJ1" models an 1-junction. The interface of this simulation block is summarized in Tab. B.13 and the parameters of this simulation block are explained in Tab. B.14. The sign corrections being parameters of this block are constants having one of the values $\{-1; +1\}$ to implement sign convention for the connected power bonds. The internal representation of the simulation block is depicted in Fig. B.8.

Table B.13: Simulation block interface

Interface	Signal
Input_1	Effort
Input_2	Effort
Input_3	Effort
Input_4	Effort
Input_5	Effort
Input_6	Flow
Output_1	Flow
Output_2	Flow
Output_3	Flow
Output_4	Flow
Output_5	Flow
Output_6	Effort

Table B.14: Simulation parameters of the block

Parameter name	Parameter meaning
s_1	Sign correction $(-)$
s_2	Sign correction $(-)$
s_3	Sign correction $(-)$
s_4	Sign correction $(-)$
s_5	Sign correction $(-)$

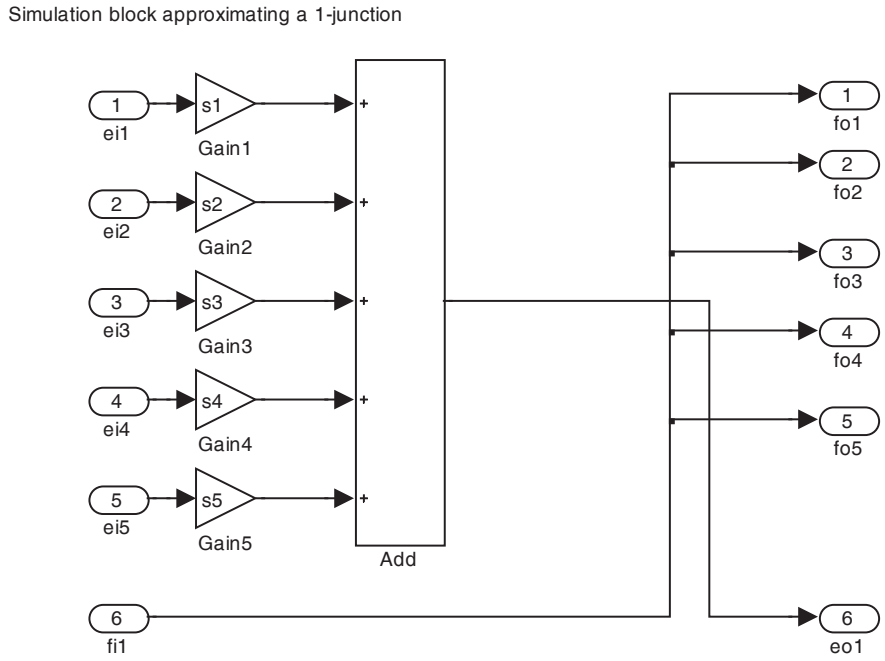

Figure B.8: Internal representation of the simulation component 1-junction "simJ1".

Simulation Block "simPipeA" Approximating a Real Pipe

Pipes realize resistance in hydraulic systems. In the addressed use-case, the inductance of the pipe is neglected. In case of very long pipes with significant value of liquid flow, this feature should be considered, however, it leads to a 2-port component. The interface of this simulation block is summarized in Tab. B.15, the parameters of this simulation block are explained in Tab. B.16, and the internal representation of the simulation block is depicted in Fig. B.9.

Table B.15: Simulation block interface

Interface	Signal
Input	Effort
Output	Flow

Table B.16: Simulation parameters of the block

Parameter name	Parameter meaning
L	Length of the pipe (m)
D	Diameter of the pipe (m)

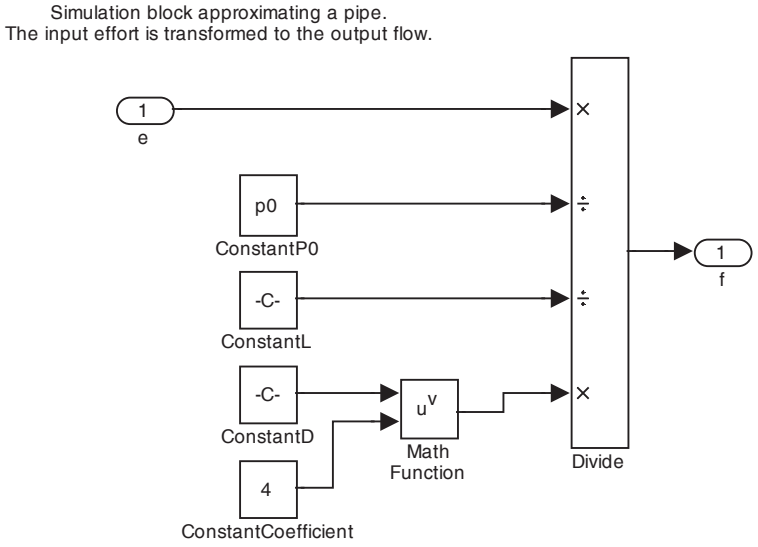

Figure B.9: A simulation block modeling a pipe in hydraulic systems.

Simulation Block "simPipeB" Approximating a Real Pipe

Whereas the simulation block "simPipeA" block transforms effort to flow (i.e., the pressure difference to the volumetric flow), the simulation block "simPipeB" functions vice-versa. The interface of this simulation block is summarized in Tab. B.17, the parameters of this simulation block are explained in Tab. B.18, and the internal representation of the simulation block is depicted in Fig. B.10.

Table B.17: Simulation block interface

Interface	Signal
Input	Flow
Output	Effort

Table B.18: Simulation parameters of the block

Parameter name	Parameter meaning
L	Length of the pipe (m)
D	Diameter of the pipe (m)

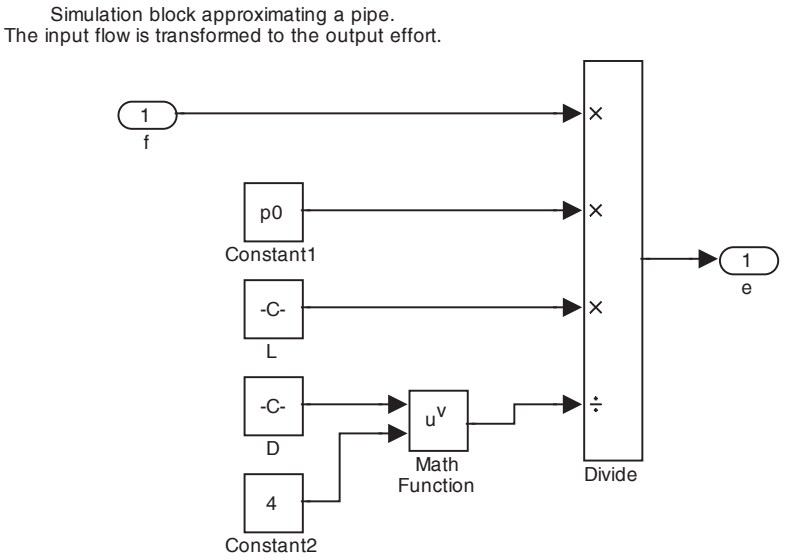

Figure B.10: A simulation block modeling a pipe in hydraulic systems.

Simulation Block "simValve" Approximating a Real Valve

The simulation block "simValveA" is one of the simulation blocks modeling a valve, in particular the easiest one. The interface of this simulation block is summarized in Tab. B.19. The dual representation would not be applicable for the entirely closed position of the valve, thus other valves in the library do not differ in assignment of signal ports to the hydraulic power port, but they differ in ways of the openness setting. The parameters of this simulation block are explained in Tab. B.20, where we can see that the openness ratio is given as a parameter and thus it is constant along the entire duration of the simulation. The internal representation of the simulation block is depicted in Fig. B.11.

Table B.19: Simulation block interface

Interface	Signal
Input_1	Effort
Output_1	Flow

Table B.20: Simulation parameters of the block

Parameter name	Parameter meaning
L	Length of the valve (m)
D	Diameter of the valve (m)
oc	Openness of the valve (-)

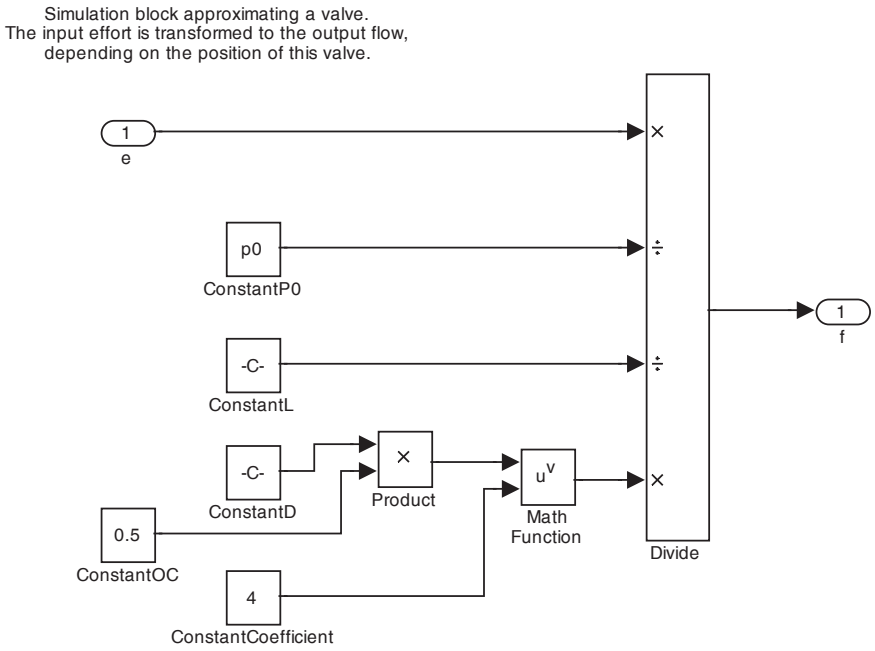

Figure B.11: A simulation block modeling a valve in hydraulic systems.

Simulation Block "simValveOC" Approximating a Real Valve

The simulation block "simValveOC" is a simulation block modeling a valve. It is featured with an input signal representing the openness of the valve, which has to lie in the interval between 0 (i.e., the valve is closed) and 1 (i.e., the valve is open). The interface of this simulation block is summarized in Tab. B.21. The parameters of this simulation block are explained in Tab. B.22, and the internal representation of the simulation block is depicted in Fig. B.12.

Table B.21: Simulation block interface

Interface	Signal
Input_1	Effort
Input_2	Open signal (0–1)
Output_1	Flow

Table B.22: Simulation parameters of the block

Parameter name	Parameter meaning
L	Length of the valve (m)
D	Diameter of the valve (m)

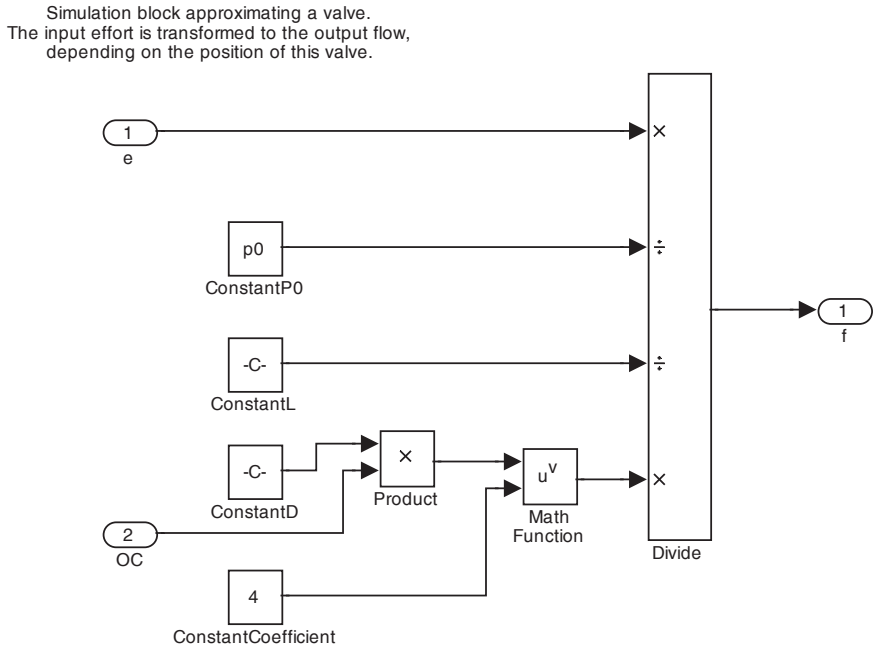

Figure B.12: A simulation block modeling a valve in hydraulic systems.

Simulation Block "simValveOCOptimized" Approximating a Real Valve

The simulation block "simValveOCOptimized" is a simulation block modeling a valve, which has openness as an input signal and which is optimized in terms of the optimized constraint handling as it was discussed in Sec. 5.13. The interface of this simulation block is summarized in Tab. B.23. The parameters of this simulation block are explained in Tab. B.24, and the internal representation of the simulation block is depicted in Fig. B.13.

Table B.23: Simulation block interface

Interface	Signal
Input_1	Effort
Input_2	Open signal (0–1)
Input_3	Inter-locking for the forward flow
Input_4	Inter-locking for the backward flow
Output_1	Flow

Table B.24: Simulation parameters of the block

Parameter name	Parameter meaning
L	Length of the valve (m)
D	Diameter of the valve (m)

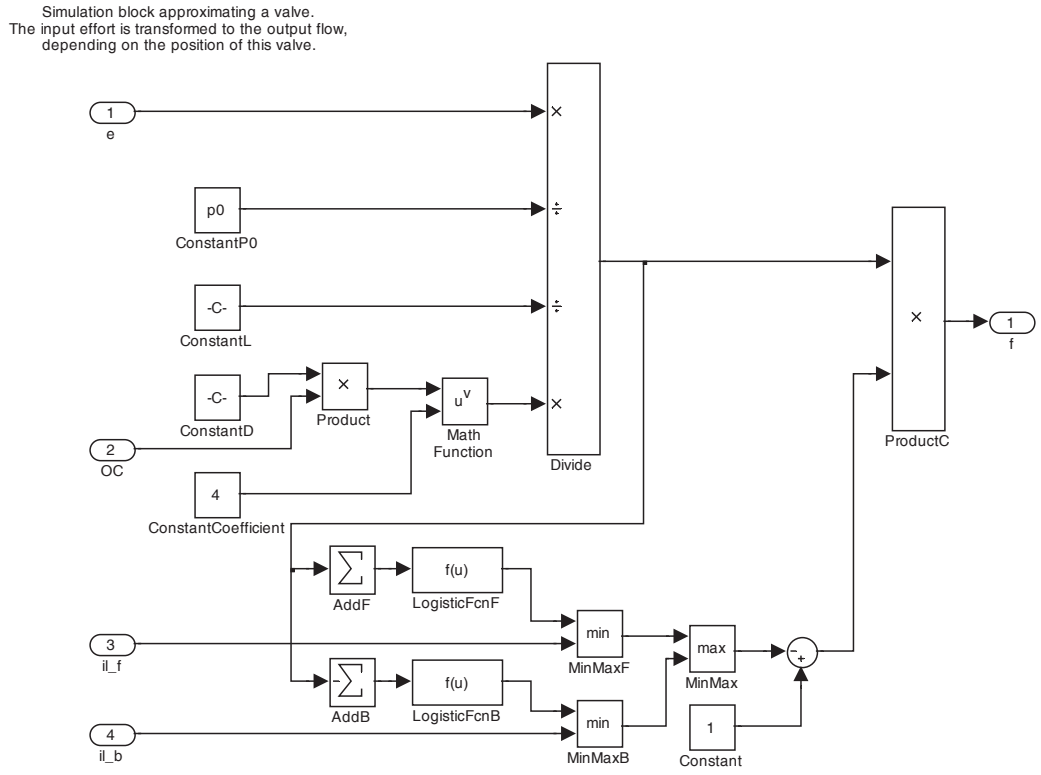

Figure B.13: A simulation block modeling a valve in hydraulic systems.

Simulation Block "simPumpA" Approximating a Real Pump

The simulation block "simPumpA" models a pump as an ideal source of effort. The interface of this simulation block is summarized in Tab. B.25. Parameters of this simulation block are explained in Tab. B.26. The internal representation of the simulation block is depicted in Fig. B.14.

Table B.25: Simulation block interface

Interface	Signal
Input_1	Flow (in the role of a formal input)
Input_2	Power signal (0–1)
Output_1	Effort

Table B.26: Simulation parameters of the block

Parameter name	Parameter meaning
E_{max}	Maximal effort (Pa)

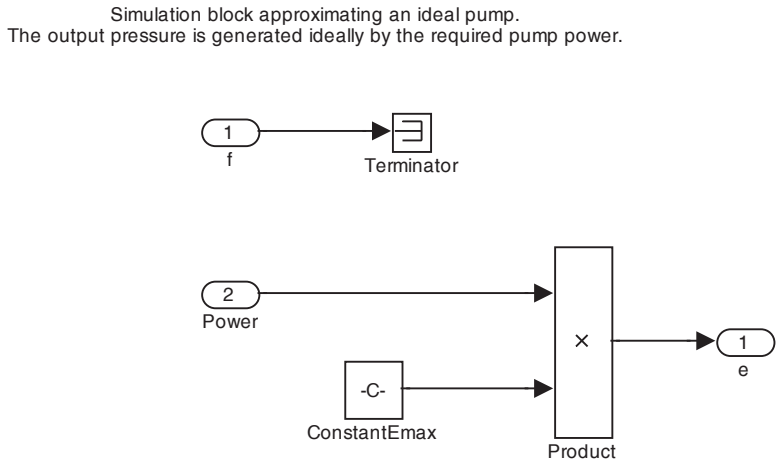

Figure B.14: A simulation block modeling a pump in hydraulic systems.

Simulation Block "simPumpB" Approximating a Real Pump

The simulation block "simPumpB" models a pump as an ideal source of flow. The interface of this simulation block is summarized in Tab. B.27. Parameters of this simulation block are explained in Tab. B.28. The internal representation of the simulation block is depicted in Fig. B.15.

Table B.27: Simulation block interface

Interface	Signal
Input_1	Effort (in the role of a formal input)
Input_2	Power signal (0–1)
Output_1	Flow

Table B.28: Simulation parameters of the block

Parameter name	Parameter meaning
F_{max}	Maximal flow $(m^3 \cdot s^{-1})$

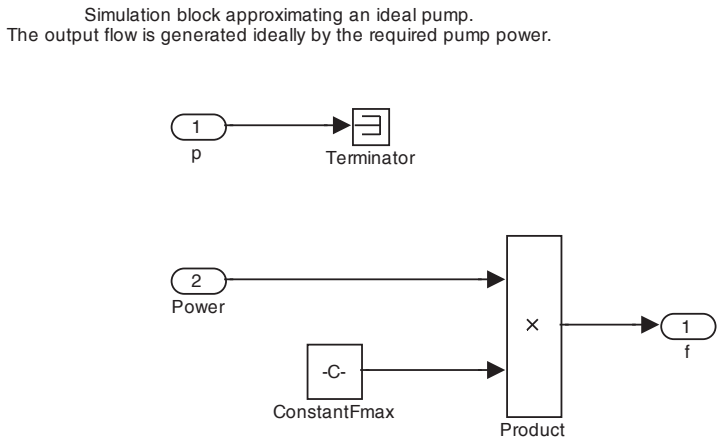

Figure B.15: A simulation block modeling a pump in hydraulic systems.

Simulation Block "simPumpOptimizedA" Approximating a Real Pump

This simulation block models a pump as an ideal source of effort. The block implements the optimized constraint handling according to Sec. 5.13. The interface of this simulation block is summarized in Tab. B.29. Parameters of this simulation block are explained in Tab. B.30. The internal representation of the simulation block is depicted in Fig. B.16.

Table B.29: Simulation block interface

Interface	Signal
Input_1	Flow (in the role of a formal input)
Input_2	Power signal (0–1)
Input_3	Inter-locking for the forward effort
Input_4	Inter-locking for the backward effort
Output_1	Effort

Table B.30: Simulation parameters of the block

Parameter name	Parameter meaning
E_{max}	Maximal effort (Pa)

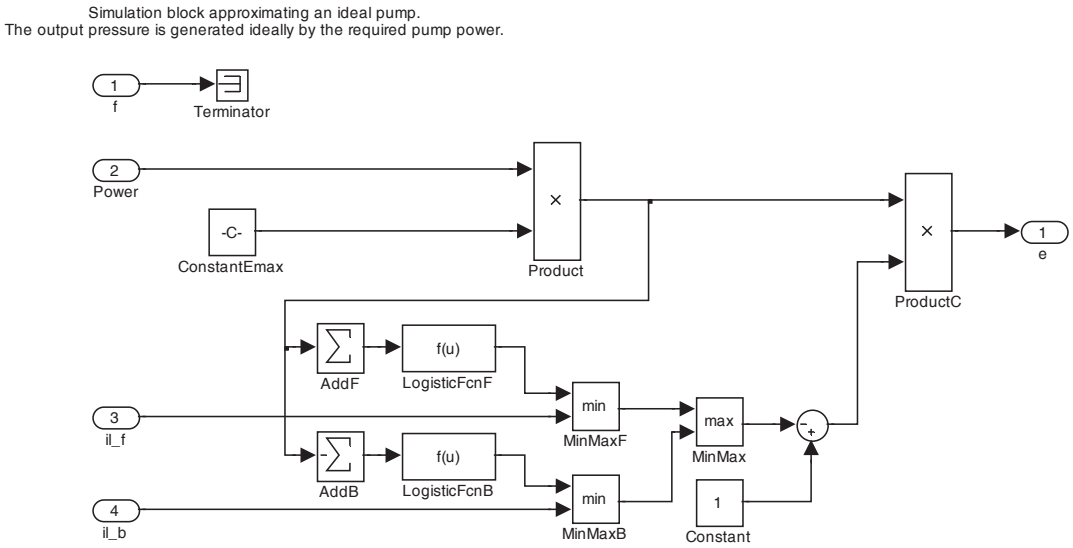

Figure B.16: A simulation block modeling a pump in hydraulic systems.

Simulation Block "simPumpOptimizedB" Approximating a Real Pump

This simulation block models a pump as an ideal source of flow. The block implements the optimized constraint handling according to Sec. 5.13. The interface of this simulation block is summarized in Tab. B.31. Parameters of this simulation block are explained in Tab. B.32. The internal representation of the simulation block is depicted in Fig. B.17.

Table B.31: Simulation block interface

Interface	Signal
Input_1	Effort (in the role of a formal input)
Input_2	Power signal (0–1)
Input_3	Inter-locking for the forward flow
Input_4	Inter-locking for the backward flow
Output_1	Flow

Table B.32: Simulation parameters of the block

Parameter name	Parameter meaning
F_{max}	Maximal flow $(m^3 \cdot s^{-1})$

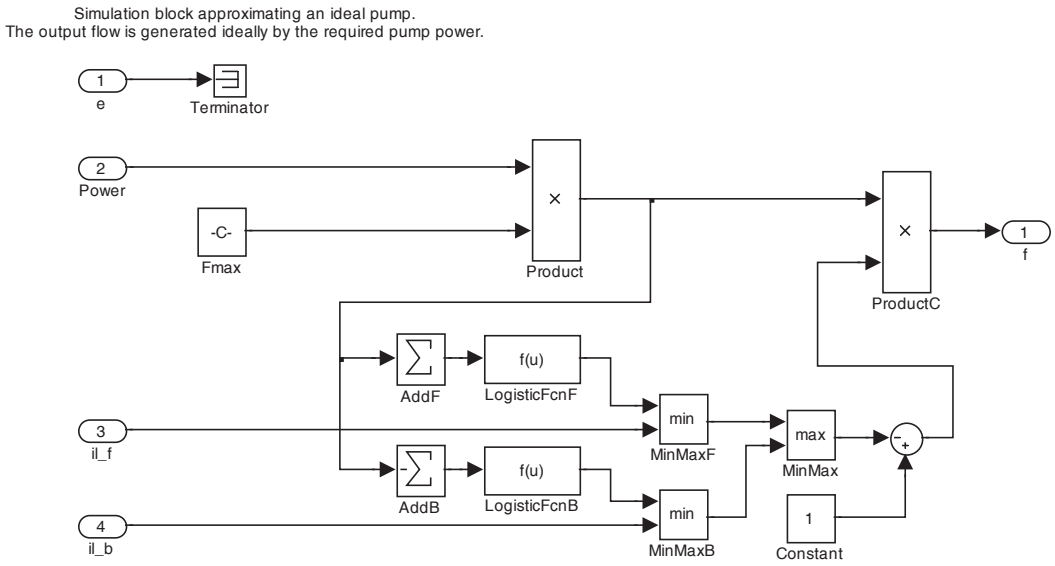

Figure B.17: A simulation block modeling a pump in hydraulic systems.

Simulation Block "simTankA" Approximating a Real Tank

The simulation block "simTankA" approximates a real tank that has one connection point. The interface of this simulation block is summarized in Tab. B.33, the parameters of this simulation block are explained in Tab. B.34, and the internal representation of the simulation block is depicted in Fig. B.18. The simulation block utilizes the basic constraint handling according to Sec. 5.13. The internal realization of the saturation block is depicted in Fig. B.19.

Table B.33: Simulation block interface

Interface	Signal
Input_1	Flow_1
Output_1	Effort_1
Output_2	Liquid level

Table B.34: Simulation parameters of the block

Parameter name	Parameter meaning
A	Area of the tank base (m^2)
h_0	Height of the tank base related to the reference level (m)
h_1	Height of the first connection point related to the tank base (m)

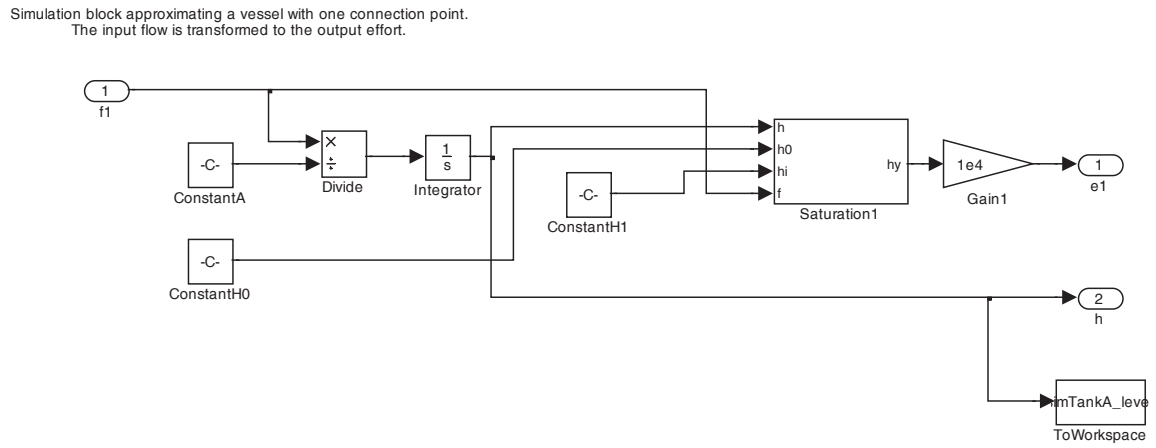

Figure B.18: A simulation block modeling a tank in hydraulic systems.

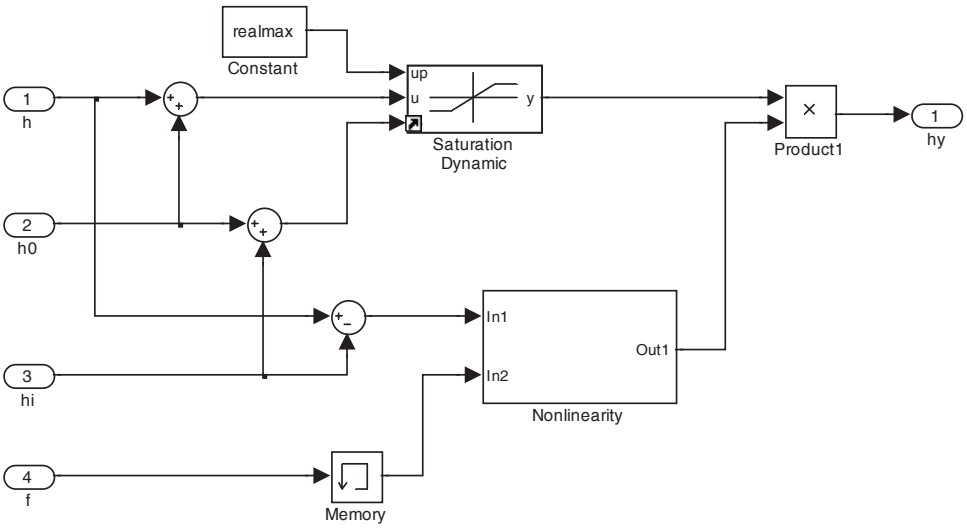

Figure B.19: The internal realization of the saturation block.

Simulation Block "simTankB" Approximating a Real Tank

The simulation block "simTankB" approximates a real tank having two connection points. The interface of this simulation block is summarized in Tab. B.35, the parameters of this simulation block are explained in Tab. B.36, and the internal representation of the simulation block is depicted in Fig. B.20. The simulation block utilizes the basic constraint handling according to Sec. 5.13.

Table B.35: Simulation block interface

Interface	Signal
Input_1	Flow_1
Input_2	Flow_2
Output_1	Effort_1
Output_2	Effort_2
Output_3	Liquid level

Table B.36: Simulation parameters of the block

Parameter name	Parameter meaning
A	Area of the tank base (m^2)
h_0	Height of the tank base related to the reference level (m)
h_1	Height of the first connection point related to the tank base (m)
h_2	Height of the second connection point related to the tank base (m)

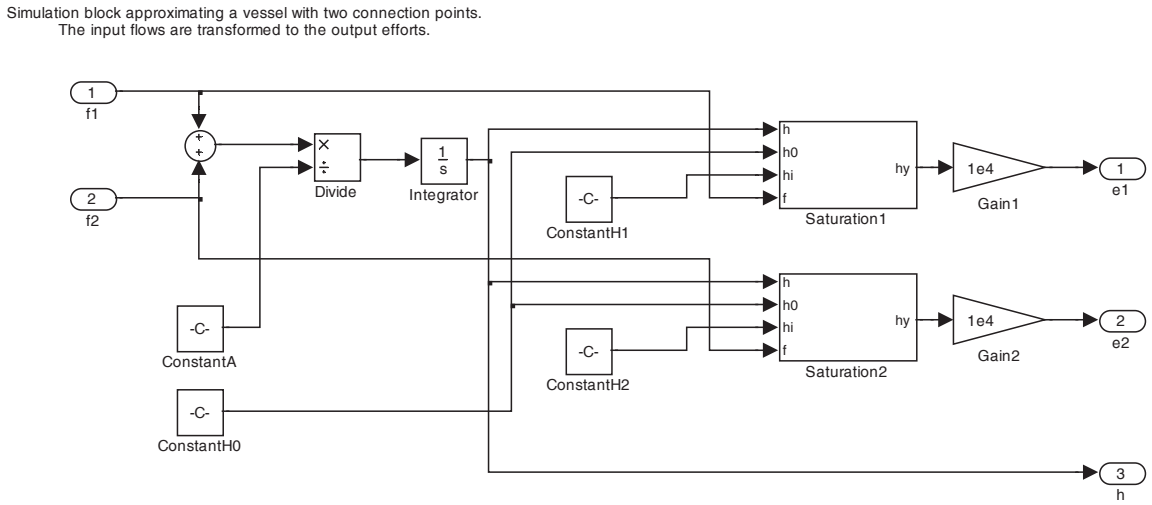

Figure B.20: A simulation block modeling a tank in hydraulic systems.

Simulation Block "simTankC" Approximating a Real Tank

The simulation block "simTankC" approximates a real tank having three connection points. The interface of this simulation block is summarized in Tab. B.37, the parameters are explained in Tab. B.38, and the internal representation is depicted in Fig. B.21. The simulation block utilizes the basic constraint handling according to Sec. 5.13.

Table B.37: Simulation block interface

Interface	Signal
Input_1	Flow_1
Input_2	Flow_2
Input_3	Flow_3
Output_1	Effort_1
Output_2	Effort_2
Output_3	Effort_3
Output_4	Liquid level

Table B.38: Simulation parameters of the block

Parameter name	Parameter meaning
A	Area of the tank base (m^2)
h_0	Height of the tank base related to the reference level (m)
h_1	Height of the first connection point related to the tank base (m)
h_2	Height of the second connection point related to the tank base (m)
h_3	Height of the third connection point related to the tank base (m)

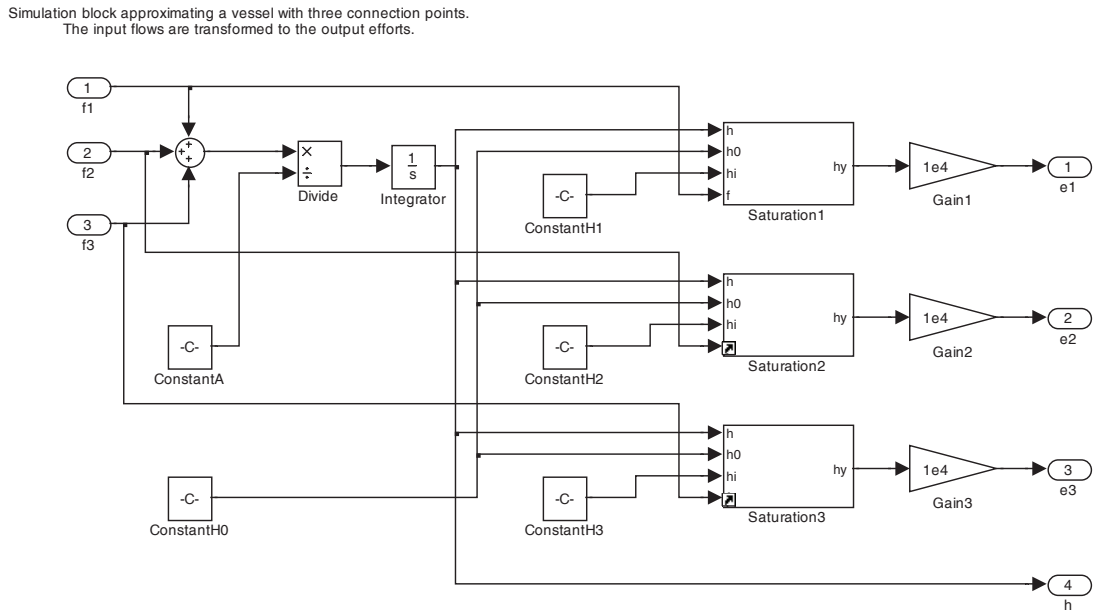

Figure B.21: A simulation block modeling a tank in hydraulic systems.

Simulation Block "simTankOptimizedA" Approximating a Real Tank

The simulation block "simTankOptimizedA" approximates a real tank that has one connection point. The interface of this simulation block is summarized in Tab. B.39, the parameters of this simulation block are explained in Tab. B.40, and the internal representation of the simulation block is depicted in Fig. B.22. The simulation block utilizes the optimized constraint handling according to Sec. 5.13. The internal realization of saturation blocks for the optimized tank blocks are depicted in Fig. B.23.

Table B.39: Simulation block interface

Interface	Signal
Input_1	Flow_1
Output_1	Effort_1
Output_2	Liquid level
Output_3	Inter-locking for $\text{Input}_1/\text{Output}_1$

Table B.40: Simulation parameters of the block

Parameter name	Parameter meaning
A	Area of the tank base (m^2)
h_0	Height of the tank base related to the reference level (m)
h_1	Height of the first connection point related to the tank base (m)

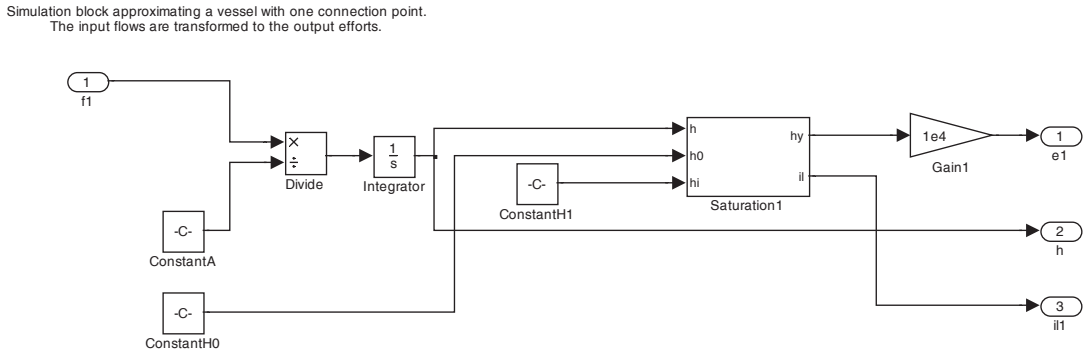

Figure B.22: A simulation block modeling a tank in hydraulic systems.

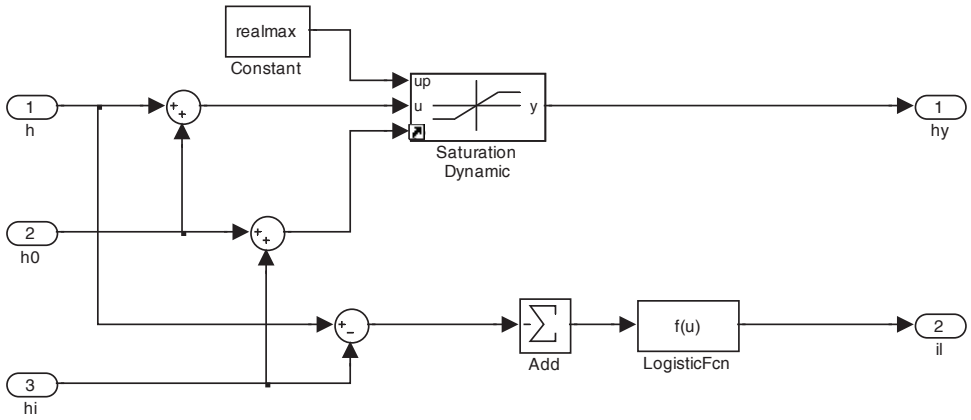

Figure B.23: The internal realization of the saturation block in case of saturation blocks.

Simulation Block "simTankOptimizedB" Approximating a Real Tank

The simulation block "simTankOptimizedB" approximates a real tank having two connection points. The interface of this simulation block is summarized in Tab. B.41, the parameters of this simulation block are explained in Tab. B.42, and the internal representation of the simulation block is depicted in Fig. B.24. The simulation block utilizes the optimized constraint handling according to Sec. 5.13.

Table B.41: Simulation block interface

Interface	Signal
Input_1	Flow_1
Input_2	Flow_2
Output_1	Effort_1
Output_2	Effort_2
Output_3	Liquid level
Output_4	Inter-locking for $\text{Input}_1/\text{Output}_1$
Output_5	Inter-locking for $\text{Input}_2/\text{Output}_2$

Table B.42: Simulation parameters of the block

Parameter name	Parameter meaning
A	Area of the tank base (m^2)
h_0	Height of the tank base related to the reference level (m)
h_1	Height of the first connection point related to the tank base (m)
h_2	Height of the second connection point related to the tank base (m)

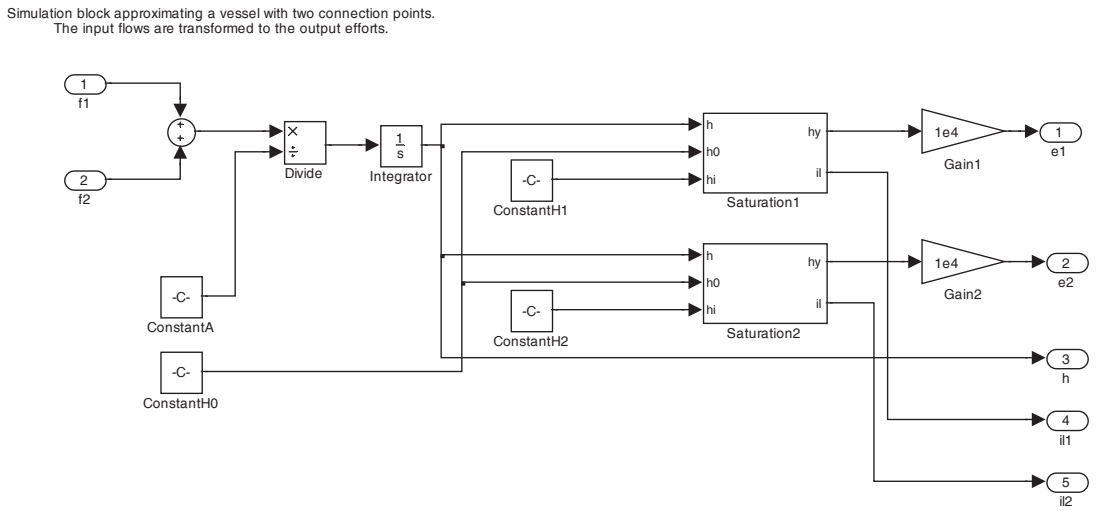

Figure B.24: A simulation block modeling a tank in hydraulic systems.

Simulation Block "simTankOptimizedC" Approximating a Real Tank

The simulation block "simTankOptimizedC" approximates a real tank having three connection points. The interface of this simulation block is summarized in Tab. B.43, the parameters of this simulation block are explained in Tab. B.44, and the internal representation of the simulation block is depicted in Fig. B.25. The simulation block utilizes the optimized constraint handling according to Sec. 5.13.

Table B.43: Simulation block interface

Interface	Signal
Input_1	Flow_1
Input_2	Flow_2
Input_3	Flow_3
Output_1	Effort_1
Output_2	Effort_2
Output_3	Effort_3
Output_4	Liquid level
Output_5	Inter-locking for $\text{Input}_1/\text{Output}_1$
Output_6	Inter-locking for $\text{Input}_2/\text{Output}_2$
Output_7	Inter-locking for $\text{Input}_3/\text{Output}_3$

Table B.44: Simulation parameters of the block

Parameter name	Parameter meaning
A	Area of the tank base (m^2)
h_0	Height of the tank base related to the reference level (m)
h_1	Height of the first connection point related to the tank base (m)
h_2	Height of the second connection point related to the tank base (m)
h_3	Height of the third connection point related to the tank base (m)

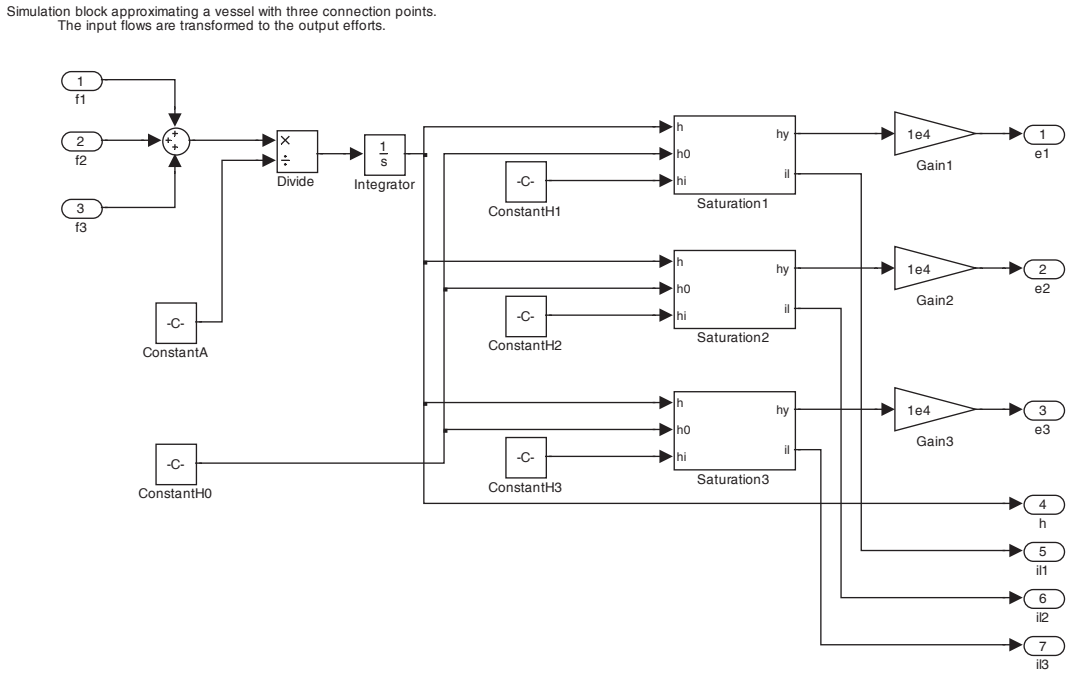

Figure B.25: A simulation block modeling a tank in hydraulic systems.

Appendix C

Screenshots of the Tool Support

This section illustrates the developed tool connector for Microsoft Visio. Fig. C.1 depicts the main page of the exporter that serializes information from the drawing into the automation ontology. A generated global tag list XML file for the configuration of the simulation integration framework for the case of the hydraulic tank model addressed in Sec. 7.2 is depicted in Fig. C.2. The second configuration XML file is the set of parameters and their values, which is depicted in Fig. C.3.

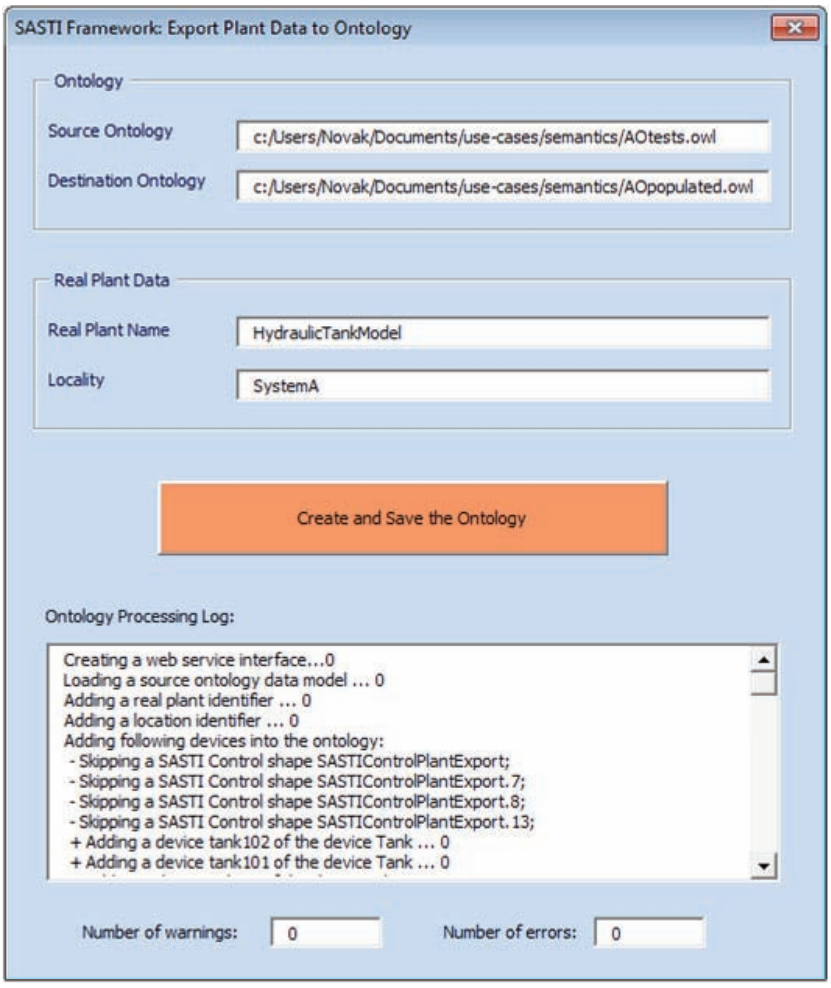

Figure C.1: Screenshot of the export dialog as part of the Microsoft Visio connector for exporting information from engineering plans into the automation ontology.

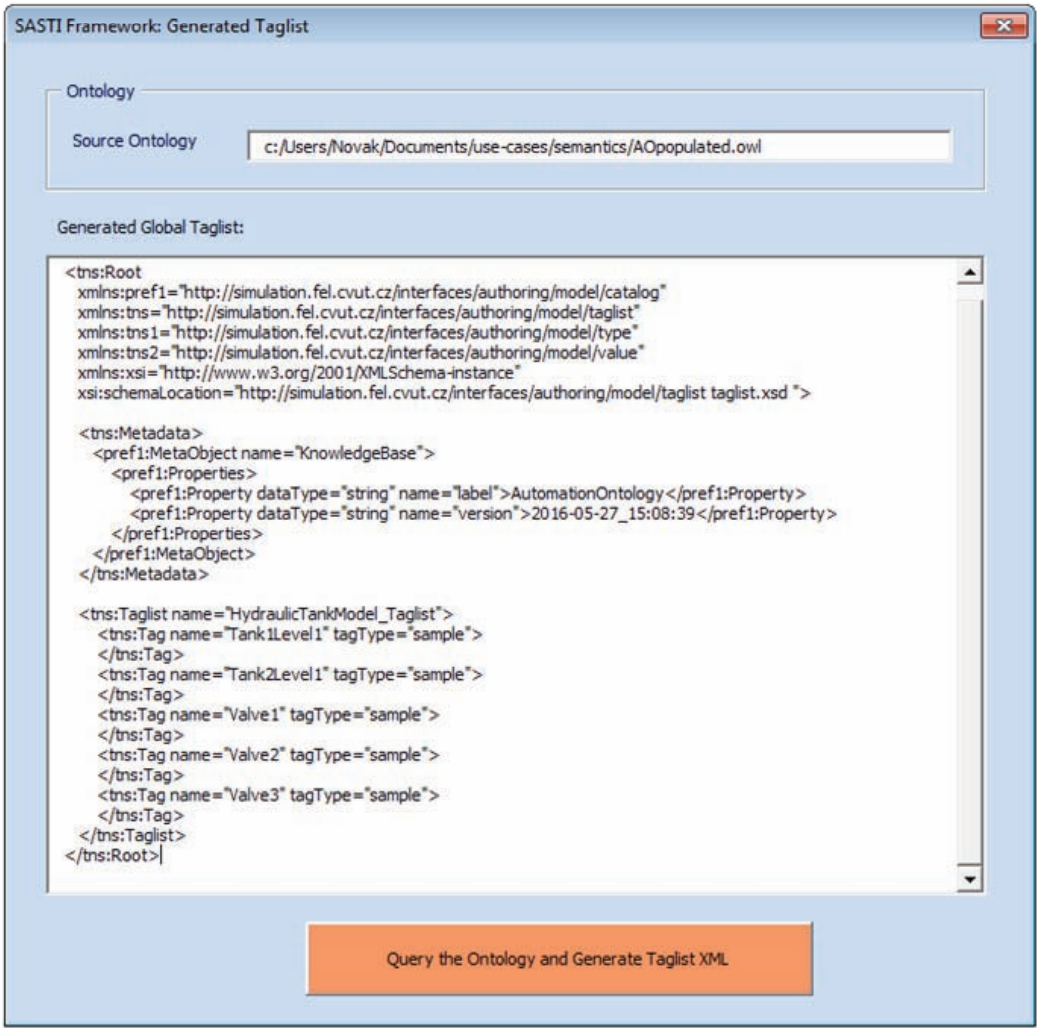

Figure C.2: Screenshot of the Microsoft Visio connector showing the XML representation of the global tag list.

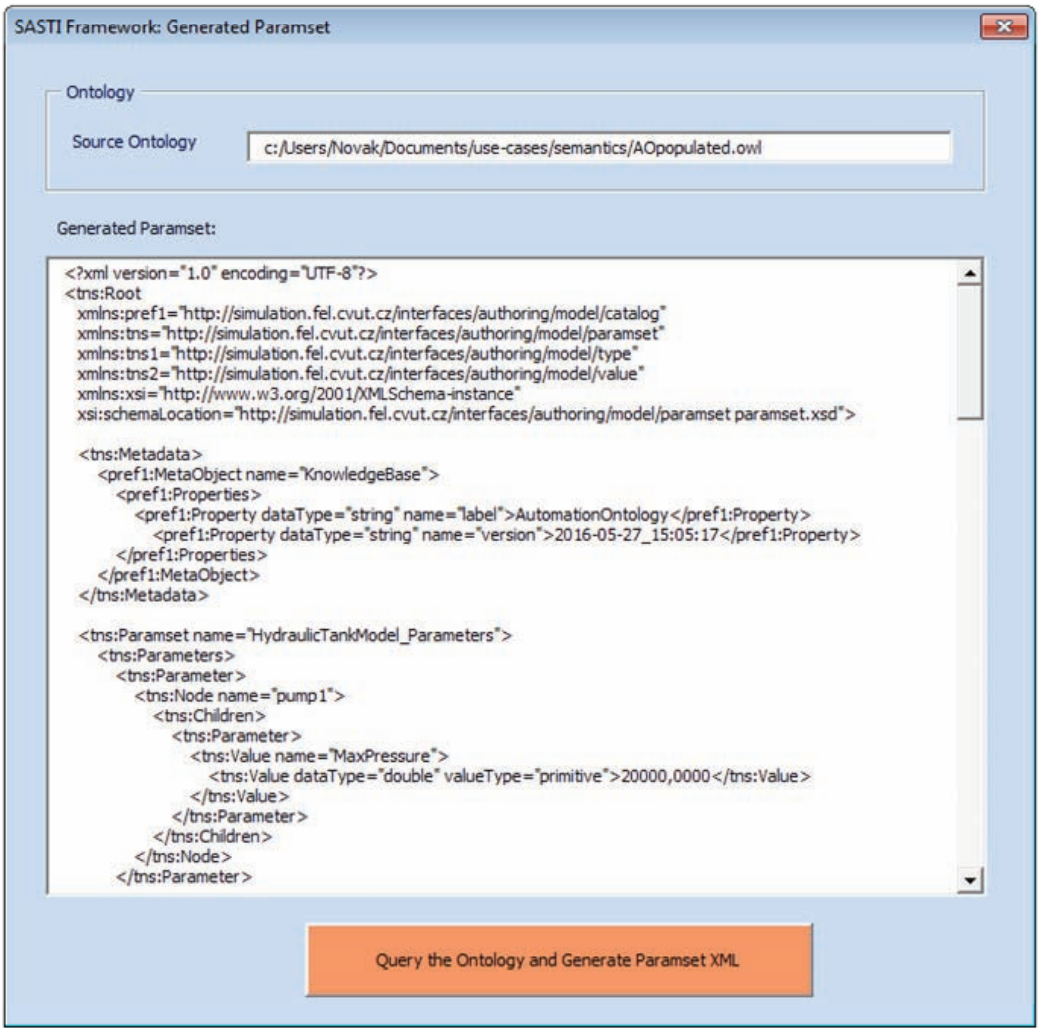

Figure C.3: Screenshot of the Microsoft Visio connector showing the XML representation of the list of parameters and their values.

Appendix D

List of the Author's Publications

Related to the Thesis Topic

Publications in Journals with Impact Factor

1. Novák, P. (80%), Serral, E. (10%), Mordinyi, R. (5%), and Šindelář, R. (5%): Integrating heterogeneous engineering knowledge and tools for efficient industrial simulation model support. *Advanced Engineering Informatics (2015).* Journal by Elsevier Science, Volume 29, 2015, Pages 575–590, DOI 10.1016/j.aei.2015.05.001.

2. Novák, P. (85%), Šindelář, R. (10%), and Mordinyi, R. (5%): Integration framework for simulations and SCADA systems. *Simulation Modelling Practice and Theory.* Journal by Elsevier Science, Volume 47, 2014, Pages 121–140, DOI 10.1016/j.simpat.2014.05.010.

   The article has been cited by the following publication:

   - Restecka, M.: IT systems in aid of welding processes. Biuletyn Instytutu Spawalnictwa w Gliwicach, Volume 59, No. 3, 2015, Pages 6–20, ISSN 2300-1674.

Publications in Reviewed Journals

1. Kadera, P. (70%), Novák, P. (20%), Jirkovský, V. (5%), and Vrba, P. (5%): Performance models preventing multi-agent systems from overloading computational resources. *Automation, Control and Intelligent Systems.* Volume 6, 2014, Pages 96–102.

Publications Excerpted in ISI

1. Novák, P. (90%), Šindelář, R. (10%): Ontology-Based Industrial Plant Description Supporting Simulation Model Design and Maintenance. In *Proc. of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013).* 6 pages, Vienna, 2013.

   The paper has been cited by the following publication:

   - Rishi, N., Gill, J. S.: Recognition of the image by using plant diseases ontology in image processing. *International Journal of Modern Computer Science (IJMCS)*, Volume 3, Issue 3, October, 2015, ISSN: 2320-7868.

2. Novák, P. (67%), and Šindelář, R. (33%): Design and Verification of Simulation Models of Passive Houses. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2012).* Krakow, 2012.

   The paper has been cited by the following publication:

- Jirkovský, V., Obitko, M.: Semantic Heterogeneity Reduction for Big Data in Industrial Automation. In *Proc. of the 14th Conference on Information Technologies Applications and Theory (ITAT 2014) with selected papers from Znalosti 2014*. 10 Pages, ISSN 1613-0073.

3. Novák, P. (80%), and Šindelář, R. (20%): Semantic design and integration of simulation models in the industrial automation area. In *Proc. of the 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2012) – 2nd Workshop on Industrial Automation Tool Integration for Engineering Project Automation (iATPA 2012)*. Krakow, 2012, pp. 1–8.

4. Novák, P. (67%), and Šindelář, R. (33%): Applications of ontologies for assembling simulation models of industrial systems. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops*. Hersonissos, Springer, 2011, pp. 148–157.

   The paper has been cited 9 times by the following publications:

   - Briola, D., Caccia, R., Bozzano, M., Locoro, A.: Ontologica: Exploiting ontologies and natural language for railway management. Design, implementation and usage examples. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 17(1), 2013, pages 3-15.
   - Dai, W., Dubinin, V. N., Vyatkin, V.: Automatically generated layered ontological models for semantic analysis of component-based control systems. *IEEE Transactions on Industrial Informatics*, Volume 9(4), November 2013, Pages 2124-2136, ISSN 1551-3203.
   - Davidovsky, M., Ermolayev, V., Tolok, V.: Application of an Instance Migration Solution to Industrial Ontologies. In *Proceedings of the 9th International Conference ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer (ICTERI 2013)*, June 2013, Pages 99-107.
   - Davidovsky, M., Ermolayev, V., Tolok, V.: Evaluation of the Ontology Instance Migration Methodology and Solution in Industrial Settings. Bookchapter in *Information and Communication Technologies in Education, Research, and Industrial Applications*. Volume 412 of the series Communications in Computer and Information Science. Springer International Publishing 2013, Pages 163-189.
   - Lange, A.-K., Jahn, C., Pirovano, G., Rossi, T.: Meta-Modellierung und Werkzeug-Integration in der Simulation von Seehafen Container-Terminals. In *Proc. of the Simulation in Production and Logistics 2015*, Fraunhofer IRB Verlag, Stuttgart 2015.
   - Lange, A.-K., Pirovano, G., Pozzi, R., Rossi, T.: Development of a Container Terminal Simulation Ontology. *Simulation Notes Europe (SNE)* 24(2), 2014, Pages 79-86, DOI 10.11128/sne.24.tn.102243.
   - Ponnusamy, S. S., Thebault, P., Albert, V.: Towards an Ontology-Driven Framework for Simulation Model Development. In *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France, January 2016.
   - Sabou, M., Kovalenko, O., Ekaputra, F., Biffl, S.: Beiträge des Semantic Web zum Engineering für Industrie 4.0. Vogel-Heuser, B., Bauernhansl, T., Hompel, M. t. (Eds.): *Handbuch Industrie 4.0*. Springer-Verlag Berlin Heidelberg 2015.
   - Tolk, A., Ihrig, M., Page, E. H., Szabo, C., Heath, B. L., Padilla, J. J., Suarez, E. D., Weirich, P., Yilmaz, L.: Epistemology of modeling and simulation. In *Proceedings of the 2013 Winter Simulation Conference (WSC 2013)*, IEEE, December 2013, Pages 1152-1166.

5. Šindelář, R. (67%) and Novák, P. (33%): Simulation integration framework. In *Proc. of the 10th IEEE International Conference on Industrial Informatics (INDIN 2012)*. Beijing 2012, pp. 80–85.

Other Publications

1. Jirkovský, V. (25%), Obitko, M. (25%), Novák, P. (25%), Kadera, P. (25%): Big Data Analysis for Sensor Time-Series in Automation. In *Proc. of the 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Barcelona, 2014.

   The paper has been cited 3 times by the following publications:

   - Kamola, M.: Analytics of Industrial Operational Data Inspired by Natural Language Processing. In *IEEE International Congress on Big Data (BigData Congress 2015)*, IEEE, 2015, Pages 681-684.
   - Obitko, M., Jirkovský, V.: Big Data Semantics in Industry 4.0. In *Industrial Applications of Holonic and Multi-Agent Systems*, Volume 9266 of the series Lecture Notes in Computer Science, Springer International Publishing, 2015, Pages 217-229.
   - Ringsquandl, M., Lamparter, S., Brandt, S., Hubauer, T., Lepratti, R.: Semantic-Guided Feature Selection for Industrial Automation Systems. In *The Semantic Web – ISWC 2015*, Volume 9367 of the series Lecture Notes in Computer Science, Springer International Publishing, 2015, Pages 225-240.

2. Kadera, P. (50%), Novák, P. (50%): Automatic Compilation of Performance Models for Industrial Multi-Agent Systems. In *Proc. of the 20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2015)*. Luxembourg, September 8-11, 2015.

   The paper has been cited by the following publication:

   - Salsinha Neves, P. M.: *Reconfiguration Methodology to improve the agility and sustainability of Plug and Produce Systems.* Doctoral thesis, KTH School of Industrial Engineering and Management, Stockholm, Sweden 2016.

3. Novák, P. (20%), Kadera, P. (20%), Jirkovský, V. (20%), Vrba, P. (20%), Biffl, S. (20%): Engineering of Coupled Simulation Models for Mechatronic Systems. In *Borangiu, T., Thomas, A., Trentesaux, D.: Service Orientation in Holonic and Multi-agent Manufacturing - Studies in Computational Intelligence 594*. Part I, Springer-Verlag Berlin Heidelberg, 2015, pp. 3–11.

   The paper has been cited by the following publication:

   - Vodenčarević, A., Fett, T.: Data analytics for manufacturing systems. In *IEEE 20th Conference on Emerging Technologies and Factory Automation (ETFA 2015)*, Luxembourg, September 2015.

4. Novák, P. (50%), Kadera, P. (20%), Jirkovský, V. (20%), Vrba, P. (5%), Biffl, S. (5%): Engineering of Coupled Simulation Models for Mechatronic Systems. In *Preprints of the International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing (SOHOMA'14)*. Nancy, France, November 5-6, 2014, pp. 1–9.

5. Novák, P. (70%), Kadera, P. (15%), Vrba, P. (10%), Šindelář, R. (5%): Architecture of a Multi-Agent System for SCADA Level in Smart Distributed Environments. In *Proc. of the 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2013)*. 8 pages, Cagliari, 2013.

   The paper has been cited 2 times by the following publications:

   - Diaconescu, E., Enescu, F. M.: A laboratory infrastructure for the evaluation of the use of multi-agent technologies in industrial SCADA systems. *Applied Mechanics and Materials*, Vol. 656, Trans Tech Publications, October 2014, pp. 432-441.

- Prostejovsky, A., Gehrke, O., Kosek, A. M., Coffele, F., Zaher, A. S. A. E.: Distributed framework for prototyping of observability concepts in Smart Grids. In *International Symposium on Smart Electric Distribution Systems and Technologies (EDST 2015)*, IEEE, September 2015, Pages 593-600.

6. Novák, P. (17%), Melik-Merkumians, M. (17%), Steinegger, M. (17%), Moser, T. (17%), Šindelář, R. (17%), and Zoitl, A. (15%): Semantic runtime interface description based on engineering knowledge. In *Proc. of the 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*. May 2012.
7. Novák, P. (50%), Melik-Merkumians, M. (10%), Steinegger, M. (10%), Moser, T. (10%), Šindelář, R. (10%), and Zoitl, A. (10%): Extraction of Automation System Engineering Knowledge for Mapping Plant and Simulation Interfaces. Bookchapter in *Borangiu, T., Thomas, A., Trentesaux, D.: Service Orientation in Holonic and Multi Agent Manufacturing and Robotics - Studies in Computational Intelligence 472*. Vol. 3, pp. 247 - 261. Springer-Verlag Berlin Heidelberg, 2013.
8. Novák, P. (95%), and Mordinyi, R. (5%): Runtime integration of industrial automation system tools based on engineering service bus. In *Proc. of the 2015 IEEE International Conference on Industrial Technology (ICIT 2015)*. Seville, 2015.
9. Novák, P. (90%), Šindelář, R. (10%): Component-Based Design of Simulation Models Utilizing Bond-Graph Theory. In *Proc. of the 19th IFAC World Congress (IFAC 2014)* Cape Town, South Africa, 2014, pp. 9229–9234.
10. Novák, P. (67%) and Šindelář, R. (33%): Integrated design of simulation models for passive houses. In *CEUR workshop proceedings (2011)*. Vol. 821, pp. 13–18.

    The paper has been cited by the following publication:

    - Suharto, E., Gomes, T., Bekbatyrova, A., Sigurjónsdóttir, S. B.: Energy Efficient Village: A Decision Enhancement Studio [Technical report], University of Groningen, 2013.

11. Sabou, M. (33,3%), Kovalenko, O. (33,3%), Novák, P. (33,3%): Semantic Modelling and Acquisition of Engineering Knowledge. Bookchapter in the publishing process in Biffl, S., Sabou, M. (Eds.): *Semantic Web Technologies for Intelligent Engineering Applications*. Springer-Verlag Berlin Heidelberg, 2016.
12. Šindelář, R. (65%), Novák, P. (35%): Ontology-based simulation design and integration. Bookchapter in the publishing process in Biffl, S., Sabou, M. (Eds.): *Semantic Web Technologies for Intelligent Engineering Applications*. Springer-Verlag Berlin Heidelberg, 2016.
13. Šindelář, R. (50%), and Novák, P. (50%): Framework for simulation integration. In *Proceedings of the 18th IFAC World Congress (Bologna, 2011)*. Vol. 18, IFAC, pp. 3569–3574.

    The paper has been cited 2 times by the following publications:

    - Pietilä, J., Kaartinen, J., Reinsalo, A.-M.: Parameter estimation for a flotation process tracking simulator, In *16th IFAC Symposium on Control, Optimization and Automation in Mining, Minerals and Metal Processing*, IFAC Proceedings Volumes, Volume 46, Issue 16, 2013, Pages 122-127, ISSN 1474-6670.
    - Sibois, R., Muhammad, A.: *State of the art in modelling and simulation* [Research report]. VTT Technical Research Centre of Finland, 2015.

Other Publications Unrelated to the Thesis Topic

1. Novák, P.: Modelling and Control of Environmental Parameters of Passive Houses. In *Proc. of the 14th International Student Conference on Electrical Engineering POSTER 2010 – CD-ROM*. ČVUT v Praze – FEL, 2010.

2. Novák, P.: Dynamické modely environmentálních veličin pasivních domů. In: Pasivní domy 2009, pp 68 - 71. Centrum pasivního domu, Brno, 2009.