

# Dependability in Multi-Agent Systems for Smart Grid Applications

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

**Doktor der Technischen Wissenschaften**

by

**Dipl.-Ing. Thomas Frühwirth, BSc.**

Registration Number 0927088

to the Faculty of Informatics  
at the TU Wien

Advisor: Dipl.-Ing. Dr.techn. Wolfgang Kastner

The dissertation has been reviewed by:

---

Wilfried Elmenreich

---

Thomas Strasser

Vienna, 4<sup>th</sup> August, 2021

---

Thomas Frühwirth



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

# Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Thomas Frühwirth, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. August 2021

---

Thomas Frühwirth



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

# Danksagung

Die Erstellung einer Dissertation ist oft ein langer Prozess, der ohne entsprechende Betreuung und Mitwirkung nur schwer möglich scheint. Der größte Dank gilt daher Prof. Wolfgang Kastner, der durch seine unermüdliche inhaltliche sowie moralische Unterstützung maßgeblich zum erfolgreichen Abschluss dieser Arbeit beigetragen hat. Sein Beitrag zum Gelingen dieser Dissertation kann kaum hoch genug eingeschätzt werden.

Ebenso dankbar bin ich allen Kollegen und Freunden aus der Automation Systems Group, anderen Instituten der TU Wien und des Austrian Center for Digital Production für die zahlreichen Diskussionen und die spannende und hoffentlich noch längerfristig anhaltende Zusammenarbeit in den verschiedensten Projekten.

Und schließlich wirken sich die Anstrengungen, vor allem im Zusammenhang mit der Verschriftlichung der Ergebnisse, auch auf den persönlichen Lebensbereich und in vielen Momenten sicher auch auf den Gemütszustand aus. Über die bedingungslose Unterstützung meiner Eltern, Verwandten und Freunde aus Oberösterreich, Wien und anderswo bin ich daher besonders dankbar.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

# Kurzfassung

Die *digitale Transformation*, also die anhaltende Verbreitung von Informations- und Kommunikationstechnologien in vielen verschiedenen Bereichen unseres täglichen Lebens, ist im elektrischen Stromnetz unter der Bezeichnung Smart Grid bekannt. Die zunehmend regionale Erzeugung und Einspeisung von Energie führt zu einer Dezentralisierung des Stromnetzes. Darüber hinaus verursacht steigender Energieverbrauch z.B. aufgrund von Elektromobilität eine höhere Belastung existierender Verteilnetze. Durch diese und ähnliche Entwicklungen werden Smart-Grid-Anwendungen zunehmend wirtschaftlich attraktiv und teilweise sogar notwendig, um einen effizienten Betrieb von Verteilnetzen zu ermöglichen.

In dieser Arbeit wird ein Systems Engineering Life Cycle vorgestellt und diskutiert, der Attribute der Verlässlichkeit bei der Entwicklung von Multiagentensystemen für Smart-Grid-Anwendungen einbezieht. Der dahinterliegende Prozess baut stark auf bestehenden Methodiken aus den Bereichen Smart Grid, Multiagentensystem und Ontologie-Entwicklung auf. Die einzelnen Agenten und Ontologien werden in insgesamt elf Aktivitäten, welche in fünf Systems Engineering Life Cycle Phasen gegliedert sind, ständig verfeinert. Im resultierenden Multiagentensystem ist jeder Agent mit einer Wissensbasis ausgestattet, die es ihm ermöglicht, die darin gespeicherten Attribute in seinen Entscheidungsprozessen zu berücksichtigen.

Leistungsschalter in städtischen Verteilnetzen ermöglichen es, die Netzkonfiguration dynamisch zu ändern, beispielsweise um Fehler zu isolieren. Die Schalterzustände (offen oder geschlossen) beeinflussen jedoch auch die Transformator- und Leitungsverluste, die durch die Versorgung der angeschlossenen Lasten entstehen. Daraus ergibt sich unmittelbar ein Schaltoptimierungsproblem, welches als begleitender Anwendungsfall in dieser Arbeit dient. Schaltoptimierung zielt darauf ab, Verluste zu reduzieren, indem das Verteilnetz dynamisch umstrukturiert wird, wenn sich Lasten zwischen verschiedenen Bereichen, z.B. von Wohnhäusern zu Bürogebäuden, verschieben.

Das aus der Anwendung des Systems Engineering Life Cycle auf das Schaltoptimierungsproblem entstehende Multiagentensystem wird mithilfe eines Co-Simulationsframeworks evaluiert. Dadurch lässt sich die mögliche Energieeinsparung abschätzen, die von einem verteilten Multiagentensystem zur Schaltoptimierung erreicht werden kann. Darüber hinaus zeigen die Ergebnisse, dass die Einbeziehung von Attributen der Verlässlichkeit in die Entscheidungsprozesse von Agenten die Leistung von Multiagentensystemen im Smart-Grid-Bereich verbessern kann.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



# Abstract

Over the past decades, the spread of information and communication technologies has affected many different domains and industries and is known under equally many keywords. This ongoing *digital transformation* has also been affecting the electric grid and led to the notation of Smart Grids. With the shift towards decentralization caused by adopting distributed energy resources and increased power consumption partly driven by the popularity of electric vehicles, Smart Grid applications are becoming economically viable and even necessary to cope with increasing energy demands in existing power distribution networks.

This thesis presents and discusses a systems engineering life cycle that incorporates dependability attributes in developing multi-agent systems for Smart Grid applications. It builds upon existing methodologies from Smart Grid, multi-agent system, and ontology development. The individual agents and ontologies are developed and refined during eleven activities structured into five main phases of the systems engineering life cycle. Each agent is equipped with a knowledge base, which allows the agent to consider the dependability attributes stored therein in its decision-making process.

Switches and circuit breakers in urban power distribution networks allow to dynamically change the network configuration, e.g., to isolate faults. However, the switch states (open or closed) also affect the transformer and line losses caused by supplying the connected loads. This insight gives rise to the switching optimization problem, which serves as an ongoing use case throughout this thesis. It aims at reducing losses by dynamically restructuring the distribution network as loads shift between different areas, e.g., from residential homes to office buildings.

An existing co-simulation framework is extended to support agents and ontologies that resulted from applying the systems engineering life cycle to the switching optimization problem. The evaluation results provide an indication of energy savings achievable by a distributed switching optimization approach. Furthermore, they show that incorporating dependability in the decision-making processes of agents can improve the performance of multi-agent systems in the Smart Grid domain.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

# Contents

<b>Kurzfassung</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Contents</b>	<b>xi</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Smart grids . . . . .	2
1.2 Dependability . . . . .	5
1.3 Multi-agent systems . . . . .	7
1.4 Knowledge representation . . . . .	8
1.5 Research question and research objectives . . . . .	11
1.6 Methodology and structure of this thesis . . . . .	13
1.7 Scientific publications . . . . .	15
<b>2 State of the art</b>	<b>17</b>
2.1 Smart grid standards . . . . .	17
2.2 Smart grid design . . . . .	21
2.3 MAS design . . . . .	32
2.4 Ontology design . . . . .	50
2.5 Related scientific work . . . . .	56
<b>3 Systems engineering life cycle definition</b>	<b>65</b>
3.1 SELC phases . . . . .	65
3.2 Definition of SELC activities . . . . .	66
3.3 Methodologies supporting the SELC activities . . . . .	68
<b>4 Planning</b>	<b>71</b>
4.1 Use case description . . . . .	71
4.2 Requirements description . . . . .	81
	xi

<b>5</b>	<b>Analysis</b>	<b>83</b>
5.1	Switching optimization algorithms . . . . .	83
5.2	Dependability requirements analysis . . . . .	86
<b>6</b>	<b>Design</b>	<b>99</b>
6.1	System architecture definition . . . . .	99
6.2	MAS design . . . . .	106
6.3	Ontology design . . . . .	114
<b>7</b>	<b>Implementation</b>	<b>131</b>
7.1	Agent implementation . . . . .	131
7.2	Ontology instantiation . . . . .	137
<b>8</b>	<b>Evaluation</b>	<b>145</b>
8.1	Smart grid simulation . . . . .	145
8.2	Functional evaluation . . . . .	148
8.3	Non-functional evaluation . . . . .	151
<b>9</b>	<b>Conclusion and future work</b>	<b>157</b>
9.1	Effects of the research objectives on the results . . . . .	158
9.2	Critical reflection . . . . .	160
9.3	Future work . . . . .	161
	<b>Acronyms</b>	<b>163</b>
	<b>Bibliography</b>	<b>171</b>

# List of Figures

1.1	Illustration of today’s power grid topology . . . . .	2
1.2	Single-line diagram of a power distribution network . . . . .	3
1.3	SGAM layers, domains, and zones . . . . .	4
1.4	Dependability tree . . . . .	5
1.5	Centralized, decentralized, and distributed (communication) networks . . . . .	8
1.6	VOWL notation exemplified . . . . .	10
1.7	SELC of SG applications . . . . .	12
1.8	Methodological approach of this thesis . . . . .	14
2.1	IEC SG standards and their mapping to the SGAM . . . . .	19
2.2	IEC 62559 use case methodology process . . . . .	22
2.3	SGAM Toolbox methodology and its mapping to MDE . . . . .	27
2.4	NISTIR 7628 user’s guide activities . . . . .	28
2.5	NIST IR 7628 logical reference model . . . . .	31
2.6	MAS design methodologies . . . . .	33
2.7	Gaia agent design methodology . . . . .	34
2.8	MaSE agent design methodology . . . . .	37
2.9	Passi agent design methodology . . . . .	40
2.10	Life-cycle of FIPA specifications . . . . .	46
2.11	FIPA abstract architecture and concrete realizations . . . . .	47
2.12	FIPA agent management reference model . . . . .	47
2.13	Ontology development 101 design methodology . . . . .	51
2.14	UPON ontology design methodology . . . . .	54
2.15	Interplay of UPON cycles, phases, iterations, and workflows . . . . .	54
3.1	Extended SELC . . . . .	66
3.2	SELC phases and activities . . . . .	67
4.1	Planning phase: activities and tools . . . . .	71
5.1	Analysis phase: activities and tools . . . . .	83
5.2	Failure rate of a technical system – the bathtub curve . . . . .	88
5.3	NIST IR 7628 logical reference model for switching optimization . . . . .	94

6.1	Design phase: activities and tools . . . . .	99
6.2	Administration shell defined by RAMI 4.0 . . . . .	101
6.3	Agent controlling a power system component . . . . .	102
6.4	Distribution network with MAS-based component control . . . . .	103
6.5	PASSI agent identification diagram for the switching optimization use case . . . . .	107
6.6	PASSI roles identification diagram for automated switching optimization . . . . .	108
6.7	PASSI task specification diagram for the Switch Agent . . . . .	109
6.8	PASSI roles description diagram for automated switching optimization . . . . .	111
6.9	FIPA agent interaction protocol diagram for the automated switching optimization . . . . .	113
6.10	Methodology for reusable ontology design . . . . .	115
6.11	Three-level methodology for the switching optimization MAS ontology . . . . .	118
6.12	Step 1: Identifying and adding domains to the graph . . . . .	119
6.13	Step 2: Adding fragments to the graph and assigning them to domains . . . . .	121
6.14	FIPA fragment ontology . . . . .	122
6.15	Dependability tree fragment ontology . . . . .	123
6.16	Metrics and scales fragment ontology . . . . .	124
6.17	Step 3: Adding fragment ratings to the graph . . . . .	125
6.18	Step 4: Removing unused fragments from the graph . . . . .	126
6.19	Rating ontology domain . . . . .	127
6.20	Agents and services domain ontology . . . . .	128
6.21	Switching optimization application-specific ontology . . . . .	129
7.1	Implementation phase: activities and tools . . . . .	131
7.2	PASSI MASD diagram for automated switching optimization . . . . .	132
7.3	PASSI SASD diagram for the Switch Agent . . . . .	133
7.4	PASSI MABD diagram for savings estimation . . . . .	134
7.5	PASSI SABD diagram for the GetNeighboringAgents method . . . . .	135
7.6	Application-specific ontology instantiation for Switch Agent 1 . . . . .	138
7.7	Modeling service availability . . . . .	140
7.8	Modeling service reliability . . . . .	141
7.9	Modeling service maintainability . . . . .	142
7.10	Modeling communication protocol integrity . . . . .	142
7.11	Modeling communication protocol scalability . . . . .	143
8.1	Evaluation phase: activities and tools . . . . .	145
8.2	FNCS architecture with GridLAB-D and NS3 . . . . .	147
8.3	Switch Agent and Transformer Agent implementation in GridLAB-D . . . . .	148
8.4	Functional evaluation setup and messages exchanged during optimization . . . . .	149
8.5	24 h load profiles of the commercial and residential loads used for simulation . . . . .	150
8.6	Supplied power and transformer losses of the functional evaluation scenario . . . . .	151
8.7	Non-functional evaluation setup . . . . .	152
8.8	Switch Agent 2 ontology for the non-functional evaluation . . . . .	154
8.9	Supplied power and transformer losses of the non-functional evaluation . . . . .	155

# List of Tables

2.1	Services defined by IEC 62559 . . . . .	25
2.2	NIST CIA impact levels definitions . . . . .	30
2.3	Gaia role schema template . . . . .	35
2.4	Feature analysis of MAS design methodologies . . . . .	45
2.5	FIPA communicative acts . . . . .	48
2.6	Dependability attributes and relevant literature in the context of MASs . . . . .	63
3.1	Methodologies and the SELC phases they support . . . . .	68
4.1	Use case identification . . . . .	72
4.2	Version management . . . . .	72
4.3	Scope and objectives of use case . . . . .	72
4.4	Narrative of use case . . . . .	73
4.5	KPIs . . . . .	74
4.6	Use case conditions . . . . .	74
4.7	Further information to the use case for classification/mapping . . . . .	75
4.8	General remarks . . . . .	75
4.9	Diagrams of use case . . . . .	76
4.10	Actors . . . . .	77
4.11	References . . . . .	77
4.12	Overview of scenarios . . . . .	78
4.13	Steps for Scenario 1 – Automated switching optimization . . . . .	79
4.14	Information exchanged . . . . .	80
4.15	Common terms and definitions . . . . .	80
4.16	Custom information . . . . .	81
4.17	Requirements . . . . .	82
5.1	Evaluation of dependability attributes in various application scenarios . . . . .	87
5.2	Smart grid organizational business functions . . . . .	91
5.3	Organizational business function risk profile . . . . .	92
5.4	Inventory of mission and business processes that support and interface with identified organizational business functions . . . . .	92
5.5	SG systems inventory . . . . .	93
5.6	SG systems inventory with CIA impacts . . . . .	94

5.7	SG systems inventory with organizational impacts, unique technical requirements, and requirement enhancements . . . . .	96
5.8	SG systems inventory with assessment scores and assessment gaps . . . . .	97
8.1	Software and versions used for evaluation . . . . .	146



# CHAPTER 1

## Introduction

In our everyday life, Information and Communication Technology (ICT) is ubiquitous and has led to an increased interconnection and interaction between the “cyber-world” and the “physical-world”, resulting in the notion of Cyber-Physical Systems (CPSs). An important field of application for CPSs is the monitoring and control of critical infrastructures, such as the power distribution network. In particular, the rise of Distributed Energy Resources (DERs) and power generation at consumer premises requires a more distributed approach to power systems control and causes devices in the distribution networks to become increasingly intelligent. This new generation of distribution networks constitutes an essential part of the Smart Grid (SG). SG applications are manifold, including substation automation, transmission line monitoring, customer energy management, advanced metering infrastructure, and many more [1].

The control of critical infrastructures requires particular precautions to be taken, as any malfunction imposes potential physical harm to people or damage to property. Thus, CPSs have to be planned and operated in a dependable fashion. Furthermore, today’s trend of connecting control systems to the Internet of Things (IoT) or even operating them via the IoT offers additional opportunities regarding flexibility and new application scenarios but exposes these systems to a large number of potential threats, in particular concerning information security and associated attack scenarios.

Individual components of a CPS may offer and invoke services among each other. In particular in the scientific literature, the predominant architectural styles for realizing CPSs in the SG field are Multi-Agent Systems (MASs), in which each device is considered as an independent, self-contained entity with well-defined goals. Each agent handles certain, simple functionalities on its own but seeks cooperation with other agents to solve more complex tasks. Deploying MASs in the SG environment requires a Systems Engineering Life Cycle (SELC) that takes dependability considerations into account from

the very beginning throughout the entire process, i.e., during planning, analysis, design, implementation, and evaluation. Thereby, Knowledge Representation (KR) provides a suitable means to express these dependability considerations. The main hypothesis of this thesis is that extending MASs with sophisticated KR functionality enables the individual agents to exchange semantically meaningful information and to consider dependability requirements in their decision-making process. This knowledge can be used to improve the performance of SG applications and to improve the dependability of the overall SG.

## 1.1 Smart grids

Although DERs are currently gaining importance, relatively few large power stations still produce the majority of electrical energy. From there, it is distributed to a vast number of consumers. The power grid follows this topology and can roughly be divided into the *transmission network* and the *distribution network*, as illustrated in Figure 1.1.

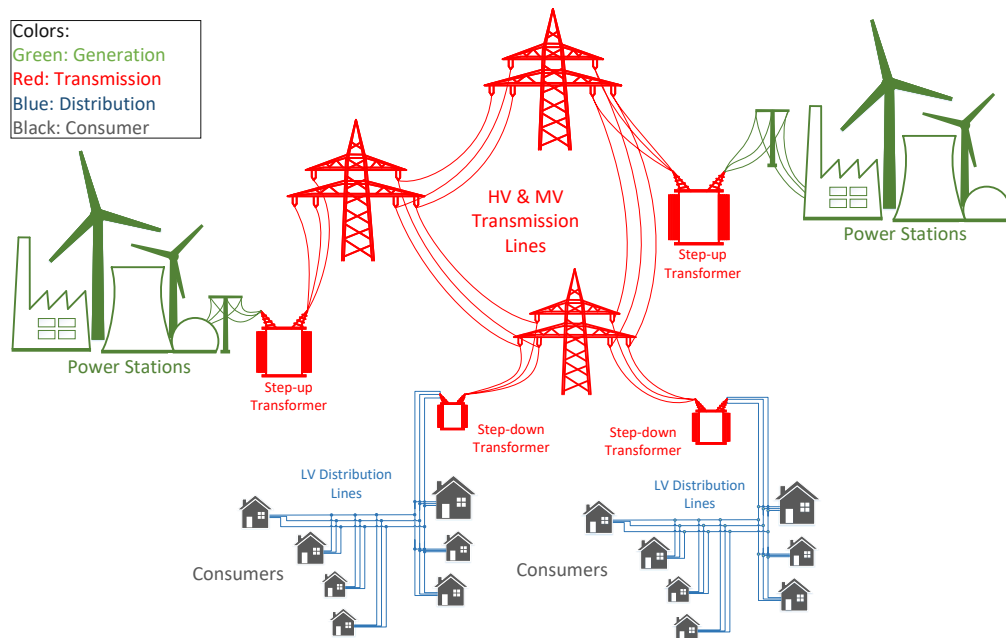


Figure 1.1: Illustration of today's power grid topology, adapted from [2]

The transmission network handles power transmission over long distances at High Voltage (HV) ( $\geq 35$  kV) or Medium Voltage (MV) ( $\leq 35$  kV,  $\geq 1$  kV), while the distribution network connects individual homes and other customers to the grid at Low Voltage (LV) ( $\leq 1$  kV) [3]. Transformers convert the voltage between different levels. Availability and reliability requirements differ between the two network types, and so do the predominant network topologies. Redundant paths (in many cases mesh or ring topologies) ensure operation of the transmission network even in the case of a single fault as an outage of the transmission network would have far-reaching consequences and affect a large number

of consumers. Contrary, the focus in distribution networks is mainly on cost efficiency by keeping the amount of necessary cabling low. For this reason, distribution networks on MV and LV level often follow a *radial structure*.

From a graph-theoretical point of view, an individual radial structure used in distribution networks is a tree having a transformer as its root node, busbars as internal nodes, and consumers (loads) as leaf nodes. A distribution network supplying a specific geographic area via multiple transformers forms a forest [4]. The graphical notation used to illustrate distribution networks throughout this thesis also highlights this forest structure, as depicted in Figure 1.2. The distribution network shown in the figure is based on the power grid structure of Figure 1.1. Power generation and the power transmission network are thereby abstracted and represented only by a red, hatched box. Transformers are illustrated via two overlapping circles, busbars via straight, thick lines, cabling via thin lines, and loads via triangles.

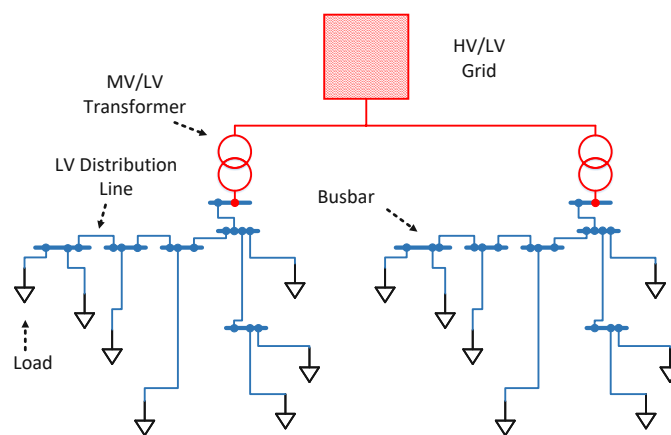


Figure 1.2: Single-line diagram of a power distribution network

During the past decades, the application of ICT in the power grid has led to a transformation towards the SG. Components are becoming more intelligent and cooperative while the grid itself shifts from the traditional generation - transmission - distribution structure towards a more decentralized structure. This development causes a change in the power grid architecture. The three-dimensional Smart Grid Architecture Model (SGAM) serves as a framework and allows assigning SG use cases, processes, products, and utility operations to three different dimensions in a technology-agnostic way. The SGAM framework is depicted in Figure 1.3.

The first dimension of the SGAM framework consists of six different *zones*. The *process* zone includes transformation and transportation of energy, and the equipment involved (e.g., generators, transformers, switches, cables and loads); the *field* zone includes equipment for process protection, control, and monitoring (e.g., protection relays and bay controllers); the *station* zone covers field data aggregation (e.g., data concentrators,

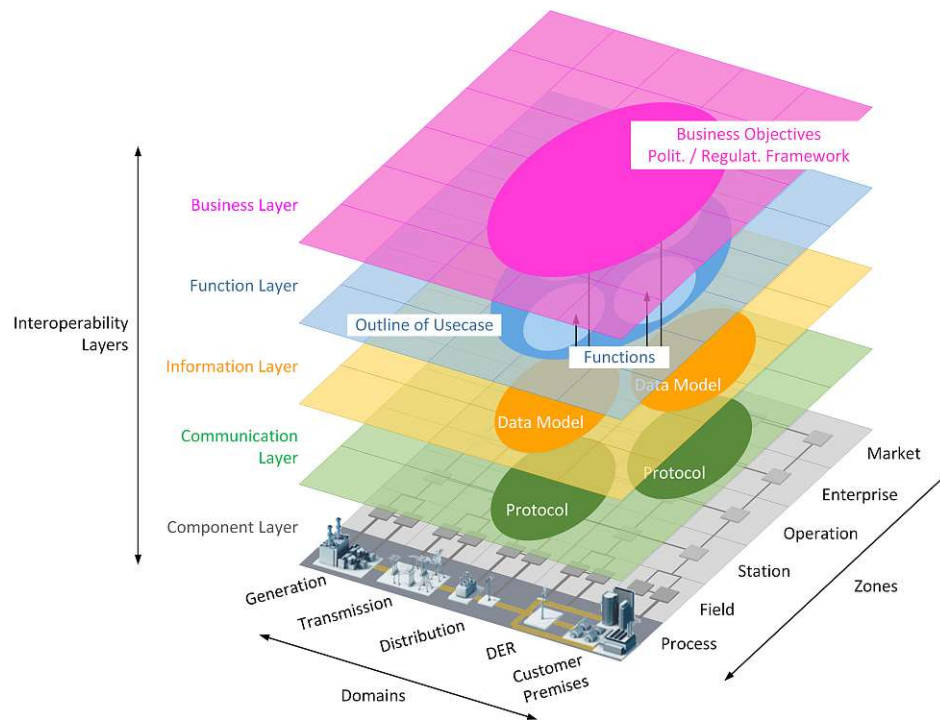


Figure 1.3: SGAM layers, domains, and zones [5]

substation automation, and local Supervisory Control and Data Acquisition (SCADA)); the *operation* zone hosts power system control operations (e.g., Distribution Management Systems (DMSs) and Energy Management Systems (EMSs)); the *enterprise* zone reflects commercial and organizational processes (e.g., asset management and logistics); and the *market* zone covers market operations (e.g, energy trading).

In its second dimension, the SGAM framework is divided into five different domains: *generation* covering energy generation in large power stations (e.g, nuclear, fossil, and large-scale renewable); *transmission* and *distribution* as previously mentioned; *DER* including small-scale, mainly renewable energy sources (small solar or wind plants); and *customer premises* covering consumption as well as production on a consumer level (e.g., PhotoVoltaic (PV) and Electric Vehicle (EV) storage).

Finally, the third dimension of the SGAM framework is built from five different layers: the *component* layer includes the physical distribution of all SG components; the *communication* layer describes protocols and mechanisms for information exchange; the *information* layer covers the exchanged information and the associated data models; the *function* layer describes functions and services independently from underlying equipment and protocols; and the *business* layer presents the business view, which includes regulations, market structures, and policies.

## 1.2 Dependability

Traditional fields of dependable applications are X-by-wire systems, e.g., steer-by-wire in the automotive industry or fly-by-wire in the aerospace industry, operation of industrial plants, and many other applications where human health is at risk if the (computer) system fails. All these systems are employed in well-controlled environments with clear limits regarding their physical expansion and the number and types of interconnected devices. These assumptions are no more valid for dependable, highly distributed computing systems found in the IoT. Therefore, new ideas are required to bring the know-how gathered from years of research and development in the field of dependability to a larger scale. The dependability tree [6] allows the various aspects of dependability to be structured in a comprehensive manner. Figure 1.4 illustrates the dependability tree. It covers threats, attributes, and means. The following paragraphs briefly discuss each of these aspects.

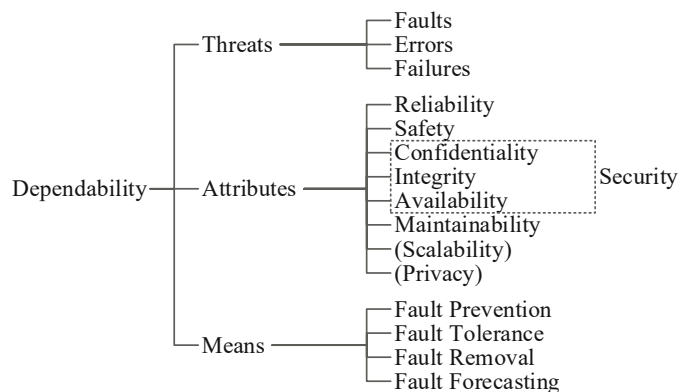


Figure 1.4: Dependability tree, adapted from [6]

Any computing system, whether or not it needs to operate in a dependable fashion, is subject to *threats*. The root cause of a malfunction is called *fault*. Faults are either human-made or caused by physical effects. Human-made faults may be intentional (e.g., security attacks) or unintentional (e.g., design faults, improper operation, or incorrect maintenance). A fault is active if it produces an error; otherwise, it is dormant. Furthermore, faults can be transient or permanent. In contrast to permanent faults, which are easier to find and repair, transient faults appear and disappear spontaneously. A fault may lead to a corrupted system state if it is activated. This corrupted system state is called *error*. An error is still internal to the affected computing system, i.e., it is not visible by other connected systems. If neither the fault nor the error is detected and corrected, the system might deviate from its specified behavior. This situation is then called a *failure*. In any dependable system, special precautions shall avoid failures whenever possible. Furthermore, a failure of a subsystem (e.g., a temperature sensor) becomes a fault to the supersystem (e.g., a controller), where it again might cause an error. This causal relationship is known as *fundamental chain* [6].

Avizienis, Laprie, Randell, et al. defined six dependability *attributes*. *Availability* specifies the system's uptime in percentage, i.e., a system having 0.99 availability is operational 99% of the time. Thus, availability gives a metric for the readiness of usage for a service and is a key attribute regarding dependability. Equation 1.1 specifies how to calculate availability from the system's up- and downtime. On the other hand, *reliability* gives the probability that a system conforms to its specification up to a given time  $t$ . Mathematically, reliability is therefore specified as a complementary Cumulative Distribution Function (CDF), called the *reliability* or *survival function*. It is defined by Equation 1.2, whereby  $f(t)$  is the Probability Density Function (PDF) specifying the system's failure probability, and  $F(t)$  is the corresponding CDF. With the definitions provided by Avizienis, Laprie, Randell, et al., only the availability and the reliability *attribute* can be defined quantitatively. *Safety* addresses the impact of a system on its environment. It is defined as the absence of catastrophic consequences on the user and the environment. *Confidentiality* refers to preventing information from unauthorized disclosure. The absence of improper system state modifications is called *integrity*. Confidentiality, integrity, and availability are often subsumed under the term *security*. *Maintainability* is the system's capability to be repaired and modified. *Scalability* and *privacy*, which are not present in the classical dependability tree but of particular interest for the IoT, were added to the dependability tree in [7]. Several definitions of scalability exist, but none of them is particularly precise [8]. For this thesis, scalability shall be defined as "the ability of a system to be extended with additional components". Privacy is often used interchangeably with confidentiality. However, this is not quite accurate. Confidentiality is rather a technical prerequisite to achieve privacy, which is defined as "the interest an individual has to control how information about them is collected, used, and shared" [9].

$$Availability[\%] = 100 * \frac{uptime[s]}{uptime[s] + downtime[s]} \quad (1.1)$$

$$Reliability(t) = 1 - F(t) = \int_t^{\infty} f(x) * dx \quad (1.2)$$

Developing dependable systems is motivated by four *means*. Development guidelines, international standards, and other procedures improve *fault prevention*. However, it is impossible to prevent faults with absolute certainty. The key to developing high dependable systems is *fault tolerance*, which is most commonly achieved by redundancy. The number of replicas required to tolerate one or even multiple faults varies depending on the failure-mode [10]. Once a fault is detected, *fault removal* may be applied. Finally, *fault forecasting* techniques enable statements about the type and frequency of faults expected to occur in the future, which is a crucial aspect of maintenance.

## 1.3 Multi-agent systems

There are different architectural styles known for realizing CPSs, e.g., based on a Service-Oriented Architectures (SOAs), but MASs gain more and more importance in the SG field. The exact definitions of a MAS and agents themselves vary. McArthur et al. [11, 12] analyzed the available literature on this topic with a strong focus on power system applications. They heavily build upon Wooldridge’s definitions of basic and intelligent agents: an agent is “a software (or hardware) entity that is situated in some environment and is able to autonomously react to changes in that environment” [13]. i.e., agents are *autonomous* and *reactive*. These properties do not set agents apart from many existing embedded systems [14]. Therefore, the notions of *intelligent agents* and *flexible autonomy* were introduced, whereby intelligent agents show flexible, autonomous behavior. This is reflected by two additional agent properties: *pro-activeness* and *social ability*. Building upon these definitions, an intelligent agent is an agent that has four characteristics:

- *Autonomy*: An intelligent agent can “make decisions about what to do” based on some internal state, “without the direct intervention of humans or others” [15]. Furthermore, when an agent makes a decision and takes the corresponding action is not pre-defined but depends on external circumstances.
- *Reactivity*: An intelligent agent can “react to changes in its environment”. Therefore, an agent must be able to observe and interact with its environment, e.g., via sensors and actuators in a physical environment or via software interfaces in a computational environment.
- *Pro-activeness*: An intelligent agent can “change its behavior in order to achieve its goals”. It can search for alternative courses of action. For example, if a particular communication partner is unreachable, it can search for another agent offering the required service.
- *Social ability*: An intelligent agent can “interact with other intelligent agents” by exchanging messages. Thereby, they communicate using a well-defined Agent Communication Language (ACL). Furthermore, agents interact via conversations, i.e., a sequence of exchanged messages, rather than simply exchanging data. They keep track of the state of each ongoing conversation.

Strictly following the above definitions, intelligence is not mandatory for agents to interact in a MAS. However, for many applications, and also in the course of this thesis, it is beneficial to agree that a MAS is a system comprising two or more intelligent agents. Ye, Zhang, and Vasilakos point out that the “capacity of a single agent is limited by its knowledge, its computing resources, and its perspectives” [16]. Therefore, no single agent in a MAS can have a full understanding of the complete system. Instead, the information is distributed. They also recognize that, when “interdependent problems arise, the agents in the system must coordinate with one another to ensure that interdependencies are properly managed”. Thus, agents within a MAS must be able to communicate. The

behavior of a MAS emerges from the behavior of the individual agents, rather than from following an overall system goal [11].

Intelligent agents are embedded in physical nodes and use communication networks to cooperate and exchange messages. These communication networks are generally classified into centralized, decentralized and distributed [17] topologies, as illustrated in Figure 1.5. They differ regarding the number of nodes that may fail without interrupting the communication between the remaining agents. For a centralized network (a), a failure in the central node immediately disrupts all communication in the network. A single node failure in any decentralized network (b) only leads to a separated network in which the individual agents in each of the segments are still able to communicate with one another. And, finally, in a distributed network, there is no distinct subset of nodes that, if these nodes fail, completely cuts communication between agents. If a sufficient number of nodes fail, the network is separated. However, the remaining subsets are still able to communicate and perform their tasks. All nodes in the distributed network are roughly equally important for its functionality. It is furthermore noteworthy that the logical topology does not necessarily equal its physical communication network structure. For example, there may as well be a central controlling instance in a physically distributed communication network.

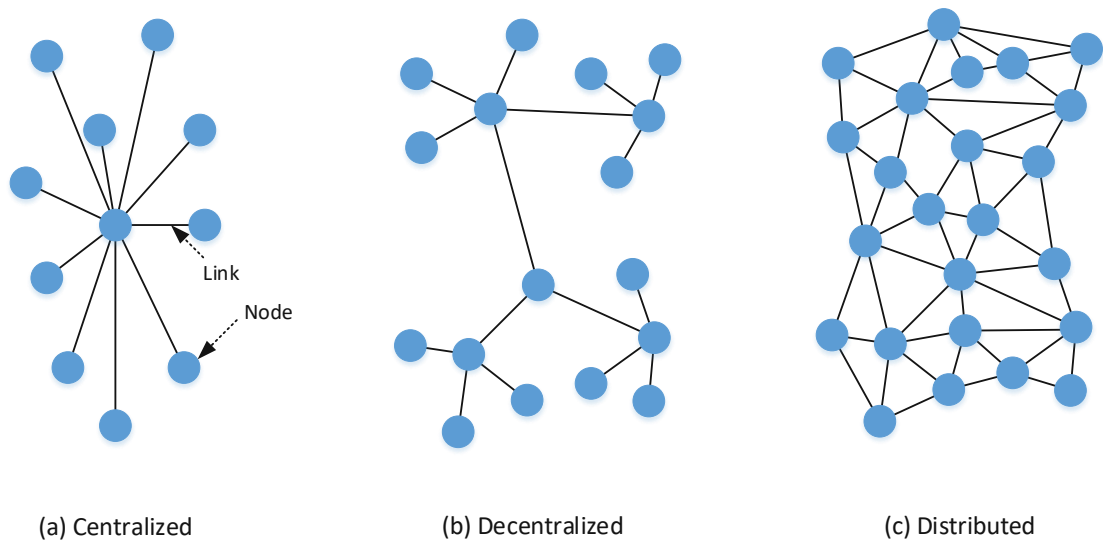


Figure 1.5: Centralized, decentralized, and distributed (communication) networks [17]

## 1.4 Knowledge representation

According to the above definition, intelligent agents can interact with their environment and communicate with other agents. For this purpose, they require some mechanism to refer to their environment, services, and attributes in a semantically meaningful way. The



corresponding field of research providing this functionality is Knowledge Representation (KR) [18]. The expressiveness of the various KR systems differs:

- *Catalog*: collection of terms without any additional information about the terms themselves or relations between them
- *Glossary*: collection of terms amended by human-readable explanations; such as this list
- *Classification/Taxonomy*: model for assigning terms to classes; classes are often hierarchically ordered
- *Semantic network*: collection of terms and arbitrary relations between them, without formal semantics
- *Ontology*: collection of terms and relations between them, including formal semantics

The predominant KR system for the IoT are ontologies as defined by the World Wide Web Consortium (W3C). Thereby, an ontology is simply a graph consisting of nodes representing specific attributes (*literal data*), objects (*individuals*), or *classes* of objects as well as edges representing relations between nodes, which are called *properties*. Classes, in the context of ontologies, are often called *concepts*. The Terminological Box (TBox) contains elements of the conceptual level that describe the principle mechanisms of the domain. The Assertional Box (ABox) contains a set of real-world instantiations of the elements defined in the TBox. A combination of a *NameSpace (NS)* and name uniquely identifies each element of the ontology. Each relation in conjunction with the nodes it connects constitutes a *triple*. Triples are of the form *Subject Predicate Object*, as in “*Paper*” *hasColor* “*White*”. A knowledge graph is an elegant and illustrative way of representing the triples of an ontology.

Even though ontologies are listed as a separate class of systems for KR, the expressiveness of ontologies themselves differ w.r.t. the Semantic Web language used for their representation (storage and exchange). Ordered from least expressive to most expressive these are Resource Description Framework (RDF), Resource Description Framework Schema (RDFS), RDFS-plus, and Web Ontology Language (OWL) [19]. And, finally, the most commonly used Semantic Web language OWL comes in different versions: OWL Lite, OWL Description Logic (DL), and OWL Full.

Ontologies and knowledge graphs are principal components of this thesis. Therefore, they are briefly introduced in the following. Figure 1.6 depicts a simple ontology in Visual Notation for OWL Ontologies (VOWL) [20]. It exemplifies the modeling concepts most relevant for this thesis and their graphical representation. The following list discusses these concepts. The numbers (except for 0) used for enumeration correspond to the numbers in the figure.

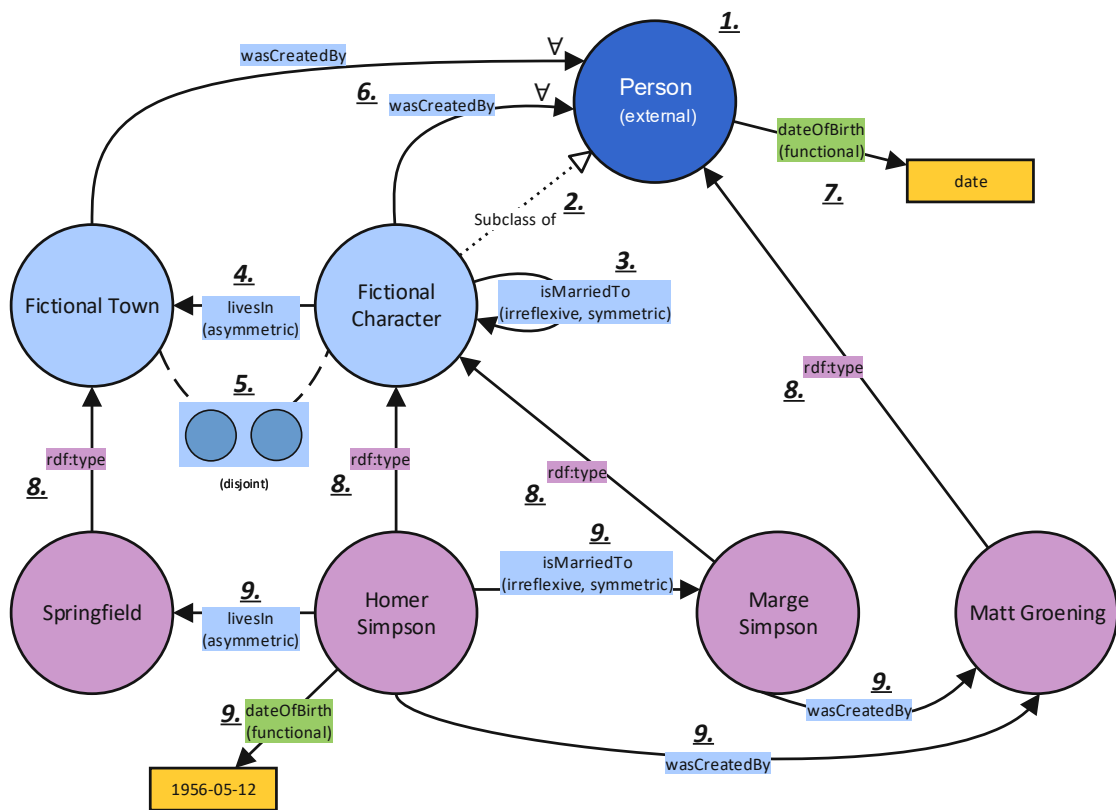


Figure 1.6: OWL notation exemplified

0. *Person*, *Fictional Town*, and *Fictional Character* are classes, while *Springfield*, *Homer Simpson*, *Marge Simpson*, and *Matt Groening* are individuals.
1. *Person* is marked as *external*. It is defined in a different NS. NSs are omitted in OWL.
2. A *Fictional Character* inherits all properties of *Person*, which is indicated by the *Subclass of* property.
3. *Fictional Characters* can be married to one another, which is indicated by the *isMarriedTo* property. This property is *irreflexive*: an individual of class *Fictional Character* cannot be married to itself. Additionally, the *isMarriedTo* property is *symmetric*: if individual *A* *isMarriedTo* individual *B*, it directly follows that *B* *isMarriedTo* to *A*.
4. A *Fictional Character* can live in a *Fictional Town*.
5. An individual can have multiple types and, therefore, combine properties of various classes. The keyword *disjoint* explicitly states that *Fictional Character* and *Fictional Town* are different classes and no individual can be of both types simultaneously.

6. For both, *Fictional Characters* and *Fictional Towns*, their creator can be specified using the *wasCreatedBy* property. Properties relating classes (and their individuals) to one another are called *object properties*.
7. Classes cannot only be related to each other but also to data values. For example, the *dateOfBirth* property enables specifying a data value for the birthday of a *Person*. Its type is *xsd:date*. Properties relating classes (and their individuals) to data values are called *data properties*.
8. The *rdf:type* property is used to specify the corresponding type for each individual (*Springfield*, *Homer Simpson*, *Marge Simpson*, and *Matt Groening*).
9. Properties that have formally been defined within the TBox on the class level (3., 4., 5., 6., 7.), are used to relate individuals among each other and to their specific data values. This creates the ABox.

To make ontologies easily accessible via the Web, they are often stored in ontology documents (typically in eXtensible Markup Language (XML) file format), just as Hyper-Text Markup Language (HTML) documents. The process of generating such a textual representation of an existing ontology is called *serialization*. Naturally, textual representations of ontologies have to be *de-serialized/parsed* to load them into programs and perform operations on them. As mentioned, ontologies are a collection of triples. Thus, databases holding these triples are typically called *triple stores* or *RDF stores*. A powerful mechanism integrated into ontologies is the possibility to *import* other, already existing ontology files by referring to their online-accessible textual representation. Once another ontology is imported, its triples can be used directly, expanded with further information, or even linked to other imported ontologies.

Due to their expressiveness, the design of ontologies is a rather complex and time-consuming task. A detailed discussion of all aspects and design principles would exceed the scope of this thesis. [21, 22, 23] provide a good starting point for this topic and list many best practices for ontology design.

## 1.5 Research question and research objectives

The design and implementation of complex systems inevitably follow a sequence of steps, known as the SELC. A SELC commonly applied in the SG domain has been identified by Andr en [24] and is depicted in Figure 1.7. It starts with the *design phase*, in which the system requirements are specified and the intended functionality is defined. The system is implemented according to its specification in the *implementation phase*. For SG systems, this certainly includes software development and may require the development of additional hardware. As SG applications often include the risks of personal damage and high economic costs in case of failure, the system needs to be validated before it can be deployed. Simulation is a common means to verify the functionality during the *validation phase*. The system is then deployed, i.e., necessary devices and software are installed in

the field during the *deployment phase*. Finally, while no engineering is performed in the *operation phase*, it is sometimes also considered part of the SELC, as the system needs to be monitored during operation to identify problems and possible improvements.



Figure 1.7: SELC of SG applications

The steps required in each of the phases of the SELC vary depending on the domain, the system that shall be developed, and the system requirements. This gives rise to the research question that will be answered in the course of this thesis:

Research question:

*Which activities need to be conducted during the Systems Engineering Life Cycle to incorporate dependability in Multi-Agent Systems for Smart Grid applications? How can state-of-the-art methodologies support this process?*

In the following, several problem statements regarding the design of such systems are given. Each problem statement results in a research objective. These research objectives guided the definition of the activities for the SELC presented in this thesis.

Methodologies are an essential tool to guide developers in systematically creating their applications. Applying a suitable methodology has several benefits: a better understanding of the problem setting, a more comprehensive analysis of the application scenario, improved system design, reduced implementation effort, and improved documentation. A vast amount of methodologies for many different kinds of problems and applications already exists. However, researchers often tend to create a new methodology rather than combining or improving existing ones.

Research objective 1:

*Whenever possible, existing methodologies shall be studied, compared, and the most suitable ones shall be applied during the systems engineering process.*

Likewise, developers often tend to create new frameworks, technologies, tools, software, and protocols rather than investing time and effort in investigating existing solutions and applying them to their specific problem. While this approach yields fast results at relatively low costs, it often turns out to be problematic, and the initial benefits vanish later in the development process. This may be caused by underestimating the required implementation effort, errors manifesting later in the development process, undiscovered security vulnerabilities, or many other reasons.

Research objective 2:

*Whenever possible, existing frameworks, technologies, tools, software, and protocols shall be studied, compared, and the most suitable ones shall be applied during the systems engineering process.*

Many SG applications found in the scientific community primarily focus on defining and realizing functional requirements, i.e., the tasks that shall be performed by a system. While this certainly is the most important aspect, including non-functional requirements in decision-making processes often would improve the overall system behavior. In particular, dependability attributes are examples of non-functional requirements.

Research objective 3:

*In addition to functional requirements, non-functional requirements (in particular dependability attributes) shall be considered in decision-making processes.*

It is commonly approved that the creation of inherently secure computing systems requires security considerations to be an integral part of every step of the systems engineering process. This idea is known as the “security by design” principle. A similar mindset is required for designing dependable systems.

Research objective 4:

*The systems engineering process shall follow a “dependability by design” principle.*

Including existing ontologies is an integral part of almost any ontology design methodology and a crucial step towards reducing the time required for ontology engineering. Furthermore, including existing ontologies increases interoperability between different applications and should be considered whenever designing a new ontology.

Research objective 5:

*Existing ontologies shall be included in the ontology design process but, equally important, ontologies created during the design process shall themselves be reusable in other applications and domains.*

## 1.6 Methodology and structure of this thesis

The methodological approach that has been applied to answer the research question is depicted in Figure 1.8. It starts with performing a literature study to identify existing work that can be incorporated in the SELC. Next, the SELC itself is defined and required

activities for each phase are identified. The result is an adapted version of the classical SELC for SGs. The main phases of the SELC and the corresponding activities are then extensively discussed and conducted based on a SG example use case. The thesis concludes with a critical discussion summarizing the main results and future work. Additionally, the methodological approach serves as a basis for the structure of this thesis, which is organized as follows.

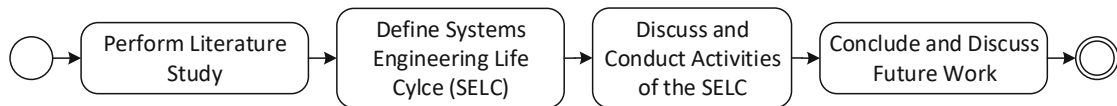


Figure 1.8: Methodological approach of this thesis

Chapter 2 presents the state of the art, starting with an overview of existing SG standards. SG, MAS, and ontology design methodologies are summarized with a focus on their suitability for SG applications. Furthermore, a suitable MAS framework is presented, and related scientific work in the field of MAS-based SG applications and dependability in MASs are discussed.

Chapter 3 builds upon the state of the art and combines SG, MAS, and ontology design methodologies in a SELC. The resulting SELC consists of five main phases: planning, analysis, design, implementation, and evaluation. Deployment & operation is added as an additional sixth phase.

Chapter 4 starts the SELC with the planning phase. It provides some general information about distribution networks, their main components, and their basic structure. Additionally, the switching optimization use case, which serves as a motivating example throughout this thesis, is described following the IEC 62559 use case methodology.

Chapter 5 continues with the analysis phase. Functional (algorithmic) and non-functional (dependability) requirements of the previously described use case are analyzed. Thereby, functional requirements are covered by presenting algorithmic approaches to the switching optimization problem. Non-functional requirements are covered by performing a dependability requirements analysis.

Chapter 6 covers the design phase. Based on the findings of the previous analysis phase, a distributed MAS is chosen as system architecture. Therefore, the design phase is split into two major parts: the MAS design and the ontology design. The MAS design builds upon the Process for Agent Societies Specification and Implementation (PASSI) methodology, resulting in the definition of several agent types, their services, and an agent interaction protocol defining the communication pattern. Additionally, a new ontology design methodology focusing on reusability is introduced and followed to create the required ontologies on a class level.

Chapter 7 covers the implementation phase. A software architecture for agents, including their interfaces to external components, is defined and implemented based on several existing open-source software libraries, including a library that serves as triple store and, therefore, provides the means to store and access the necessary ontologies. The ontologies themselves are instantiated, i.e., extended with individuals.

Chapter 8 corresponds to the evaluation phase. Simulation is used to illustrate how agents interact and how they include dependability considerations in their decision-making process. Furthermore, the performance of the switching optimization use case is evaluated based on a near real-world consumption scenario to provide estimations about achievable energy savings.

Chapter 9 concludes this thesis. It provides some critical reflection upon a distributed MAS as a system architecture for SG applications, the potential of switching optimization in LV power grids, and a discussion about dependability of MASs for this and similar use cases. Furthermore, ongoing and future work are presented.

## 1.7 Scientific publications

The main contributions of this thesis rest on several scientific publications:

T. Frühwirth, L. Krammer, and W. Kastner. “Dependability Demands and State of the Art in the Internet of Things”. In: *Proceedings of the 20th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Luxembourg, Sept. 2015, pp. 1–4

T. Frühwirth, A. Einfalt, K. Diwold, and W. Kastner. “A distributed multi-agent system for switching optimization in low-voltage power grids”. In: *Proceedings of the 22nd IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Limassol, Cyprus, Sept. 2017, pp. 1–8

T. Frühwirth, L. Krammer, and W. Kastner. “A methodology for creating reusable ontologies”. In: *Proceedings of the 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. Saint Petersburg, Russia, May 2018, pp. 65–70

G. Steindl, T. Frühwirth, and W. Kastner. “Ontology-Based OPC UA Data Access via Custom Property Functions”. In: *Proceedings of the 24th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Zaragoza, Spain, Sept. 2019, pp. 95–101

D. Herbst, M. Lager, R. Schürhuber, E. Schmutzner, L. Fickert, A. Einfalt, H. Brunner, D. Schultis, T. Frühwirth, and W. Prügler. “Zukünftige Anforderungen an NS-Netze und deren Lösungsansätze am Beispiel PoSyCo”. In: *16. Symposium Energieinnovation*. Graz, Austria, Feb. 2020, pp. 1–11



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



# State of the art

The goal of this thesis is to combine existing technologies, protocols, and methodologies to incorporate dependability in MASs for SG applications, which requires a detailed analysis of the state of the art in multiple fields. Standards created by various standardization organizations provide a reliable source of information about the electric grid in general and SGs in particular. Thus, this chapter starts with examining existing standards and assigning them to the different layers of the SGAM. It continues by introducing existing methodologies that support the planning phase and the analysis phase of SG applications, as well as methodologies for designing MASs and ontologies. Furthermore, some information about MAS platforms is presented to provide the technical background for the implementation and the evaluation phases. The chapter concludes with a selection of existing related scientific work in the area of SG MASs, and existing approaches to increase the dependability of MASs.

## 2.1 Smart grid standards

Standards often define the legal requirements that have to be considered when creating applications for the SG field. Furthermore, they serve as a valuable source of information regarding existing knowledge about the specific field of application in general and provide best practices, terminology, and information models. Among the existing standardization organizations that publish material relevant for SG applications are:

- Institute of Electrical and Electronics Engineers (IEEE)
- National Institute of Standards and Technology (NIST)
- International Society of Automation (ISA)
- American National Standards Institute (ANSI)
- International Organization for Standardization (ISO)

- Deutsches Institut für Normung (DIN)
- Comité Européen de Normalisation Électrotechnique (CENELEC)
- European Telecommunications Standards Institute (ETSI)
- International Electrotechnical Commission (IEC)

The IEC uses the domains (x-axis) and zones (y-axis) defined by the SGAM framework to arrange the vast number of SG-related standards in its “Smart Grid Standards Mapping Tool”. It is available online as an interactive chart [30]. The IEC standards map currently contains more than 300 standards, which is still only a fraction of the existing IEC standards, not to mention material provided by other standardization organizations. However, it provides a good starting point for engineers to filter for standards relevant to their device or application. Standards are often relevant in multiple domains and zones. In addition, according to the IEC, four *crosscutting functionalities* must be taken into account in all SG applications: telecommunication, security, EMS, and power quality.

The IEC standards map is a two-dimensional tool and, thus, cannot associate standards to one of the five vertical SGAM interoperability layers directly. To compensate for this, the following brief introduction of the standards that are related to the contents of this thesis is based on the SGAM interoperability layers from bottom to top. It also includes IEC standards that have not been added to the IEC standards map yet, as well as several relevant standards from other standardization organizations. For any standard series (also called standard family), the discussion only provides a reference to the first relevant document of the series. Figure 2.1 illustrates a selection of these standards and their association to the various SGAM layers.

Standards on the SGAM component layer specify properties of physical SG components. *IEC 60038* [3], *IEC 60059* [31], and *IEC 60196* [32] define standard voltages, standard currents, and standard frequencies, respectively. The *IEC 60076* [33] standard series provides information about different transformer types, winding types, naming conventions, operating conditions such as altitude and temperature, transformer ratings, measurement procedures to determine characteristic values of transformers such as winding resistance, short-circuit impedance and no-load losses, and many more different aspects relevant for planning, building, and operating power transformers. In addition, definitions for these terms are given. Likewise, relevant specifications for overhead lines primarily include recommended materials and alloys for overhead line conductors (*IEC 60104* [34], *IEC 60889* [35], *IEC 61394* [36]), insulators (*IEC 60305* [37], *IEC 60383* [38], *IEC 61109* [39], *IEC 61325* [40], *IEC 61466* [41], *IEC 61467* [42], *IEC 61211* [43], *IEC 61952* [44]), and testing (*IEC 60652* [45]). *IEC 60105* [46] and *IEC 60114* [47] recommend materials to be used for busbars. Information about electrical switches is provided in *IEC 61020* [48] and *IEC 61058* [49]. Safety is another important topic on the component layer, as improper design and operation of equipment imposes a threat to people and the infrastructure. Most important in this regard is the *IEC 61508 – Functional safety of electrical/electronic/programmable electronic safety-*

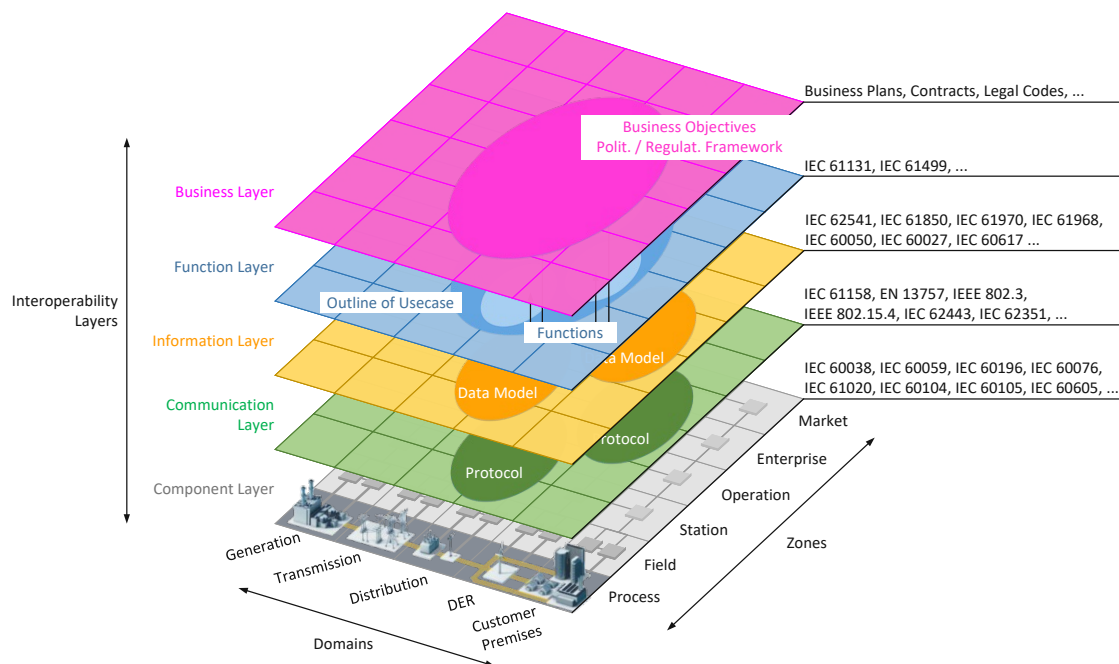


Figure 2.1: IEC SG standards and their mapping to the SGAM, adapted from [5]

related systems [50] standard series, which introduces, among many general aspects about safety, the Safety Integrity Levels (SILs). These SILs provide a metric that combines the likelihood and severity of failures for a component or a complete system into a single value from SIL 1 (lowest requirements) to SIL 4 (highest requirements). SILs play an important role in certification. Information associated to reliability of SG equipment can be found in *IEC 60605* [51], *IEC 61709* [52], and *IEC 61163* [53]. Availability is addressed in *IEC 61070* [54], and maintainability of equipment is covered, e.g., by the *IEC 60706 – Maintainability of equipment* [55] standard series. Furthermore, *IEC 61703* [56] defines *Mathematical expressions for reliability, availability, maintainability, and maintenance support terms*. Standards defining properties of communication equipment, e.g., for optical fibers (*IEC 60793* [57], *IEC 60794* [58]), can also be assigned to the component layer.

The SGAM communication layer enables devices of the component layer to exchange messages by making use of communication protocols. For example, the *IEC 61158* [59] in combination with the *IEC 61784* [60] set of standards specifies Process Field Bus (PROFIBUS) (*IEC 61784-5-3* [61]), Ethernet for Control Automation Technology (EtherCAT) (*IEC 61784-5-12* [62]), MODBUS (*IEC 61784-5-15* [63]), and many other industrial communication protocols. In addition, *IEC 62734* [64] defines the ISA100.11a wireless communication protocol (adopted from the International Society of Automation (ISA)). However, other standardization organizations than the IEC are more active in the field

of communication protocols. For example, the Meter-Bus (M-Bus) (*EN 13757-2* [65], *EN 13757-3* [66]), and Wireless Meter-Bus (Wireless M-Bus) (*EN 13757-4* [67]) protocols are particularly used in Advanced Metering Infrastructure (AMI) applications. As IoT protocols are increasingly being adopted in SG applications, the *IEEE 802* standard series gains importance, including well-known technologies such as the *IEEE 802.3* [68] set of standards specifying Ethernet with all its different physical layers, the *IEEE 802.11* [69] set of standards specifying Wireless Local Area Network (WLAN), the *IEEE 802.15.4* [70] set of standards providing the lower layers for protocols such as ZigBee [71], IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [72], wireless Highway Addressable Remote Transducer protocol (wirelessHART) [73], and the *IEEE 802.1* set of standards providing, among other features, the groundwork for Time-Sensitive Networking (TSN). An important topic on the communication layer is security, which is covered e.g., by the *IEC 62443 – Industrial communication networks – Network and system security* [74] standard series and *IEC 62351 – Power systems management and associated information exchange – Data and communications security* [75] standard series, but is of course also addressed by other standardization organizations, e.g., in *NISTIR 7628 – Guidelines for Smart Grid Cyber Security* [76] with its corresponding user's guide [77].

The information layer provides the means to model information and to use these data models for communication. Thereby, the communication layer handles the exchange of messages, but the information layer defines the semantics of these messages. As the communication layer and the information layer are so closely related, many protocols such as the OPC Unified Architecture (OPC UA) (*IEC 62541* [78]) include both aspects. Likewise, the *IEC 61850* [79] standard series also covers aspects of the communication as well as the information layer. It defines an Ethernet-based, object-oriented communication standard to be used by Intelligent Electronic Devices (IEDs) in SG systems. The objective of this standard series is to increase interoperability between devices by standardizing the content of messages. Other standards only focus on information modeling aspects and rely on mechanisms of the communication layer for message exchange: the *IEC 61970* [80] standard series defines the Common Information Model (CIM), which contains terms and definitions for many components and concepts of SGs. Moreover, part *IEC 61970-501* [81] of the series specifies an RDFS format for CIM. *IEC 61970* is accompanied by the *IEC 61968* [82], which extends the CIM with information specifically related to electrical distribution networks. The corresponding RDF representation can be found in *IEC 61986-13* [83]. There are many more standards that might at least partly be assigned to the information layer even though they are primarily used for information exchange between professionals rather than between IEDs: *IEC 60050 – International Electrotechnical Vocabulary* [84] standard series, *IEC 60027 – Letter symbols to be used in electrical technology* [85] standard series, and *IEC 60617 – Graphical symbols for diagrams* [86].

Standards on the SGAM function layer provide the means to specify the functionality of the SG system independently of the underlying technologies. The *IEC 61131 – Industrial-*

*process measurement and control – Programmable controllers [87]* standard series, in particular *IEC 61131-3: Programming languages [88]*, provide languages to describe the functionality of industrial automation systems by making use of a cyclic execution model. The *IEC 61499 – Function blocks [89]* standard series is based on *IEC 61131* but it encapsulates the functionality in Function Blocks (FBs) and replaces the cyclic execution model by an event-driven execution model, in which FBs interact by exchanging events and data. Furthermore, FBs can be nested to be able to model a system on different levels of abstraction.

Finally, the SGAM business layer focuses on economic and legal aspects. It is not discussed in detail here for two reasons: (1) the corresponding documents are business plans, contracts, and legal codes rather than technical specifications and standards, and (2) these topics are out of the scope of this thesis.

## 2.2 Smart grid design

The first type of design methodologies covered by this thesis is targeting the SG domain. Most SG design methodologies only focus on specific aspects of the systems engineering process, such as the use case description or security guidelines. Model-Driven Engineering (MDE) can be applied to support and partly automate this process [24]. All SG design processes, whether or not they are MDE-based, should start with a proper definition of the intended use cases. Furthermore, many MAS design methodologies and ontology design methodologies presume that the requirements have already been analyzed a priori to applying the MAS/ontology design methodology itself. Therefore, the use case description can serve as valuable input for subsequent phases in the systems engineering process.

### 2.2.1 IEC 62559 use case methodology

The IEC 62559 use case methodology [90, 91] can be used for use case specification in various domains. However, its main field of application is the SG domain. It provides a template to describe the static and dynamic aspects of the system under development [92]. Thereby, a description of the involved actors covers the static aspects, while detailed descriptions of use cases and the necessary interactions between actors cover the dynamic aspects. The template contains eight template sections and additional template subsections. Each template subsection of IEC 62559-2 (or the template section itself if it does not have template subsections) specifies a table to be filled in by domain experts or future users of the system. Some of these tables are considered optional.

The IEC 62559 use case methodology follows an almost completely linear process, i.e., the tables can almost strictly be filled in one after another, as illustrated in Figure 2.2. Only template subsection 1.8, as well as template sections 7 and 8 are accompanying the otherwise linear process and allow to collect additional information. Additionally, the IEC 62559 use case methodology supports storage, exchange, and reuse of use case descriptions by specifying an XML schema. This enables use case descriptions

to be collected in use case repositories, such as the Use Case Management Repository (UCMR) [93]. The following paragraphs briefly discuss the template sections and their purposes. The names of the tables and the corresponding table fields are *emphasized*.

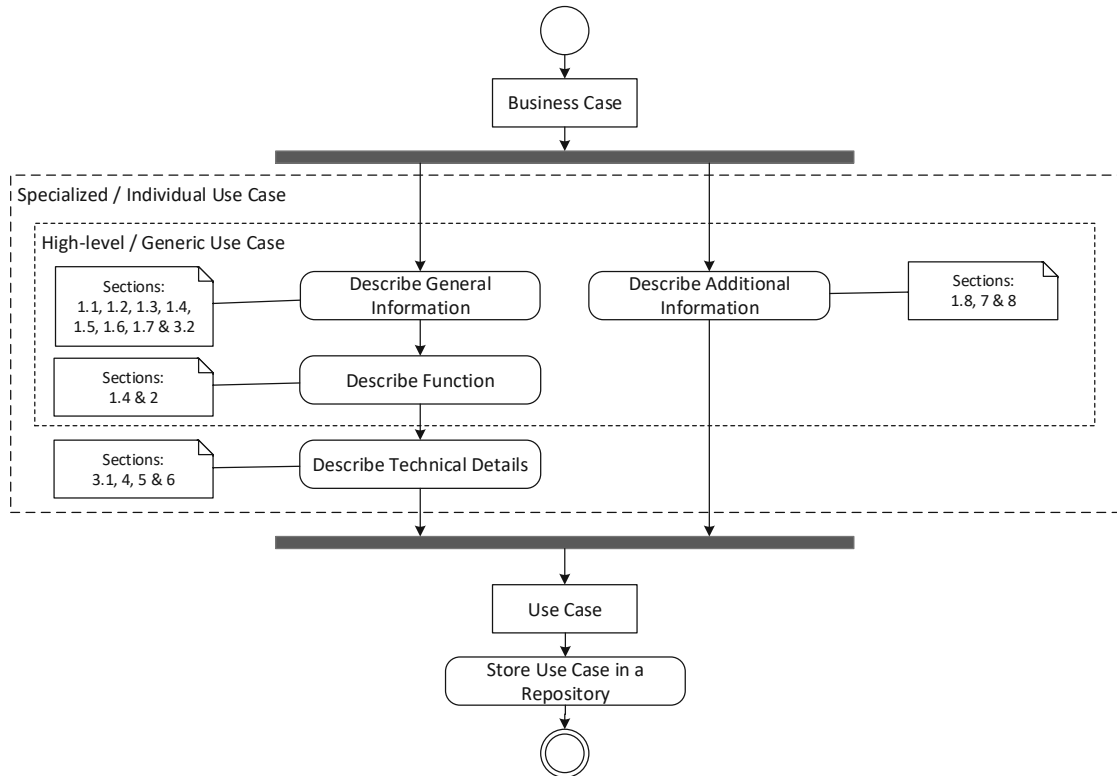


Figure 2.2: IEC 62559 use case methodology process [92]

### Template section 1: Description of the use case

Template section 1 provides the most general information about the use case. It consists of eight template subsections, and, therefore, eight tables. Template subsection 1.1. (*Use case identification*) specifies an *ID* for the use case that should be unique within the project, the corresponding *Area/Domain(s)/Zone(s)*, and a name (*Name of use case*). If the use case is related to the SG area, it is suggested to use the corresponding areas and domains of the SGAM. Template subsection 1.2. (*Version management*) is intended to document changes in the use case description and defines the *Version No.*, *Date*, *Name of author(s)*, *Changes*, and *Approval status* fields. Template subsection 1.3. (*Scope and objectives of use case*) gives some background information and motivation about the use case by defining its *Scope*, *Objective(s)* and *Related business case(s)*. Template subsection 1.4. (*Narrative of use case*) consists of a *Short description* and a *Complete description* of the use case. These descriptions should help non-domain-experts to understand the use case in such a way as to enable them to decide if the use case description fits their specific problem.

The short description is in textual form and should not exceed 150 words. The complete description can be more comprehensive and use drawings for additional explanation. Template subsection 1.5. (*Key performance indicators (KPI)*) defines metrics to be able to evaluate the use case after its implementation. Each Key Performance Indicator (KPI) has an *ID*, a *Name*, a *Description*, and some *Reference to mentioned use case objectives*, i.e., the objectives of template subsection 1.3 that are measured by the KPI. Template subsection 1.6. (*Use case conditions*) defines *Assumptions* and *Prerequisites* of the use case. Thereby, assumptions are general conditions at the design time of the use case, e.g., about existing systems and their configurations. Prerequisites define specific conditions, e.g., states of sensors and actuators, that should be met before initiating the use case. Template subsection 1.7. (*Further information to the use case for classification/mapping*) defines the *Relation to other use cases*, e.g., if the use case is an alternative to another use case, *Level of depth* of the use case description, e.g., high level, generic, or specialized, *Prioritisation*, typically High/Medium/Low or a similar scale, *Generic, regional, or national relation*, e.g., if the use case describes national-specific circumstances, the *Nature of the use case*, e.g., if it is a technical or a business use case, and a list of *Further keywords for classification*. Template subsection 1.8. (*General remarks*) can be used for additional comments that do not fit into any of the previous template subsections.

### Template section 2: Diagrams of use case

Template section 2 (*Diagram(s) of use case*) complements the use case description with additional diagrams, usually Unified Modeling Language (UML) diagrams. A typical and obvious starting point for this section is a use case diagram. Additional diagrams like sequence diagrams and activity diagrams then refine the use case. The purpose of this section is to provide a better understanding of the use case regarding its interactions and procedures.

### Template section 3: Technical details

Template section 3 consists of two subsections. In template subsection 3.1 (*Actors*), the actors that are involved in the use case are described. This description also involves *Grouping* (a name for the group of actors) and a short *Group description*. Actors may be grouped hierarchically, e.g., by using a group name like “InputOutput.Actuators”. Each group is a dedicated table starting with the group-specific information and then listing the corresponding actors by specifying an *Actor name*, *Actor type*, *Actor description*, and *Further information specific to this use case*. Examples for actor types are “Human”, “Device”, “Software program”, and “Organization”. *Grouping* and *Actor type* provide orthogonal classifications, e.g., the group “InputOutput.Actuators” may include actors of type “Device”, “Human”, etc., while actors of type “Device” may also be members of other groups, e.g., “InputOutput.Sensors”. The *Further information specific to this use case* field can be used to describe the actor and the properties relevant for this use case in more detail. Template subsection 3.2 (*References*) is used as a collection of relevant literature. Each reference should have a *No.*, *References type*, e.g., “Standards” or “Laws”,

the actual *Reference* to identify the document, e.g., the standard number and title, the *Status* of the referenced document, the expected *Impact on the use case*, i.e., how the use case is affected by the referenced document, the *Originator/organization* that created the referenced document, and possibly a *Link*, i.e., the Uniform Resource Locator (URL) to the referenced standard or legal paragraph.

### Template section 4: Step by step analysis of use case

Template section 4 adds additional information to the description of the use case in the form of describing possible scenarios step by step. The various scenarios can be derived from, and should comply with, the *Complete description* field of template subsection 1.4 and the diagrams of template section 2. In addition to the normal success scenario, additional scenarios like alternative solutions and failure scenarios may be specified. Template section 4 contains two template subsections. Template subsection 4.1 (*Overview of scenarios*) summarizes all scenarios of the use case in a single table and defines a *No.* for each scenario, the *Scenario name*, a short *Scenario description*, the *Primary actor* (or multiple) which triggers the scenario, the corresponding *Triggering event(s)*, *Pre-condition(s)*, and *Post-condition(s)*. After this general overview of the scenarios, a dedicated template subsections 4.2 (*Scenarios*) describes each scenario step by step. Each table links to the scenario it describes by stating the *Scenario name*, which is followed by the list of steps. Steps are defined starting with a sequential *Step No.*, which is a number optionally followed by a letter. Steps are hierarchically organized by separating them with a dot, e.g., with *Step No.* 1b.3a. The *Step No.* is followed by the *Event* triggering the step, e.g., the completion of the previous step, the *Name of process/activity* that can be used in a process diagram to illustrate the overall scenario, a *Description of process/activity* focusing on the interactions and information flow rather than technical details, the *Service* describing the “nature of the information flow” (typically one of Services defined in Table 2.1), the *Information producer (actor)*, the *Information receiver (actor)*, the *Information exchanged (IDs)* (cf. template section 5), and the *Requirement, R-IDs* field to state requirements (cf. template section 6) relevant for the step.

### Template section 5: Information exchanged

Template section 5 provides details about the information exchanged while executing a step of one of the scenarios defined in template section 4. It does not have any template subsections. The table (*Information exchanged*) consists of a unique *Information exchanged ID*, which is also used in template subsection 4.2, a short but unique *Name of information*, a *Description of information exchanged*, and a *Requirement, R-IDs* field to state requirements (cf. template section 6) relevant for the information exchange.

### Template section 6: Requirements

Template section 6 defines the requirements that were used in the various steps of the scenarios of template section 4. It does not have any template subsections. The template



Table 2.1: Services defined by IEC 62559, adapted from [90]

<i>Service</i>	<i>Description</i>
GET	The Receiver requests information from the Producer. GET is the default value if no service is defined.
CREATE	The information object is to be created at the Producer.
CHANGE	The information object is to be updated, i.e., the Producer updates the Receiver's information.
DELETE	The information is to be deleted, i.e., the Producer deletes information from the Receiver.
CANCEL/ CLOSE	Triggers a stop action related to processes, such as the closure of a work order or the cancellation of a control request.
EXECUTE	A complex transaction is being conveyed using a service, which potentially contains more than one verb.
REPORT	Unsolicited or asynchronous information is transferred from the Producer to the Receiver.
TIMER	Defines a waiting period in the scenario. When using the TIMER service, the Producer and Receiver fields shall refer to the same actor.
REPEAT	A series of steps is repeated until a condition or trigger event. The condition is specified as the text in the "Event" column for this row or step. The first and last step to be repeated are defined in parentheses after the keyword REPEAT in the form REPEAT(X-Y).

section is optional. Alternatively, a separate document can be used to collect and describe the requirements in more depth. It uses a separate table (*Requirements (optional)*) for each requirement category, whereby each table contains a list of requirements of the specific category. Therefore, it starts with the definition of the category. Categories have a *Categories ID*, a *Category name for requirements*, and a short *Category description*. The *Category IDs* can again be hierarchically organized. The category-related information is followed by a number of requirements that fit into this category. Each requirement is defined by a *Requirement R-ID*, a *Requirement name*, and a *Requirement description*.

### Template section 7: Common terms and definitions

Template section 7 provides a glossary for terms and definitions. It does not have any template subsections but only uses a simple table (*Common terms and definitions*) with one row for each *Term* and its corresponding *Definition*. If terms are used in multiple use cases, they should be added to template section 7 of each use case description.

### Template section 8: Custom information

Template section 8 is optional. It should only be included if it contains additional relevant information that does not fit into one of the other seven template sections. The table *Custom information (optional)* defines a unique *Key* to identify the information, a

*Value* (typically a short description, number value, etc.), and a *Refers to section* field to reference the corresponding template section.

### 2.2.2 SGAM Toolbox

SG applications are generally considered to be complex distributed systems. Their complexity mainly arises from the many stakeholders, technologies, and protocols involved. The SGAM Toolbox [94], which was introduced in [95] and described in detail in [96], applies MDE techniques to cope with this complexity. It is available as plug-in for Enterprise Architect (EA) [97], a software modeling tool created by Sparx Systems Ltd., and supports its users in the typical phases of MDE: creating the Computational Independent Model (MDE-CIM)<sup>1</sup>, Platform Independent Model (PIM), Platform Specific Model (PSM), and finally the Platform Specific Implementation (PSI). The corresponding models build upon UML. UML *profiles* are used to cover domain-specific aspects. Furthermore, the SGAM Toolbox provides a number of SGAM templates to support and simplify the modeling process. The individual models are created during three subsequent phases: the System Analysis Phase, the System Architecture Phase, and the Design & Implementation Phase, as illustrated in Figure 2.3. This development process is based on the Use Case Mapping Process (UCMP) introduced in [5] and briefly presented in the following.

#### Phase 1: System analysis

During the system analysis phase, stakeholder needs and system requirements are determined by conducting a use case analysis. Thereby, using the SGAM Toolbox, use case diagrams are created and detailed using activity diagrams and sequence diagrams. These diagrams make up the MDE-CIM and cover the top two layers of the SGAM: the business layer and the functional layer.

#### Phase 2: System architecture

Based on the MDE-CIM, the PIM is developed during the system architecture phase. It starts with identifying the relevant components. Actors derived from the use case analysis are mapped to physical components of the SGAM component layer or an application hosted by one of these components. According to the SGAM specification, the component layer should also include the communication hardware that allows the physical components to exchange messages. However, the developers of the SGAM Toolbox argue that it might be more useful to define the network topology and architecture rather than the devices and communication media on the component layer. On the SGAM information layer, information exchange between the physical components as well as the corresponding data model standards are defined. The necessary communication paths and protocols between components are added to the SGAM communication layer.

---

<sup>1</sup>Note that for the Computational Independent Model, MDE-CIM is used as an acronym here to distinguish it from the Common Information Model

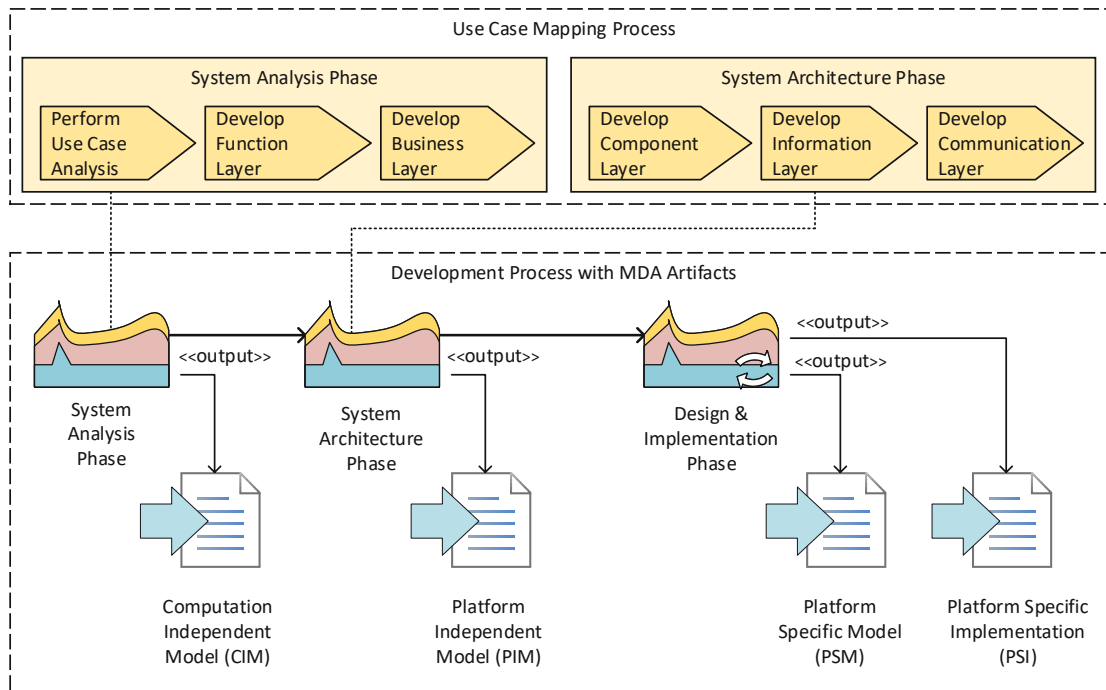


Figure 2.3: SGAM Toolbox methodology and its mapping to MDE [95]

### Phase 3: System design & implementation

The system design and implementation phase is an iterative process, whereby each iteration extends the PSM and the PSI by an additional use case. Thus, implementation is performed vertically across the SGAM layers, rather than layer per layer. The implementation is based on the diagrams and information derived during the previous two phases of the development process. Features of EA can be used to generate parts of the required code automatically. Generating not only structural elements of the code like classes and methods but also the corresponding functionality based on the specifications corresponding to the SGAM function layer is envisioned [96].

#### 2.2.3 NISTIR 7628 guidelines for smart grid cybersecurity

Security is an important aspect of dependability as security flaws not just affect the attributes confidentiality, integrity, and availability directly but may also impact most remaining dependability attributes. In almost any case, exploitation of a security vulnerability causes the system to deviate from its intended behavior, thus, affecting its reliability. Depending on the type of system, attackers may disable safety mechanisms or even directly cause damage, e.g., by overloading individual power system components. They may also target maintainability, e.g., by disabling remote access, scalability by occupying computational resources, or privacy by collecting user-specific information.

Therefore, several standards and documents specifically target security considerations of SG applications. Many of them, such as the *IEC 15408 – Common Criteria* [98] and the *IEC 27000 – Information technology – Security techniques – Information security management systems – Overview and vocabulary* [99] set of standards specify the terminology and give general recommendations, but do not provide a full guide for system developers to follow to derive a secure system. In this regard, the *National Institute of Standards and Technology Interagency Report (NISTIR) 7628 – Guidelines for Smart Grid Cybersecurity* [76] with its corresponding user’s guide [77] are more useful. The National Institute of Standards and Technology (NIST) identified seven different SG domains: *marketing, operations, service provider, bulk generation, transmission, distribution, and customer*. Depending on the specific use case, only some of these domains and connections might be subject to the NISTIR 7628 security analysis. Identifying the relevant domains and connections is an integral part of the methodology.

NISTIR 7628 is separated into eight activities, each containing several steps to be performed. The activities and steps are conducted one after another in strict order, which is illustrated in Figure 2.4. The activities are briefly summarized in the following. Each of the steps generates a specific outcome. This outcome typically is the selection of a responsible person, creation of a specific document, or creation/extension of a table. Outcomes of the individual steps are *emphasized*. It should be noted that NISTIR guidelines and the individual steps are intended to conduct a security assessment of an existing SG system, rather than supporting the development of new systems.

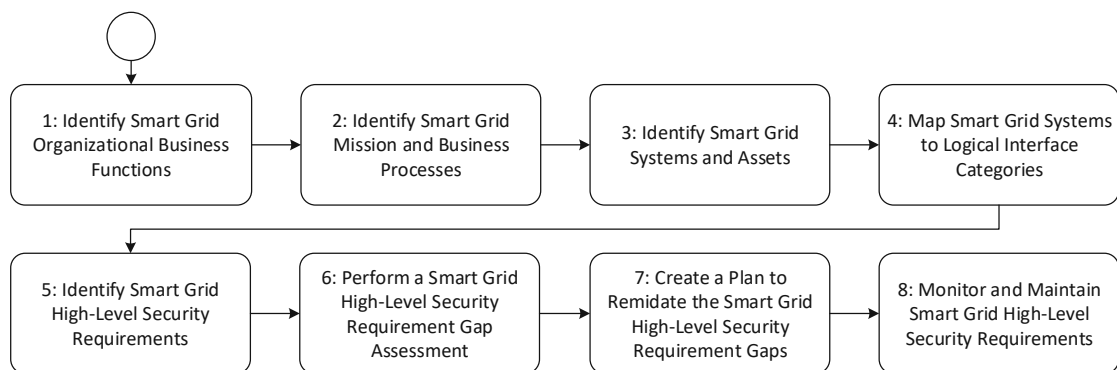


Figure 2.4: NISTIR 7628 user’s guide activities

### Activity 1: Identify smart grid organizational business functions

This activity addresses risk management on an organizational level. The company defines an *Executive Sponsor for Cybersecurity Risk Management Governance*. The executive sponsor selects the *Executive Cybersecurity Risk Management Governance Team*, which is responsible for implementing and monitoring the Risk Management Process (RMP). The executive cybersecurity risk management governance team identifies *Smart Grid Organizational Business Functions* and creates an *Organizational Business Function Risk*

*Profile Table*. Thereby, the possible threats and vulnerabilities for each business function are identified and rated. Furthermore, a *Priority Rating* is created for all organizational business functions, which reflects the business functions' criticality for the organization.

### Activity 2: Identify smart grid mission and business processes

In this activity, a *group of managers and subject matter experts* are assigned to the previously defined business functions. They identify *Supporting Business Processes (Dependencies)*, i.e., mission and business processes that support the business functions.

### Activity 3: Identify smart grid systems and assets

First, the *Smart Grid Systems Inventory* is created by associating SG systems to the individual business processes of the previous activity. Next, a *Risk Prioritization* determines the impact on the security attributes (Confidentiality, Integrity, and Availability (CIA)), their probability, and the risk for each system. Ratings (including impact levels, risks, and probabilities) in NISTIR 7628 typically use a metric based on three different values: H .. High, M .. Moderate, and L .. Low. The meanings of the various levels are defined in detail in Table 2.2. In a third step, information about available hardware for each of the SG systems is collected in a *Smart Grid Asset Inventory* table, which includes the assets' names, locations, serial numbers, logical addresses, and other information.

### Activity 4: Map smart grid systems to logical interface categories

Activity 4 is supported by NISTIR 7628 "Logical Reference Model" depicted in Figure 2.5. It is based on the seven different SG domains identified in NIST (cf. Section 2.2.3). Additionally, it specifies common actors for each of the domains, e.g., actor "27 – Distribution Management System". Actors that typically interact with each other are connected via logical interfaces. These logical interfaces have a unique number (starting with "U"), and each of them is associated to a Logical Interface Category (LIC), e.g., logical interface U9 between actor "27 – Distribution Management System" and actor "29 – Distribution SCADA" is member of LIC "5. Interface between control systems within the same organization", as highlighted by the red box in Figure 2.5. Detailed descriptions of all domains, actors, and LICs can be found in [76]. Even though the Logical Reference Model is fairly comprehensive, it is not intended to be complete. The system name specified in the Smart Grid Systems Inventory of Activity 3 may differ from the names of the actors specified in Figure 2.5. However, in many cases it is still possible to assign each system to an existing actor. Otherwise, an actor and the corresponding logical interfaces have to be added. Thus, within Activity 4, the Smart Grid Systems Inventory table of the previous activity is extended by three additional columns: *Actor(s)*, *Logical Interfaces*, and *Logical Interface Category(s)*.

Table 2.2: NIST CIA impact levels definitions [76]

<i>Potential Impact Levels</i>			
<i>CIA attribute</i>	<i>Low</i>	<i>Moderate</i>	<i>High</i>
<b>Confidentiality</b> Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information.	The unauthorized disclosure of information could be expected to have a <b>limited</b> adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized disclosure of information could be expected to have a <b>serious</b> adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized disclosure of information could be expected to have a <b>severe or catastrophic</b> adverse effect on organizational operations, organizational assets, or individuals.
<b>Integrity</b> Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity.	The unauthorized modification or destruction of information could be expected to have a <b>limited</b> adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized modification or destruction of information could be expected to have a <b>serious</b> adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized modification or destruction of information could be expected to have a <b>severe or catastrophic</b> adverse effect on organizational operations, organizational assets, or individuals.
<b>Availability</b> Ensuring timely and reliable access to and use of information.	The disruption of access to or use of information or an information system could be expected to have a <b>limited</b> adverse effect on organizational operations, organizational assets, or individuals.	The disruption of access to or use of information or an information system could be expected to have a <b>serious</b> adverse effect on organizational operations, organizational assets, or individuals.	The disruption of access to or use of information or an information system could be expected to have a <b>severe or catastrophic</b> adverse effect on organizational operations, organizational assets, or individuals.

### Activity 5: Identify smart grid high-level security requirements

NISTIR 7628 recommends CIA security levels for each of the LICs derived in the previous activity. These CIA security levels are termed the *NIST CIA Impact* and added as an additional column to the Smart Grid Systems Inventory of the previous activity. Additionally, a new metric – the *Organizational CIA Impact* – is derived by choosing either the CIA ranking from the Risk Prioritization of Activity 3 or the NIST CIA Impact from the column that has just been added. The choice should be made in a way such that it reflects the organization’s opinion about the risk associated with the specific SG system.

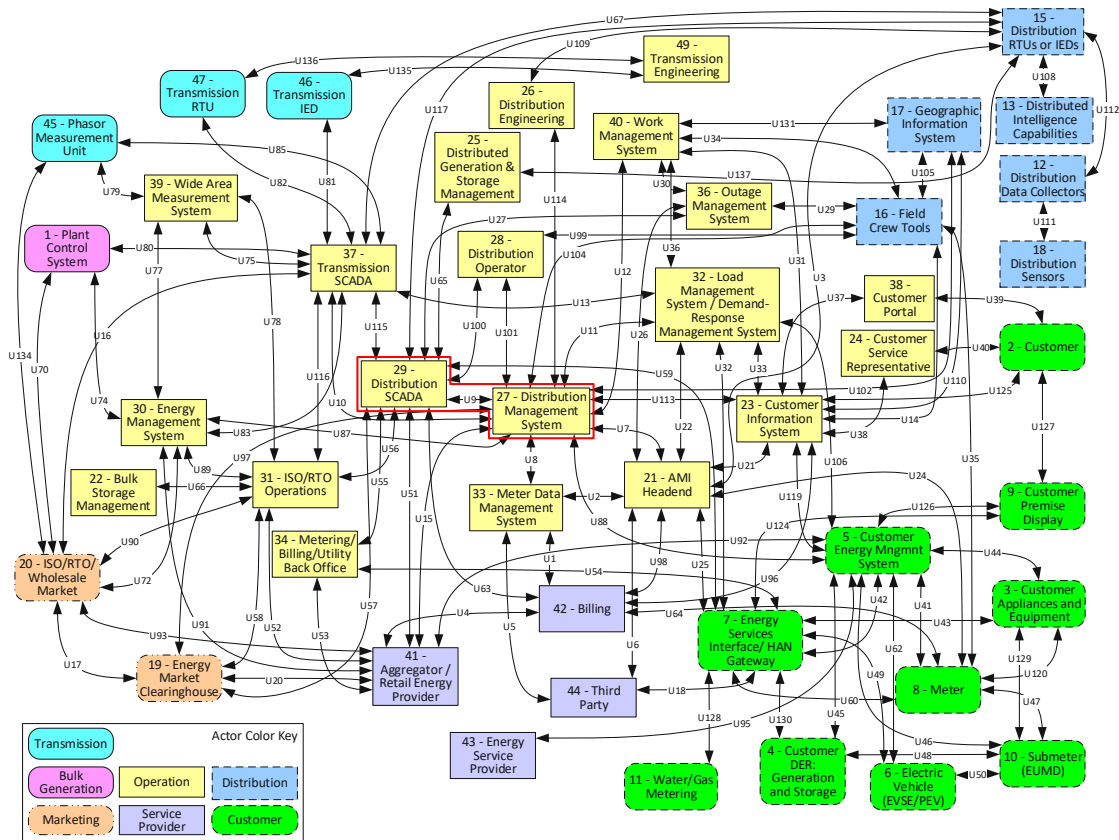


Figure 2.5: NIST IR 7628 logical reference model [76]

The Organizational CIA Impact is added to the table. Any differences between the NIST CIA Impact rating and the Organizational CIA impact rating should be investigated as they may indicate shortcomings in the organization’s risk assessment process.

Next, the *Requirements Type* and the *Requirements for Each System* columns are added to the Smart Grid Systems Inventory table. NIST distinguishes between three types of requirements: “Governance, Risk, and Compliance (GRC) Smart Grid Requirements”, “Common Technical Requirements (CTRs)”, and “Unique Technical Requirements (UTRs)”. For each combination of LIC (1-22) and Organizational CIA Impact (H, M, L), NISTIR 7628 specifies GRC Smart Grid Requirements, CTRs, and UTRs that should be fulfilled by the system. These requirements and their types are inserted into the two columns that have just been added. Furthermore, a *Consolidated UTR Reqs. for each system* column is added that combines only the UTRs of all LICs of the system.

### **Activity 6: Perform a smart grid high-level security requirement gap assessment**

From the previous activity, a list of relevant requirements (GRC Smart Grid Requirements, CTRs, and UTRs) has been derived. Next, it has to be checked to which extent these requirements are already fulfilled by the system. They can either be satisfied (“S”) or other than satisfied (“O”). Therefore, the *Assessment Ratings (S or O)* column is added to the table, and each requirement is rated. Additional information about the *assessment gaps* between the requirement and the current situation has to be collected for all requirements not satisfied yet. Therefore, the *Assessment Gaps* column is added and filled with the corresponding information for each unsatisfied requirement.

### **Activity 7: Create a plan to remediate the smart grid high-level security requirement gaps**

The objective of this activity is to define the necessary steps towards closing the gaps identified in the previous activity. The *Proposed Mitigations* column is added to the table to document the risk response actions and provide a short description for each Assessment Gap of the unsatisfied requirements. Specific actions to be taken are not provided by the standard but have to be specified by the organization individually. Under certain conditions, e.g., if the costs for closing a particular gap outweigh the benefits, the organization may decide to omit the corresponding requirement. Furthermore, a *Priority (H, M, L)* is added in a separate column to each of the Proposed Mitigations. Ordered by priority, detailed *plans to address selected gaps* are created based on the proposed mitigations.

### **Activity 8: Monitor and maintain smart grid high-level security requirements**

In this final activity, the progress of executing each of the plans to address selected gaps specified in the previous activity is monitored and documented. The standard does not prescribe how the documentation of the progress should be implemented but suggests the use of automated tools and documented processes whenever possible. In the event of significant changes, e.g., the identification of new security threats, the whole process (Activities 1-8) shall be repeated, and the corresponding documentation shall be revised.

## **2.3 MAS design**

Even though the definition of MASs does not restrict agents to being software components (an agent is “a software (or hardware) entity” [13]), most agents are implemented in software, possibly controlling hardware. In many cases, presupposed that the programming language chosen provides the possibility, agents are objects in the sense of Object-Oriented Programming (OOP). Thus, OOP techniques are often applied when designing and implementing agents. However, a comprehensive MAS design method-



ology has to cover additional aspects, e.g., distributing functionality across different types (classes) of agents and the design of agent interaction protocols required for agent communication. According to Urbano, Wagner, and Göhrner [100], existing MAS design methodologies cover different phases of the systems engineering process. The most prominent MAS design methodologies, and which phases of the development process they cover, are illustrated in Figure 2.6. A distinct selection of the most widely used MAS design methodologies is discussed in the following. However, many other methodologies (e.g., Agent UML [101] and Decentralized Iterative Multiagent Open Networks Design (DIAMOND) [102]) exist.

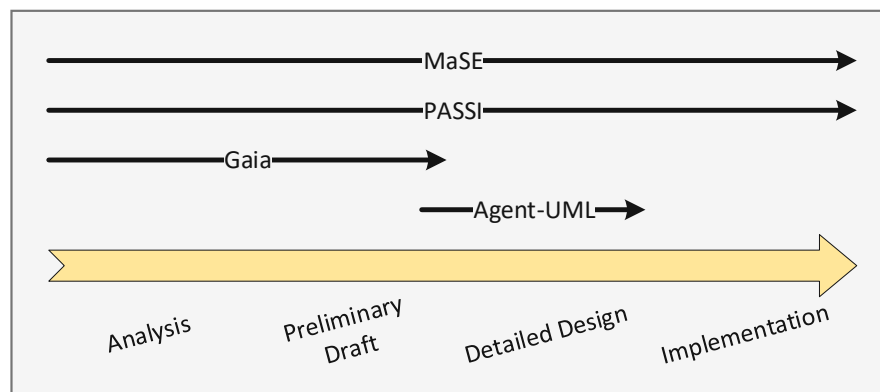


Figure 2.6: MAS design methodologies [100]

### 2.3.1 Gaia

The Gaia agent-oriented analysis and design methodology [103] owes its name to the *Gaia hypothesis*, which proposes that all living organisms interact in a self-regulating system that maintains conditions for life on the planet [104]. This point of view is closely related to the idea of MASs, in which individual, self-organizing agents cooperate to achieve a common goal. Gaia considers a MAS to be “an artificial society or organization” [103]. It comes with several assumptions and limitations:

- Agents can make use of “significant computational resources”.
- Agents cooperate to achieve a common goal, e.g., optimizing or minimizing some property. There are no true conflicts, in which the common goal cannot be achieved because of the self-interests of individual agents.
- Agents are heterogeneous, i.e., they may be implemented using different hardware, programming languages, and architectures.
- Inter-agent relationships are static, i.e., they do not change during runtime.
- The abilities of agents are static.
- The number of different agent types is limited (less than 100).

The Gaia methodology is separated into two stages: the Analysis stage and the Design stage. Each stage yields several models. Figure 2.7 illustrates how these models relate to one another. Concepts for identifying and describing requirements are out of the scope of the Gaia methodology. Thus, the following discussion starts with the analysis stage. Thereby, the individual models are *emphasized*.

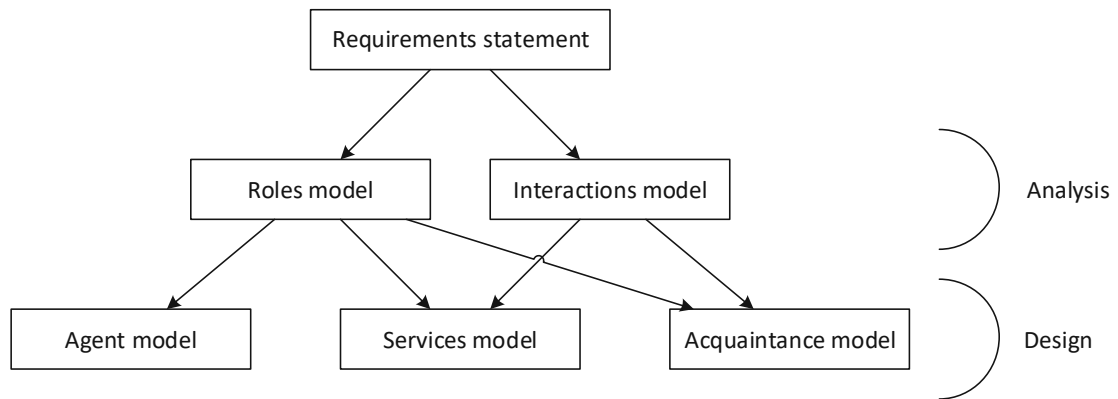


Figure 2.7: Gaia agent design methodology [103]

### Stage 1: Analysis

The analysis stage is implementation-independent and intended to develop a basic understanding of the MAS. Following the view of MASs as a society or organization, the overall system is constructed from a set of roles. These roles are covered by the *roles model*. Interactions between roles yield the *interactions model* (cf. Figure 2.7). Thereby, roles are abstract concepts, i.e., there is no individual (computing system or person) defined that takes on a specific role. An example of such a role is “President”: at this stage of development, it is irrelevant which specific person currently is president.

The *roles model* consists of all the roles that participate in the MAS. A role is defined by its responsibilities, permissions, activities, and protocols. Responsibilities describe the functionality of the role, i.e., the tasks it is designed to fulfill. Thereby, Gaia distinguishes between liveness responsibilities, which state that “something good eventually happens” [105], i.e., the agent is alive and performs specific tasks, and safety responsibilities, which state that “some bad thing does not happen”. The agent is required to have a set of permissions, e.g., to access specific data or to interact with its environment by exchanging messages, reading sensor values, and controlling actuators. Activities are associated with a specific role, but in contrast to responsibilities, activities are internal operations invisible to other agents. Finally, protocols are the link between roles and interactions. Gaia defines a template for role schemata (cf. Figure 2.3) to support the developer in this aspect of the process. The template provides fields for responsibilities, permissions, activities, and protocols as well as placeholders for some additional information, in particular the role’s name and its description.

Table 2.3: Gaia role schema template [103]

<i>Role schema</i>	<i>Name of role</i>
Description	Short description of the role
Protocols and activities	Protocols and activities in which the role plays a part
Permissions	“Rights” associated with the role
Responsibilities	
Liveness	Liveness responsibilities
Safety	Safety responsibilities

Interactions define the way a role can interact with other roles on a high level. Therefore, the *interactions model* consists of several protocol definitions. Details like the type, structure, and sequence of exchanged messages are omitted at this point. Instead, each protocol definition only contains information about the purpose of the protocol (in the form of a few words), the initiating role, other (responding) roles, inputs (information provided by the initiating agent at the start of the protocol), outputs (information provided by/to the responding agent during protocol execution), and processing (a short textual description about processing performed by the initiating agent during protocol execution). Unfortunately, Gaia does not provide a protocol specification template.

### Stage 2: Design

The Design stage is the second and final stage of the Gaia methodology. The objective of the Design stage is to refine the role model and the interaction model, which were created during the Analysis stage, to a level allowing traditional software development techniques to be used to implement the specified agents. Thereby, three more models are created: the *agent model*, the *services model* and the *acquaintance model* (cf. Figure 2.7). They are briefly discussed in the following.

The *agent model* is the set of all agent types. An agent type is thereby built from a single or a combination of agent roles. While there is a one-to-one correspondence between agent roles and agent types in many cases, this is not mandatory. A single agent type may also incorporate multiple roles. However, a single role cannot be split among multiple agent types. For example, the agent role “President” requires the definition of a corresponding agent type, e.g., “PresidentAgent”. Assuming there is another agent role “HeadOfGovernment”, the PresidentAgent may combine both roles into a single type. However, as an individual role can never be split among multiple types, only a single PresidentAgent can incorporate the President role. Additional agent types may be introduced, which combine the functionality of several other agent types. This structure can be thought of and documented as a tree structure, having agent roles as leaves and agent types as inner nodes. The number of instances of each agent type is specified using annotations. This annotation may be an exact number ( $n$ ), a range of possible numbers ( $m..n$ ), zero or more ( $*$ ), or at least one ( $+$ ).

For every activity defined in the roles model during the Analysis stage, an associated service is defined during the Design stage and added to the *services model*. Additionally, there may also be services defined that do not originate from the roles model. A service can best be described as a function of an agent or a “single, coherent block of activity in which an agent will engage” [103]. For each service, four properties have to be specified: inputs, outputs, pre-conditions, and post-conditions. Typically, a table is used to document the services and their properties.

Finally, the *acquaintance model* simply defines which communication relationships exist between agent types. The acquaintance model is a simple directed graph, having an arc  $a \rightarrow b$  if agent type  $a$  can send a message to agent type  $b$ . This model completes the Gaia methodology. The derived *agent model*, *services model*, and *acquaintance model* can now be implemented.

### 2.3.2 MaSE

In contrast to Gaia, the Multiagent Systems Engineering (MaSE) methodology [106, 107] also covers the detailed design and implementation phases of the agent development life-cycle. It comes with a number of limitations regarding the suitable types of MASs:

- The MAS has to be closed, meaning that all communication happening within the system is only performed directly between agents. However, agents themselves may have interfaces to communicate with entities outside of the MAS.
- Systems in which agents can be added, removed, or changed during runtime are out of scope.
- MaSE does, in general, assume that all conversations between agents follow a one-to-one scheme. However, multicast may still be used for message exchange at a lower level.
- It is suggested to use the MaSE methodology only for MASs with a small number (typically ten or less) of different agent types, as it has not been validated for larger systems.

MaSE is fundamentally based on the fact that agents are significantly more complex than software objects known from Object-Oriented Design (OOD). Therefore, it specifies a sophisticated methodology, which is illustrated in Figure 2.8. Arrows between the individual models show their interdependencies, i.e., how models within different phases are built upon one another. The individual phases of this methodology and their corresponding graphical models are discussed in more detail in the following. Like Gaia, MaSE also assumes that the requirements description is given. Therefore, the requirements description is not covered by the methodology.

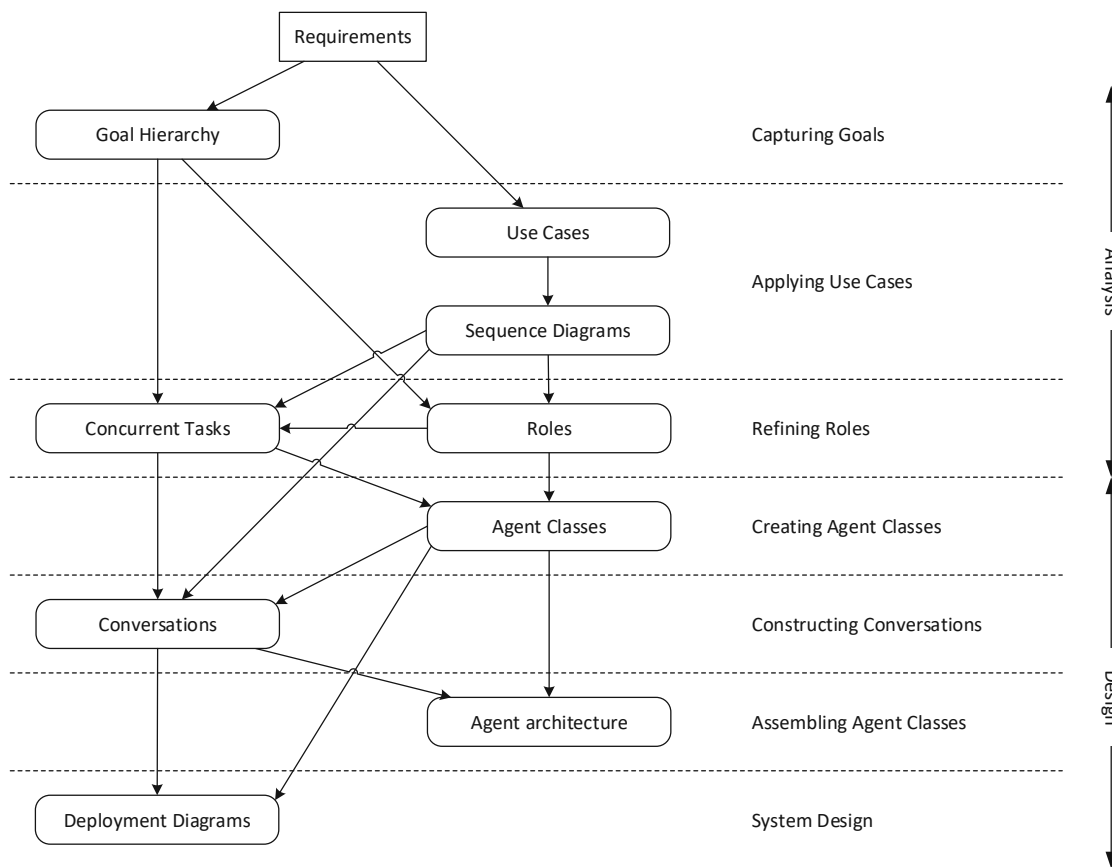


Figure 2.8: MaSE agent design methodology [106]

### Phase 1: Capturing goals

During this phase, the overall system goals are defined. Furthermore, they are refined in the sense that complex goals are subdivided into simpler, more fine-grain ones. Each goal is identified by a unique number. A hierarchy diagram (the *Goal Hierarchy* model) is used to keep track of how the goals build upon one another. Each fine-grain goal may contribute to achieve multiple, more complex goals. Thus, the hierarchy diagram is not necessarily a tree-like structure.

### Phase 2: Applying use cases

*Use cases* are derived directly from the system requirements. During this phase of the methodology, the use cases are defined in terms of messages exchanged between individual agent roles. *Sequence diagrams* are used for this purpose. For each identified use case, there should be at least one sequence diagram. However, there may be multiple sequence diagrams per use case, if it covers several possible scenarios.

### Phase 3: Refining roles

Goals of the MAS are accounted for by associating them to roles. Typically, there are one-to-one associations between roles and goals. However, it may also be the case that a single role handles multiple goals, but also that a single goal requires multiple roles to be created. As a starting point for refining roles, a conventional Object Role Model (ORM) is used to denote the roles (as rectangles), their associated goals (within the rectangles), and possible communication paths (using undirected and directed arcs for bidirectional and unidirectional message exchange between roles, respectively).

MaSE also defines a more detailed version of a conventional ORM, the MaSE role model. It adds the notion of tasks. Tasks are derived from the goals of each agent role, added as additional nodes in the form of ovals to the diagram, and connected to their responsible roles. Tasks cannot be shared among multiple roles. Instead, additional roles and tasks have to be added in this case. Furthermore, communication paths are illustrated in more detail than in a conventional ORM diagram. Communication paths link specific tasks to one another, rather than the corresponding roles. Communication links are illustrated with directed arcs, pointing from the initiator to the responder, and are annotated with the names of the communication protocols involved. Communication links between concurrent tasks of the same role are illustrated with a dashed arc. Thus, the MaSE role model covers *Roles* as well as *Concurrent Tasks*. Finally, a detailed state diagram is defined for each task. Thereby, transitions between the individual states are typically triggered by sending messages to/receiving messages from other agents.

### Phase 4: Creating agent classes

During this phase, *Agent Classes* are created, and the identified agent roles are assigned to these classes. The resulting agent class diagram is based on the previously created MaSE role model, whereby all roles are consolidated into the same number or fewer classes. Arcs between classes represent conversations. These conversations are derived from the MaSE role model, whereby, if a role was participating in a specific communication protocol in the MaSE role model, the class subsuming this role now has to participate in the corresponding conversation. Regarding the directions of arcs, the same rule as in the MaSE role model applies.

### Phase 5: Constructing conversations

The *Conversations* identified in the previous phase are now modeled using a pair of state diagrams for each conversation, one state diagram for each participant. The state diagrams have to be consistent with all sequence diagrams derived in Phase 2. Phase 5 is closely related to the subsequent Assembling agent classes phase, as any activity causing a state transition has to be added as a method to the corresponding agent class.

### Phase 6: Assembling agent classes

Agent classes are defined in detail during this phase. An agent typically consists of various components, e.g., a controller, Input/Output (I/O)-interfaces, and communication interfaces. Robinson introduces five templates to describe the various components [108]. The MAS developer may build upon these templates or specify completely new ones. Additionally, MaSE uses a class-diagram-like model to define the components, their internal variables, operations (methods), connections among each of the components of an agent, and connections to external resources like sensors, actuators, and other agents. The interactions between components can be described in more detail, e.g., by using state diagrams. All diagrams combined provide a detailed description of the *Agent Architecture*.

### Phase 7: System design

The System design phase is considered to be the most straight-forward phase of the MaSE methodology. Therein, agent classes are instantiated as individual agents. Instantiating agents from agent classes in MaSE is essentially equivalent to instantiating objects from classes in OOP. *Deployment diagrams* are used to show the “numbers, types, and locations of agents within a system” [106]. Lines connect agents participating in the same conversations. Furthermore, agents sharing the same physical platform are placed within a dashed box. This completes the MaSE design process. Code generation is suggested to transform the derived models to code fragments automatically. However, code generation is not mandatory and, therefore, not part of MaSE itself.

#### 2.3.3 PASSI

Another commonly used approach for MAS design is the PASSI [109] methodology. It builds upon concepts known from OOD and Artificial Intelligence (AI). In contrast to the previously discussed methodologies Gaia and MaSE, the PASSI approach has been designed with a specific target environment in mind: the Foundation for Intelligent Physical Agents (FIPA) MAS framework (cf. Section 2.3.5). It uses slightly extended UML diagrams for specifying the MAS. The PASSI methodology guides the user in step-wise improving and detailing the various MAS models. Furthermore, code generation is supported by the corresponding tools. Figure 2.9 illustrates the models and phases of the PASSI methodology. The specification of the PASSI methodology primarily builds upon its models. Therefore, also the next sections follow this structure. Phases conducted to derive these models are *emphasized*.

#### Model 1: System requirements

During the *Domain Requirements Description* phase, a functional description of the MAS is created. This description is based on use case diagrams on an abstract level. The individual scenarios are discussed in more detail using sequence diagrams at a later point in the methodology.

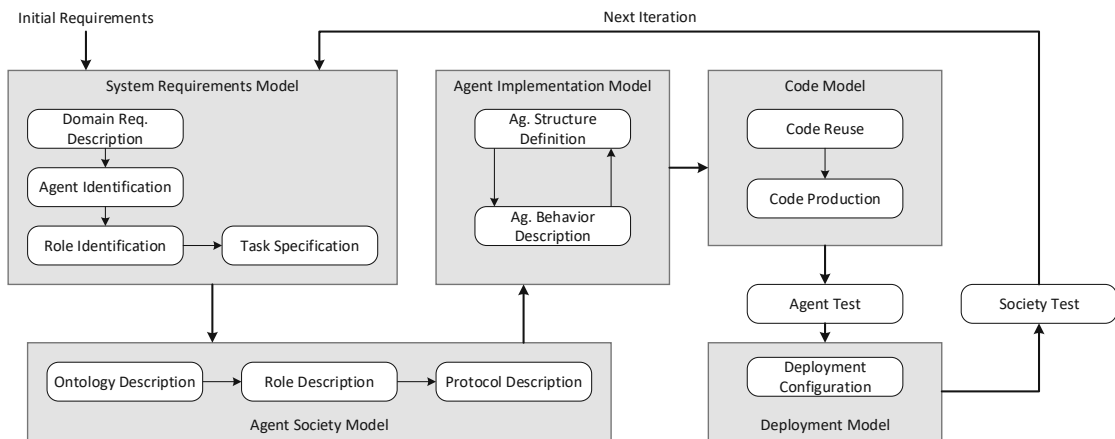


Figure 2.9: Passi agent design methodology [109]

Already from the very beginning, the *Agent Identification* phase tries to identify which agents might exist in the final MAS by grouping closely related use cases from the use case diagrams to packages. Each of these packages defines the functionalities of an agent and is named accordingly. The «include» and «extend» relationships between the use cases remain unless the connected use cases are associated with different packages, i.e., agents. In these cases, the relationships are renamed to «communicate», as agent communication is required if the corresponding use case is triggered. These relations shall point from the initiator to the participant.

Each communication path in the modified use case diagram, i.e., each sequence of «communicate» relationships, is a specific interaction scenario. Thus, the set of communication paths is equal to the set of scenarios. However, many of these scenarios are not relevant for real applications. These are not examined any further. The remaining scenarios form the set of possible scenarios. Agents play specific roles when participating in possible scenarios. They may even participate in the same scenario multiple times by taking several different roles simultaneously. This distinction between agents and roles is made during the *Role Identification* phase. Possible scenarios are defined using sequence diagrams, whereby the objects of the sequence diagrams are named using the following syntax: «role\_name» : «agent\_name». Agent roles are derived from the objects participating in possible scenarios.

Next, during the *Task Specification* phase, agents are described in more detail by breaking their roles down to a set of tasks. A task is some well-defined, encapsulated functionality. For this purpose, PASSI introduces task specification diagrams, a special type of activity diagram. These task specification diagrams have exactly two swimlanes, whereby the right swimlane contains activities having a one-to-one relation to the agent's tasks, and the left swimlane contains activities of other interacting agents. One task specification diagram is created for each agent.



## Model 2: Agent society

Agents have to share the same notion of semantics to be able to communicate. In PASSI, ontologies are used to explicitly model semantics. Approaches using UML diagrams for ontology design exist in the literature [110, 111] and are adopted by the PASSI methodology in the *Ontology Description* phase in the form of a domain ontology diagram and a communication ontology diagram. Both are class-diagram-like diagrams, and classes can be related via arbitrary relationships. In the domain ontology diagram, each class implements one of the following interfaces:

- *Concept*: A concept represents some entity of the domain. Concepts can have attributes as their class members, e.g., a book has a title and a publisher.
- *Predicate*: Predicates are “assertions on properties of concepts” [109]. For example, to successfully purchase a book, there has to be an associated successful negotiation about a specific price and delivery time.
- *Action*: Actions describe events that already happened. Thereby, they typically refer to concepts of the ontology and probably provide some extra information. For example, a book has been delivered at a specific date.

The communication ontology diagram is derived from the results of the *Agent Identification* phase. As a starting point, each agent is represented by a dedicated class. For each agent, the data structures required for the interactions it participates in are specified as class attributes. Relationships between classes represent interactions between agents. The direction of the relationship remains unchanged, i.e., it still points from the initiator to the participant. As agents play a specific role when participating in an interaction, these roles are stated at the beginning and the end of each relationship arc. Additional information, modeled as an association class, is added to each of the relationships to define three interaction properties:

- *Protocol*: Interactions in MASs follow a specific sequence, i.e., order and type of messages. This sequence is called interaction protocol, communication protocol, or simply protocol. As mentioned, PASSI uses FIPA as MAS framework. Therefore, FIPA communication protocols (cf. Section 2.3.5) shall be used as protocols whenever suitable.
- *Ontology*: The ontology used during the interaction is stated. This ensures semantically meaningful messages, as it eliminates the risk that the same term may have different meanings in different ontologies.
- *Language*: The language attribute defines message encoding. Multiple languages may be used in a single MAS.

The *Roles Description* phase describes the life-cycle of agents (how they change roles), the conversations they participate in, and their collaborations. Therefore, PASSI defines the roles description diagram, which is again similar to a class-diagram but uses agent

roles as classes. These roles and the corresponding classes can easily be derived from the communication ontology diagram, where they have been stated on the relationships. Roles originating from the same agent are grouped into packages. Tasks, which already have been defined based on agents in the *Task Specification* phase, are now assigned to the individual roles by putting them in the operations compartment of the corresponding class. Additional methods may be added if they are required from an implementation point of view. Of course, tasks may be used by more than one role of an agent and, therefore, put into multiple classes. If an agent incorporates multiple roles and these roles depend on one another (e.g., a Purchaser agent has to act as Negotiator and, if successful, then acts as OrderPlacer), a dashed, directed edge is drawn from the first to the second role and marked with the keyword [ROLE CHANGE]. Conversations between roles are also derived from the communication ontology diagram. Finally, roles may depend on one another to achieve their goals. Such a dependency may indicate a service that another agent has to offer or a resource that another agent has to provide. Dependencies are added to the roles description diagram in form of additional relationships named “service” or “resource”, respectively.

Each conversation between agents follows a specific sequence of messages. FIPA already specified several sequence diagrams, which can be used by the developer for this purpose. If no suitable conversation is provided by FIPA, developers can define sequence diagrams during the *Protocol Description* phase.

### Model 3: Agent implementation

In any MAS design methodology, the question arises whether it should focus on the structural or the behavioral definition of agents first. As they are typically closely intertwined, it is hardly possible to consider all relevant aspects of one, while entirely postponing the other. PASSI proposes an iterative approach to address this issue. The corresponding phases are the *Agent Structure Definition* and the *Agent Behavior Definition*. They are conducted in parallel to form the agent implementation model. Furthermore, they both operate on two different structural levels, which themselves are suggested to be developed iteratively: the multi-agent and the single-agent level.

The Multi-Agent Structure Definition (MASD) uses a class diagram to depict all agents and their connections on a rather high level of abstraction. Thereby, classes represent agents. Methods (still representing tasks at this point) of the corresponding roles are simply combined and listed in the agents’ operation compartments. Communication relations between agents are indicated via directed, unnamed relationships. Furthermore, relations to the outside world, e.g., via sensors or actuators are added to the diagram in the form of actors. Relationships/interactions among actors and agents are illustrated via directed edges in the direction of causality from the initiator to the participant.

During the Single-Agent Structure Definition (SASD), a class diagram is designed for each agent. Each class diagram describes an agent’s internal structure and functionality

in detail and forms the basis of later code generation and implementation. Thus, it already has to take the Agent Platform (AP) into account. PASSI assumes FIPA as an AP. The class diagram consists of two parts: the description of the agent's main class (its attributes and methods), and one class per task, i.e., per method in the MASD. Attributes and methods required for implementing each task are added to the classes. The classes representing tasks from the MASD are also called the agent's inner classes.

The Multi-Agent Behavior Description (MABD) consists of several activity diagrams defining the necessary communication between agents as well as method invocations within agents. Each activity diagram describes a scenario that may occur in the MAS. The starting point of each scenario is a function call that can be triggered by an actor, a timeout, or some other event. Each swimlane of such an activity diagram represents a task of an agent that participates in the scenario.

The Single-Agent Behavior Description (SABD) defines how agent and task methods shall be implemented. No specific diagram type is suggested by PASSI for this purpose. Any methods known from classical OOP like flow charts, state diagrams, and textual descriptions may be applied.

#### Model 4: Code

PASSI tries to largely automate the code generation process by reusing predefined design patterns of agents and tasks during the *Code Reuse* phase. Thereby, patterns are considered to be “pieces of design and code to be reused in the process of implementing new systems” [109]. PASSI identifies three different types of patterns:

- Patterns concerning models about the static structure of agents
- Patterns concerning models about the dynamic behavior of agents
- Patterns concerning the agents' program code

Additional literature elaborates on patterns in the PASSI methodology in more detail, e.g., [112, 113]. Two software components that support developers in the design process and encourage the use of patterns were developed in accordance with the PASSI methodology: the PASSI Tool Kit (PTK) [114] and the PASSI Agent Factory [115]. Thereby, the PTK is used to compile the PASSI diagrams and to generate code skeletons, which are then filled by the PASSI Agent Factory. While most of the PASSI methodology itself is in principle independent from the choice of an AP, the PTK and Agent Factory are designed to be compatible with the Java Agent DEvelopment framework (JADE) platform [116], which itself builds upon the FIPA specifications.

Program code that cannot be automatically generated, e.g., because it is specific to the MAS's purpose and, therefore, no pattern exists, has to be written manually by programmers during the *Code Production* phase. However, they can build upon the code

skeleton generated by the PTK and, therefore, fully concentrate on implementing the methods rather than building the complete code structure themselves.

After having implemented the individual agents, their functionality is tested during the *Agent Test* phase. These tests are carried out based on the initial requirements specification. PASSI does not address this phase in detail but leaves it to standard software testing techniques.

### Model 5: Deployment

During the *Deployment Configuration* phase, the locations, movements, and communication support (which depends on the processing unit they are executed on) of agents are specified using a PASSI deployment configuration diagram, which builds upon a standard UML deployment diagram. Processing units are represented by boxes, containing the agents they host as components. A simple syntax extension, the introduction of directed, dashed lines with a `move_to` stereotype, allows to additionally represent agent mobility, i.e., the possibility of moving agents from one AP to another during runtime.

Finally, the PASSI methodology is completed by performing a *Society Test*. In contrast to the *Agent Test*, the *Society Test* is performed after the MAS has been deployed. It validates the compliance of the complete system with the requirements specification.

#### 2.3.4 MAS methodology comparison

With over 80 MAS design methodologies [117], there are plenty of options available to design and implement a specific MAS use case. The difficulty primarily lies in selecting a methodology that is suitable for the present application. For this reason, various publications [118, 119, 120] define lists of criteria used to analyze existing MAS design methodologies. These lists include hard facts, such as the SELC phases supported by each methodology and the number of steps within these phases, but also more subtle properties like clear notation, adequateness and expressiveness of the language, clarity of concepts, easiness to learn, and easiness to use. Tran, Low, and Williams [118] define such a list of criteria and provide the results for five different MAS design methodologies, including Gaia and MaSE. Table 2.4 summarizes a subset of the criteria for these two methodologies. Additionally, the PASSI methodology, which is not present in the original publication, has been evaluated based on the information provided in Section 2.3.3 and added to the table.

#### 2.3.5 FIPA MAS framework

MASs are deployed in many different fields of application, including SGs, energy markets, manufacturing, and telecommunications. The predominant MAS framework for applications in all of these domains is the FIPA framework. FIPA consists of a set of specifications, some of which are considered mandatory when designing a FIPA-compliant

Table 2.4: Feature analysis of MAS design methodologies, extended from [118]

<i>Evaluation criteria</i>	<i>GAIA</i>	<i>MaSE</i>	<i>PASSI</i>
<b><i>Process-related criteria</i></b>			
Development life cycle	Iterative across all phases	Iterative across all phases	Iterative across all phases
Coverage of the life cycle	Analysis & Design	Analysis & Design	Analysis, Design, Implementation & Evaluation
Application domain	Any	Any	Any, focus on FIPA platform
Size of MAS	$\leq 100$ agents	$\leq 10$ agents	not specified
Usability of the methodology	Medium. Missing many important steps	High, except for internal agent modeling	High
Approach towards MAS development	object-oriented, role-oriented, organisation-oriented	object-oriented, role-oriented, goal-oriented	object-oriented, role-oriented, use-case-oriented
<b><i>Model Related Criteria</i></b>			
Completeness	Medium	High	High
Formalization/Preciseness	High	High	High
Ease of understanding	High	High	High
Adaptability	No	No	No
Communication ability	No	Yes	Yes
Concurrency	No	Yes	Yes
Human-computer interaction	No	No	Yes
Models reuse	Yes	Yes	Yes
<b><i>Supportive Feature Criteria</i></b>			
Software and methodological support	No	Yes	Yes
Open systems and scalability	Yes	No	Yes
Dynamic structure	Yes	No	Yes*
Agility and robustness	No	No	No
Support for conventional objects	No	No	Yes
Support for mobile agents	No	No	Yes*
Support for self-interested agents	Yes	No	not specified
Support for ontology	No	No	Yes

\* provided by the AP

MAS. Others, e.g., specifications defining details about agent communication such as the various agent interaction protocols, are optional and, typically, only implemented if required for the specific use case.

FIPA rigorously keeps track of the different states and versions of the specifications. The FIPA specification life-cycle is illustrated in Figure 2.10. Most of the 25 specifications that made it to standardization so far can roughly be assigned to three different categories: FIPA basic concepts, agent communication, and ontologies. In the following, a separate section is devoted to each of these three categories. Additionally, the FIPA Nomadic Application Support Specification [121] specifies how to implement agent mobility, and FIPA Design Process Documentation Template [122] provides information about the MAS design process essentially by referring to the PASSI methodology (cf. Section 2.3.3).

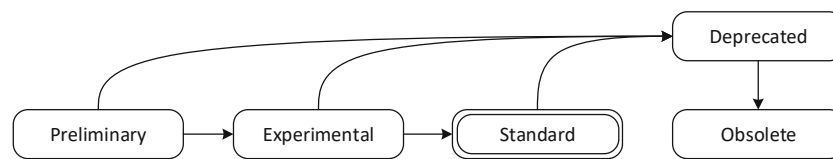


Figure 2.10: Life-cycle of FIPA specifications [123]

### FIPA basic concepts

The main objective of FIPA is not to define a specification that can directly be implemented in software but to identify architectural elements of MASs and their relationships. This information is summarized in the FIPA Abstract Architecture [124] specification. The FIPA Abstract Architecture and possible mappings to concrete realizations are illustrated in Figure 2.11. The minimum set of architectural elements that should be supported by any FIPA-compliant MAS consists of *Message Transport*, *Agent Directory*, *Service Directory*, and *ACL*. Agent Directory and Service Directory allow agents and services to be registered and discovered. Message Transport and ACL are outlined in more detail in the following section.

According to the FIPA Agent Management Reference Model, which is defined in the FIPA Agent Management Specification [125], agents require an AP to be executed on. The AP subsumes the machine, operating system, agent support software, Directory Facilitator (DF), Agent Management System (AMS), Message Transport System (MTS), and the agents themselves. It is illustrated in Figure 2.12, and its components are briefly discussed in the following.

The first component of the AP is the Agent. FIPA defines an agent as a “a computational process that implements the autonomous, communicating functionality of an application” [125]. However, it is not clearly defined, whether the Agent itself should be considered an integral part of the AP, or just be executed on the AP. Both view points are common, and their distinction is not particularly relevant in practice.

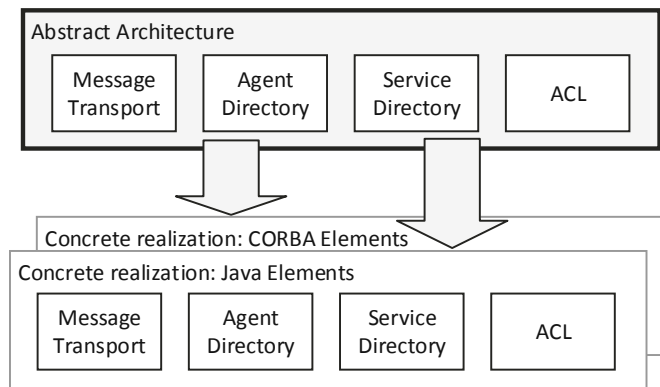


Figure 2.11: FIPA abstract architecture and concrete realizations [124]

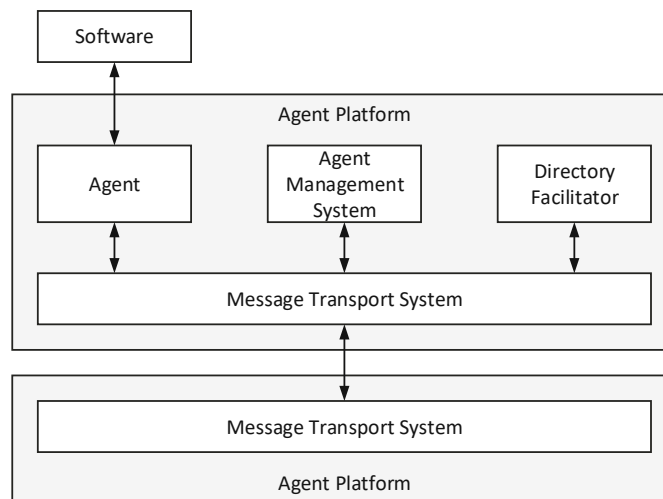


Figure 2.12: FIPA agent management reference model [125]

The AP itself is controlled by the AMS, which is a mandatory component on each AP. It manages how the AP may be accessed by other agents and used by agents that are executed on the same AP. Each agent has to register with an AMS to receive its unique Agent Identifier (AID).

Agents may register themselves, including their full description (cf. Section 2.3.5) and descriptions of their services, at the DF. The DF is an optional component and, as such, need not be implemented on every AP. If present, it can be used (also by agents that are executed on other APs) to register and discover agents and services.

Finally, the mandatory MTS provides the communication mechanisms. It enables communication among the individual components of the AP but also to other APs. The

MTS may use different encoding mechanisms depending on the destination of the message. For example, if the message is sent over a wireless connection, the MTS may choose a very efficient encoding scheme in terms of message size to minimize power consumption.

### Agent communication

Most of the FIPA specifications (20 out of 25) define various aspects of agent communication. Some of them specify FIPA-ACL, a concrete realization of the ACL architectural element. Others specify the message transport, i.e., how FIPA-messages can be encoded in XML, binary format, or plain text [126], and how messages are transported, e.g., via HyperText Transfer Protocol (HTTP).

The FIPA-ACL Message Structure Specification [127] is of particular interest. It lists the basic components and the structure of FIPA-ACL messages. FIPA specifies a total number of 13 different fields for a FIPA-ACL message, some of which are optional. The first and most important field of a FIPA message is the performative denoting the type of the *Communicative Act (CA)*. All predefined CAs are defined in the FIPA Communicative Act Library Specification [128]. A selection is listed in Table 2.5.

Table 2.5: FIPA communicative acts [128]

<i>Communicative act</i>	<i>Description (simplified)</i>
query-ref	Ask another agent for some specific information.
inform-ref	Inform another agent about some specific information.
inform	Inform another agent that a specific proposition is true.
call for proposal	Ask for proposals to perform a specific action.
propose	Submit a proposal as a response to a call for proposal.
accept proposal	Accept a received proposal.
reject	Reject a received proposal.
request	Request a specific action from another agent. Note that a request for inform-ref is the same as query-ref.
agree	Agree to perform some requested action.
refuse	Refuse to perform some action or to offer a proposal.
not understood	The agent received a message it did not understand.

Often, the sequence of CAs follows a predefined *agent interaction protocol*, which is denoted in the FIPA message by the `protocol` field. Popular examples are the FIPA Query Interaction Protocol [129] and the FIPA Contract Net Interaction Protocol [130]. Messages can be associated to a specific interaction by making use of the `conversation-id` field. Using different `conversation-ids` allows an agent to participate in multiple interactions simultaneously without much additional programming effort.



The `performative` specifies the FIPA message type, whereas additional information is stored in the `content` field. The information in this field, and also the way it is encoded, is completely application specific. Nevertheless, FIPA provides a specification for the semantically meaningful communication language FIPA Semantic Language (SL) [131], which may be used. The encoding used for the information within the `content` field is defined by the `language` field.

### FIPA ontologies

FIPA does not provide its ontologies in a standard W3C format, e.g., RDF or OWL, but rather specifies them using several tables per ontology. Ontologies are primarily used to enable agents to exchange semantically meaningful messages. FIPA defines three main ontologies: `fipa-agent-management`, `fipa-device`, and `fipa-qos`.

The basic structure of FIPA agents is specified by the `fipa-agent-management` ontology, which is provided by the Agent Management Specification [125]. It specifies the classes required to describe the agent itself, including its AID, addresses, supported protocols, supported ACLs, as well as the classes required to describe the agent's services, such as the name, type, and the ownership. This information is of particular interest when it comes to registering and searching agents and services at the DF.

The `fipa-device` ontology is defined by the FIPA Device Ontology Specification [132]. It provides classes to describe many different hardware and software aspects of computational devices, such as their available memory, Central Processing Unit (CPU), user interfaces, Operating System (OS), and AP. It can be used by agents when communicating about these aspects, e.g., because an agent wants to evaluate if a computational device is capable of performing a specific task.

Finally, the FIPA Quality of Service Specification [133] defines the `fipa-qos` ontology. It allows to model additional aspects of the transport protocols and communication channels, e.g., the throughput, delay, and frame error rate. The `fipa-qos` can be used by agents when communicating about Quality of Service (QoS).

### FIPA-compliant APs

An AP has to realize the four architectural elements Message Transport, Agent Directory, Service Directory, and ACL to be FIPA-compliant. The technologies used to implement these architectural elements are not prescribed. Developers can build their own AP based on existing technologies such as HTTP for message transport, Lightweight Directory Access Protocol (LDAP) for agent directory and service directory functionality, and possibly a custom ACL. However, in most cases it is beneficial to build upon existing APs such as JADE [116], Mobile-C [134], Aglets [135], Agent Tcl [136], Concordia [137], D'Agents [138], ZEUS [139], or Mole [140]. They reduce the implementation effort by providing all the general services like message transport, agent registration, discovery,

and agent life-cycle management. Therefore, developers can focus on implementing the agents themselves and their application-specific functionality. JADE and Mobile-C are briefly introduced in the following.

JADE [116] is an AP implemented in the Java programming language. It is the most commonly used AP, in particular in the field of SGs. JADE implements the AMS, DF, and MTS components of the FIPA Agent Management Reference Model (cf. Section 2.3.5). In addition, it implements the Agent Communication Channel (ACC) as an additional component designed to handle message exchange with agents running on another AP. The AMS, DF, and ACC are themselves implemented as agents. The following list provides a subset of additional features that are implemented by JADE and are often very useful to the agent programmer:

- Graphical User Interface (GUI) to manage the AP and the corresponding agents
- AP can be split among multiple hosts to increase performance
- Possibility to run several DFs on the same host to enable multi-domain applications
- MTS enables lightweight message exchange for agents running on the same host
- FIPA agent interaction protocol implementations
- Message encoding and decoding
- Agent mobility [141]

Mobile-C [134, 142] is another FIPA-compliant AP implementation. It is implemented in C programming language, specifically targeting resource constrained, embedded devices. Mobile-C agent code can either be compiled and executed directly on the host, or interpreted by integrating the C/C++ interpreter *Ch* [143]. The ability to transfer and interpret code enables agent mobility and easy update of agents in Mobile-C. The Mobile-C AP is built from a number of separate Application Programming Interfaces (APIs), including the AMS, DF, and ACC APIs. Furthermore, it provides an optional security API, the Agent Security Manager (ASM).

### 2.4 Ontology design

Earlier ontology design methodologies started by following rather simple, step-wise approaches, as in [144, 145, 21]. Later on, as ontologies became more extensive and more complex, so did the corresponding ontology design methodologies by adopting ideas from software engineering and other disciplines. Ontology design methodologies became incremental, i.e., adding only small portions of information at a time, and iterative, i.e., performing the different phases of ontology design (e.g., analysis, modeling, and test) for each increment, as in [146, 147]. An overview of existing scientific work on ontology design can be found in [148]. Furthermore, ontology design can be supported by MDE techniques. An overview of existing work on this topic, as well as a novel approach

that identifies and avoids inconsistencies during the design phase, is provided in [149]. The following sections briefly present “Ontology Development 101: A Guide to Creating Your First Ontology” [21] as an example for step-wise approaches, and UP for ONtology (UPON) – “A software engineering approach to ontology building” [147] as an example for incremental and iterative approaches.

### 2.4.1 Ontology development 101

In “Ontology Development 101: A Guide to Creating Your First Ontology” [21], Noy, McGuinness, et al. provide some motivation for the use of ontologies, outline the differences to classical OOD, give a short introduction to most crucial building blocks of ontologies, and present Protégé [150], the most commonly used ontology design tool, to create their examples. Additionally, some general advice on how class hierarchies are built efficiently, naming conventions, and other aspects of ontology design is given. Thus, they provide an excellent starting point for ontology development in general. However, the focus here shall be on their ontology design methodology. Therefore, the seven (almost strictly consecutive) steps their methodology consists of are briefly discussed in the following. The steps of the methodology are illustrated in Figure 2.13.

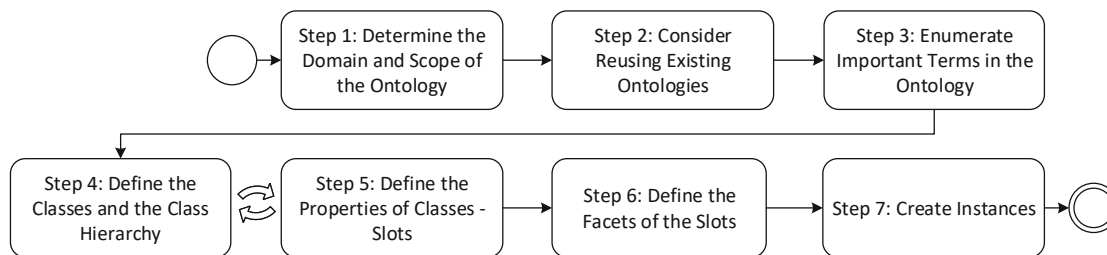


Figure 2.13: Ontology development 101 design methodology

#### Step 1: Determine the domain and scope of the ontology

The design process starts with answering several general questions used to define the domain and scope of the ontology to be developed, as a single ontology cannot cover a domain entirely and from all perspectives. Thus, this first step gives a basic orientation about which information should be modeled, and the level of detail required. This view of the domain might evolve during the subsequent steps of the methodology. However, it should not change drastically. In addition to these general questions, formulating competency questions [151] has become a proven means to ensure the quality of the designed ontology and is also suggested by the Ontology development 101 methodology. Thereby, competency questions are simple and human-readable questions that should be answerable by making use of the ontology after the design process. The list of competency questions is intended to be exemplary rather than exhaustive.

### **Step 2: Consider reusing existing ontologies**

As ontology design becomes increasingly popular, the probability increases that parts of the desired ontology are already available. Thus, developers should consider incorporating this existing knowledge in their ontology, which improves compatibility and reduces the amount of work involved with building ontologies. The authors of the Ontology development 101 methodology limit the reuse of existing knowledge to existing ontologies, which can be found, for example, in online libraries such as Ontolingua [152], and DARPA Agent Markup Language (DAML) ontology library [153].

### **Step 3: Enumerate important terms in the ontology**

At this step of the methodology, a comprehensive list of terms that should appear in the final ontology is created. This list is based on the general questions and the competency questions that have been defined during Step 1. There is neither a hierarchy defined on these terms, nor are there any relations specified.

### **Step 4: Define the classes and the class hierarchy**

Defining the classes and the class hierarchy is closely intertwined with the subsequent Step 5, defining the properties of the classes. Thus, Step 4 and Step 5 are typically not performed in strict order. Instead, developers should jump back and forth, first defining several classes and their hierarchical order, then specifying relations between them in the form of properties.

The order in which classes and their hierarchy should be defined can follow one of three approaches: top-down, bottom-up, or a combination of these two [154]. The top-down approach follows the idea of specialization. It starts with specifying the most general concepts first and adds related, more specific ones along the way. Contrary, the bottom-up approach follows the idea of generalization. It adds specific concepts first, grouping them to more general concepts during the process. The combination of the top-down and the bottom-up approach defines the most important, salient concepts first and applies specialization and generalization on them simultaneously. None of the three approaches can be considered best in any perspective, but the choice is subject to the personal preferences of developers and their view of the domain.

Independent of the chosen approach, Step 4 starts with adding terms identified in the previous step as classes. These terms should identify types of objects (e.g., Transformer Agent) rather than properties of these objects (e.g., address, name). Superclass/subclass relationships are added between classes, just like inheritance in OOP: if class B (e.g., Transformer Agent) is a subclass of class A (e.g., Agent), then every individual of class B is also an individual of class A. Furthermore, B inherits all properties of A. Additionally, some practical advice on how to organize classes in a class hierarchy is provided in [21].

### Step 5: Define the properties of classes – slots

During this step, terms that represent properties are added to the ontology by assigning them to classes. At this point in the process, no distinction is made between object properties and data properties. The term *slot/slots* is used for both. As properties are inherited according to the class hierarchy, they should be added to the most general applicable class. For example, the property “name” is applicable for any type of agent. Therefore, it should be added to the class Agent, rather than to all its subclasses.

### Step 6: Define the facets of the slots

Facets specify additional information about slots like their type, allowed values, and cardinality. The cardinality defines how many instances a slot can have, e.g., an agent must have precisely one name and one to infinitely many addresses. During this step, it is determined whether a slot is a data property or an object property according to its data type: data properties relate classes (and their instantiated individuals) to data values while object properties relate classes among each other. The domain and range are used to add additional information about slots. Thereby, the domain specifies to which classes a property can be attached to. The range specifies allowed data types and classes the property can point to. The domain and range should contain all relevant classes but not be overly generous.

### Step 7: Create instances

As a last step of the Ontology development 101 methodology, individuals are added to the ontology as instances of classes. The slots of each individual are filled with concrete values, e.g., the actual name of an agent.

## 2.4.2 UPON

The UPON ontology design methodology [147] is heavily inspired by the Unified Software Development Process / Unified Process (UP) software engineering approach [155]. UPON is intended to be used for the development of large-scale ontologies and, therefore, specifies an iterative and incremental development process. Furthermore, UPON is use-case-driven, i.e., it produces an ontology for a specific use case rather than a general ontology for a complete domain. Like UP, UPON uses the notion of cycles, phases, iterations, and workflows as illustrated in Figure 2.14. Each cycle adds additional details and produces a new version of the ontology. There are four phases executed during each cycle: 1. *inception* (capturing requirements) 2. *elaboration* (identifying and sketching fundamental concepts), 3. *construction* (detailed design and implementation), and 4. *transition* (test of the ontology and collection of information for the next cycle). Each phase may involve multiple iterations over five workflows. Figure 2.14 only exemplifies the workflows for the construction phase. However, the same five workflows are executed during each of the four phases.

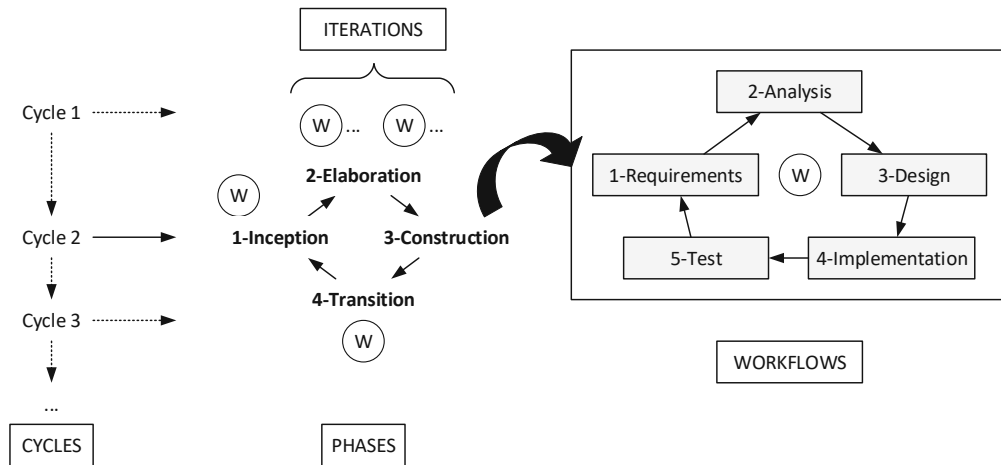


Figure 2.14: UPON ontology design methodology

As the interplay of cycles, phases, iterations, and workflows is the central aspect of the UPON methodology, it shall be discussed in more detail based on Figure 2.15. Each of the five workflows is executed in every iteration, and, therefore, also in each of the four phases. However, their contribution to each phase differs depending on how far the design of the ontology has already evolved. For example, the test workflow does not contribute during the inception phase, as there is nothing to be tested at this point. The contribution of each workflow across the various phases is sketched in Figure 2.15.

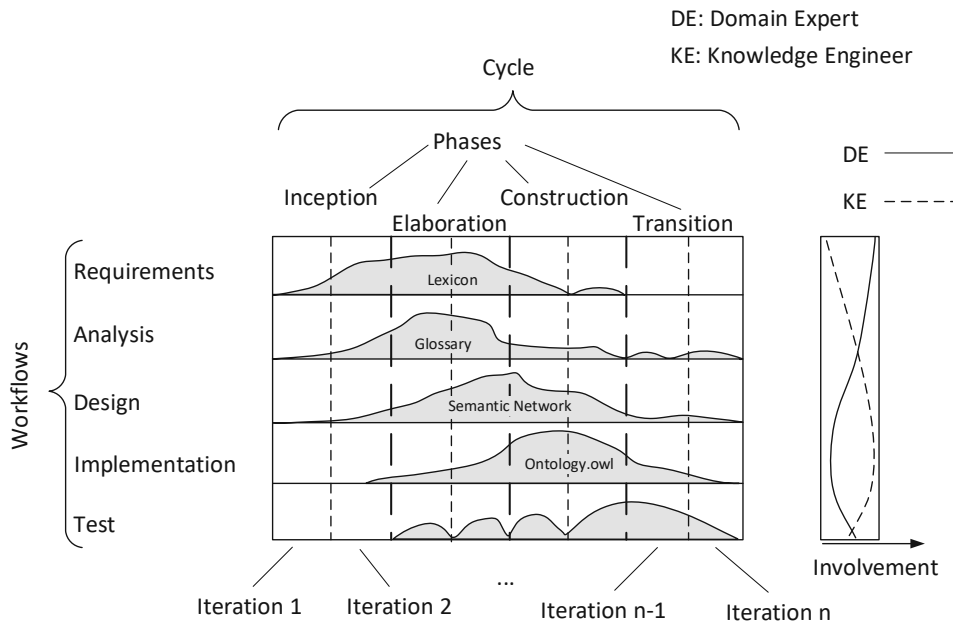


Figure 2.15: Interplay of UPON cycles, phases, iterations, and workflows

Each of the UPON phases produces a different outcome: lexicon, glossary, semantic network, and finally, the ontology. However, for the sake of not over-complicating things, it may be helpful to think about the lexicon, glossary, and semantic network as unfinished ontologies. Furthermore, two main characters are expected to participate in the process: a Domain Expert (DE) and a Knowledge Engineer (KE). Their involvement in each of the workflows differs, as illustrated in the figure. For example, specifying the system's requirements is almost exclusively done by the DE, while the KE gains importance later in the process when it comes to modeling the concepts. While the cycles, phases, and iterations provide the structure of the UPON methodology, the actual work is done by executing the workflows. Therefore, they are briefly discussed in the following. Each of the workflows produces/extends several outcomes like lists of terms and diagrams. These outcomes are *emphasized* and numbered (i, ii, iii, ...).

### Workflow 1: Requirements

The goal of the requirements workflow is to capture the needs of future users of the ontology. Therefore, (i) the domain and scope of the ontology are defined by identifying the *most important concepts and properties*. (ii) The *business purpose/motivating scenario* of the ontology is defined based on the users' objectives. (iii) Motivating scenarios are outlined in more detail using *storyboards*, i.e., short written or drawn descriptions of the steps comprising the scenario. (iv) A preliminary list of the relevant terms, an *application lexicon* in UPON terminology, is created from the information given in the storyboards. (v) *Competency questions* are formulated (cf. Section 2.4.1). (vi) Simple *use case diagrams* are created from the competency questions and prioritized.

### Workflow 2: Analysis

During the analysis workflow, the requirements identified in the previous workflow are refined. The application lexicon is enriched step by step until a reference glossary, i.e., a list of terms and their definitions in the domain context, is derived. Therefore, (i) a *domain lexicon*, i.e., a collection of terms based on information from existing material like standards, manuals, and reports, is built. (ii) The application lexicon and the domain lexicon are used as a basis to create the *reference lexicon*. Thereby, all terms included in both of these lexica are added to the reference lexicon. Whether or not terms that are present only in either the application lexicon or the domain lexicon, but not in both, shall be added to the reference lexicon has to be decided individually. (iii) The information given by the use case diagram is detailed based on *class diagrams* and *activity diagrams* to create models of the application scenarios that can be used for validation of the ontology later on. (iv) The *reference glossary* is generated by extending the reference lexicon with explanations of the terms.

### Workflow 3: Design

During the design workflow, “ontological structure” is given to elements of the reference glossary using the Object, Process, Actor modelling Language (OPAL) [156]. OPAL is an ontology modeling framework providing a limited number of design patterns to simplify the process of building ontologies. (i) *Concepts from the reference glossary are assigned to one of five categories*: business actor, business object, business process, message, or attribute. Additional information about these categories can be found in [147] and [156]. (ii) *Relationships between concepts are modeled*. As a first step, a class hierarchy is established using the top-down, bottom-up, or a combined approach. Furthermore, aggregation relationships and associations are added between concepts. The design workflow results in several class diagrams that represent the ontology.

### Workflow 4: Implementation

During the implementation workflow, the (i) *diagrams derived in the design workflow are encoded in a formal language*, such as RDF or OWL. Therefore, this task is primarily performed by the KE. Additional ontology design constructs (e.g., range, domain, cardinality, and disjunction) are also added during the implementation workflow.

### Workflow 5: Test

The test workflow ensures the quality of the derived ontology in terms of four different characteristics. (i) *Syntactic quality* ensures that the ontology file is syntactically correct and is typically done by automated tools, e.g., by verifying the ontology file against the corresponding XML Schema Definition (XSD). (ii) *Semantic quality* mainly focuses on the absence of logical contradictions within the ontology like an individual having type  $A$  and  $B$  simultaneously, where  $A$  and  $B$  are disjoint classes. (iii) *Pragmatic quality* refers to “the ontology context and usefulness for users” [147]. (iv) *Social quality* provides a measure of how well an ontology is accepted, in particular by determining how many other ontologies link to it and how developers use it within and outside of the community. This workflow concludes the execution of one iteration of the five workflows. The next iteration, phase, or cycle may follow if the ontology needs further refinement.

## 2.5 Related scientific work

The MAS used as an example in this thesis shall cover functional as well as non-functional (in particular dependability) aspects of actors in the SG field. Therefore, this section addresses related scientific work regarding MASs for SG applications first. The corresponding MASs are thereby grouped by their organizational structure, i.e., centralized, decentralized, and distributed. After that, existing scientific work about the way MASs address dependability is presented based on the various dependability attributes.



### 2.5.1 Functional aspects: MASs for SG applications

According to a survey paper of Kantamneni et al. [157], MASs in SG applications can be categorized into centralized, distributed, and two/three-level hierarchical approaches. Two-level and three-level structures are predominant [158], and, according to the definitions provided in Section 1.3, fall into the class of decentralized communication systems. The individual approaches found in the scientific literature cover many different applications. In the following, work available on MASs in SGs is discussed with a focus on systems designed to minimize losses in the distribution network.

#### Centralized MASs

*Centralized approaches* are heavily inspired by traditional control structures: agents that are deployed in the field (field agents) report directly to and get instructions from a central control agent only. There is no direct communication between field agents. However, they are intelligent and can perform actions autonomously, as far as these actions do not require the agent to coordinate with other agents. An approach of this type is presented in [159]. It implements an energy management algorithm that optimizes the operation of water pumps based on the available energy in batteries across multiple houses. Thereby, agents are categorized into *Reactive Agents* and *Cognitive Agents*, whereby Reactive Agents are simpler and can react to changes in the environment more quickly while Cognitive Agents are more sophisticated and used for applications that require “increased intelligence” [159]. Intelligent Load Controllers (ILCs) are placed at the power line outside of each building. Each ILC is equipped with an intelligent agent and hosts a web server for remote access. Major, non-local functionalities of the control system, e.g., coordinating battery charging among multiple houses, are implemented in the Cognitive Agent.

#### Two- and three-level decentralized MASs

In [160], Merdan et al. present a MAS for fault detection and correction in distribution networks. The architecture comprises six types of agents: *System Agent*, *Automation Agent*, *Data Agent*, *Substation Agent*, *Switch Agent*, and *Generation/Load Agent*. Conceptually, the approach is a *two-level hierarchical approach*: the *Management Layer* (containing high-level agents) is responsible for network stability and functionality while the *Execution Layer* (containing low-level agents) provides access to the system’s equipment. If an agent detects a fault, it informs the System Agent, which then calculates options for network reconfiguration. The main objective is to optimize (maximize) the number of supplied loads. In the case of multiple possible configurations, the one that causes minimum losses is chosen. Based on this result, the System Agent requests switching actions from the Switch Agents. It furthermore informs other agents of the system to update their world models. As the paper focuses on the MAS architecture, it remains unclear whether the System Agent has to know the complete structure of the distribution network to perform its algorithm and which optimization algorithm is used. *Three-level hierarchical*

*approaches* are conceptually similar to two-level hierarchical approaches but add mid-level agents for tasks such as fault location and grid-connected/islanded mode switching [157].

### Distributed MASs

As one of few examples for a *distributed control approach*, Jiang presents an “Agent-Based Control Framework for Distributed Energy Resources Microgrids” [161]. The test system is made up of three agent types: *Energy source agent*, *Energy storage agent* and *Load agent*. Furthermore, it comprises a *Directory Service*, which enables agent registration and discovery. All agents are equal and not organized in any type of hierarchy. They rather coordinate their actions via a simple agent interaction protocol. The overall system goal is voltage regulation. Thereby, safety considerations regarding a battery for energy storage (which is controlled by the Energy storage agent) are of particular importance. Simulation results presented in the paper indicate that the approach is well suited for the control of distributed energy resources in microgrids. The effects of the communication network are neglected in this early stage of development.

#### 2.5.2 Non-functional aspects: Dependability in MASs

In the following discussion, existing scientific work in the field of dependable MASs is grouped by the dependability attribute addressed. Even though the nomenclature varies, e.g., the terms “dependable” and “robust” are often used interchangeably, most papers can very well be related to one of the dependability attributes defined in Section 1.2.

#### Reliability in MASs

Measures to increase the *reliability* of MASs can generally be applied before as well as during runtime. Techniques applied before runtime include static verification (e.g., model checking) and other software testing approaches. An overview of “Specification and Verification of Multi-agent Systems” is given by Dastani, Hindriks, and Meyer in [162]. In [163], Lomuscio, Qu, and Raimondi present their Model Checker for Multi-Agent Systems (MCMAS). The Interpreted Systems Programming Language (IPSL) is thereby used to describe the environment and the individual agents, regarding their initial states, variables, agent interaction protocols, or actions. MCMAS provides a GUI that supports the user in checking this description of the MAS against Alternating-time Temporal Logic (ATL) [164], Computation Tree Logic (CTL) [165], and other types of specifications. Similar model-checking approaches for MASs are presented in [166, 167, 168, 169, 170].

A common measure to increase the *reliability* of MASs during runtime is *self-checking*. This is typically achieved on the MAS level, i.e., agents check one-another to verify that the counterpart is still operating reliably and its behavior can be trusted. A general overview of the ideas and existing work on this and related topics (e.g., self-optimization and self-healing) is presented in [16]. In [171, 172], Costantini suggests to use a simple Interval Temporal Logic (ITL) [173] to describe constraints on the behavior of an agent,

including the notion of time. ITL allows to define that some action has to be performed before or after a specified time. If an agent violates such a constraint, it can be concluded that the agent is no longer functioning properly. Additionally, suitable reactions can be defined for violations of each of the constraints, which provides the basis for self-healing capabilities. In [174], Haegg introduces *sentinels* as a special agent type that is responsible for monitoring agent communication within the MAS, building models of other agents, and ultimately performing specific actions (e.g., informing the operator) if any abnormal behavior is detected. The MAS is thereby implemented first, and sentinels are added as an additional control system later.

In [175], Matsumoto, Maruo, and Tanimoto argue that large MASs suffer from additional communication overhead introduced by the self-checking mechanisms (e.g., [176]) and a scalable approach to this problem is required. They suggest to address this problem by introducing *middle agents*, each responsible for carrying communication between a subset of *client agents* – so-called *local groups*. Local groups contain three to four client agents. Client agents only test one another within local groups, and middle agents are aware of the test results. Middle agents exchange information about possibly faulty client agents. Furthermore, client agents are switched between local groups from time to time to account for the fact that more than one client agent of a local group may be faulty.

Lockemann and Nimis present a different approach. They address “agent dependability as an architectural issue” [177]. Thereby they refer to a software agent as dependable “if it maintains its service according to specifications even if disturbances occur within the agent” [178], which, according to the definitions used in this thesis (cf. Section 1.2), actually better fits the term reliability than dependability. Lockemann and Nimis propose to build upon a five-layer reference architecture for agents and to define a dependability model for each of the layers. Each dependability model precisely states the services of the layer, which errors may occur, if they are handled and how, or if they are propagated to the next higher layer.

### Safety in MASs

The amount of scientific work addressing *safety* in MASs is plentiful, though not as vast as on reliability. According to Musliner, Durfee, and Shin [179], a key challenge in designing intelligent systems for safety-critical applications is the trade-off between flexibility and predictability, in particular regarding real-time aspects of safety-critical applications. In their Cooperative Intelligent Real-Time Control Architecture (CIRCA), they separate the *task level goals*, which require intelligence and are, therefore, hardly predictable regarding real-time guarantees, from the *control level goals*, which cover safety-critical operation, among other functionality. Communication between both parts of the architecture is enabled by a special asynchronous interface, which ensures that no timing-guarantees of the control level are violated. The drawback of their approach is that it provides timing guarantees only within a single component, but not among components, which would be preferable for MASs.

Giese et al. [180] address this issue by separating the MAS design into a *macro architecture* and a *micro architecture* part. The macro architecture defines the overall system structure and behavior. The system structure is thereby defined using class diagrams, instance diagrams, and social diagrams. The behavioral description is based on the definition of possible behavioral patterns using an extended real-time variant of state machines. Behavioral patterns include information about timing, in particular upper limits for how long each component can reside in a specific state before a timeout is triggered and appropriate actions are performed. The micro architecture refines the internal behavior of agents. Thereby, all applied refinements have to conform to the time-limits prescribed by the macro architecture. A similar approach to guaranteeing real-time capabilities of MASs is presented by Hayes-Roth in [181].

Additionally to the real-time aspects of safety-critical applications, few scientific publications address safety explicitly during the MAS analysis and design phases. For example in [182], Jamont and Ocello extend their DIAMOND [102] methodology by an additional activity that guides the developer in identifying safety-related (and other) non-functional requirements. Agent design accounts for these requirements by defining corresponding states and actions to be taken if one of these requirements is violated. Jamont and Ocello stress out that, although in a MAS the individual agents typically do not have access to all information, for safety-critical incidents, it is essential that all agents “react collectively”. This behavior is achieved by introducing a state of *alertness* that is shared among all agents.

### Security and privacy in MASs

*Security* and *privacy* are very much related and build upon the same technical means. Surveys about existing work on security in MASs can be found in [183, 184, 185, 186]. In general, MASs suffer from the same security issues as any other computing system, including distributed systems, but also impose several new security issues. This is particularly the case if agents act on behalf of some owner, are executed on a potentially compromised AP, are mobile (i.e., they can migrate between multiple APs), or if the number and types of agents of the MAS are unknown at design time. All of these scenarios raise trust issues. Several possible solutions to these and additional considerations are discussed in [187, 183, 184, 186].

To ensure that an agent is indeed representing its owner and the owner is accountable for the agent’s actions, the agent should possess some secret of its owner (typically a Personal Identification Number (PIN) or private key). Furthermore, the owner’s identity has to be known to all participants of the MAS interacting with the specific agent [187].

If agents, including mobile agents, are executed on an AP, sandboxing techniques may be applied to ensure that agents do not misuse the AP’s resources. Thereby, agents may only use a predefined API to interact with the AP, thus limiting its potential for misuse but also its functionality [188]. This approach was implemented for JADE in [189].

In dynamic systems, where new agents and even new agent types connect and disconnect during runtime, agents may keep track of previous interactions and the degree of trust they place in each of their interaction partners to decide if they continue cooperating [190]. Prerequisites for establishing trust among multiple agents are authentication, authorization, and access control. These topics have been studied extensively [191, 192, 193] and can typically be handled adequately by applying existing solutions. Likewise, communication security in MASs is essentially achieved by applying existing solutions, as done in [194].

Paruchuri et al. [195] address another aspect of MAS security, namely that adversaries often try to analyze the behavior and communication of a system they are going to attack. Thus, they suggest *policy randomization*, i.e., the individual agents should not show deterministic behavior but rather randomize their actions.

Few scientific publications exist on integrating security into the MAS design process from the very beginning rather than adding it to an existing MAS. Examples are [196, 195]. They both extend existing MAS design methodologies: the *Tropos* [197] MAS design methodology and the FAME Agent-oriented Modeling Language (FAML)<sup>2</sup>, respectively.

### Maintainability in MASs

*Maintainability* is hardly addressed explicitly in the scientific literature about MASs. However, some aspects of MASs discussed so far also contribute to their maintainability. Based on the definition of agents as self-contained entities, their functionality and interfaces are precisely defined. Therefore, it is fairly easy to replace or update individual agents. If supported, which is the case for both APs presented in Section 2.3.5, agent mobility can be used for remote updates.

Furthermore, applying MAS design methodologies (cf. Section 2.3) produces, as a side-effect, a precise documentation of the MAS, including the agent types, their instantiations, and locations. This documentation as well as additional services provided by the AP (e.g., the agent directory service) can serve as a valuable source of information when it comes to repairing or modifying agents.

### Scalability in MASs

*Scalability* in the context of MASs is typically addressed regarding the number of agents in a MAS rather than regarding the complexity of individual agents. Thereby, scalability means that adding agents to the MAS should have no noticeable effect on the performance or administrative complexity. In [198], Wijngaards, Van Steen, and Brazier provide an overview of MAS scalability. They identify three basic techniques to improve scalability, which can also be applied to MASs: (1) Communication latencies should be hidden, which

<sup>2</sup>Framework for Agent-oriented Method Engineering (FAME) is the acronym of the project under which FAML has been developed.

is generally known as *asynchronous communication scheme*, i.e., agents should be able to continue their operation while they are waiting for a response to a previous request. (2) Processing complexity and large data sets should be *distributed* across multiple servers or, in the context of MASs, across multiple agents. (3) Likewise, data sets, services, and possibly agents as a whole, which are heavily used, should be *replicated*. Furthermore, the basic MAS services like the naming and the directory service should be scalable, which can easily be achieved by applying existing technologies that already take scalability into account, like the Domain Name System (DNS), and LDAP.

Turner and Jennings [199] use an exemplary trading scenario, in which customer agents buy products that are sold by supplier agents. They use this scenario to address scalability by analyzing three different organizational forms and agent interactions. Two of them can be classified as distributed approaches and the third as two-level hierarchical approach, in which an intermediary agent keeps everyone updated about current product availability and prices. Turner and Jennings conclude that hierarchical approaches reduce the overall number of machine instructions because the intermediate agent can perform a task, which otherwise would have to be performed by all customer agents or all supplier agents individually. However, the latency of information sharing increases compared to the distributed approaches.

In [200], Rana and Stout examine metrics for measuring scalability in MASs. They propose a “Complexity Metric” to be able to reason about the performance of MASs with varying sizes, coordination mechanisms, and APs. The complexity metric is constructed from several performance values. These performance values may be rated based on domain-dependent constraints and summed up. The performance values represent processing times required for starting an agent, activating and deactivating an agent, message reception from/delivery to an agent on a specific AP, and similar parameters.

### Summary

Table 2.6 summarized the scientific publications categorized by the dependability attributes they address. It shows that, even though dependability is not covered as a whole by any referenced paper, various ideas and techniques exist to address each attribute individually.

The cited literature covers many different solutions for increasing dependability in MASs. However, depending on the particular system requirements, not all of these solutions are actually relevant for each MAS. Thus, a SELC addressing dependability in MASs needs to identify requirements early in the process. Furthermore, it needs to be flexible enough to incorporate the necessary measures to achieve these requirements.

Table 2.6: Dependability attributes and relevant literature in the context of MASs

<i>Dependability attribute</i>	<i>Relevant literature</i>
Reliability	Survey: [162] Model-checking: [169], [170], [163], [166], [168], [167] Self-checking: [171], [172], [174], [176], [175], [16] Architectural-based: [177], [178]
Safety	Separation of criticality: [180], [181], [179] Methodology-based: [182]
Security & Privacy	Survey: [183], [185], [186], [184] Trust: [183], [184], [187] Sandboxing: [188], [189] Accountability: [194], [191], [190], [192], [193] Randomization of behavior: [195] Methodology-based: [196], [195]
Maintainability	MAS platforms: [140], [116], [134], [142], [124], [136], [135], [137], [139] Documentation: [109], [106], [103]
Scalability	Survey: [198] MAS structure: [199] Expressing scalability: [200]



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



# Systems engineering life cycle definition

The information gathered during the previous chapter, in particular the presented SG, MAS, and ontology design methodologies, shall now be incorporated into a comprehensive SELC. For this purpose, a suitable SELC is selected from existing scientific literature and adapted to emphasize the importance of earlier phases. Next, activities that need to be conducted in each phase are identified, and a corresponding process specifically for MASs incorporating dependability is defined. Finally, it is discussed to which extent existing methodologies can support the individual activities, how they are interlinked, and where gaps may exist in the process.

## 3.1 SELC phases

In most scientific literature, the terms system(s) engineering and system(s) development are used interchangeably. They describe a “robust approach to the design, creation, and operation of systems” [201]. This definition already indicates that systems engineering accompanies the system that shall be developed across all the system’s life cycle phases. Models that describe these phases are termed SELCs and Systems Development Life Cycles (SDLCs), respectively. SELCs are iterative processes, which enable improving the system by reverting to earlier phases of the SELC after the system has already been developed and deployed. This iterative nature is reflected by the life *cycle* aspect of SELCs and is the case for all SELCs. However, it is not always explicitly stated in favor of simpler models.

Moreover, in many cases the SELC may be processed incrementally, i.e., a base version of the system with reduced functionality may be created to gain experience and then improved upon following an incremental approach. For these reasons, systems engineering

is not a linear process, and the phases of SELCs do not need to be conducted in strict order. Traversing back and forth along the phases of the SELC in an agile manner is expected.

The phases and activities that need to be conducted vary greatly among different SELCs depending on the type of system, the requirements, the domain, and other factors. An extensive overview and comparison of various SELCs is provided in [202]. Andr en [24] identified the SELC depicted in Figure 1.7 as a suitable model for the SG domain. It is also used as a starting point for the SELC specified in this thesis. However, the planning and the analysis phase were added to emphasize the work required in preparation for the design phase. Furthermore, deployment and operation are not fully covered here, as the system resulting from this thesis is only evaluated using simulation but not deployed or operated in the field. The final SELC is depicted in Figure 3.1. It consists of five main phases: planning, analysis, design, implementation, and evaluation. Deployment & operation is added as an additional sixth phase, but, as mentioned, not covered in detail. The phases and the related activities within each phase are discussed in more detail in the next section.



Figure 3.1: Extended SELC, adapted from Figure 1.7

## 3.2 Definition of SELC activities

Figure 3.2 illustrates the activities that need to be conducted in each of the five main SELC phases. As for the SELC itself, there is no need to follow the process in strict order, but an iterative and incremental approach may be applied. The definition of these activities is driven by the research objectives defined in Section 1.5, existing SELCs, and state of the art methodologies. In the following discussion, the activities of the various SELC phases are *emphasized*.

During the planning phase of the SELC, a detailed *use case description* is formulated to define the system’s expected functionality. The use case description also includes a description of the components and actors involved and describes the information that needs to be exchanged. The scope of the problem and the objectives that should be achieved by the system are defined. KPIs are identified, which can be used in the evaluation phase to verify that the objectives have been met. Additionally, identifying and specifying the system requirements during the *requirements description* activity is of particular importance as the subsequent phases of the SELC heavily rely on the appropriateness of this information. The planning phase should focus on a detailed description of the problem, rather than on possible solutions (technical or otherwise). These are developed in later phases of the process.

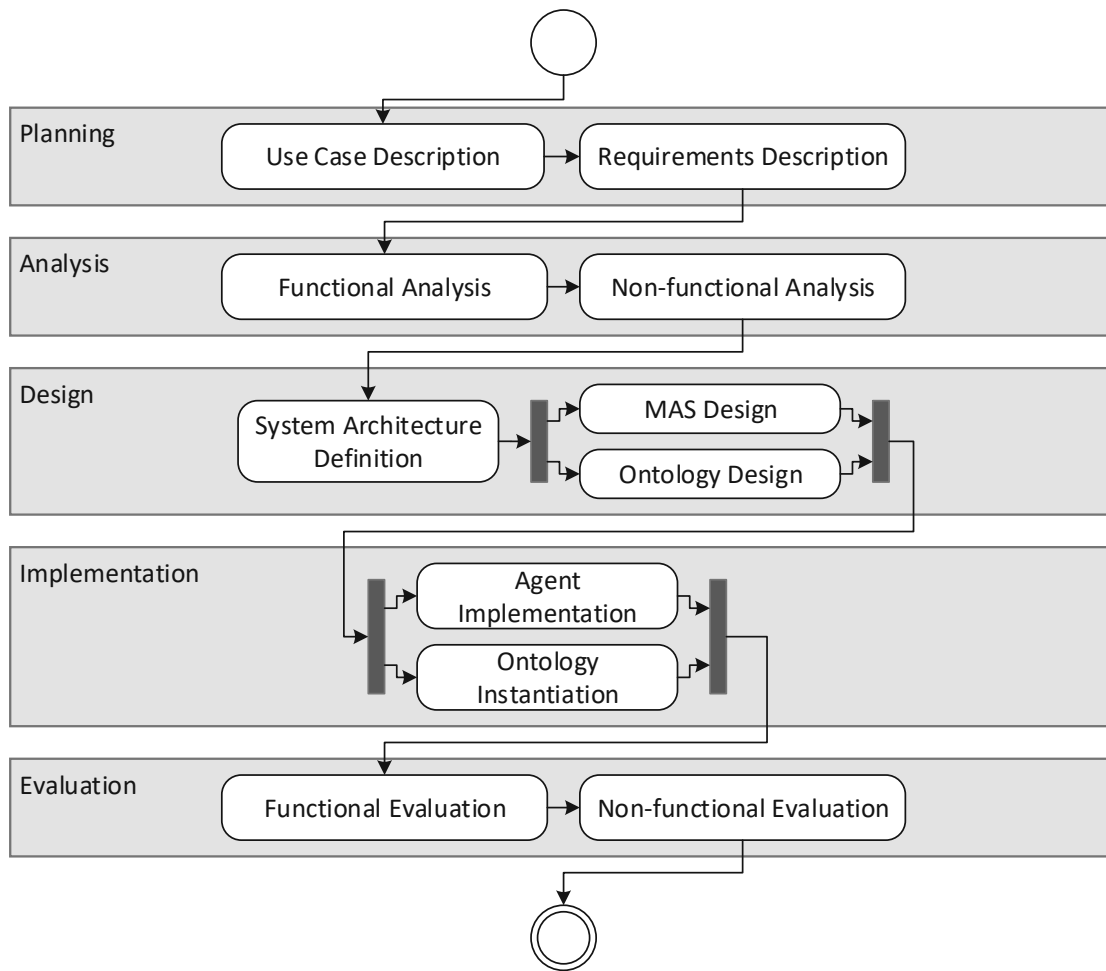


Figure 3.2: SELC phases and activities

During the analysis phase, preliminary work to support the subsequent design phase is conducted. Requirements are analyzed in more detail than in the previous phase to gain additional insights into the overall system. The *functional analysis* investigates possible solutions (mostly algorithms) that fulfill the functional requirements. The *non-functional analysis* defines the importance of each non-functional requirement (e.g., availability, reliability, and maintainability) for the use case. Furthermore, it identifies means allowing to define and reason about non-functional attributes.

During the *system architecture definition* activity, the type of MAS (centralized, decentralized, or distributed), the required agent types, the main services, and their internal structure, i.e., their basic software and hardware components, are specified. Furthermore, the general idea of how agents interact to solve the given problem is discussed. Based on this knowledge, the MAS and the corresponding ontology are designed. As the support

for ontologies is an important aspect but typically out of scope in the MAS design methodology [118], a hybrid design approach consisting of the *MAS design* activity and the *ontology design* activity is followed.

The agents are then implemented following a classical software design process during the *agent implementation* phase. An existing MAS framework, or parts of such a framework, may be used to reduce the implementation effort. Furthermore, each agent is equipped with its configuration and an initial knowledge base during the *ontology instantiation* activity.

Finally, a *functional evaluation* and a *non-functional evaluation* are conducted using simulation. The functional evaluation ensures that each agent operates as specified during the design process and that the overall MAS fulfills its intended task. During the non-functional evaluation activity, the effect of agents considering non-functional requirements in their decision-making process on the system's overall performance is evaluated.

### 3.3 Methodologies supporting the SELC activities

As specified by research objective 1, whenever possible existing methodologies shall be applied to support the individual phases and activities of the SELC. Therefore, Table 3.1 provides an overview of the methodologies presented in Chapter 2 and the SELC phases they can be assigned to. As already noted, the exact meaning of the various phases varies across different SELC definitions. The association shown in Table 3.1 refers to the SELC and the activities as defined in the previous section. Furthermore, the methodologies listed do not cover all relevant aspects of each SELC phase and have to be substituted with additional tools and methodologies. These gaps are also briefly addressed in the following and covered more extensively in the remaining chapters.

Table 3.1: Methodologies and the SELC phases they support

<i>Methodology</i>	<i>Planning</i>	<i>Analysis</i>	<i>Design</i>	<i>Impl.</i>	<i>Eval.</i>
IEC 62559	+				
NISTIR 7628		+			
Gaia			+		
MaSE			+	+	
PASSI			+	+	+
Ontology 101			+	+	
UPON			+	+	+

Particularly for SG applications, the IEC 62559 standard provides a comprehensive methodology for the planning phase through use case descriptions and a requirements description. As mentioned, the functional analysis requires a detailed discussion of

possible algorithmic solutions to the specific problem that shall be solved. The non-functional analysis is partly covered by the NISTIR 7628 standard, which provides a detailed security analysis methodology. However, additional discussion is required for the remaining non-functional dependability requirements. Furthermore, possible metrics for dependability attributes are identified to support ontology design. Functional aspects of the design, implementation and partly the evaluation phase are well-covered by several MAS design methodologies, e.g., PASSI, which has been identified as a suitable MAS design methodology in Section 2.3.4. However, non-functional aspects are mostly neglected in MAS design. Furthermore, existing ontology design methodologies are only partly suitable to be applied in MASs. For these reasons, the non-functional aspects of the design and implementation phases require a dedicated ontology design methodology that allows creating ontologies that can easily be reused across multiple agents and applications. This methodology is introduced and applied during the design phase. The resulting ontology is instantiated during the implementation phase. Finally, the functional and non-functional evaluation are conducted using SG simulation. The following Chapters 4 to 8 cover the individual activities of the SELC, including a more detailed discussion of appropriate methodologies and tools.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

# Planning

The activities that need to be conducted during the planning phase of the SELC are depicted in Figure 4.1. The figure also includes appropriate tools/methodologies supporting each activity. Throughout the rest of this thesis, the switching optimization use case serves as a motivating example of a SG application. It is introduced during the use case description activity based on template sections 1-5 and 7 of the IEC 62559 use case methodology. The requirements are identified during the requirements description phase, which is covered by IEC 62559 template section 6.

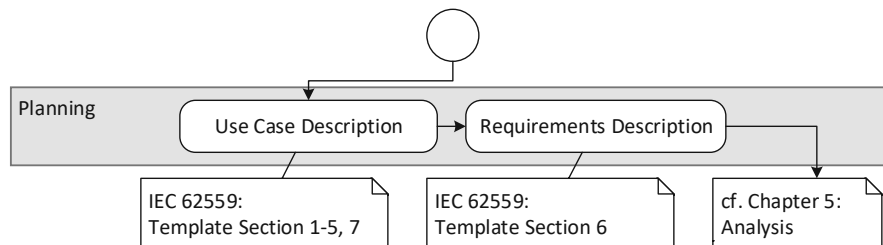


Figure 4.1: Planning phase: activities and tools

## 4.1 Use case description

As discussed in Section 2.2.1, the IEC 62559-2 [90] standard provides numerous templates to be filled in when following the methodology. Furthermore, the standard also provides an example to illustrate how to use these templates. Additional examples can be found in [24, 92]. In the following, the switching optimization use case is presented based on these templates. The structure follows the IEC 62559 template sections, each containing one or multiple template subsections/tables.

### 4.1.1 Description of the use case

This template section of the IEC 62559 use case methodology provides information relevant for project management and general information about the use case (cf. Section 2.2.1). Results are provided in Tables 4.1 to 4.8.

Table 4.1: Use case identification

<i>Use case identification</i>		
<i>ID</i>	<i>Area/Domain/Zone(s)</i>	<i>Name of use case</i>
UC1	<ul style="list-style-type: none"> <li>• <i>Areas (Layers):</i> Component, Communication, Information, Function</li> <li>• <i>Domains:</i> Distribution</li> <li>• <i>Zones:</i> Process, Field, Station, Operation</li> </ul>	LV switching optimization

Table 4.2: Version management

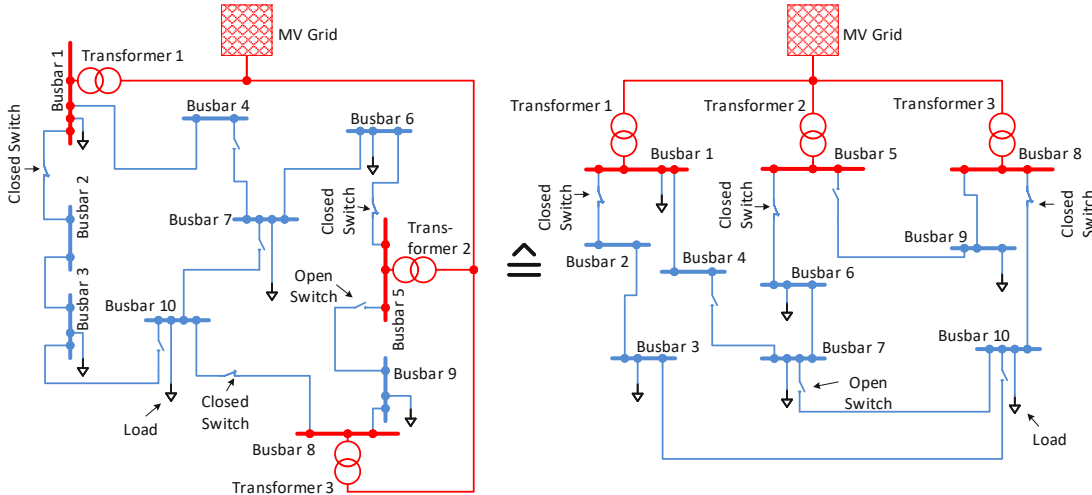
<i>Version management</i>				
<i>Version No.</i>	<i>Date</i>	<i>Name of author(s)</i>	<i>Changes</i>	<i>Approval status</i>
0.1	January 11, 2019	TF	Initial version	Draft

Table 4.3: Scope and objectives of use case

<i>Scope and objectives of use case</i>	
<i>Scope</i>	Power losses in transmission and distribution networks account for about 6% of electric energy consumption in Europe in 2010 [203]. Thereby, the main sources of power losses are transformers and cables [204]. Distribution networks often contain switches, e.g., to change the power grid topology in case of faults. The states of these switches (open or closed) determine the grid topology and, therefore, influence the losses caused by the distribution network. Thus, determining which switches should be closed and which should be opened constitutes an optimization problem. This switching optimization problem is typically solved offline during the distribution network planning phase. However, people following their daily routine and other effects cause loads to shift from one area of the distribution network to another. Adjusting the switch states more dynamically can reduce power losses, which is the focus of this use case.
<i>Objective(s)</i>	<ul style="list-style-type: none"> <li>• Reduce losses: Losses caused by the distribution network to supply a given test area shall be reduced</li> <li>• Incorporate dependability: Dependability requirements shall be considered in the decision-making process</li> </ul>
<i>Related business case(s)</i>	<ul style="list-style-type: none"> <li>• Overload prevention of line segments and transformers</li> <li>• Improved fault detection and response</li> <li>• Distributed fault record provisioning</li> <li>• Service restoration acceleration in case of faults</li> </ul>



Table 4.4: Narrative of use case

<i>Narrative of use case</i>
<b>Short description</b>
Adding ICT to components deployed in the distribution network allows dynamically adjusting the grid topology by opening and closing switches during runtime. Therefore, such systems can react to short-term changes in power consumption and continuously adjust the grid topology so that the losses caused by the distribution network are minimal. Such systems have to fulfill several functional (e.g., connectedness) and non-functional (e.g., availability) requirements.
<b>Complete description</b>
<p>Distribution networks generally follow a forest structure. Each load within the distribution network is thereby supplied by one transformer only, as effects of faults (e.g., tripped fuses in case of a short circuit) shall be kept local. Each transformer may supply numerous loads. Thus, in this analogy, transformers correspond to the forest's roots, loads correspond to leaves, and busbars correspond to inner nodes. Particularly in urban areas, a certain degree of redundancy is integrated into the distribution network by adding connections between the forest's individual trees. To maintain the locality of faults, normally open switches are added to the connections between the individual trees. Open switches are closed only in abnormal situations. Thus, strictly speaking, the graph-theoretical structure of distribution networks is often not a forest anymore, but a mesh network switched and operated as a forest, as exemplified in the figure below. Both graphs represent the same distribution network, whereby the individual elements in the right graph have been rearranged so that the forest structure becomes visible. Bus bars and loads are thereby placed underneath the transformer supplying them. In this configuration, Transformer 1 supplies Busbars 1-4 (and their connected loads), Transformer 2 supplies Busbars 5-7, and Transformer 3 supplies Busbars 8-10.</p>  <p>The distribution network structure illustrated above constitutes a valid configuration: no transformers and no lines are operated above their rated levels, all loads are connected to a transformer, and there are no closed cross-connections between trees, i.e., the forest structure is preserved. Requirements defining valid configurations are called functional requirements.</p>

Depending on the power consumption of the various loads, the distribution network topology illustrated above might be non-optimal regarding losses caused by transformers and lines. Under most circumstances, transformer losses are minimal if the loads are evenly distributed among the available transformers. Assuming identical line properties, line losses are minimal if short lines are favored over long lines. Furthermore, dependability considerations such as the reliability of components are called non-functional requirements and shall be considered in the decision-making process.

Table 4.5: KPIs

<i>Key performance indicators (KPI)</i>			
<i>ID</i>	<i>Name</i>	<i>Description</i>	<i>Reference to mentioned use case objectives</i>
KPI-01	Relative loss reduction	Loss reduction achieved by the system for a specific distribution network and load profile; measured in absolute values or in percentage of total energy losses of the same setup when the loss optimization system is disabled; may be improved by considering non-functional requirements	Reduce losses

Table 4.6: Use case conditions

<i>Use case conditions</i>
<b><i>Assumption</i></b>
The transmission network is continuously monitored and controlled. Therefore, effects on the transmission network do not have to be considered.
<b><i>Prerequisite</i></b>
Initially (before enabling the switching optimization system), the distribution network is in a state such that no equipment ratings (e.g., current levels) are violated, all loads are connected, and the network topology is a forest.
Transformers and switches are controlled by IEDs.
The distribution network topology is known and can be provided to the IEDs.
IEDs have a decent amount of memory, processing capabilities, and at least one communication interface.
Voltage and current can be measured by all IEDs.

Table 4.7: Further information to the use case for classification/mapping

<i>Further information to the use case for classification/mapping</i>
<i>Relation to other use cases</i>
N/A
<i>Level of depth</i>
Detailed
<i>Prioritisation</i>
Medium
<i>Generic, regional, or national relation</i>
Generic
<i>Nature of the use case</i>
System use case
<i>Further keywords for classification</i>
Switching optimization, overload prevention, automated reconfiguration, load shift

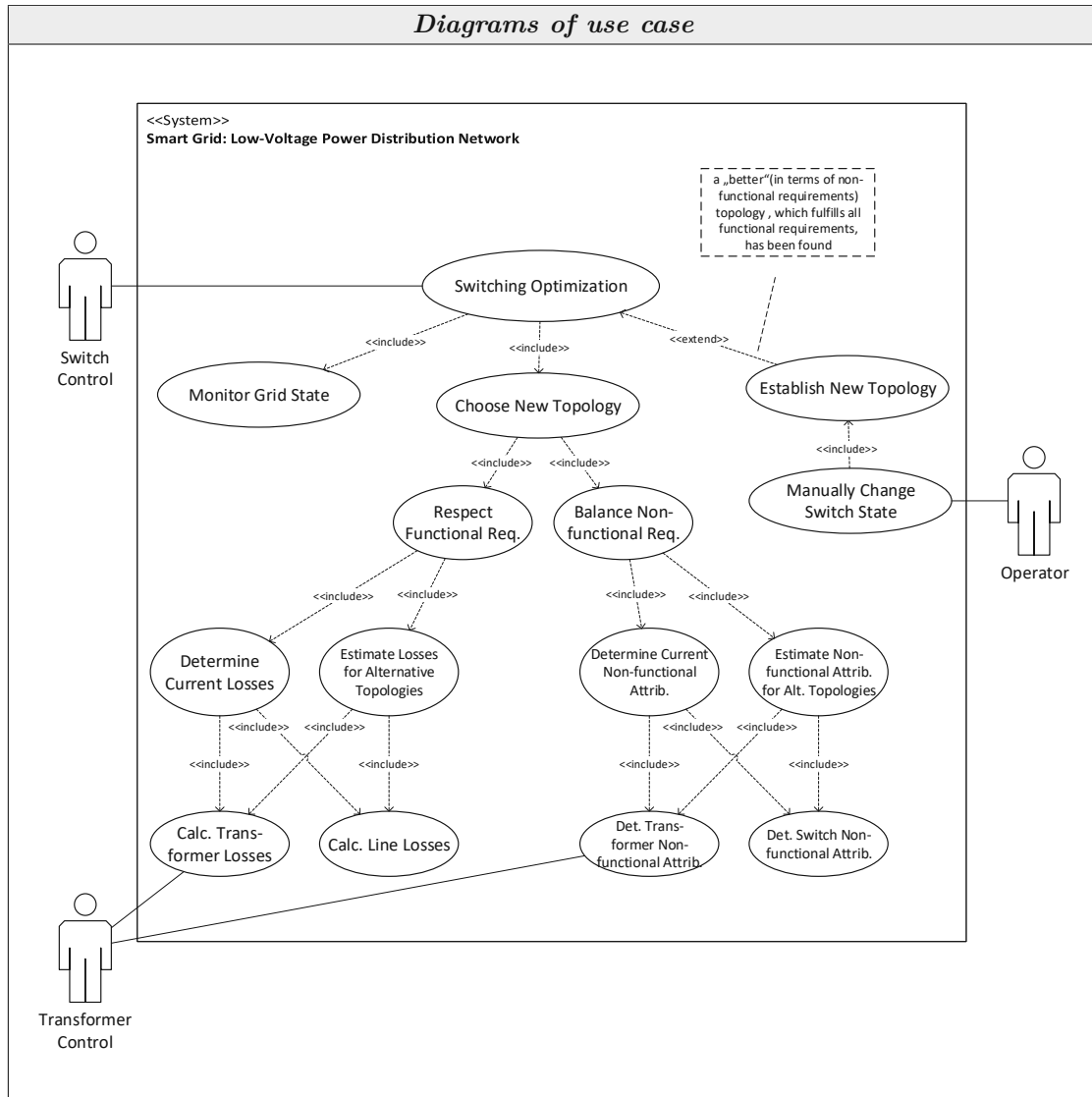
Table 4.8: General remarks

<i>General remarks</i>
The use case focuses on its realization from an ICT point of view. It shall serve as a motivating example to analyze the suitability of distributed MASs in SG applications. Therefore, challenges arising from an electrical engineering perspective are not addressed in detail.

#### 4.1.2 Diagrams of use case

This template section of the IEC 62559 use case methodology provides diagrams about the use case, particularly in the form of a use case diagram (cf. Section 2.2.1). Results are provided in Table 4.9. The main actors in the use case diagram are the switch control and the transformer control. They are both executed on IEDs and operate autonomously to implement switching optimization and perform the required actions and calculations. Furthermore, an operator has been added to the use case diagram to indicate how the use case could also support manual intervention, e.g., manually changing a switch state in case of a fault. However, this second scenario is not the main focus of this work and is eventually not further pursued.

Table 4.9: Diagrams of use case



### 4.1.3 Technical details

This template section of the IEC 62559 use case methodology focuses on the technical aspects of the use case. It describes the involved actors and relevant technical documentation (cf. Section 2.2.1). Results are provided in Table 4.10 and in Table 4.11.

Table 4.10: Actors

<i>Actors</i>			
<i>Grouping</i>		<i>Group description</i>	
Computational actors		IEDs, being computational devices, possess a certain amount of intelligence and can participate in the use case or trigger the use case.	
<i>Actor name</i>	<i>Actor type</i>	<i>Actor description</i>	<i>Further information specific to this use case</i>
Switch control	Device	A switch control is an IED that controls an electrical switch.	
Transformer control	Device	A transformer control is an IED that controls a transformer.	
<i>Grouping</i>		<i>Group description</i>	
Human actors		The use case is typically executed by computational actors only. However, in case of abnormal situations, human actors can overrule the automated system.	
<i>Actor name</i>	<i>Actor type</i>	<i>Actor description</i>	<i>Further information specific to this use case</i>
Operator	Human	Personnel (typically employed by the DNO company) responsible for distribution network operation.	The operator can open/close any switch at any time by requesting this functionality from the corresponding switch control.

Table 4.11: References

<i>References</i>						
<i>No.</i>	<i>References type</i>	<i>Reference</i>	<i>Status</i>	<i>Impact on the use case</i>	<i>Originator/organization</i>	<i>Link</i>
[203]	Report	Power Statistics & Trends 2012	Published	Scope, Motivation	Eurelectric	<a href="https://www3.eurelectric.org/powerstats/2012/key-documents/">https://www3.eurelectric.org/powerstats/2012/key-documents/</a>
[204]	Scientific paper	Towards decentralization: A topological investigation of the medium and low voltage grids	Published	Scope, Motivation	Pagani, Andrea Giuliano, Marco Aiello	<a href="https://ieeexplore.ieee.org/document/5783965">https://ieeexplore.ieee.org/document/5783965</a>
[3]	Standard	IEC standard voltages (IEC 60038)	Published	Terms & Definitions	IEC TC 8	<a href="https://webstore.iec.ch/publication/153">https://webstore.iec.ch/publication/153</a>

#### 4.1.4 Step by step analysis of use case

This template section of the IEC 62559 use case methodology discusses different scenarios of the use case (cf. Section 2.2.1). Results are provided in Table 4.12 and Table 4.13.

Table 4.12: Overview of scenarios

<i>Overview of scenarios</i>						
<i>No.</i>	<i>Scenario name</i>	<i>Scenario description</i>	<i>Primary actor</i>	<i>Triggering event(s)</i>	<i>Pre-condition(s)</i> <sup>1</sup>	<i>Post-condition(s)</i> <sup>1</sup>
1	Automated switching optimization	The grid topology shall be changed to be more efficient.	Switch control	Change/Shift in power consumption	FR-01 – FR-03	FR-01 – FR-03
2	Manual switch control	The operator wants to set the state of a switch manually. Post-conditions may be overruled, e.g., isolating a fault may require violating FR-02.	Operator	Scheduled maintenance, fault	FR-01 – FR-03	FR-01 – FR-03

The focus of the motivating example for this thesis lies on the automated switching optimization. Therefore, only Scenario 1 is fully covered from hereon. Scenario 2 could easily be analyzed using the same process but would not provide much additional insights. Throughout the rest of this thesis, Scenario 2 it is only addressed scarcely, e.g., during the security analysis.

The IEC 62559 standard provides numerous predefined services to describe the individual steps of scenario tables (cf. Table 2.1). These services cover the steps that involve communication between two actors very well. However, the standard does not suggest any service that allows to express internal operations of individual actors like complex calculations or optimizations. For this reason, a new service called PROCESS is introduced. It is described as “used to indicate that the step is performed within a single actor (no communication with other actors is required); the actor performing the process/activity shall be stated in the Information producer and the Information receiver fields”. It is used in Steps 1 and 4 of the scenario described in Table 4.13.

<sup>1</sup>Functional Requirements (FR) and Non-Functional Requirements (NFR) are defined in Table 4.17

Table 4.13: Steps for Scenario 1 – Automated switching optimization

<i>Scenario</i>								
<i>Scenario name</i>		No. 1 — Automated switching optimization						
<i>Step No.</i>	<i>Event</i>	<i>Name of process/activity</i>	<i>Description of process/activity</i>	<i>Service</i>	<i>Inf. producer (actor)</i>	<i>Inf. receiver (actor)</i>	<i>Information exchanged (IDs)<sup>1</sup></i>	<i>Requirements, R-IDs<sup>2</sup></i>
1	-	Monitor grid state	Power consumption is permanently monitored	PROCESS	Switch control	Switch control	N/A	FR-01, NFR-01, NFR-08
2	Load shift	Determine losses	Power consumption has shifted between grid parts; Losses are determined	GET, EXECUTE	Switch control, Transformer control	Switch control	InfEx-01, InfEx-02	NFR-03 – NFR-05, NFR-07, NFR-08
3	Losses determined	Estimate losses for alternative topologies	Transformer losses and line losses caused by other valid configurations are estimated	GET, EXECUTE	Switch control, Transformer control	Switch control	InfEx-03, InfEx-04	FR-01, NFR-03 – NFR-05, NFR-07, NFR-08
4	Estimated losses determined	Choose new topology	Possible new topologies are evaluated w.r.t. functional and non-functional requirements	PROCESS	Switch control	Switch control	N/A	FR-01 – FR-03, NFR-01 – NFR-08
5	Improved topology possible	Establish new topology	A new topology is established	EXECUTE, CHANGE	Switch control	Switch control	InfEx-05	FR-01 – FR-03, NFR-01 – NFR-08

#### 4.1.5 Information exchanged

This template section of the IEC 62559 use case methodology specifies the information exchanged by actors during each step of the various scenarios (cf. Section 2.2.1). Results are provided in Table 4.14.

<sup>1</sup>Information Exchanged (InfEx) items are defined in Table 4.14

<sup>2</sup>Functional Requirements (FR) and Non-Functional Requirements (NFR) are defined in Table 4.17

Table 4.14: Information exchanged

<i>Information exchanged</i>			
<i>Information exchanged ID</i>	<i>Name of information</i>	<i>Description of information exchanged</i>	<i>Requirements, R-IDs<sup>1</sup></i>
InfEx-01	Transformer losses	Losses caused by a transformer for supplying a specific area in the current topology	NFR-01, NFR-03 – NFR-05, NFR-08
InfEx-02	Line losses	Losses caused by all lines for supplying a specific area in the current topology	NFR-01, NFR-03 – NFR-05, NFR-08
InfEx-03	Estimated transformer losses	Losses estimated to be caused by a transformer for supplying a specific area in a new topology	NFR-03 – NFR-05, NFR-08
InfEx-04	Estimated line losses	Losses estimated to be caused by all lines for supplying a specific area in a new topology	NFR-03 – NFR-05, NFR-08
InfEx-05	Requested switch state	New switch state (open / closed) requested to be set by a switch control	FR-01 – FR-03 NFR-03 – NFR-05, NFR-08

#### 4.1.6 Common terms and definitions

This template section of the IEC 62559 use case methodology is intended to serve as a glossary (cf. Section 2.2.1). Results are provided in Table 4.15.

Table 4.15: Common terms and definitions

<i>Common terms and definitions</i>	
<i>Term</i>	<i>Definition</i>
Switch control	A switch control is an IED. Its purpose is to control an electric switch in a low-voltage distribution network.
Transformer control	A transformer control is an IED. Its purpose is to control a transformer in a low-voltage distribution network.
Busbar	A busbar is a metal bar (typically copper, brass, or aluminum) used to connect multiple distribution lines.
Transmission network	The transmission network is typically operated at HV/MV.
Distribution network	The distribution network is typically operated at LV.

<sup>1</sup>Functional Requirements (FR) and Non-Functional Requirements (NFR) are defined in Table 4.17



### 4.1.7 Custom information

This template section of the IEC 62559 use case methodology provides the possibility to add additional information about the use case that does not fit into one of the previous template sections (cf. Section 2.2.1). Results are provided in Table 4.16.

Table 4.16: Custom information

<i>Custom information</i>		
<i>Key</i>	<i>Value</i>	<i>Refers to section</i>
HV	High Voltage $\geq 35$ kV (IEC 60038)	3.4 Narrative of use case (Table 4.4), 3.7 Common terms and definitions (Table 4.15)
MV	Medium Voltage $\leq 35$ kV, $\geq 1$ kV (IEC 60038)	3.4 Narrative of use case (Table 4.4), 3.7 Common terms and definitions (Table 4.15)
LV	Low Voltage $\leq 1$ kV (IEC 60038)	3.4 Narrative of use case (Table 4.4), 3.7 Common terms and definitions (Table 4.15)

## 4.2 Requirements description

Requirements description is either covered in a separate document or by template section 6 of the IEC 62559 use case methodology and the corresponding table (cf. Section 2.2.1). For the simple switching optimization use case, the template section suffices. Therefore, the results of the requirements description are provided in Table 4.17. Functional requirements have to be met by each valid solution of the switching optimization algorithm, while non-functional requirements are considered in the decision-making process when selecting the best solution. The identified requirements determine possible solutions and serve as a starting point for the subsequent analysis phase.

Table 4.17: Requirements

<i>Requirements</i>		
<i>Category ID</i>	<i>Category name</i>	<i>Category description</i>
F-RQ	Functional requirements	Switching optimization shall reduce losses while not having any adverse effects on consumers and SG components. These are strict functional requirements and have to be fulfilled by any switching optimization algorithm/system.
<i>Req. ID</i>	<i>Req. name</i>	<i>Requirement description</i>
FR-01	Respect ratings	Line and transformer ratings must not be violated at any time.
FR-02	Connect-edness	All consumers have to be connected to at least one transformer at any time.
FR-03	Forest structure	The network topology before and after any reconfiguration process has to be a forest with exactly one transformer at each tree's root. During reconfiguration, two trees are temporarily connected.
<i>Requirements</i>		
<i>Category ID</i>	<i>Category name</i>	<i>Category description</i>
NF-RQ	Non-functional requirements	Solutions to the switching optimization problem have to be within the boundaries set by the functional requirements. Multiple solutions may exist within these boundaries, and the best solution shall be chosen based on the non-functional requirements.
<i>Req. ID</i>	<i>Req. name</i>	<i>Requirement description</i>
NFR-01	Reliability	The system has to conform to its specification. Deviations thereof shall be determined, and appropriate actions shall be taken.
NFR-02	Safety	The system may not endanger people or damage equipment.
NFR-03	Confidentiality	Measures to ensure unauthorized disclosure of information shall be taken.
NFR-04	Integrity	Measures to ensure unauthorized altering of information of information shall be taken.
NFR-05	Availability	Measures to ensure that the system stays operational shall be taken.
NFR-06	Maintainability	Even though the system may contain many self-contained devices distributed in the field, maintainability shall be ensured. Relevant aspects are keeping track of devices' location and configuration and enabling remote software updates.
NFR-07	Scalability	The system shall be scalable regarding the size of the distribution network, i.e., the number of transformers, switches, lines, busbars, and consumers.
NFR-08	Privacy	No data that allows deducing any information about the behavior of individual customers shall be collected or stored.

# Analysis

As illustrated in Figure 5.1, the system analysis is performed according to the two requirement categories identified in the previous phase: functional and non-functional requirements. Thereby, functional requirements are covered by a more detailed analysis of the technical aspects of the switching optimization problem and a discussion of possible algorithms, including optimal solutions and greedy heuristics. Non-functional requirements are covered by the subsequent dependability requirements analysis with a particular focus on security requirements analysis according to the NISTIR 7628 – Guidelines for Smart Grid Cybersecurity [76] (cf. Section 2.2.3). Finally, possible system architectures are discussed, and a distributed MAS is selected as a suitable approach.

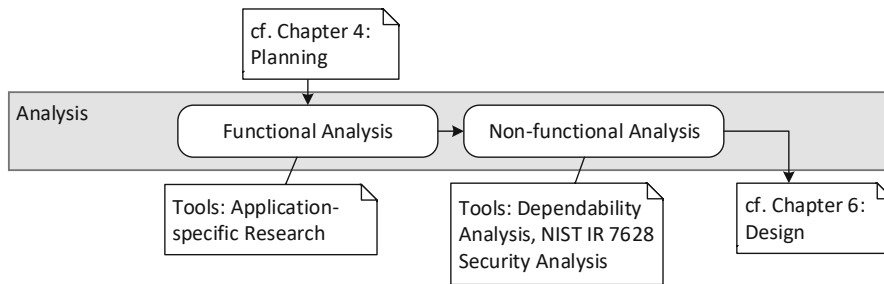


Figure 5.1: Analysis phase: activities and tools

## 5.1 Switching optimization algorithms

The main sources of power losses within the distribution network are transformer and line losses [204]. Losses caused by busbars are negligibly small and, therefore, typically not considered by switching optimization algorithms. The power losses of transformers

and lines are second-order functions of the current [205] and, thus, the switching optimization problem constitutes a non-linear optimization problem. Furthermore, individual characteristics like ratings and electrical impedance have to be considered. Mathematical modeling and analysis show that switching optimization, in general, is a Non-deterministic Polynomial-time (NP)-hard [206, 207, 208] problem.

### 5.1.1 Centralized, graph-theory-based approaches

As already argued and exemplified in Figure 1.2, power distribution networks are typically represented as graphs. In [204], Pagani and Aiello define power grid graphs as nodes (substations, transformers, and consuming units) that are connected via edges (physical cables). Adding the possibility to represent cable properties, particularly the cables' resistance, resulted in weighted power grid graphs. The definitions formulated by Pagani and Aiello are given in Definition 5.1.1 and Definition 5.1.2. A more comprehensive analysis of how power grids can be represented using graphs and various definitions related to this topic are provided by Pagani and Aiello in [209].

**Definition 5.1.1.** *Power Grid Graph:* A power grid graph is a graph  $G(V, E)$  in which each element  $v_i \in V$  is either a substation, transformer, or consuming unit of a physical power grid. If there is an edge  $e_{i,j} \in E$  between two nodes, there is a physical cable connecting directly the elements represented by  $v_i$  and  $v_j$ . [204]

**Definition 5.1.2.** *Weighted Power Grid Graph:* A weighted power grid graph is a power grid graph  $G_w(V, E)$  with an additional function  $f : E \rightarrow \mathbb{R}$  associating a real number with an edge representing the resistance, expressed in Ohm, of the physical cable represented by the edge. [204]

The resulting graphs are models of the power distribution network and, as such, can only provide a simplified abstraction of the real world. In particular, as the weights assigned to each edge are limited to the set of real numbers  $\mathbb{R}$ , it is only possible to represent the real part of the impedance  $Z$ , i.e., the resistance  $R$ , but not its imaginary part, i.e., the reactance  $X$ . Nevertheless, weighted power grid graphs provide a sufficiently detailed model to start addressing problems like switching optimization in LV power grids.

A popular approach to solving graph-based problems is mapping them to already existing graph algorithms. The number of available graph algorithms and their variations to choose from is enormous, and a profound analysis cannot be performed in this thesis. Nevertheless, a first analysis regarding the nature of the switching optimization problem and the requirements imposed resulted in the class of transshipment problems to be identified as a closely related problem. In addition to the problem definition of the classical transshipment problem, however, the switching optimization problem adds at least two more requirements:

- *Capacities*: Ratings of transformers and lines limit the amount of current that can be *transported*, i.e., the flow on each edge is capacitated.
- *Non-linear cost functions*: Line and transformer losses are quadratic functions of the current, i.e., the cost function is non-linear.

The corresponding variant of the switching optimization problem could therefore be named *capacitated quadratic transshipment problem*. Capacitated quadratic transportation problems<sup>1</sup> [210, 211] and capacitated (but linear) transshipment problems [212] have already been solved by Khurana and Verma. However, the capacitated quadratic transshipment problem still needs to be addressed. A comprehensive survey about variants of the transshipment problem is provided in [213].

A more straightforward (and less formal) approach to solving the switching optimization problem is applying a simple greedy algorithm. Thereby, the algorithm starts with a graph having all switches opened and closes one switch after the other until all consumers are supplied while respecting line and transformer ratings if possible. The quality of the strategy which switch to close next determines the quality of the solution. If this selection can be computed in linear time, such a greedy algorithm falls into the class of polynomial-time algorithms and (unless P was equal to NP) cannot deliver an optimal solution to the switching optimization problem, which is NP-hard. Such approximations have already been implemented in commercial software like Siemens' PSS®SINCAL [214] net planning tool.

Both of the mentioned graph-based approaches assume that the entire LV distribution network, including its current state and all necessary parameters, is modeled in a single weighted power grid graph and stored on a computer executing the switching optimization algorithm. In many cases, particularly during the planning phase and in small power distribution networks, this is a valid assumption. However, if the optimization shall be performed during runtime in large networks, this imposes several challenges and limitations, as all necessary data have to be communicated to, stored, and processed by this central unit. In addition to computational limitations, limiting the size of power distribution networks that can be processed in a reasonable amount of time, a central unit constitutes a Single Point of Failure (SPoF).

### 5.1.2 Decentralized & distributed approaches

Particularly in the scientific community, the problems arising from centralized approaches are most commonly approached by implementing decentralized solutions. For SG applications, these are often two-level or three-level hierarchical MASs (cf. Section 2.5.1). Agents within the lowest level have limited processing and storage capabilities and mainly forward information to or receive commands from the higher levels. Optimization problems like

<sup>1</sup>Transportation problems are simplified transshipment problems. Transportation problems only consider supply and demand locations but no intermediate nodes.

the switching optimization problem are solved for subareas of the distribution network by more powerful middle-level agents. Middle-level agents thereby use the same graph-based algorithms already discussed, but on a smaller scale. Optional top-level agents coordinate the actions of middle-level agents and handle more complex tasks, e.g., fault recovery. Decentralized approaches are well suited for many SG applications because they can easily be integrated into the existing distribution network structure. The optimization problem each middle-level agent needs to solve is simplified, and, from a system-wide perspective, the SPoF is eliminated. However, the failure of a single middle-level agent still causes a service outage for the corresponding subarea of the distribution network. While decentralized systems are superior over centralized approaches in many aspects, they still use the same concepts on a smaller scale and suffer from similar problems.

An alternative approach is to assume that all components of the SG are equal in the sense that they may have different capabilities in processing power, storage, and connected hardware, but no component exercises direct control over any other component. In such distributed settings, the switching optimization algorithm itself is not executed on a single component, but the individual components exchange messages and cooperate closely to find a switching configuration that is superior to the current situation. Therefore, information about the grid topology and the current state does not need to be consolidated at a single component but can remain distributed among all components. Depending on the optimization algorithm that is cooperatively executed by the individual components, this approach may increase communication overhead and lead to less-optimal results but eliminates the SPoF and is superior regarding the scalability of the system.

## 5.2 Dependability requirements analysis

As dependability is not a single property but a combination of the various dependability attributes, there is no simple metric to rate an application's dependability. Furthermore, the nature of the individual attributes themselves is very diverse: while, for example, availability can easily be expressed in quantitative terms, this is not as straight-forward for maintainability or scalability. To address this issue, Krammer [7] uses a three-value scale ( $-$ ,  $\sim$ ,  $+$ ) to compare the dependability properties of various communication protocols found in the building automation domain. Similarly in [25], another three-value scale (1, 2, 3) is used to discuss dependability requirements of various IoT applications. NISTIR 7628 relies on Federal Information Processing Standard (FIPS) 199's [215] and ISA 99's [216] definitions (L .. Low, M .. Moderate, H .. High) for their SG security analysis and risk assessment (cf. Table 2.2). Based on these references, the following three-value scale is introduced to rate the importance of each dependability attribute for applications like switching optimization.

- *L .. Low*: The dependability attribute is of low relevance for the application. No special measures have to be taken to ensure that the application's requirements regarding the dependability attribute are met.

- *M .. Moderate*: The dependability attribute is of moderate relevance for the application. Appropriate measures have to be taken to meet the requirements regarding the dependability attribute. The application only provides benefits to its users if the requirements regarding the dependability attribute are met. However, violating the requirements does not have a severe negative effect on assets or individuals.
- *H .. High*: The dependability attribute is of high relevance for the application. Appropriate measures have to be taken to meet the requirements regarding the dependability attribute. Violating the requirements is expected to have a severe or catastrophic adverse effect on assets or individuals.

In Table 5.1, the priority of dependability attributes in applications of the SG, Intelligent Transportation System (ITS), and Ambient Assisted Living (AAL) domains are given. A brief discussion of the top three applications is provided in [25]. The table has been extended with the switching optimization application. In the following subsections, the significance of each dependability attribute for switching optimization is stated. Furthermore, the considerations taken into account when deriving these ratings are outlined.

Table 5.1: Evaluation of dependability attributes in various application scenarios. L .. Low, M .. Moderate, H .. High priority, adapted from [25]

<i>Domain / Appl. Scenario</i>	<i>Reliability</i>	<i>Safety</i>	<i>Confidentiality</i>	<i>Integrity</i>	<i>Availability</i>	<i>Maintainability</i>	<i>Scalability</i>	<i>Privacy</i>
SG / Smart Meter	M	L	M	H	M	H	H	M
ITS / Variable Speed Limit	H	H	L	H	L	M	H	L
AAL / Emergency Call System	H	H	M	H	H	L	M	L
SG / Switching Optimization	H	L	L	H	M	H	M	L

### 5.2.1 Reliability

The reliability of a system is defined as the probability that the system conforms to its specification up to a given time  $t$ . A suitable metric to measure reliability is therefore a complementary CDF, expressed as  $1 - F(t)$ . The corresponding PDF  $f(t)$ , which describes the failure probability, can be one of many different distribution functions. Most commonly, an exponential distribution function or a Weibull distribution function is applied [217]. A closer look at failure rates of systems that are available and tested in sufficient quantity (e.g., Complementary Metal-Oxide-Semiconductor (CMOS) chips)

reveals that many technical systems suffer from infant mortality, i.e., a high failure rate within the first phase of operation caused by problems like material defects and improper handling. This phase is followed by a relatively constant failure rate for most of the system's expected lifetime. Finally, in the third and last phase, the failure rate increases due to wearout. The graphical representation of the failure rates in these three phases follows a bathtub curve (cf. Figure 5.2). Bathtub curves can be mathematically described by specifying a different distribution function for each of the three phases. Furthermore, statistical methods, e.g., Markov models, provide the means to estimate the reliability of a complex system from the reliability of its components. Thereby, either simple probability matrices [218] or more complex models [219] can be used to specify how the overall system's reliability depends on the reliability of the individual components.

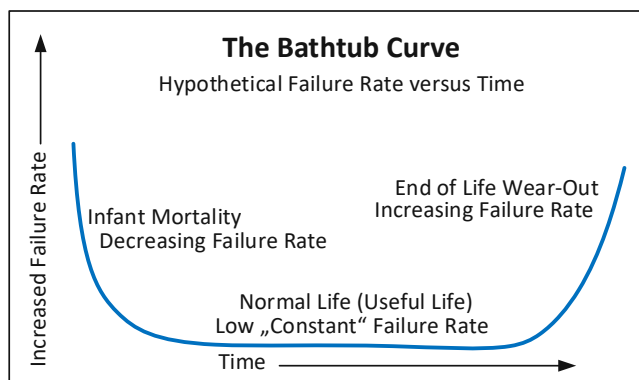


Figure 5.2: Failure rate of a technical system – the bathtub curve [220]

The main objective of switching optimization is to reduce the costs of distribution network operation. Outages caused by failures of the system defeat this objective. Therefore, it is only viable to implement switching optimization if the system is operating reliably. The exact reliability requirements on the system in terms of acceptable failure rates are subject to economic considerations.

Significance for switching optimization: H

### 5.2.2 Safety

Avizienis, Laprie, Randell, et al. define safety as the absence of catastrophic consequences on the user and the environment in the event of a failure [6]. This is a noble goal but infeasible in real-world technical systems. A more practical approach to safety is putting the likelihood of failures, i.e., the system's reliability, in perspective to their severity. The most commonly used metric for safety in technical systems, including power system components, is the notion of SILs defined in IEC 61508 [50]. Each SIL is specified by a combination of failure rate and severity. The conformity of components and systems to the individual SILs is ensured by a certification process and enforced by law. Thereby, documentation and testing requirements are more strict for higher SILs.



The combination of multiple, individually “safe” components in a new system typically requires re-certification as new risks may emerge. Additionally, *controllability*, i.e., the difficulty of managing possible accidents, may be incorporated in safety analysis [221]. Methodologies like Functional Hazard Assessment (FHA), Fault Tree Analysis (FTA), Failure Mode and Effects Analysis (FMEA), and many more exist to support developers in identifying, analyzing, and managing risks associated with their products or systems.

Switching optimization can be considered as an “add-on” to an existing distribution network. As such, its safety relies on existing equipment, e.g., enclosed switches. However, it has to be ensured that the new control mechanisms do not cause new risks to emerge.

Significance for switching optimization: L

### 5.2.3 Maintainability

Similarly to safety, Avizienis, Laprie, Randell, et al. consider maintainability to be an essential attribute of dependability but do not specify the means to express the maintainability of a component or system quantitatively. However, the theory of *repairable systems* [222] provides the necessary concepts for analyzing the maintainability of systems. A commonly used definition for maintainability is “the ability of equipment to be maintained or the probability of performing a successful repair action within a given time” [223]. Therefore, the mathematical techniques for analyzing maintainability are very similar to the ones used for reliability. Additionally, concepts originating from the field of repairable systems, such as Preventive Maintenance (PM) (servicing components before they fail) and the introduction of different repair models (e.g., perfect vs. minimal repair), enable exact quantitative analysis of maintainability [224].

While maintainability, as defined above, provides a profound mathematical analysis, other aspects should also be considered when rating the maintainability of components within the context of switching optimization. Due to the long life-span of SG devices, the number of devices, and the fact that they are distributed in the field, aspects like location information, version management, and the possibility to update devices remotely cannot be neglected.

Significance for switching optimization: H

### 5.2.4 Scalability

Most dependable systems follow a centralized or decentralized structure and are hardly changed during operation. Therefore, scalability has been of minor importance and was not included in the earlier definition of dependability [225]. However, the term is more frequently used in context with distributed dependable systems as, e.g., in the building automation domain [7]. Most commonly, systems are considered scalable if adding additional components to the system is generally possible, and a performance

gain may be expected. For example, a multi-processor computing system is perfectly scalable if its performance increases linearly with the number of processors installed.

For specific applications, the overall performance increase gained by adding additional components may even scale better than linearly. For example, considering the switching optimization problem, energy savings may improve substantially if only a few switches are added at the correct positions within the distribution network. Of course, scalability must also be considered in its more traditional sense, as individual distribution networks may grow in size and connected to other distribution networks. This has to be supported from a technological point of view, whereby the potential for additional energy savings is considered an extra benefit.

Significance for switching optimization: M

### 5.2.5 Privacy

Like scalability, privacy is not present as an attribute in the initial definition of dependability. As dependable systems may require to collect data and public awareness regarding privacy has increased over recent years, the need arises to analyze privacy in this context. Privacy is concerned about the link between peoples' identities and their data. This definition offers two approaches to ensuring privacy: protecting the person's identity or protecting the person's data [226]. In the former case, the data are collected and stored completely. However, measures are taken to avoid the possibility of linking the data to an individual's identity like anonymizing data<sup>2</sup> by not storing any unique identifier. Practical studies have shown that an adversary may still draw conclusions about an individual's identity by combining multiple attributes [226, 228]. In the latter case, the quality of the stored data itself is reduced so that the data are sufficient for the application but not unnecessarily precise. This can be achieved, for example, by only using averaged or approximated data. A combination of both approaches may be applied and is prescribed by various regulations [229]. Privacy is a rather vague concept, and the degree of privacy guaranteed by a specific solution can hardly be expressed in precise mathematical terms. Instead, multi-value metrics have been defined that allow describing to which extent an individual's privacy is assured in a specific computing system [230].

Regarding the switching optimization problem, privacy is of little concern as no information on a per-load/per-individual basis is collected or stored. However, it has to be ensured that only consolidated power consumption is measured. Thereby, the number of loads has to be above the limits prescribed by regulations. This measure reduces the risk posed by Non-Intrusive Load Monitoring (NILM) techniques, in which an adversary compares measured data to known load characteristics to draw conclusions about individuals.

Significance for switching optimization: L

---

<sup>2</sup>“anonymized” data can only be linked to its owner with “extraordinary” effort [227]

### 5.2.6 Security

The *NISTIR 7628 – Guidelines for Smart Grid Cybersecurity* [76] with their corresponding user's guide [77] are used as a methodology for conducting a detailed security analysis of the switching optimization use case. The involved steps/activities of the methodology have already been briefly discussed in Section 2.2.3. However, not all activities and steps within these activities suggested by the user's guide are relevant for this thesis. The focus of the analysis presented in the following is on technical aspects of the switching optimization use case, as seen from the operator's perspective. Therefore, organizational and business-related considerations are not covered. Furthermore, the NISTIR user's guide is intended for performing a security assessment of an existing system. Thus, some minor changes are required to identify requirements for a system under development.

**Activity 1: Identify smart grid organizational business functions:** Step 1.1 (identify an *Executive Sponsor for Cybersecurity Risk Management Governance*) and Step 1.2 (select the *Executive Cybersecurity Risk Management Governance Team*) of this activity are skipped for the reasons mentioned. Thus, the first task to be performed is Step 1.3 (identify the *Smart Grid Organizational Business Functions*). From the operator's point of view, the business function corresponding to the switching optimization use case could be termed power loss reduction. The result is documented in Table 5.2. Next, in Step 1.4, an *Organizational Business Function Risk Profile Table* is created and filled in for the business function covering power loss reduction. The meaning of the metric is H .. High, M .. Moderate, and L .. Low. The exact criteria applied to rate a risk as H, M, or L are defined in [77]. In Step 1.5, the *Priority Rating* is added to the table created in the previous step. The results of Step 1.4 and Step 1.5 are summarized in Table 5.3.

Table 5.2: Smart grid organizational business functions

<i>Organizational business functions</i>
Power loss reduction

**Activity 2: Identify smart grid mission and business processes:** Step 2.1, in which a *group of managers and subject matter experts* are assigned to the previously defined business function, is again skipped. In Step 2.2, *Supporting Business Processes (Dependencies)* are identified. In this example, switching optimization is the supporting business process that enables the power loss reduction business function. For switching optimization, the topology of the distribution network has to be known. It is assumed that the distribution network operation business function provides this information. Additionally, manual intervention (e.g., opening a specific switch by the operator for maintenance) is also enabled by this business function. Thus, the distribution network operation process contributes to the power loss reduction business function and is added to Table 5.4.

**Activity 3: Identify smart grid systems and assets:** The *Smart Grid Systems Inventory* is compiled during Step 3.1 and lists all device classes (or systems) but not the

Table 5.3: Organizational business function risk profile

<i>Org. business functions</i>	<i>Threats</i>	<i>Vulnerabilities</i>	<i>Impact</i>	<i>Probability</i>	<i>Constraints</i>	<i>Tolerances</i>	<i>Risk rating</i>	<i>Risk response</i>	<i>Priority rating</i>
Power loss reduction	Unauthorized control of system components	Accessibility of devices deployed in the field	H – Power outage, negative financial impact	M – Physical security measures may be bypassed	Budget – depending on cost-saving potential due to switching optimization	Results have to be convincing in simulation before deployment	H	Implementing adequate cybersecurity mechanisms	M

Table 5.4: Inventory of mission and business processes that support and interface with identified organizational business functions

<i>Prioritized list of organizational business function(s)</i>	<i>Supporting business processes (dependencies)</i>
1. Power loss reduction	a. Switching optimization b. Distribution network operation

individual devices. In Step 3.2, a *Risk Prioritization* is performed for each system. The results of Step 3.1 and Step 3.2 are provided in Table 5.5. This table is extended during each of the following activities. The individual devices, i.e., assets, alongside additional information such as their names, locations, and serial numbers, are usually collected during Step 3.3 in the *Smart Grid Asset Inventory* table. As the switching optimization system has not been deployed yet, there is no content for the *Smart Grid Asset Inventory*, and Step 3.3 is skipped in this analysis activity.

**Activity 4: Map smart grid systems to logical interface categories:** For each system of Table 5.5, the *associated actor* is chosen from the actors stated in Figure 2.5. Figure 5.3 illustrates the parts of the NISTIR logical reference model relevant for this use case. To support this, the exact meaning of actors is specified in [76]. As already mentioned, the naming may differ. However, in many cases, it is still possible to map each system to an actor. According to this description, switches and transformers both include functionalities of actors *Distribution Data Collector* (actor number 12), *Distributed Intelligence Capabilities* (actor number 13), *Distribution Remote Terminal Unit/Intelligent Electronic Device* (actor number 15), and *Distribution Sensor* (actor number 18). Furthermore, the *Distribution Management System* (actor number 27) together with the *Distribution SCADA* (actor number 29) provide information about the distribution network topology and enable distribution network control. These actors are

Table 5.5: SG systems inventory

Steps: 3.1			3.2				4.1	4.2	4.3	
Priority busi- ness function(s)	Business processes	System name(s)	<i>Risk prioritization</i>				Actors	Logical interfaces	Logical inter- face category(s)	
			Impact			Probability				Risk ranking
			<i>C</i>	<i>I</i>	<i>A</i>					
Power loss reduction	a. Switch- ing optimi- zation	Switch control	L	H	M	M	H	12, 13, 15, 18	U108 U111 U112	12 11 12
		Transformer control	L	H	M	M	H	12, 13, 15, 18	U108 U111 U112	12 11 12
	b. Distri- bution network operation	Distrib. net- work topology	L	M	M	L	M	15, 27 29	U9 U117	5 1
		Distrib. net- work control	L	H	H	M	H	15, 27, 29	U9 U117	5 1

therefore added to the table during Step 4.1. Next, in Step 4.2, each *logical interface* connecting two of the identified actors is documented in an additional column. These interfaces are derived from Figure 2.5 with additional information provided in [76]. Additionally, the standard associates each logical interface to one or multiple LICs. During Step 4.3, the corresponding *LICs for each logical interface are selected* and added to the table in an additional column. In this example, these are *Interface between control systems and equipment with high availability, and with compute and/or bandwidth constraints* (LIC 1), *Interface between control systems within the same organization* (LIC 5), *Interface between sensors and sensor networks for measuring environmental parameters* (LIC 11), and *Interface between sensor networks and control systems* (LIC 12). The relevant results of this step are then added to the new Table 5.6.

**Activity 5: Map smart grid systems to logical interface categories:** The standard [76] defines a CIA impact for each LIC. These *CIA impacts are added* to Table 5.6 in Step 5.1. Next, in Step 5.2, the *Organizational CIA Impact is specified* by combining the results from Step 3.2 and Step 5.1 according to the criteria defined in Section 2.2.3. The CIA Impact is added to the table. The standard [76] prescribes *Requirements for Each System* for each combination of LIC and Organizational CIA impact. Requirements are categorized into three different *Requirement Types*: GRC Smart Grid Requirements, CTRs, and UTRs. Therefore, GRCs rows and the relevant GRC requirements themselves

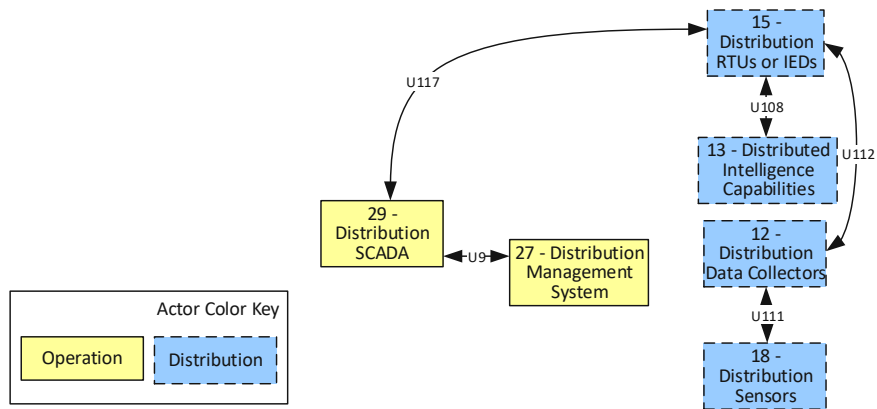


Figure 5.3: NIST IR 7628 logical reference model for switching optimization (excerpt of [76])

Table 5.6: SG systems inventory with CIA impacts

Steps: 3.1	3.2					4.1	4.2	4.3	5.1			5.2					
	<i>Risk prioritization</i>								<i>Actors</i>	<i>Logical interfaces</i>	<i>Logical interface category(s)</i>	<i>NISTIR CIA impact</i>			<i>Organizational CIA impact</i>		
	<i>Impact</i>	<i>Probability</i>	<i>Risk ranking</i>	<i>C</i>	<i>I</i>							<i>A</i>	<i>C</i>	<i>I</i>	<i>A</i>		
Switch control	L	H	M	M	H	12,	U108	12	L	M	M	L	H	M			
						13,	U111	11	L	M	M	L	M	M			
						15,	U112	12	L	M	M	L	H	M			
						18	U117	1	L	H	H	L	H	M			
Transformer control	L	H	M	M	H	12,	U108	12	L	M	M	L	H	M			
						13,	U111	11	L	M	M	L	M	M			
						15	U112	12	L	M	M	L	H	M			
						18	U117	1	L	H	H	L	H	M			
Distribution network topology	L	M	M	L	M	15,	U9	5	L	H	H	L	H	H			
						27,	U117	1	L	H	H	L	H	M			
Distribution network control	L	H	H	M	H	15,	U9	5	L	H	H	L	H	H			
						27,	U117	1	L	H	H	L	H	M			
29																	

are added to the table in Step 5.3. The same process is repeated for CTRs in Step 5.4 and UTRs in Step 5.5. With over 130 requirements in this category, GRC contributes by far the most requirements. However, they focus on an organizational and personnel level, including security-relevant topics such as documentation about access control, interaction with external information systems, how to manage publicly accessible content, security awareness and training of employees, conducting audits, and many more topics that are highly relevant on an organizational level but are skipped in this analysis. CTRs and UTRs, contributing the more technical aspects of SG security, are determined and listed in Table 5.7. Additionally, the standard defines requirement enhancements for some requirements of any of the three requirement types. These *requirement enhancements are added* next to the corresponding requirement (within parentheses) in Step 5.6. Steps 5.3 to 5.6 are rather complex in practice, and it is recommended to consult the NISTIR 7628 user’s guide [77] for instructions. A slightly different table format than suggested by the user’s guide has been chosen for Table 5.7, resulting in Step 5.7 (*consolidating UTRs for each system*) being unnecessary. Therefore, this step is omitted.

**Activity 6: Perform a smart grid high-level security requirement gap assessment** In Step 6.1 each of the *requirements* identified during the previous activity *are rated satisfied (“S”) or other than satisfied (“O”)*. As the system is still in its analysis phase, all requirements are currently other than satisfied (cf. Table 5.8). Furthermore, Step 6.2 (*conducting the gap assessment*) and Step 6.3 (*adding the gaps to the smart grid systems inventory table*) are not applicable, as the gap is always equal to the requirement in a non-existing system.

**Activity 7: Create a plan to remediate the smart grid high-level security requirement gaps** In Step 7.1 and Step 7.2, *additional columns for Proposed Mitigations and Priorities are added* (cf. Table 5.8). In Step 7.3 and Step 7.4, the requirements are analyzed, and their *priority is rated* considering aspects like potential impact, probability, constraints, and tolerances. A priority rating is documented in the table for each requirement. In Step 7.5, *proposed mitigations are identified* for each of the requirements. The NISTIR 7628 standard lists detailed information for all of the requirements and provides many useful hints about how they can be addressed.

Table 5.7: SG systems inventory with organizational impacts, unique technical requirements, and requirement enhancements

Steps: 3.1	4.3	5.2			5.3	5.3–5.6
<i>System name(s)</i>	<i>Logical inter-face category(s)</i>	<i>Organizational CIA impact</i>			<i>Requirement type</i>	<i>Requirements for each system SG.*</i>
		<i>C</i>	<i>I</i>	<i>A</i>		
Switch control / Transformer control	12	L	H	M	GRC	n/a
					CTR	AC-6, AC-7, AC-8, AC-9, AC-16, AC-17, AC-21, AU-2(1), AU-3, AU-4, AU-15, CM-7, CM-8, SA-10, SA-11, SC-11, SC-12, SC-15, SC-17, SC-18, SC-19, SC-20, SC-21, SC-22, SC-30, SI-8, SI-9
					UTR	IA-5(1–2), SC-5, SC-7(1–3), SC-8(1), SI-7(1)
	11	L	M	M	GRC	n/a
					CTR	same as for LIC 12
					UTR	SC-8(1)
	1	L	H	M	GRC	n/a
					CTR	same as for LIC 12
					UTR	AC-14(1), IA-4, IA-5(1–2), IA-6, SC-3, SC-4, SC-5, SC-7(1–3), SC-8(1), SC-29, SI-7(1)
Distribution network topology / Distribution network control	5	L	H	M	GRC	n/a
					CTR	same as for switch control LIC 12
					UTR	AC-14(1), IA-4, IA-6, SC-5, SC-6, SC-7, SC-8(1), SC-29, SI-7(1), AC-14(1), IA-4, IA-5(1–2), IA-6, SC-3, SC-5, SC-7(1–3), SC-8(1), SC-29, SI-7(1)
	1	L	H	M	GRC	n/a
					CTR	same as for switch control LIC 12
					UTR	AC-14(1), IA-4, IA-5(1–2), IA-6, SC-3, SC-5, SC-7(1–3), SC-8(1), SC-29, SI-7(1)



Table 5.8: SG systems inventory with assessment scores and assessment gaps

Steps: 5.3–5.6	6.1	6.2	7.1, 7.5	7.2– 7.4
<i>Requirements for each system SG.*</i>	<i>Assessment ratings (S / O)</i>	<i>Assessment gaps</i>	<i>Proposed mitigations</i>	<i>Priorities</i>
AC-6	O	n/a	Impl. authentication mechanism	H
AC-7	O	n/a	Impl. Role-Based Access Control (RBAC)	H
AC-8	O	n/a	Log unsuccessful auth. attempts	L
AC-9	O	n/a	Issue warning before executing commands	L
AC-14(1)	O	n/a	Authentication and authorization shall not be bypassed, even in emergency situations	H
AC-16	O	n/a	Encrypt wireless communication	H
AC-17(1–4)	O	n/a	Impl. auth. & RBAC	H
AC-21	O	n/a	Specify password rules	M
AU-2(1)	O	n/a	Not a technical requirement	-
AU-3	O	n/a	Not a technical requirement	-
AU-4	O	n/a	Not a technical requirement	-
AU-15	O	n/a	Not a technical requirement	-
CM-7	O	n/a	Impl. firewall and restrict enabled services	H
CM-8	O	n/a	Store inf. about assets in a knowledge base	L
IA-4	O	n/a	Impl. auth. & RBAC	H
IA-5(1-2)	O	n/a	Impl. device authentication mechanism	H
IA-6	O	n/a	Obscure feedback of authentication information	H
SA-10	O	n/a	Not a technical requirement	-
SA-11	O	n/a	Not a technical requirement	-
SC-3	O	n/a	Specify access rights for security functions	H
SC-4	O	n/a	Impl. auth. & RBAC	H
SC-5	O	n/a	Out of scope	-
SC-6	O	n/a	Specify priorities for resources and services	L

SC-7(1-3)	O	n/a	Out of scope	-
SC-8(1)	O	n/a	Apply state of the art cryptography mechanisms	H
SC-11(1)	O	n/a	Apply state of the art key management systems	H
SC-12	O	n/a	Apply state of the art cryptography libraries	H
SC-15	O	n/a	Apply state of the art Public Key Infrastructure	H
SC-17	O	n/a	Use state of the art VoIP if necessary	L
SC-18	O	n/a	Secure external connections	L
SC-19	O	n/a	Implement RBAC	H
SC-20	O	n/a	Apply state of the art cryptography mechanisms	H
SC-21	O	n/a	Covered by authentication mechanisms	-
SC-22	O	n/a	Out of scope	-
SC-29	O	n/a	Impl. auth. & RBAC	H
SC-30	O	n/a	Implement decentralized / distributed system	M
SI-7(1)	O	n/a	Apply state of the art cryptography mechanisms for software and databases	H
SI-8	O	n/a	Apply consistency checks & message authentication	H
SI-9	O	n/a	Categorize errors and define responsibilities	L

#### Activity 8: Monitor and maintain smart grid high-level security requirements

During this activity, the progress of each proposed mitigation is monitored and documented. This task is again primarily relevant to the organizational/management level and, therefore, omitted here.

The results provided in Table 5.7 can be combined to derive the significance of each CIA attribute for the switching optimization use case. Thereby, the highest entry of Table 5.7 for each attribute determines its overall significance.

Significance for switching optimization: Confidentiality: L, Integrity: H, Availability: M

Furthermore, Table 5.8 provides technical solutions that can be incorporated in the design and implementation phases to ensure the required level of security. This activity completes the analysis of the analysis phase. Next, the insights gained so far are used to design a system that meets the functional and non-functional requirements.

The activities and tools of the design phase are illustrated in Figure 6.1. Based on the information gathered during the planning and analysis phases, a suitable system is designed. As mentioned, for the course of this thesis, the system architecture is a distributed MAS. Some additional concepts and considerations regarding the system design are introduced and discussed. Next, the MAS design and ontology design activities are conducted in parallel. The MAS design process follows the PASSI methodology and starts with defining the system requirement model and agent society model. For the ontology design activity, a reusable ontology design methodology is introduced and followed to create the required ontologies for the switching optimization use case.

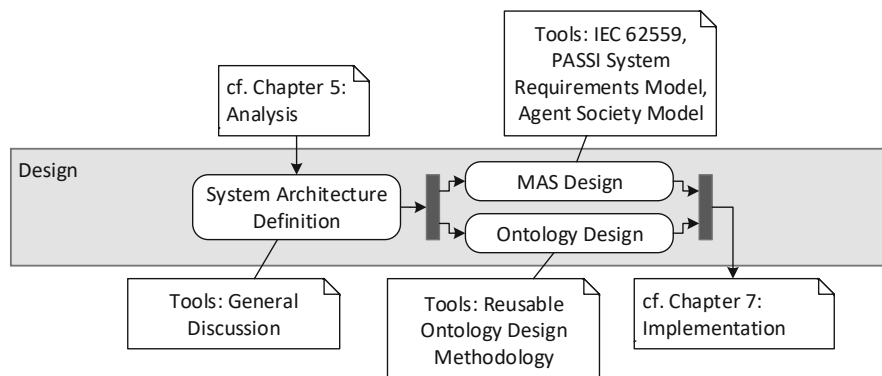


Figure 6.1: Design phase: activities and tools

## 6.1 System architecture definition

The insights gained in the previous chapter shall now be exploited to select a suitable system architecture among the various architectures that are typically found in the SG

or the industrial domain, respectively. Thereby, the focus is on distributed MASs for the reasons outlined in the following section. Afterward, the system architecture is defined and several general concepts required in the subsequent MAS design step are discussed.

### 6.1.1 Smart grid system architectures

Early power utilities employed human dispatch operators equipped with SCADA systems to manage plant control, protective relaying, transmission switching, and communication protocols, along with economic operation of large interconnected power plants. While SCADA systems offer timely and detailed monitoring of traditional grid resources, additional analysis by engineers using multiple data sources is required to obtain useful information about the power systems' state and operation. Such a manual analysis of data can be time-consuming. For example, in the event of a grid failure, SCADA systems can generate thousands of fault records in a matter of only a few hours. Real-time manual analysis of these data is unfeasible [157].

In particular with the spread of DERs, it is expected that centralized and hierarchical SCADA systems will soon reach their limits regarding scalability, computational complexity, and communication [231]. More flexible architectures like SOA and MAS have been heavily researched in the SG domain and gradually find their way into industrial applications [11]. SOA and MAS are often described as being complementary paradigms [232]. While the scientific debate about the differences and similarities of SOA vs. MAS are not settled yet, both paradigms build upon the idea of loosely coupled entities that cooperate and exchange messages to implement an application [231]. The general opinion tends towards ascribing an agent of a MAS a higher degree of independence than a service in a SOA.

Thus, a MAS is a suitable architecture for implementing many SG applications, including the switching optimization problem. However, it remains to decide whether a centralized, de-centralized, or fully distributed approach shall be followed. While the latter leads to higher complexity of the individual agents and limits the number of applicable switching optimization algorithms (cf. Section 5.1), it provides several unique advantages over the other options. A distributed approach eliminates SPoFs. If one agent fails due to an attack or a defect, all others can still fulfill their purpose. Furthermore, the autonomy of agents is preserved in the way envisioned by the definition of McArthur et al. and Wooldridge and Jennings. In centralized and in multi-level hierarchical approaches, agents are often directly controlled by more powerful agents, which, in a way, contradicts the initial idea of MASs. And finally, scalability is inherent to a distributed approach because the view of each agent can be limited to the parts of the distribution network it is connected to. Thus, extending the distribution network geographically and with additional components provides an additional potential for optimization rather than increasing the complexity.

### 6.1.2 Similarities to the industrial domain

Industry 4.0 and the SG face similar challenges, e.g., long life-cycles of components and the necessity to include existing assets in new applications. Consequently, concepts are exchanged from one domain to the other. In particular, the SGAM of the SG domain found its counterpart in the industrial domain as the Reference Architectural Model for Industrie 4.0 (RAMI 4.0) [234]. Likewise, concepts from the industrial domain like the Asset Administration Shell (AAS) may also contribute to the development of SGs. The concept of an AAS is defined as part of the RAMI 4.0 and illustrated in Figure 6.2.

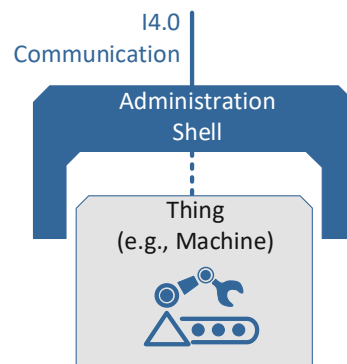


Figure 6.2: Administration shell defined by RAMI 4.0 [234]

As the name suggests, the AAS primarily focuses on the administration of assets. Such an asset can either be a software component (e.g., a database) or a hardware component (e.g., an industrial robot). The AAS provides information about and access to the asset via standardized or proprietary interfaces and data structures [18]. It shall include all data available about the asset across all phases of the asset’s life cycle, including but not limited to data about identification, engineering, configuration, and runtime [235]. Additionally, the Industry 4.0 platform glossary also assigns the AAS an active aspect and defines it as a “virtual digital and active representation of an I4.0 component in the I4.0 system” [236].

The definition of an intelligent agent provided Section 1.3, which includes the behavior, computing, and control aspects, partly coincides with the concept of AASs known from the industrial domain. In many situations, an agent can be considered a virtual representation of some *asset* or *thing*. Thereby, the agent allows other agents to access the services of the asset it controls. These insights inspired the following system architecture definition.

### 6.1.3 System architecture definition

The similarities between the industrial domain and SGs give rise to the idea of an agent controlling a power system component like a switch or a transformer enabling these devices to participate in a SG similarly to the way an AAS enables assets/things to

participate in an I4.0 system. Such an agent shall be named *Component Administration Agent (CAA)*. An architecture building upon this concept is described in more detail in the following.

A CAA and the power system component it controls are illustrated in Figure 6.3. The architecture is FIPA-compliant. However, although FIPA strictly distinguishes between the agent and the AP, this is generally not common. Instead, most scientific literature only refers to agents as a general concept subsuming the agent’s functionalities and the AP it is executed on. This allows for more natural formulations and is also the preferred approach in this thesis. Consequently, the term *agent control* is used to explicitly refer to the component that contains the application-specific agent logic.

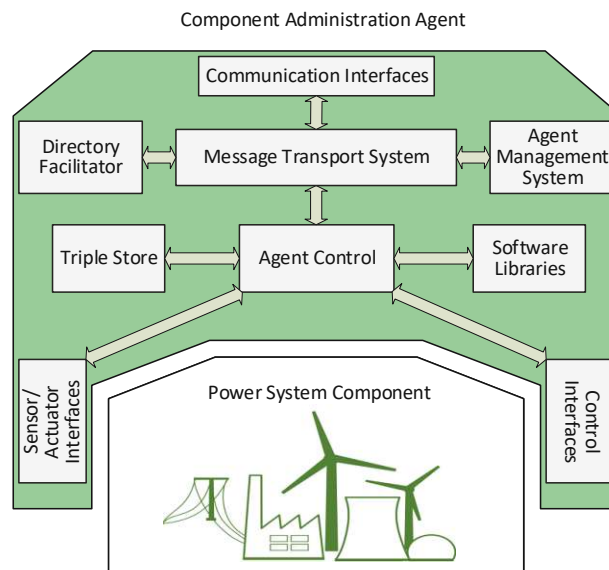


Figure 6.3: Agent controlling a power system component

As illustrated in Figure 6.3, an CAA consists of the following components:

- *Agent control*: core element containing all use-case-specific functionality
- *Message Transport System, Directory Facilitator, and Agent Management System*: components defined by FIPA (cf. Section 2.3.5)
- *Communication interfaces*: enable communication with other agents or external systems
- *Triple store*: software component used to store and manipulate ontologies
- *Software libraries*: third-party software libraries that can be used by the agent, e.g., for cryptography and optimization algorithms

- *Sensor/actuator and control interfaces*: interfaces used to interact with the power system component either directly via digital or analog I/O, or via control interfaces like bus systems or Industrial Ethernet (IE)

Figure 6.4 illustrates how components of a distribution network are equipped with agents enabling them to communicate and actively participate in the SG. In this example, only transformers and switches are equipped with agent technology, but the same concept can easily be extended to busbars, loads, additional sensors, smart meters, and other components in the transmission and distribution networks. In addition, an *area* is highlighted in the figure. An area is the smallest subset of the distribution network that can be supplied via an alternative path by opening/closing the corresponding switches. Areas play an important role in the optimization procedure and are defined more precisely in Section 6.1.5.

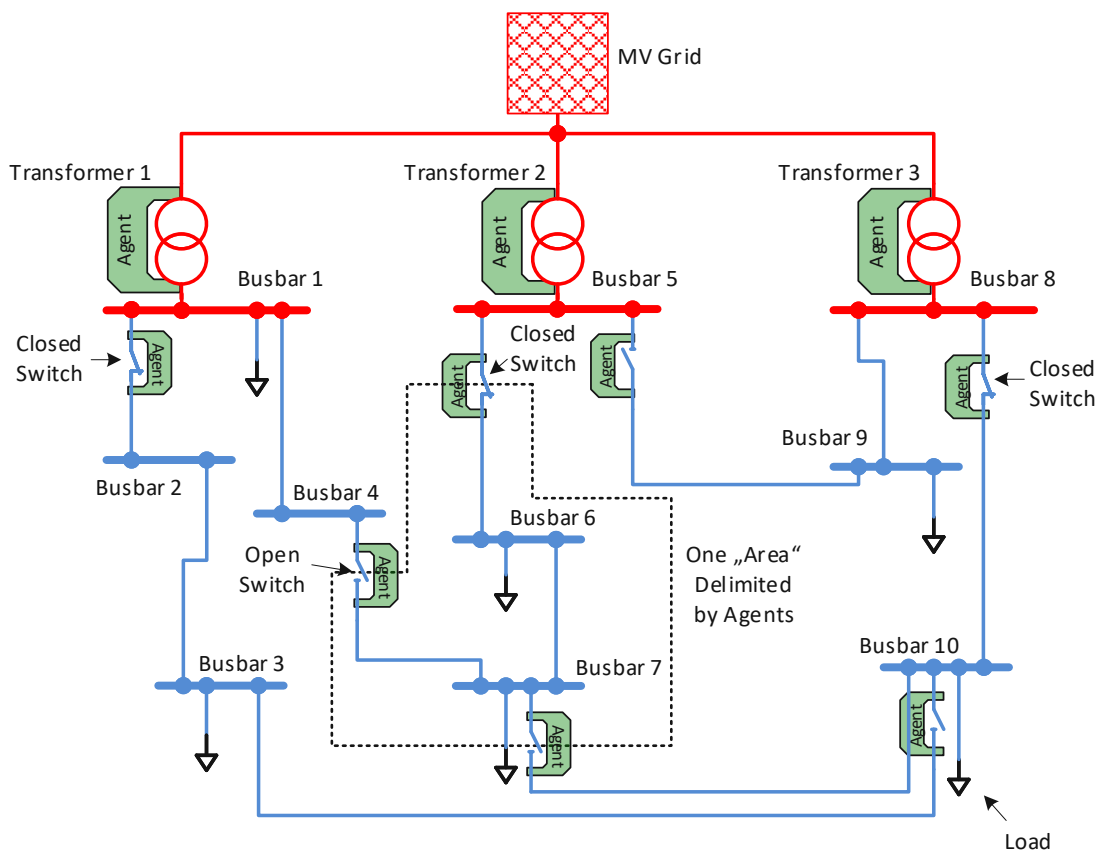


Figure 6.4: Distribution network with MAS-based component control

#### 6.1.4 Agent types and services

The base version of the MAS presented in this thesis incorporates two types of agents: *Switch Agents* and *Transformer Agents*. Naturally, switches are controlled by Switch Agents, and transformers are controlled by Transformer Agents.

Agents within the MAS may implement two types of services: the *Switching Service* and the *Loss Evaluation Service*. The Switching Service allows any Switch Agent to open or close its switch. Additionally, the service is offered to other agents, allowing agents to request opening and closing switches from any Switch Agent. In general, a Switch Agent is obligated to serve a switching request when received. The Loss Evaluation Service is offered by Switch Agents and Transformer Agents. It evaluates the losses that would be caused if an additional load was supplied by the agent offering the service and the possible savings if the load supplied by the agent was reduced by a certain amount.

#### 6.1.5 Definition of areas

It is assumed that all switches and all transformers within the LV grid of interest are equipped with agent technology. Switching optimization in the course of this thesis is performed in the granularity of *areas*. Thereby, an area is the smallest group of inner nodes (busbars) and leaf nodes (loads) delimited by Switch Agents or Transformer Agents. Delimited in this context means that each possible connection between the area and the transmission network involves at least one agent. Areas can easily be identified by using a breadth-first search. The search algorithm starts at an agent and stops when another agent is found at each branch. Agents do not need to store the complete topology of the distribution network but only the topology of the areas they are directly connected to. Agents that are connected to the same area are termed *neighbors* or *neighboring agents*.

Areas are always delimited by agents, even if there are no switches at all (and, therefore, no Switch Agents). In this case, the areas just correspond to the individual trees of the distribution network. Furthermore, each area is supplied by precisely one Transformer Agent (either directly or via exactly one other area), i.e., there is only a single path connecting the area to the transmission network. Any other possible path from the area to the transmission network must be interrupted by at least one open switch to ensure the tree topology.

It is assumed that switches (and, therefore, also Switch Agents) are placed in a way so that they connect two different areas  $A$  and  $B$ . According to the definition of an area, if a Switch Agent connected an area to itself, i.e.,  $A = B$ , there had to be a connection from  $A$  to  $B$  which does not involve a Switch Agent. While this is, in principle, possible, closing such a switch would always create a cycle and violate the tree structure. Transformer Agents are only connected to one area because they are connected to the transmission network via their primary side.



### 6.1.6 Optimization procedure

The overall optimization procedure repeatedly tries to answer the following question: “Is there an area (or multiple areas) that could be connected to the transmission network with less power losses than caused by the current configuration?”. If so, the distribution network should be restructured. In the proposed distributed MAS, where no agent knows the complete distribution network, each agent tries to answer this question only for the areas it is directly connected to. Restructuring is performed by swapping an area (possibly including other areas supplied by it) from one transformer to another.

For this purpose, the agent has to know the current losses caused by supplying an area and possible alternatives to supply it. Therefore, one *optimization run* is separated into two phases. During *Phase 1*, the agent tries to evaluate the losses caused by the current configuration, and during *Phase 2* it requests proposals for alternative configurations.

### 6.1.7 Trigger for the optimization run

Naturally, the question arises which agent should start the optimization run and when to do so. A possible approach is to let the Transformer Agents initialize the procedure, i.e., a **top-down approach**. Starting from the root nodes, the information about the current losses would propagate through the distribution network. The information should not be kept within the tree supplied by the initiating Transformer Agent but should also be forwarded to neighboring areas. This allows Switch Agents to compare the current configuration with alternative configurations (in terms of losses). However, the top-down approach has several drawbacks. (1) Each agent within a MAS should have a clear motivation for its actions. Arguably, spreading information for the sake that it is possibly useful for other agents is not considered a clear motivation. (2) Communication bandwidth is used for information that is not needed and without consequences for most agents receiving it. (3) The exact losses for alternative configurations depend on the power consumption within specific areas. However, in a top-down approach, the Transformer Agents do not have any information thereof.

There are certainly counter-arguments and measures to be taken for each of the mentioned drawbacks. However, for a first approach, a **bottom-up approach** seems more reasonable. Any agent that encounters optimization potential requests the losses caused by the energy consumed within the relevant area(s) from the agent located closer to the transmission network. Thus, the request travels bottom-up and can be answered once it reaches the responsible Transformer Agent. Only agents required to answer the request are involved in the communication process.

The question of when the Switch Agent should start the optimization procedure remains. Agents may coordinate with each other so that the optimization procedure is started by the Switch Agents one after another (i.e., **sequentially**). It may do so **periodically**, e.g., at every full hour (**periodically synchronized**), or 24 times a day scatter based

on the agent's ID (**periodically scattered**). These ideas are all valid, but neglect that time itself is not a good indicator of optimization potential. An approach that takes **changes in consumption** into account (e.g., +/- 5 % within a specific area) was chosen for this work, but in fact, when to start the optimization procedure is up to the individual agent and may vary. Consequently, an optimization procedure is only started by the Switch Agent currently supplying a specific area because only this agent can determine changes in consumption.

### 6.1.8 Resource reservation

Agents are not allowed to participate in multiple optimization runs at the same time. In particular during Phase 2, an agent proposes additional losses for supplying a given load value (i.e., the consumption within an area to be supplied) and verifies that supplying this additional load would not violate any line or transformer ratings. The proposal's validity must be ensured until the optimization run has been completed to maintain the stability of the grid. However, an agent does not know beforehand whether or not its proposal will be accepted. Thus, it has to reserve the resources until its proposal has been accepted or declined. Meanwhile, it has to refuse other Call For Proposals (CFPs).

## 6.2 MAS design

The basic ideas and concepts presented in the previous section shall now serve as a starting point for the MAS design activity. Therefore, a suitable MAS design methodology is needed. As in many cases, no MAS methodology comes out clearly beneficial over the others (cf. Section 2.3.4). However, the PASSI methodology has been identified as an appropriate MAS design methodology as it supports most of the desired features. The decisive criteria of Table 2.4 for selecting PASSI over GAIA and MaSE are:

- *Coverage of the life cycle:* PASSI includes implementation and evaluation (testing) elements.
- *Application domain:* FIPA is the most commonly used AP, particularly in the SG domain. The close relation of PASSI and FIPA will reduce the effort of mapping the results of the MAS design methodology to FIPA as a specific AP.
- *Communication ability:* The ability of PASSI to support knowledge-based communication simplifies interaction with already existing agents.
- *Human-computer interaction:* According to the use case description (cf. Chapter 4), for certain scenarios the operator shall be able to interact with the MAS. The MAS design methodology should, therefore, also consider human-computer interaction.
- *Support for ontology:* An ontology provides the necessary means for knowledge-based communication and can serve as a very flexible data store.

### 6.2.1 System requirements model

The PASSI methodology starts with the system requirements model, including the *Domain Requirements Description* in the form of a use case diagram. This diagram has already been created as part of the IEC 62559 use case description during the planning phase in Chapter 4. It is provided in Table 4.9.

Next, the use case diagram is further refined during the *Agent Identification* phase. The result thereof is depicted in Figure 6.5. The Operator is included in this figure to illustrate how personnel can interact with the MAS via a user interface agent. However, as mentioned, the focus is on the automated switching optimization scenario, and the manual switch control scenario is not discussed any further.

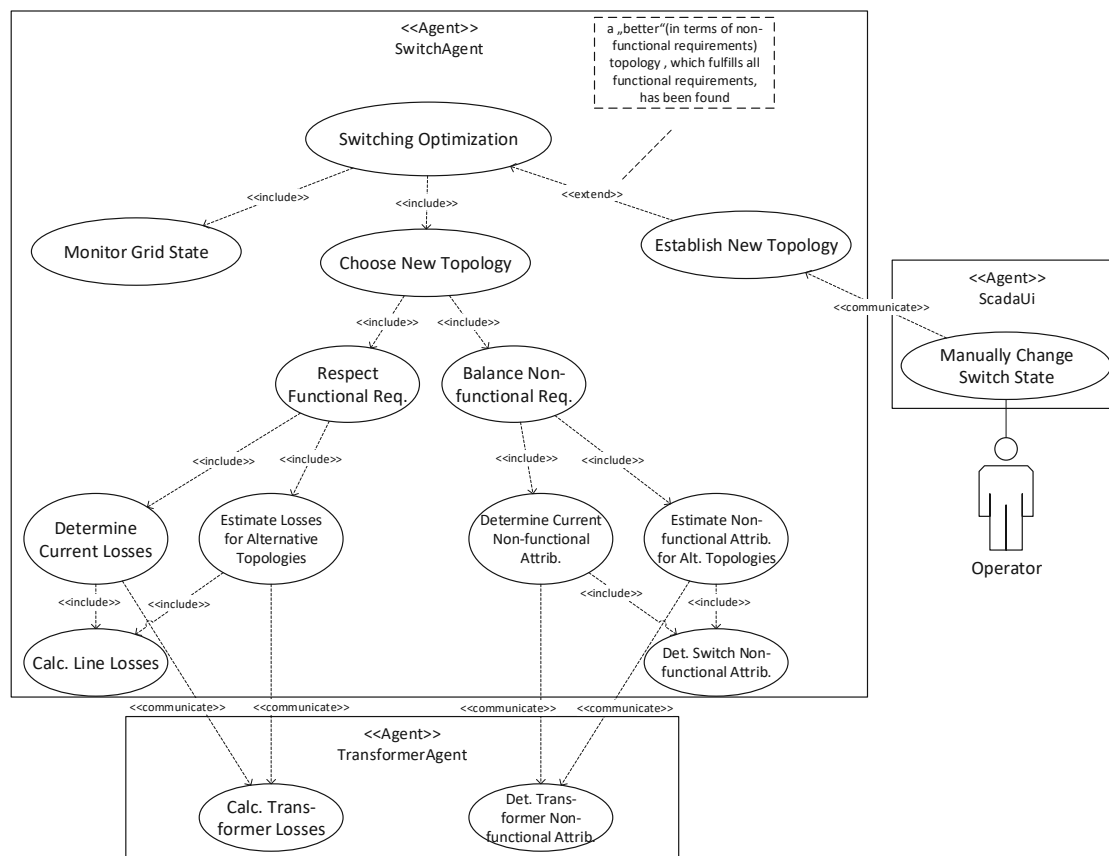


Figure 6.5: PASSI agent identification diagram for the switching optimization use case

The interactions between agents are examined in more detail during the *Role Identification* phase. Thereby, each scenario of the use case analysis is transformed into a sequence diagram or a roles identification diagram in PASSI terminology. This allows identifying the various roles each agent can take. The roles identification diagram for the automated switching optimization scenario is depicted in Figure 6.6.

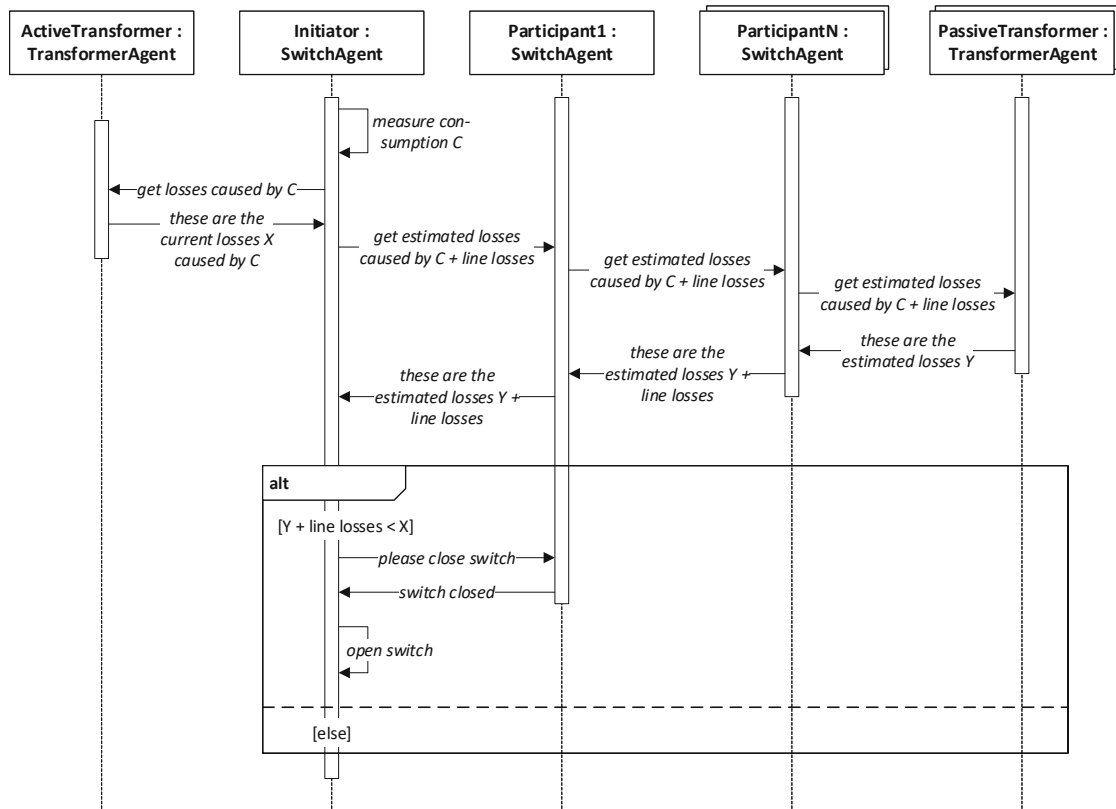


Figure 6.6: PASSI roles identification diagram for automated switching optimization

An optimization run is always initiated by a Switch Agent, taking the role of the Initiator. This is because only the Initiator is currently supplying the corresponding area(s) and can, therefore, determine changes in the power consumption and trigger the optimization run (cf. Section 6.1.7). The Initiator first has to determine the losses caused within the distribution network for supplying power to the area. It does so by sending a request to the Transformer Agent that is currently supplying the area. This Transformer Agent takes the Active Transformer role. Additional Switch Agents between the Initiator and the Active Transformer are omitted for the sake of clarity in this figure. The Active Transformer estimates the losses caused by supplying the area and responds to the Initiator.

Next, the Initiator requests all other Switch Agents to estimate the additional losses caused by supplying the area via a different path. These Switch Agents take the roles of Participants. The requests are forwarded from Participant to Participant until reaching the corresponding transformers. Because of the forest structure of the distribution network, these transformers are currently not supplying the area, thus, taking the role

of Passive Transformers. The responses are again forwarded to the Initiator, which decides if a switching action is desirable. If so, it requests closing the switch from the corresponding Participant and opens its own switch. To conclude, the roles identified during the PASSI *Role Identification* phase are the Initiator and Participant for Switch Agents, and the Active Transformer and Passive Transformer for Transformer Agents.

The next phase of the PASSI methodology is the *Task Specification* phase. The resulting task specification diagram for the Switch Agent is depicted in Figure 6.7. It summarizes all tasks required for the Switch Agent to be able to fulfill its roles. The right part of the figure provides a detailed list of all tasks for the Switch Agent, while the left part of the figure only illustrates tasks of other agents that directly interact with switch tasks via message exchange. These can be transformer tasks as well as tasks of other Switch Agents. Each agent implements a Listener task, which dispatches incoming messages. The individual tasks processing the incoming messages are not discussed in more detail, as they largely coincide with the various steps already presented for the roles identification diagram. Also, the PASSI task specification diagram for the Transformer Agent is skipped, as it is much simpler and would not provide any additional insights.

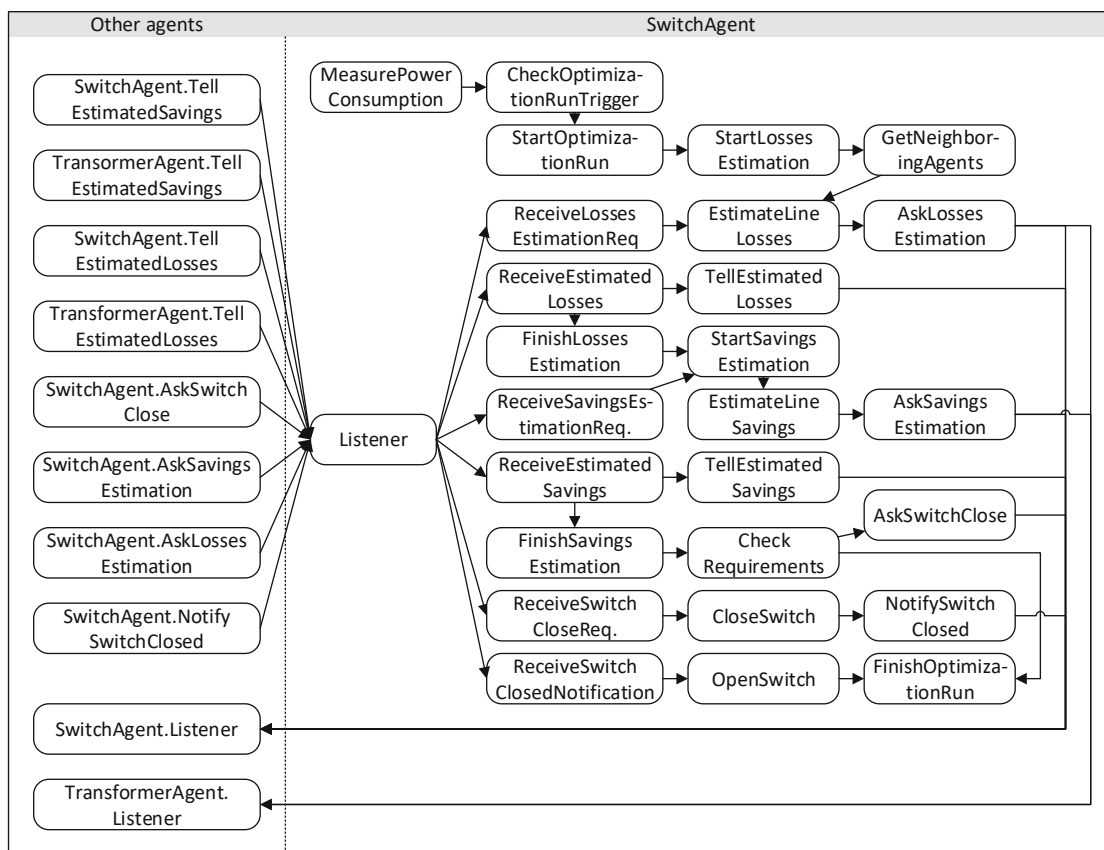


Figure 6.7: PASSI task specification diagram for the Switch Agent

### 6.2.2 Agent society model

Having identified the roles each agent can take, how they interact, and the tasks they have to implement on a very abstract level, the PASSI agent society model refines this information for later use by traditional software development techniques. It starts with the *Ontology Description* phase, which models the information required for operation in the domain ontology diagram and the information required for communication in the communication ontology diagram. The authors of the PASSI methodology provide a detailed example of each of these ontologies but do not suggest a specific methodology [109]. Considering the vast amount of existing ontology design methodologies (cf. Section 2.4), this is a very reasonable approach. While investigating ontologies in the context of MASs, however, it has been found that agents impose a particular requirement on ontology design that is not well-covered by existing methodologies: the reusability of ontologies. Reusability is required because agents that interact in a MAS, such as the Transformer Agent and the Switch Agent, share large portions of their ontologies. However, at the same time, they implement very specific functionality that is not relevant to other agents at all or to the same level of detail. A new ontology design methodology is required to cover these MAS-specific aspects. The methodology is presented in Section 6.3, and the ontology is instantiated in Section 7.2. Therefore, the *Ontology Description* phase is postponed for now.

The next step of the PASSI methodology is *Role Description* phase. The resulting roles description diagram, is depicted in Figure 6.8. Its purpose is to illustrate interactions between roles, and, thus, it is a refinement of the PASSI task specification diagrams (cf. Figure 6.7), which provide similar information but based on agents instead of roles. Each role is represented by a separate class, and the classes belonging to the same agent are grouped into packages. Each role only requires a subset of the tasks specified for the agent. These tasks are listed in the operations compartment of the corresponding role. Furthermore, interactions between different roles of the same agent can now be visualized. For example, the AskSwitchClose request can only be issued by an Initiator Switch Agent towards a Participant Switch Agent.

The interactions between agents are examined in full detail during the *Protocol Description* phase. Thereby, each scenario of the use case analysis is transformed into a sequence diagram or an agent interaction protocol in PASSI/FIPA terminology. The diagram specifying the automated switching optimization scenario is depicted in Figure 6.9. It is inspired by the FIPA Query Interaction Protocol [129] and the FIPA Contract Net Interaction Protocol [130]. Combining these two protocols in a new agent interaction protocol allows ensuring resource reservation (cf. Section 6.1.8). The agent interaction protocol is discussed in the following. Furthermore, it is stated how the agent interaction protocol addresses the functional and non-functional requirements (cf. Table 4.17).

The Initiator starts the optimization run by requesting information about the possible savings from the Active Transformer. It includes the power consumption of its area (and

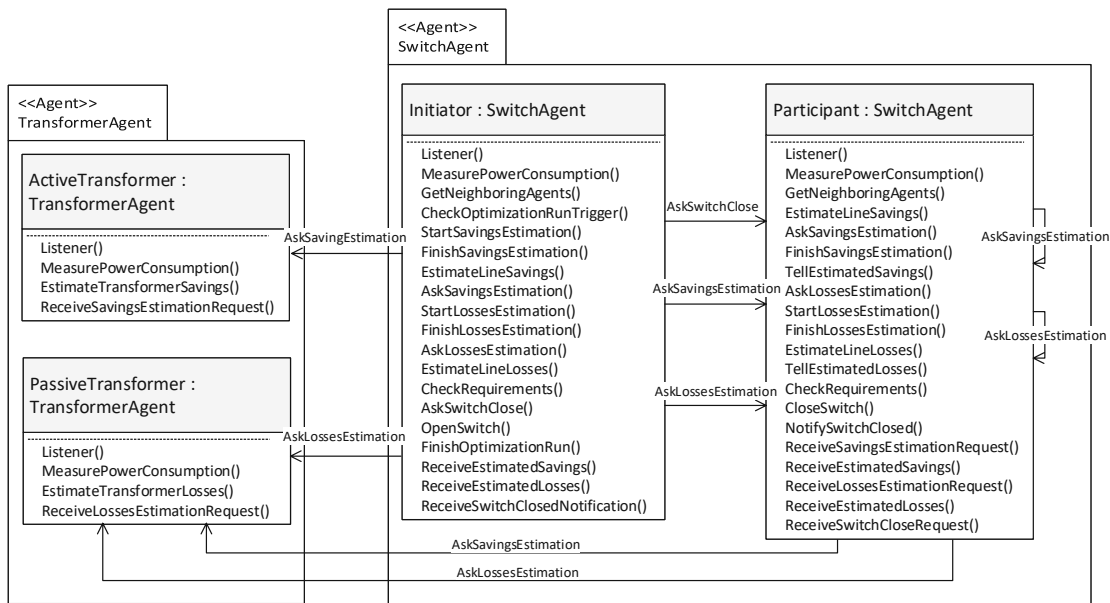


Figure 6.8: PASSI roles description diagram for automated switching optimization

possibly other areas supplied by it) in the request. Additional Switch Agents between the Initiator and the Active Transformer are again omitted. The Active Transformer may *refuse* to answer the request if it is already participating in another optimization run. Otherwise, it informs the Initiator about the transformer losses caused by supplying the load, i.e., the possible savings, using an *inform-ref* message.

The Initiator then sends a *CFP* message to its neighboring Participants. The *CFP* may include non-functional requirements, which allows Participants to refuse making an offer if they cannot meet these requirements. For example, a description of NFR-01 – NFR-08 (Reliability, Safety, Confidentiality, Integrity, Availability, Maintainability, Scalability, Privacy) can be added to the *CFP*. Participants may also *refuse* to propose to the *CFP* in case they are busy, or cannot fulfill a NFR. Otherwise, they forward the *CFP* message until it reaches the corresponding Passive Transformers.

After checking the requirements, each Passive Transformer can send a *propose* message back to the Participant it received the *CFP* message from, or *refuse* to make a proposal. The messages are forwarded to the Initiator, which collects all incoming proposals. Only the best proposal (in terms of possible savings and non-functional requirements) is accepted via an *accept-proposal* message. All other proposals are rejected via *reject-proposal* messages.

With accepting the proposal, the Initiator instructs the corresponding Participant to close its switch. The Participant responds to the Initiator once the switch is closed using

an *inform* message. In the current state of the specification, the Participant cannot reject the request to close its switch after it has made the offer. The Initiator is now safe to open its switch (FR-02: Connectedness and FR-03: Forest structure) and then *informs* the corresponding Participant that the reconfiguration is completed. Upon that, the Participant finishes the conversation corresponding to the open CFP that led to the new configuration via an *accept-proposal* message, which is in turn confirmed with an *inform* message.

The agent interaction protocol is fully distributed, i.e., it does not rely on the existence of any orchestrating agent. Furthermore, the optimization procedure is not implemented in a single agent explicitly but emerges from the interaction of the participating agents. Extending the network does not increase the amount of data a single agent has to process (NFR-07: Scalability). However, the size of areas (i.e., the number of its components) must be kept in a range still manageable by all delimiting agents.

Message loss during the optimization run is a particular problem. If, for example, the final *inform* message did not reach its destination, the optimization run would not complete, and the Transformer Agent could not participate in further optimization runs. Message losses can be reduced by using reliable communication protocols. Furthermore, redundant network interfaces may be used to address the risk of broken communication links. However, ultimately, error-free completion of the agent interaction protocol can never be guaranteed. As a last fall-back mechanism, all agents should use timeouts to detect lost messages, broken communication links, and unresponsive communication partners and start a recovery procedure.

As mentioned, the agents may include statements about non-functional requirements in their messages. For this purpose, they require a way to communicate via semantically meaningful messages, which can be achieved using ontologies. Furthermore, they use this information to select the best reconfiguration option among multiple possibilities.



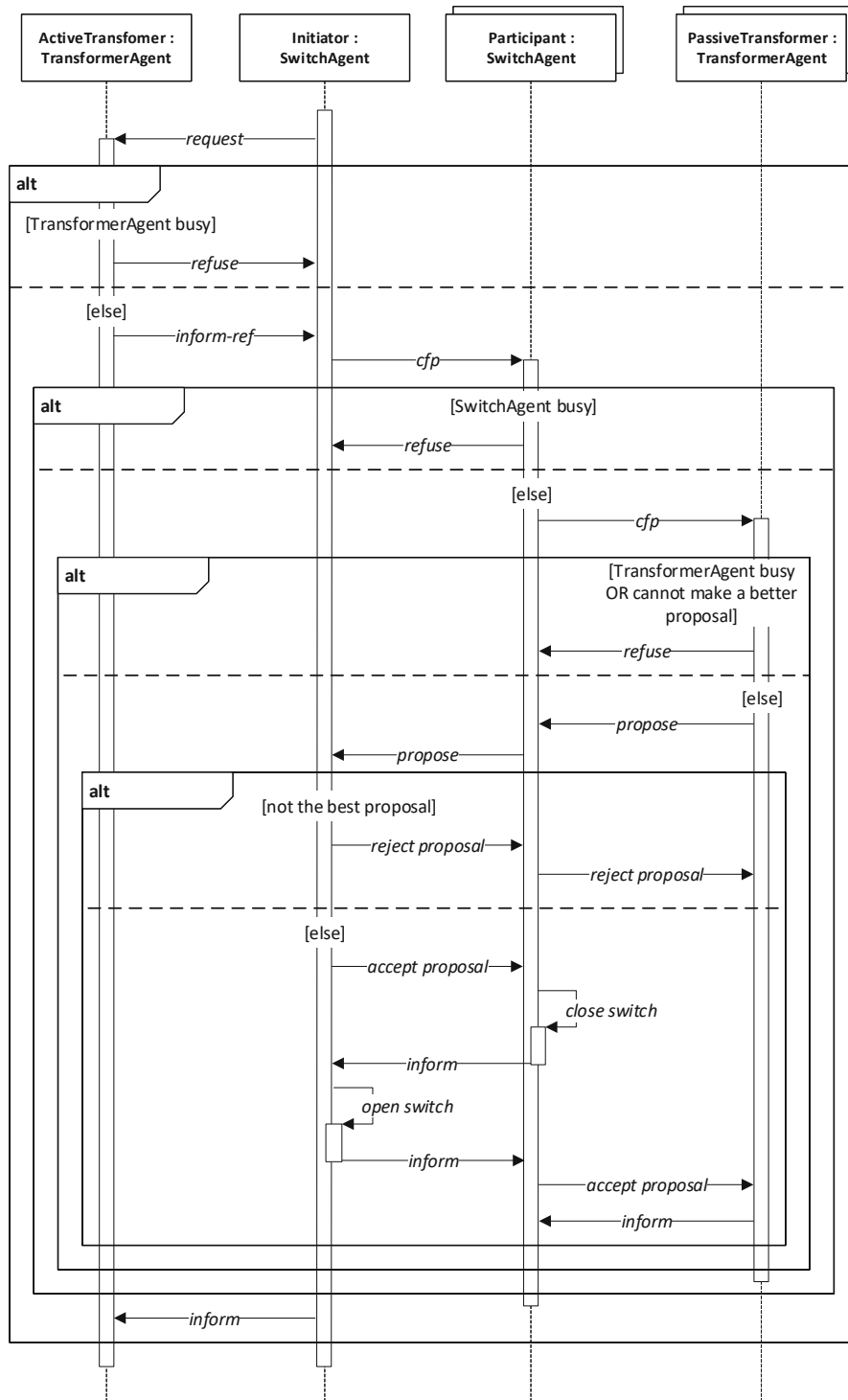


Figure 6.9: FIPA agent interaction protocol diagram for the automated switching optimization

## 6.3 Ontology design

Most ontology design methodologies target the creation of a single ontology covering the complete application scenario. These methodologies typically follow a step-wise approach, which often motivates the use of existing ontologies but, on the other hand, does not support the generation of reusable ontologies. The probably best-known example of this approach is “Ontology Development 101: A Guide to Creating Your First Ontology” [21]. It describes a methodology for ontology design structured into seven simple steps (cf. Section 2.4.1). “A lightweight methodology for rapid ontology engineering” [144] is another (six-step) approach falling into this category of ontology design methodologies.

Later on, ontology design methodologies were developed that start with a small or even empty ontology and add other ontologies and concepts incrementally. Such an iterative methodology is presented in “An Incremental and Iterative Process for Ontology Building” [237]. Again, reusing existing ontologies and other sources is a core aspect of the ontology design but deriving reusable ontologies along the way is not of particular importance. This is addressed by the reusable ontology design methodology presented in the following.

### 6.3.1 Reusable ontology design methodology

A methodology that focuses on deriving reusable ontologies as well as reusing existing ontologies is illustrated in Figure 6.10. It follows a “divide and conquer” or “divide and combine” approach using different levels. At the very beginning, the application-specific ontology is created, but no concepts are added yet. Instead, it is divided into smaller (Tier 1) ontologies, each intended to cover a well-defined subset of the required information. In general, Tier  $n$  ontologies are again divided into Tier  $n + 1$  ontologies until reasonably small ontologies are derived.

Tier  $n + 1$  ontologies are then combined (imported to) Tier  $n$  ontologies and extended with concepts that are not present in the Tier  $n + 1$  ontologies already but arise from their combination. Individual basic Tier  $n + 1$  ontologies, e.g., such modeling timing or unit concepts, may thereby be imported in multiple Tier  $n$  ontologies. Others might not be used at all, e.g., because they were found to be redundant or too complex. Combining ontologies is continued until the application-specific ontology is derived. Ontologies on each level should be self-contained to allow them to be reused in ontologies for other applications. The individual steps of the methodology are briefly discussed in the following. They can be documented, for example, using a tabular or graphical notation and adding information along the way.

#### Methodology step: divide

At the beginning of the methodology, the application-specific ontology is recursively divided into smaller parts. This process stops for a specific Tier  $n$  ontology for one

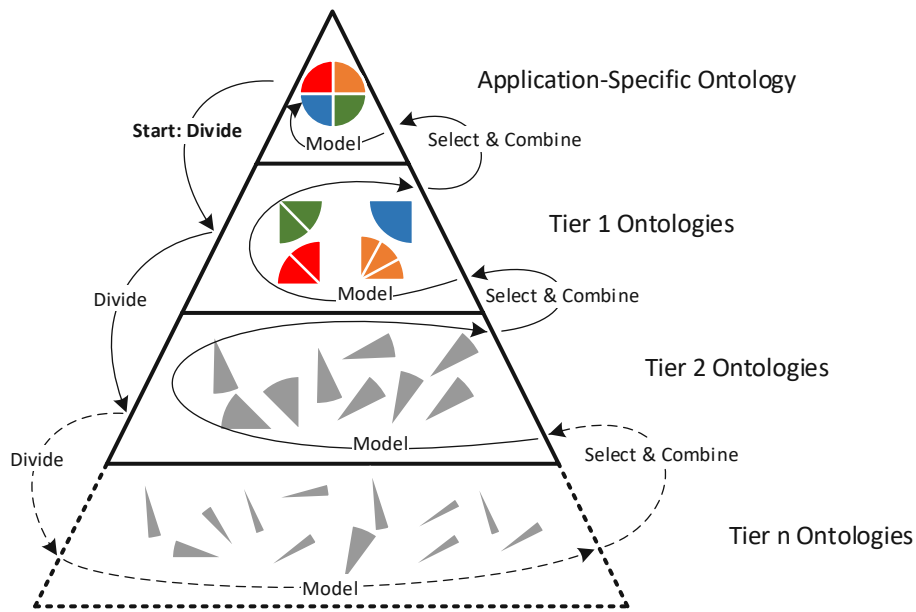


Figure 6.10: Methodology for reusable ontology design

of the following three reasons: (1) it is already covered by an existing ontology, (2) it is already covered by another source of information (e.g., books and papers), or (3) it cannot reasonably be divided any further. The quality of each potentially useful source of information is thereby rated and documented. According to the reason why the process stopped at a certain point, (1) the ontology is either used directly or (2) & (3) the ontology is created following standard ontology modeling techniques.

The divide step should put a strong focus on reusing existing ontologies and other sources of information. Thus, developers should analyze available material and choose their sub-ontologies accordingly. Thereby, information already being available as an ontology and probably in some standardized ontology document format is preferred over other kinds of information. Nevertheless, developers should not completely ignore other sources of information and consider including them in this step. In the following, some valuable sources for existing ontologies and additional valuable information are listed:

- *Conventional search engines:* Conventional search engines mainly focus on human-readable information. Nevertheless, they often include all types of online-available documents in their search results and even provide options to search for specific file types (e.g., \*.rdf or \*.owl).
- *Ontology search engines:* On the other hand, there are search engines specifically focusing on ontologies. Their benefit lies in providing meta-information about the ontology itself, e.g., the number of classes, properties, and individuals contained.

- *Scientific publications*: Ontologies still are a somewhat scientific way of structuring information, and the concepts many ontologies are built upon are published in papers and theses. Sometimes, these publications refer to the corresponding online-available ontology document.
- *Databases*: Databases and, in particular, their structure often may directly be translated to ontologies or at least provide a good starting point.
- *Human readable documents*: Finally, plenty of information about all different kinds of domains is available in standards, specifications, books, articles (scientific or otherwise), company-internal documents, diagrams, or similar sources of information. Often, this information is structured already and can easily be converted into an ontology. A typical example are existing UML diagrams.

The exact number and types of ontologies on each level depend on the application's complexity and on the different views developers might have on their specific problem. Once reasonably small sub-ontologies are derived, they are not divided any further but the model step and the select & combine step are applied alternately on each level.

#### **Methodology step: model**

Existing ontologies do not need to be re-modeled. However, in many cases, the required information is not available in a standard file format. Existing ontology modeling techniques (cf. Section 2.4) are used to generate ontologies from this information and model new concepts that are not found in any existing source. Besides, concepts that emerge from selecting and combining Tier  $n + 1$  ontologies are added in this step. The model step is performed on each level of the methodology and alternates with the select & combine step until the application-specific ontology is derived.

#### **Methodology step: select & combine**

In this step, Tier  $n + 1$  ontologies are analyzed and rated to select suitable subsets that are then combined to Tier  $n$  ontologies. Thereby, the rating does not follow a strict, predetermined scheme but is subject to the developer's experience. Multiple considerations have to be taken into account:

- Comprehensive ontologies reduce the effort of implementing missing concepts.
- On the other hand, concepts that are not used by the application cause unnecessary overhead in memory consumption and processing power.
- Preferably, existing sources should be well known and well accepted by the corresponding domain experts.
- In general, ontologies already used in other applications are preferred to others.
- Tier  $n + 1$  ontologies may contain information that is relevant for multiple Tier  $n$  ontologies.

Unfortunately, selecting the “best” subset of ontologies is not a straightforward procedure and – taking the required effort into account – in most cases, it is not practicable. However, results from the previous analysis provide a sound basis. The preferable approach for selecting and combining Tier  $n + 1$  ontologies is to select each ontology only if it adds considerable value to the corresponding Tier  $n$  ontology. It is thereby necessary to re-evaluate each ontology’s rating after each step to avoid the selection of multiple, similar ontologies. After having selected and combined all relevant Tier  $n + 1$  to Tier  $n$  ontologies, the methodology continues with the model step on the Tier  $n$  level.

### Brief discussion on the reusable ontology design methodology

The reusable ontology design methodology allows to create application-specific ontologies faster and easier by strongly encouraging the reuse of existing work, not only in the form of existing ontologies but also including books, articles, specifications, and standards. Furthermore, each of the designed ontologies generated while following the procedure, except for the application-specific ontology, focuses on being reusable by other developers or other projects. An obvious drawback of this approach is that the resulting, application-specific ontology might contain more classes, individuals, and properties than required for the specific use case. This causes additional resource consumption for storing and processing, which has to be considered if resource-constrained devices need to store these ontologies.

#### 6.3.2 Reusable ontology design

The exact number of levels of the methodology presented in the previous section is not predetermined and may also be fixed while following the methodology. The switching optimization MAS ontology uses a hierarchy of three types of ontologies, as illustrated in Figure 6.11. While the application-specific ontology keeps its name, Tier 1 ontologies are referred to as *domain ontologies* or *domains*, and Tier 2 ontologies are referred to as *fragment ontologies* or *fragments*. The individual steps of the methodology are documented using a graphical notation.

##### A. Divide application-specific ontology

This first step aims to reduce the complexity of the application-specific ontology by separating it into multiple domains. The use case description of the switching optimization use case provides a good starting point to divide the application-specific ontology into smaller domain ontologies:

- *Agents and services*: Agents in the sense of MAS-theory typically have a specific structure and several attributes like a name and possibly multiple addresses. They are often closely related to the services they offer and are, therefore, part of the same domain. Services themselves also may have multiple attributes like a name and additional information required to invoke the service.

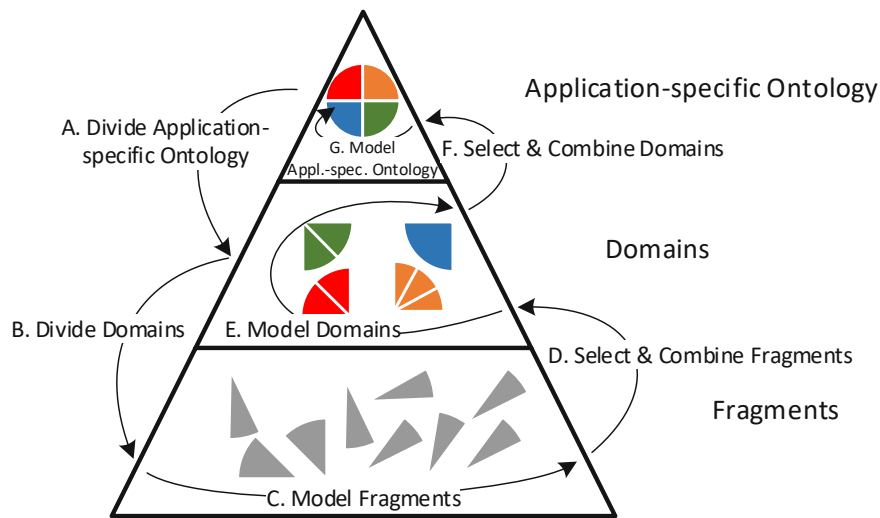


Figure 6.11: Three-level methodology for the switching optimization MAS ontology

- *Power system components:* As the intended MAS operates in the SG-field, power system components form the most crucial part of the agents' environment. Nevertheless, they are not directly related to agents and services but merely one field of application where MASs can be applied. Therefore, they form a separate domain.
- *Communication protocols:* Communication protocols form another domain because they are used not only by agents but also by other devices in the IoT.
- *Rating:* Due to the openness and flexibility of MASs, there often exist many different possibilities to achieve the same goal. The various possibilities need to be comparable to choose the best option, which can be achieved via a rating ontology. The rating domain includes QoS and dependability attributes. Furthermore, it provides concepts to assign these attributes to individuals of the other domains, e.g., the availability of a service, the reliability of an electric switch, or some communication protocol's integrity features.

As a starting point, the four identified domains are arranged in a simple graph, as shown in Figure 6.12. This graph is complemented with additional information along following the process.

### B. Divide domains

The domain ontologies are now divided into fragments. The following list briefly discusses possible sources for each domain and their fragments:

- *Agents and services domain:* Despite being a major field of application for ontologies, there are not many ontologies specifically focusing on MASs and agents

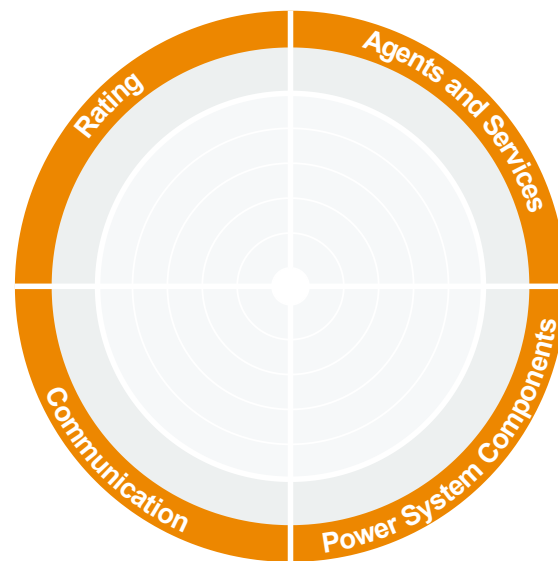


Figure 6.12: Step 1: Identifying and adding domains to the graph

on a meta-level. One example is the Onto2MAS framework [238], Donzelli et al. define an agent primarily based on its behavior and its role within the MAS, which is a reasonable approach for their intended use case of generating MASs directly from information given within the ontology. As most publications about MASs in the SG-field do, Donzelli et al. themselves refer to the FIPA specifications for specific functionality, e.g., FIPA-ACL [127] for specifying the encoding of messages exchanged between agents.

Besides the ACL, FIPA also gives a quite general description of agents in their FIPA agent management specification [125]. Although not being available in standard file formats, in this specification FIPA introduces the `fipa-agent-management` ontology, which defines the basic components of a FIPA-compliant MAS. Unfortunately, the ontology itself is not very sophisticated and would impose severe limitations if it were directly translated to an RDF or OWL ontology: most parameters are specified to be of type `string` or `set of string`. Considering, for example, the languages supported by a service, creating dedicated individuals that represent the corresponding languages and referencing them in the ontology is beneficial over listing the supported languages as a `set of strings`.

The concept of agents can also be found in other ontologies, such as the COMmon Semantic MOdel (COSMO) [239], the FIESTA-IoT [240] ontology and the DOLCE+DnS UltraLite (DUL) [241] ontology. However, they do not specify the inner structure of an agent as detailed as FIPA does and are therefore not very suitable for the present use case.

- *Power system components domain:* The CIM is probably the best-known model for power system components and very comprehensive. The standard document IEC 61970-501 [81] specifies the RDFS format for CIM.

The Prosumer-Oriented Smart Grid (ProSG) ontology [242] has a stronger focus on the consumer level and is, therefore, less comprehensive regarding power system components. However, it might be a valuable source once the example use case is extended towards DERs.

- *Communication domain:* A core aspect of any MAS is the cooperation between agents, which heavily builds upon communication. While application layer protocols are often designed for a specific MAS or AP, e.g. FIPA, they typically build upon existing lower-layer protocols like Transport Layer Security (TLS), Transmission Control Protocol (TCP), and Internet Protocol (IP), and corresponding Link layer protocols. In an open MAS, knowledge about the communication network topology and which protocols are supported by each agent is required to select a suitable communication partner. A typical source for such information are ontologies about the IoT or the Internet in general, like the IoT Network ontology, described in [243].

Another valuable starting point for an ontology about communication protocols is provided by Jablonski's "Guide to Web Application and Platform Architectures" [244]. Therein, Jablonski provides a comprehensive classification of Internet standards and technologies. The root classification entities are Interaction, Logic, Security, Data Management, Description, Presentation, Export/Import Interface, and Platform Software. These root classification entities are then further subdivided into Category Bags (CBs). The author states that the classification is incomplete and can be extended by additional CBs. Finally, many existing Internet standards and technologies are assigned to each CB, and additional items (e.g., upcoming protocols) can easily be added.

- *Rating domain:* In MASs, agents may offer their services via multiple interfaces. Therefore, some form of rating scheme is required to select the most appropriate interface. For this reason, FIPA specifies the fipa-qos ontology [245].

In regards to W3C-compliant ontologies, the Quality of Service Modeling Ontology (QoS-MO) is a very sophisticated approach [246]. It includes a variety of QoS and dependability attributes and provides concepts for rating these attributes, enabling agents to select a suitable service, e.g., based on their availability values.

When dealing with critical infrastructure, traditional QoS attributes are often extended by aspects targeting the notion of dependability as defined by Avizienis,



Laprie, Randell, et al. [6]. Their structured view on dependability in the form of a dependability tree and the corresponding statements about the relationships between dependability attributes, means, and threats provide an excellent starting point for an ontology about dependability.

A separate ontology is required to assign ratings (values) to the various non-functional attributes like dependability attributes. This ontology should combine and generalize information from various sources and domains into a comprehensive collection of metrics and scales.

The various ontologies and other sources of information identified, i.e., the fragment ontologies, can now be added to Figure 6.12. The result is depicted in Figure 6.13.

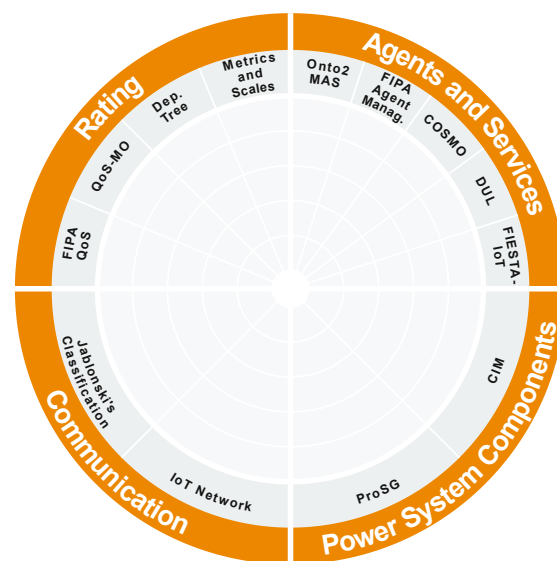


Figure 6.13: Step 2: Adding fragments to the graph and assigning them to domains

### C. Model fragments

Identified fragments that are not already available as ontologies in any standardized ontology file format are modeled using existing ontology design methodologies (cf. Section 2.4). The Ontology development 101 methodology was used in this thesis. It is sufficiently sophisticated for this purpose, as the complexity of the application-specific ontology has already been reduced by separating it into domains and fragments.

The ontologies derived until this point are not at all specific to the use case but reflect small portions of the identified domains in very abstract and general ways (e.g., attributes, threats, and means of dependability; main elements of an agent; known IoT communication protocols; power system components). This increases the likelihood of deriving

fragment ontologies that are also useful for researchers beyond the exact same field of expertise, e.g., someone who specialized in MASs for production facilities or developers who want to extend their ontologies with dependability concepts. The fragments that had to be modeled in the course of this thesis are discussed in the following. For all other fragments, the reader may refer to the sources provided in the previous section.

- *FIPA agent management fragment:* The FIPA agent management ontology is depicted in Figure 6.14. It is based on the formal definition provided in the FIPA Agent Management Specification [125]. Its intended purpose is to describe each agent in the Directory Facilitator, hence the name of its main class: DfAgentDescription. Information about the services offered by each agent is modeled via the ServiceDescription class. Among various other attributes, the specification already foresees the possibility to “discriminate” between multiple services (possibly offered by different agents) via the Property class, whereby property is not specified any further at this point. Although the original specification does not foresee this, the same concept can also be applied to the DfAgentDescription class, which allows adding properties to the agent description. Additional information about the remaining concepts illustrated in Figure 6.14 is provided by the specification [125].

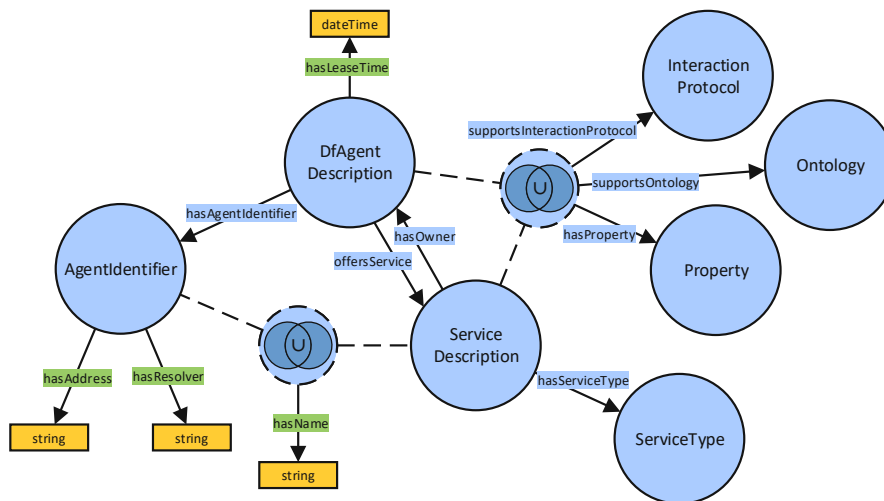


Figure 6.14: FIPA fragment ontology

- *Dependability tree fragment:* The dependability tree ontology is depicted in Figure 6.15 and based on [6]. A description of the various threats, attributes, and means is provided in Section 1.2. The illustrated ontology deviates from the original dependability tree (cf. Figure 1.4), as there are no connections between the various aspects to a central Dependability class. This is because there is no pre-defined property available in OWL that would match the required semantics, e.g. *isAspectOf*. Furthermore, such a property would not add any useful information and it is therefore not added.

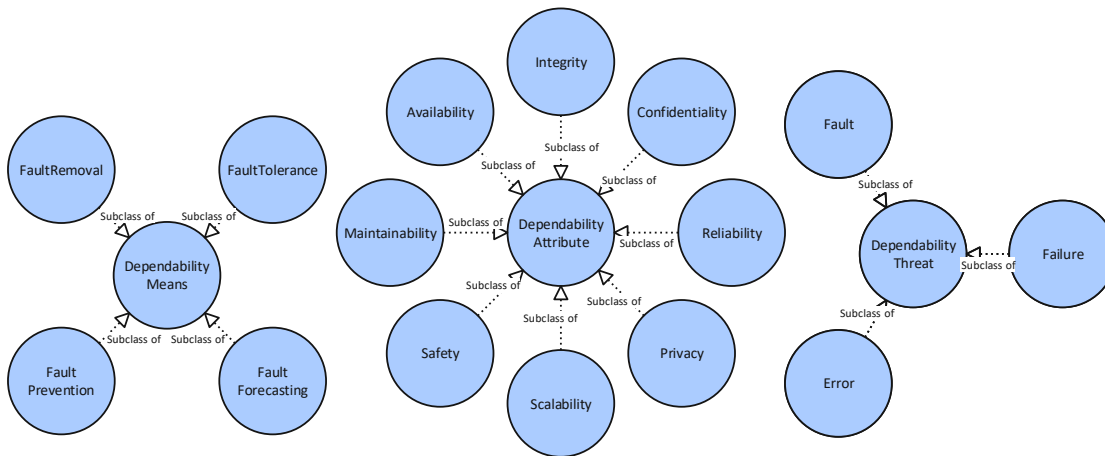


Figure 6.15: Dependability tree fragment ontology

- Metrics and Scales fragment:* The metrics and scales ontology depicted in Figure 6.16 enables the rating and comparison of non-functional attributes. It is designed to be used along with other ontologies that define the meaning of these attributes. The `RateableAttribute` is used as a base class in this ontology. However, this class should not be used as a type for individuals but an appropriate sub-class should be chosen. Thereby, the following modeling convention is used and also reflected in the naming scheme: the first level of sub-classes defines the nature of the attribute, and the second level of sub-classes defines the nature of the value that is used to rate this attribute.

A `NominalAttribute` can take one of several, typically non-overlapping nominal values. Following the naming convention, the value of a `CategoricalNominalAttribute` represents a category (e.g., “LV”, “MV”, “HV”). A `DichotomousAttribute` can only take one of exactly two categorical values and, therefore, is a further specialization of a `CategoricalNominalAttribute`. There is no intrinsic order within nominal values, i.e., one value cannot be regarded better or worse than another.

An `OrdinalAttribute` is defined as an attribute taking values that can be ordered. `LiteralOrdinalAttributes` take literal values that can be ordered in a relative manner to each other. Typically, these literal values are of some numeric data type. `CategoricalOrdinalAttributes` take `OrdinalCategoricalValues`, which are values that are represented by categories that can be ordered (e.g., “very good”, “good”, “medium”, “bad”). While `OrdinalAttributes` can be ordered relative to each other, there is no equal distance between successive values, i.e., there might be only a negligible small difference between “very good” and “good”, but a larger difference between “medium” and “bad”.

The definition of an `IntervalAttribute` adds the possibility to model the concept of equal distances, i.e., the distance of two `IntervalAttribute` values is meaningful.

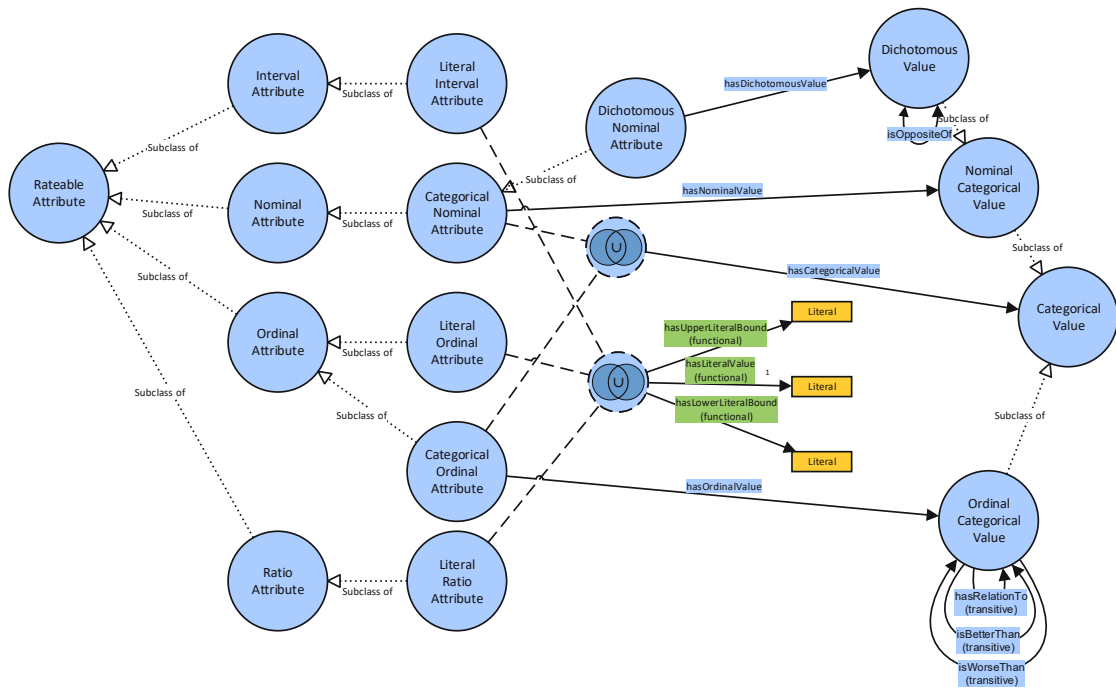


Figure 6.16: Metrics and scales fragment ontology

For example,  $40^{\circ}\text{C}$  and  $20^{\circ}\text{C}$  have the same distance as  $80^{\circ}\text{C}$  compared to  $60^{\circ}\text{C}$ . However, while the distance is meaningful, most mathematical operations, in particular multiplication, are not. This characteristic can be traced back to the lack of an absolute/true zero in interval scales. For example, by the definition of `IntervalAttributes`,  $40^{\circ}\text{C}$  is not twice as warm as  $20^{\circ}\text{C}$ , as the Celsius scale lacks a true zero.

For this reason, the `RatioAttribute` is introduced as a final class that adds the possibility to model scales with true zeros. The `LiteralRatioAttribute` takes literal values. For example, a load of 2 kW electrical power consumes twice as much power as a load with only 1 kW and the same as four loads with 500 W each.

#### D. Select & combine fragments

For selecting possibly relevant sources of information, it is useful to first collect and rate them according to their suitability for the domain. The rating for each fragment has been added to Figure 6.13, resulting in Figure 6.17. The height of each bar, starting from the center of the diagram, directly corresponds to the associated rating, i.e., high bars  $\hat{=}$  high ratings, low bars  $\hat{=}$  low ratings. For example, the `Onto2MAS` framework does not cover agents' inner structure to the same level of detail as the `FIPA` agent management specification and is, therefore, rated lower.

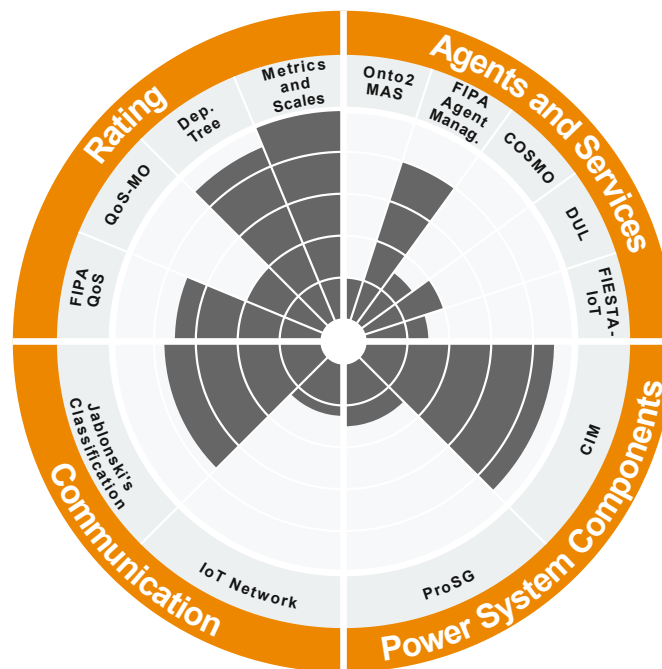


Figure 6.17: Step 3: Adding fragment ratings to the graph

Ontologies that are excluded based on their low ratings are removed from Figure 6.17, resulting in Figure 6.18. The remaining fragments are combined by importing them to the corresponding domain ontologies. It is thereby without problems to use the same fragments for multiple domain ontologies.

In this example, the Agents and services, Power system components, and Communication protocol domains consist of only one fragment. Thus, these fragments can be imported into their corresponding domain ontology, and no additional domain modeling is required. The dependability tree fragment and the metrics and scales fragment have to be imported for the rating domain. The concepts that establish the relations between both fragments are modeled in the next step.

### E. Model domains

The Dependability tree fragment and the Rating fragment shall now be combined. Appropriate rating metrics for each dependability attribute have been identified in the dependability requirements analysis phase (cf. Section 5.2). The combination of both fragments and additional concepts are illustrated in Figure 6.19. Most importantly, the DependabilityAttribute is declared a subclass of RateableAttribute, i.e., each DependabilityAttribute is also a RateableAttribute. The corresponding metrics of each DependabilityAttribute are discussed in the following.

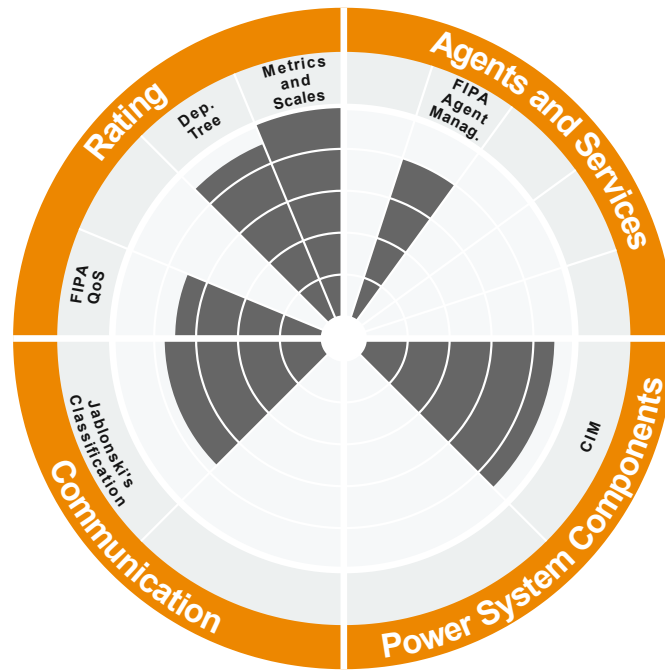


Figure 6.18: Step 4: Removing unused fragments from the graph

Reliability, maintainability, and availability are all probabilistic values. Availability is defined by a simple probability value. Reliability and maintainability are defined by a PDF. The PDF can easily be converted to a complementary CDF. Neither of these metrics has been defined in the Rating fragment ontology. Thus, they need to be added to the rating ontology domain. The matter is relatively straightforward for availability: the Probability class is simply defined as a subclass of LiteralRatioAttribute. The availability can, therefore, be defined by the hasLiteralValue property. For reliability and maintainability, the hasFunctionalValue property is introduced. It allows specifying a function, e.g., a PDF or CDF, that has to be evaluated to derive the corresponding value.

As mentioned in Section 5.2.1, a Weibull PDF is a suitable mathematical model to specify reliability and maintainability. It is defined by Equation 6.1 and, in its standard form, requires two parameters:  $k$  and  $\lambda$ . Furthermore, the time offset  $\Delta t$  has been added, allowing to take the installation time of a component into account. The corresponding properties hasK, hasLambda, and hasTimeOffset are therefore added to the rating domain ontology.

$$f(t) = \begin{cases} \frac{k}{\lambda} \left( \frac{t-\Delta t}{\lambda} \right)^{k-1} e^{-((t-\Delta t)/\lambda)^k} & t - \Delta t \geq 0, \\ 0 & t - \Delta t < 0, \end{cases} \quad (6.1)$$

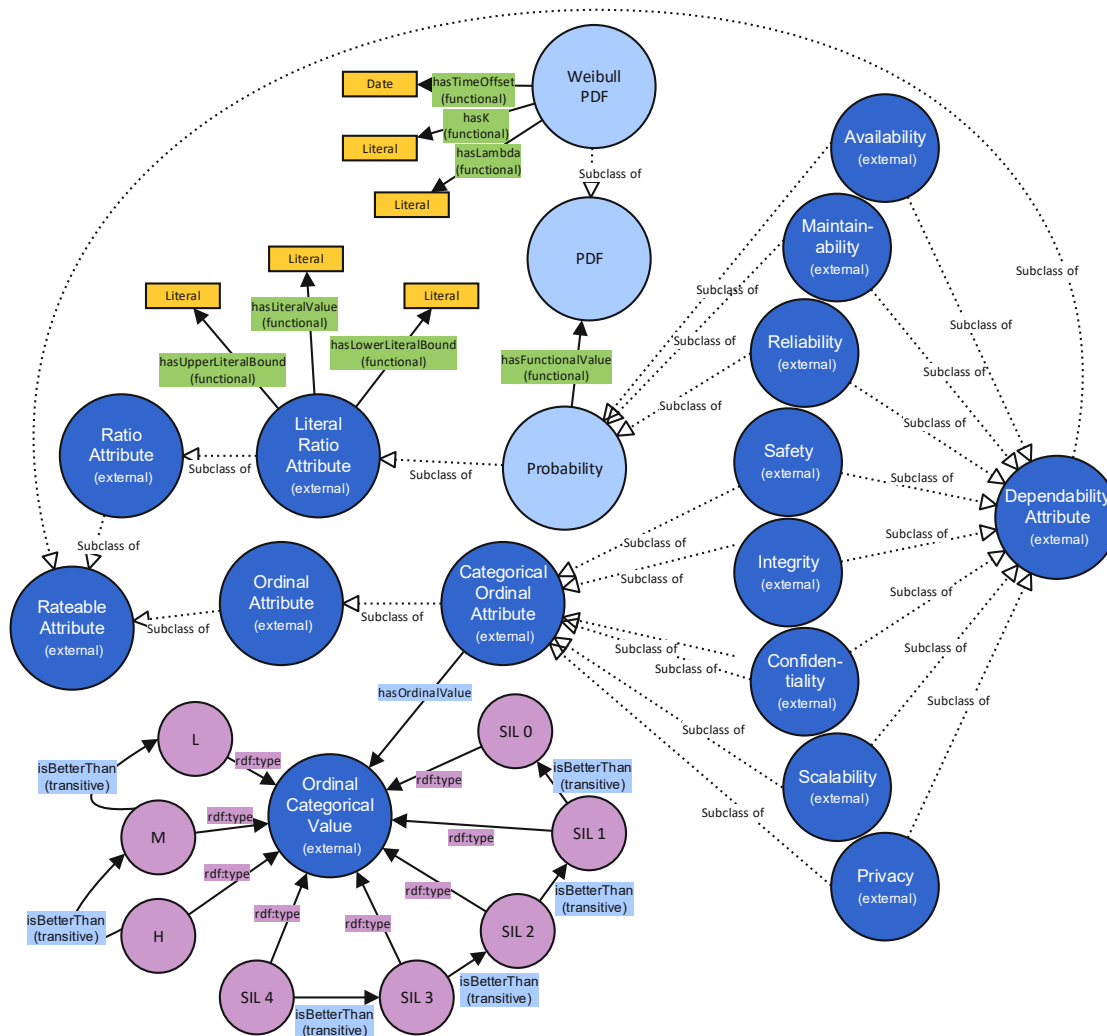


Figure 6.19: Rating ontology domain

Safety, scalability, privacy, confidentiality, and integrity are each modeled as `CategoricalOrdinalAttribute`. For all attributes except safety, H, M, and L are used as `OrdinalCategoricalValues`. Their meaning is defined in Section 5.2. Naturally, H is `isBetterThan` M, and M is `isBetterThan` L. Similarly, the SIL levels defined in IEC 61508 [50] (SIL 1 to SIL 4) are modeled as `OrdinalCategoricalValues` for safety, whereby SIL 4 is `isBetterThan` SIL 3, etc. Additionally, SIL 0 is introduced to indicate that no safety evaluation has been performed for the rated individual and, thus, no safety guarantees can be given.

The agent and services domain ontology is illustrated in Figure 6.20. The FIPA specification and, therefore, the FIPA fragment ontology allows to assign contact information to Agents via the `hasAddress` property and the `hasResolver` property. However, this

approach is limited as it does not support assigning these addresses or other properties to specific communication interfaces of the agent like LTE, LoRaWAN, or power line communication [247]. For this purpose, the CommunicationInterface class is added in the Agents and services domain ontology illustrated in Figure 6.20. The hasAddress property is used to assign addresses to CommunicationInterfaces. The hasProperty property allows model properties of CommunicationInterfaces and, therefore, enables the possibility to discriminate between multiple CommunicationInterfaces.

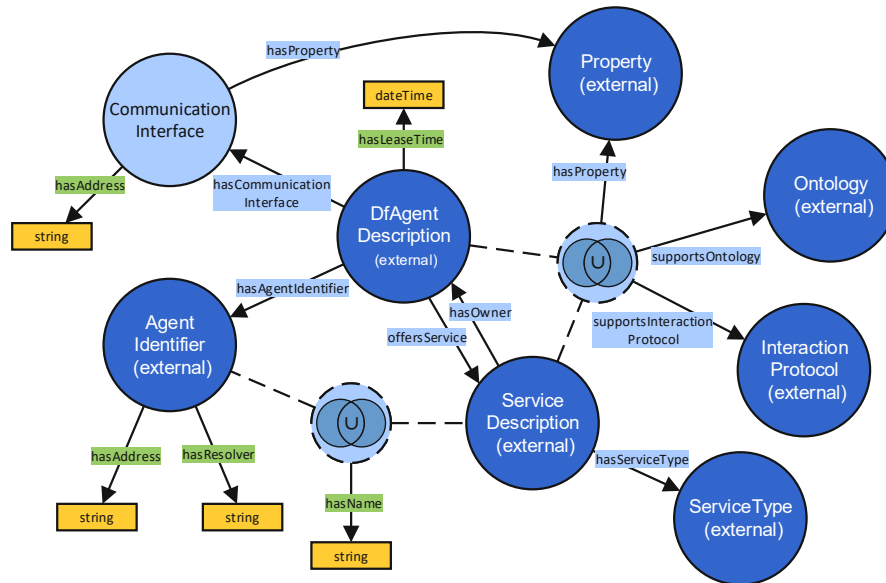


Figure 6.20: Agents and services domain ontology

## F. Select & combine domains

All of the identified domain ontologies are selected for the automated switching optimization application. Therefore, they are combined and form the application-specific ontology.

## G. Model application-specific ontology

Information about interdependencies between the individual domains is now added to the application-specific ontology. The main concepts are illustrated in Figure 6.21. Each Property discriminates a Service, an Agent, or a CommunicationInterface and, as such, is a RateableAttribute. Therefore, the Property class is declared as a subclass of RateableAttribute. Furthermore, as this is the application-specific ontology, the AutomatedSwitchingOptimization literal is added as InteractionProtocol, and the fragment and domain ontologies are added. Finally, the LossEvaluation and Switching services are also added to the ontology. This completes the ontology design, and the application-specific ontology is now ready to be instantiated for the individual agents.



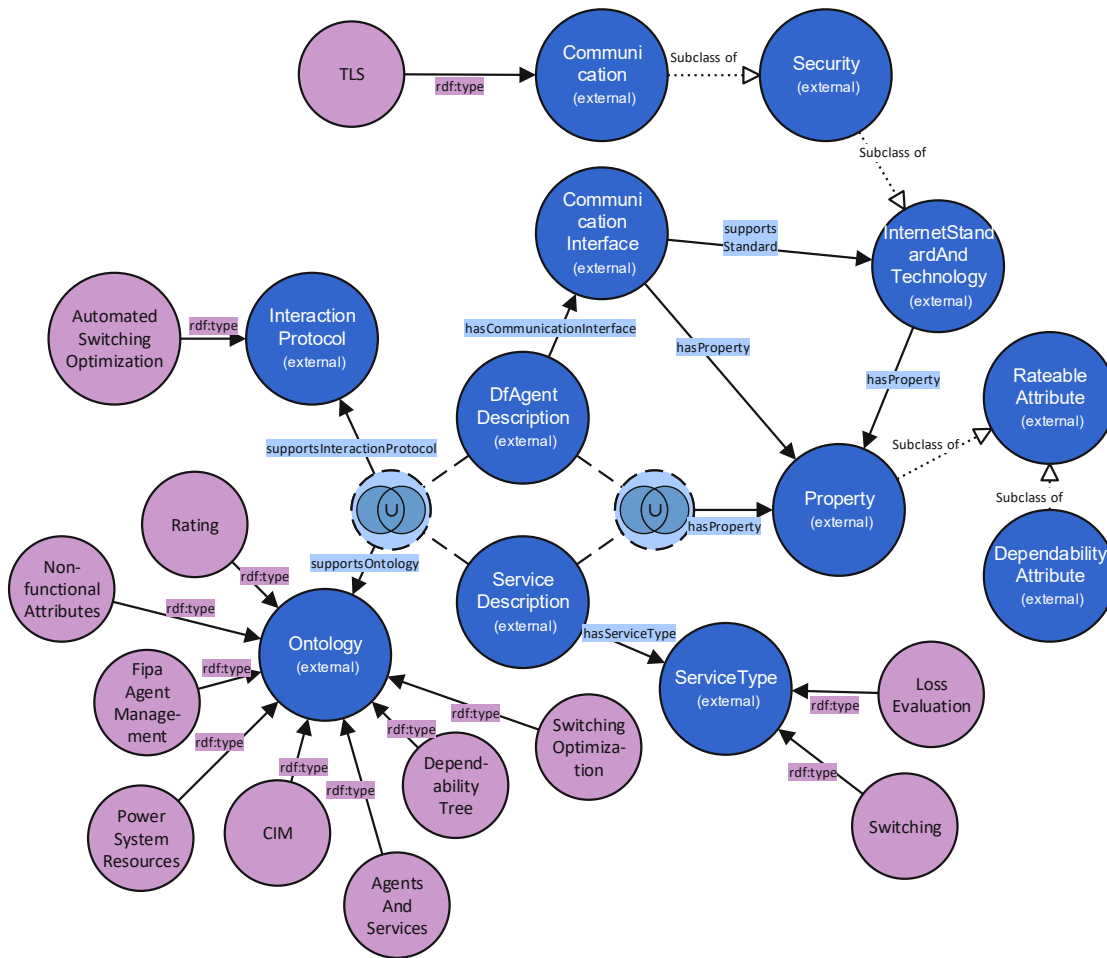


Figure 6.21: Switching optimization application-specific ontology



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

# Implementation

The implementation phase consists of the agent implementation and ontology instantiation activities, as depicted in Figure 7.1. Both activities are mostly independent of each other and can be conducted in parallel. The agent implementation activity builds upon the PASSI agent implementation model and code model. During the ontology instantiation activity, the ontology created in the previous phase is instantiated for each agent. Thereby, the required individuals and literal values are added. For non-functional attributes, ontology instantiation is exemplified based on the switching service and a communication protocol for invoking this service.

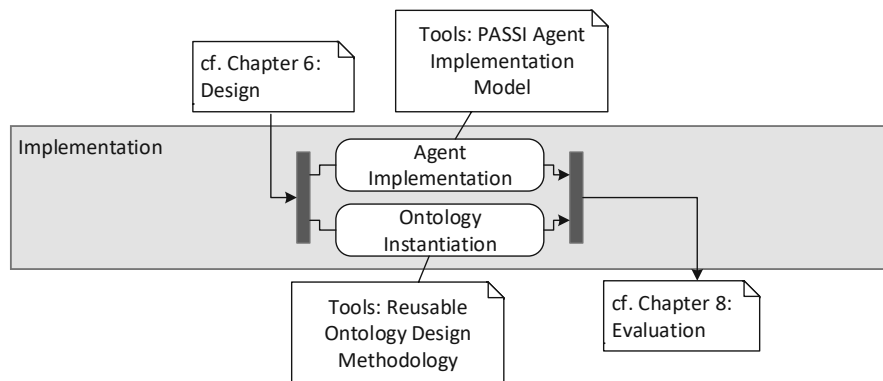


Figure 7.1: Implementation phase: activities and tools

## 7.1 Agent implementation

Agent implementation directly builds upon the results of the MAS design activity and refines them to create executable software. Thereby, the focus is primarily on developing

the individual agents and the algorithms they execute, rather than on implementing services like agent registration and discovery. These services are either provided by an existing AP (cf. Section 2.3.5) or need to be developed and implemented separately.

### 7.1.1 Agent implementation model

The PASSI methodology continues with the agent implementation model that includes several diagrams to define the structure and behavior of the overall MAS and individual agents. Thereby, the *Agent Structure Definition* phase and the *Agent Behavior Definition* phase are closely intertwined and typically conducted in parallel. The agent implementation model closes the gap between MAS development and classical software development.

The agent structure definition results in a MASD diagram for the overall MAS and multiple SASD diagrams (one per agent). The MASD diagram for automated switching optimization is depicted in Figure 7.2. It builds upon the roles description diagram of Figure 6.8. However, from a software development point of view, there is no need to separate the various agent roles, as they will be implemented in a single class. Thus, each agent is represented by a class in the MASD diagram, subsuming all the various roles it can take.

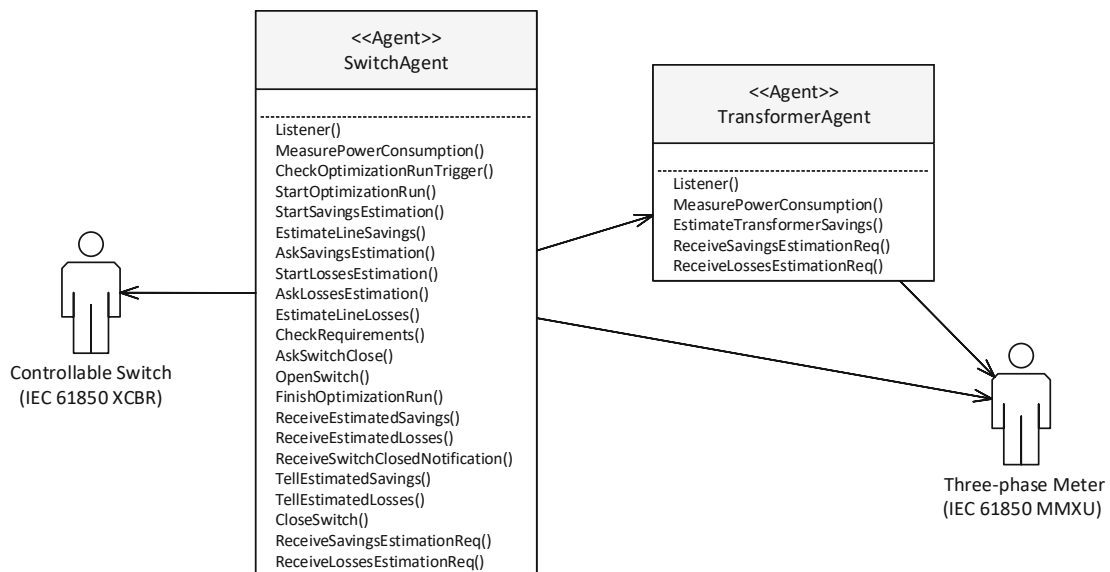


Figure 7.2: PASSI MASD diagram for automated switching optimization

Additionally, external systems the agent interacts with are added as actors. The Transformer Agent and the Switch Agent receive data from a three-phase meter (e.g., an IEC 61850 MMXU measurement node), which measures the currents and voltages on

each phase. Additionally, the Switch Agent interacts with the controllable switch (e.g., an IEC 61850 XCBR circuit breaker node) to determine and control the switch state.

The SASD diagram for the Switch Agent is depicted in Figure 7.3. It defines a class for the Switch Agent itself as a subclass of the more general FIPA\_Agent class. Additionally, the PASSI methodology suggests defining one class per task. The necessary attributes and internal methods are defined for each task. For example, the CheckOptimizationRunTrigger task requires a powerConsumptionHistory array to be able to detect changes in power consumption and trigger the optimization run, as defined in Section 6.1.7. The CheckOptimizationRunTrigger() is the class constructor and the StartTask() method actually performs the task. All other tasks listed in the MASD diagram are added to the SASD diagram in the same manner. The SASD diagram is well-suited for automated code generation if the intended programming language supports OOP. Otherwise, it is still a valuable means of documentation.

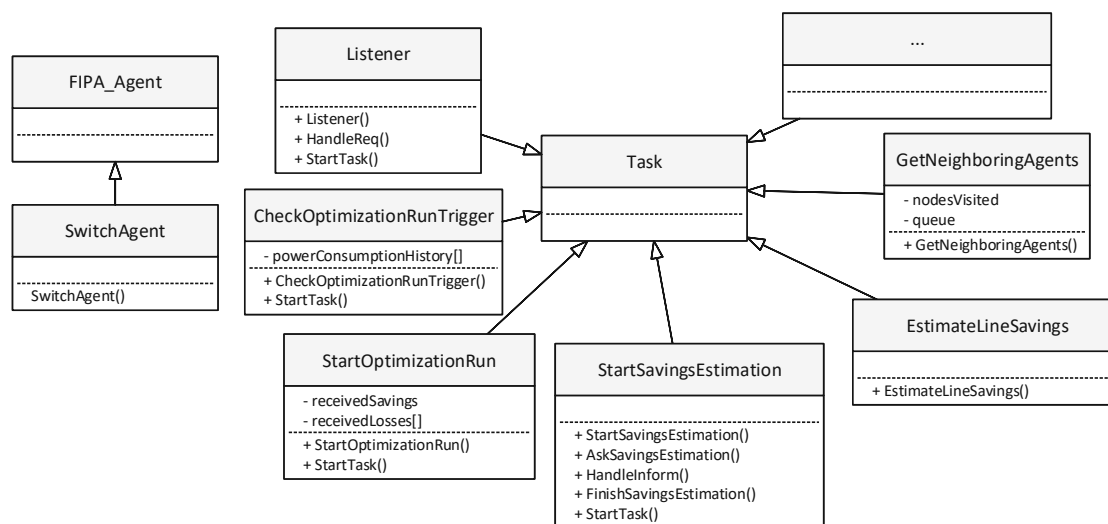


Figure 7.3: PASSI SASD diagram for the Switch Agent

The interaction between various classes within each agent and among multiple agents are examined in more detail using one or several MABD diagrams. They are intended to illustrate the flow of events and the messages exchanged between agents when executing specific scenarios. Complex tasks, potentially lengthy tasks, and tasks that require communication with other agents are implemented asynchronously, which is indicated by the *StartTask* method. Furthermore, the message type used for communication with other agents is specified in parentheses.

Figure 7.4 depicts the MABD diagram for estimating the losses caused by a distribution line and the corresponding transformer. As defined in the task specification diagram (cf. Figure 6.7), this scenario is either triggered by an Initiator Switch Agent upon starting

an optimization run or by a Participant Switch Agent upon receiving the corresponding request via a message. The MABD diagram depicted shows that a task is started for evaluating the losses. First, the Switch Agent examines the associated area to determine neighboring agents via the `GetNeighboringAgents` task. Because the distribution network follows a forest structure, it can only be supplied by a single agent, which in this example, is a transformer agent.

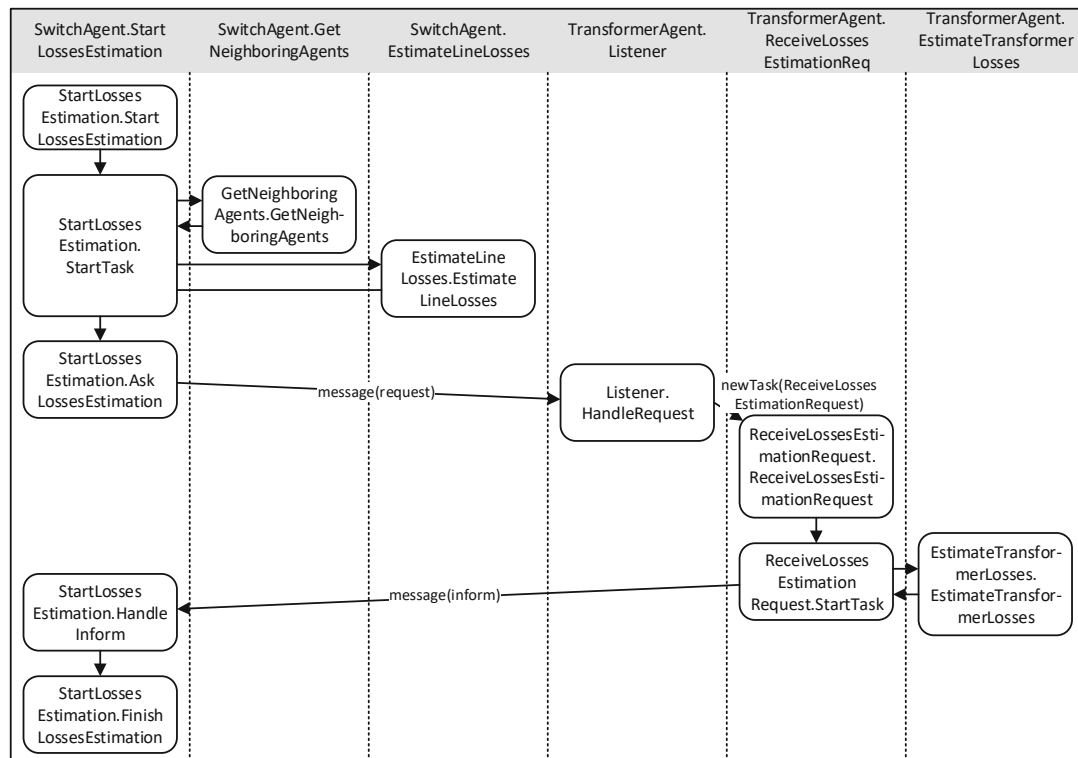


Figure 7.4: PASSI MABD diagram for savings estimation

Next, the losses caused by the line connecting the switch to the corresponding transformer are estimated by the `EstimateLineLosses` task. The Switch Agent then asks the Transformer Agent to estimate the losses caused by its transformer. The request is received by the Transformer Agent's `Listener` task and forwarded to its `ReceiveLossesEstimationRequest` task. The request is handled by estimating the transformer losses using the `EstimateTransformerLosses` task. The Switch Agent is then informed about the result and can add it to the line losses calculated before, providing an estimation about the losses caused by the distribution network in its current configuration.

As a next step towards creating executable code, the individual methods are specified by means of SABD diagrams. Figure 7.5 exemplifies this for the `GetNeighboringAgents` method. The method performs a breadth-first search to identify all neighboring agents connected to the same area and returns them as a list.

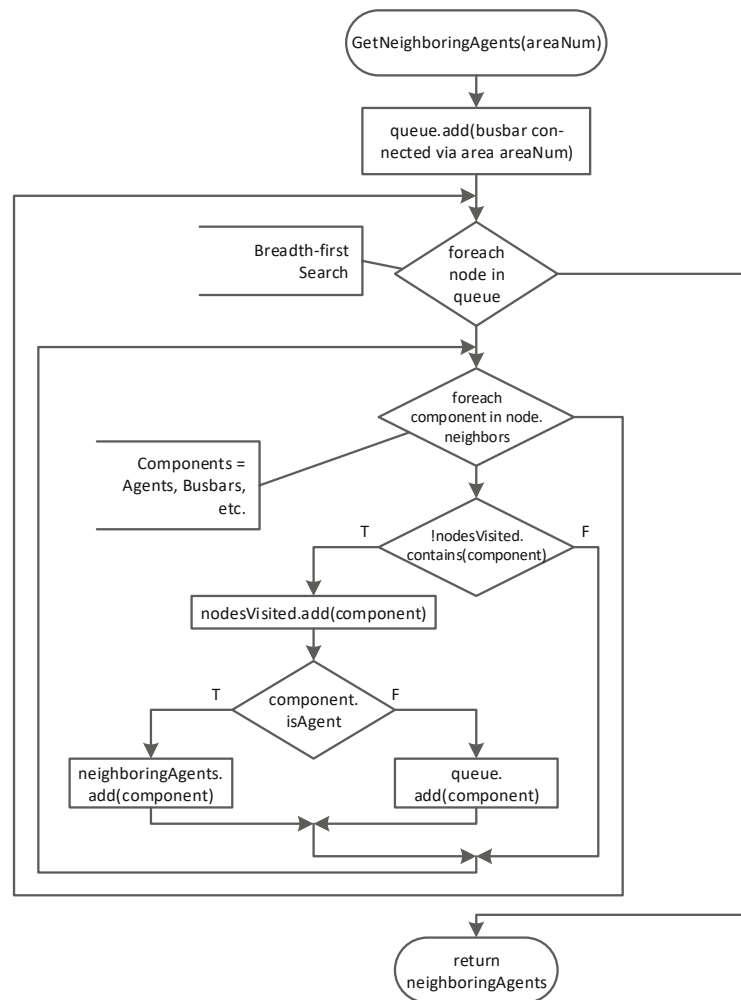


Figure 7.5: PASSI SABD diagram for the GetNeighboringAgents method

### 7.1.2 Code model

As argued, the primary focus of the PASSI methodology is on developing the agent control logic. This is also the case for the code model and supported by the PASSI PTK and the PASSI Agent Factory (cf. Section 2.3.3), which both build upon JADE as an AP. However, an agent following the concept of an CAA as defined in Section 6.1.3, includes a lot more software than an AP. Thus, while an existing AP to build upon is an important aspect when selecting a suitable programming language and computing platform, the following components and aspects should also be taken into account:

- *Triple store*: As KR is a key aspect for this thesis, an agent must include a triple store. Triple stores are available in various programming languages. However,

they differ regarding performance and features to access, serialize, de-serialize, and manipulate the information.

- *Software libraries:* The agents may require access to existing software libraries for tasks like communication, encryption, and graph algorithms.
- *Hardware and processing platform:* The agents may be required to be deployed on existing hardware and processing platforms, resulting in limitations in terms of processing and storage capabilities, available communication interfaces, and available OSs. Even if this is not the case, economic considerations may impose similar limitations.
- *Simulation environment:* Being able to test and improve agent control code in SG simulation tools and then reuse as much program code as possible in the real-world MAS has obvious benefits.
- *Sensor/actuator and control interfaces:* Sensors/actuators and existing control equipment may be connected to the agents via industrial communication systems and protocols that are not necessarily vendor-neutral or have an open specification. Therefore, the programming language used for the agent's logic must be compatible with the available APIs, or additional gateways need to be implemented.

Considering the above list, the C++ programming language has been identified as the most suitable candidate to implement the Switch Agent and the Transformer Agent. C++ is object-oriented, supports a broad range of existing software libraries, has excellent compatibility with existing hardware platforms, and supports many existing sensor/actuator and control interfaces, often implemented in C or C++. Furthermore, as will be discussed in the evaluation chapter, agents implemented in C++ can easily be tested in the Framework for Network Co-Simulation (FNCS) SG simulation frameworks.

A minor drawback of using C++ as a programming language is the limited availability of software to store and operate on ontologies. However, very basic triple stores are also available for C++, and the remaining required functionality for extracting specific information from the triple store can easily be implemented. Alternatively to using a C++ triple store with limited functionality, it would also be possible to follow a more modular programming approach, use a triple store as a standalone component, and access the information via a standardized interface like SPARQL Protocol and RDF Query Language (SPARQL)<sup>1</sup>.

---

<sup>1</sup>SPARQL Protocol and RDF Query Language (SPARQL) is a recursive acronym



## 7.2 Ontology instantiation

The ontology created in the previous chapter contains all relevant concepts between ontology classes, i.e., the TBox of the switching optimization ontology. Additionally, it contains some individuals relevant to all agents within the MAS and, therefore, already a part of the ABox of the switching optimization ontology. However, as argued, in a large distributed MAS, each agent only requires a certain subset of the complete SG information like the local grid structure, its neighboring agents, and the communication protocols it supports. Therefore, the ontology instantiation activity complements the ABox of each agent-specific ontology with additional individuals.

Agent-specific information can either be added before deploying the ontologies or during operation. However, also a combination of both approaches may be suitable: some information, e.g., the agent’s name and the services it supports, may be added in advance, while other, e.g., the switch state, can only be obtained during run-time and may change. It immediately follows that synchronizing distributed ontologies may simplify the control logic of agents in a MAS. If synchronization was handled transparently, individual agents would not need to bother requesting information from other agents. Instead, they could just rely on the information in their local triple store. Synchronizing distributed ontologies is heavily researched under the topic “Change Management for Distributed Ontologies” [248].

Furthermore, the situation may arise that agents within a MAS support entirely different ontologies, e.g., because they participate in multiple applications and require multiple application-specific ontologies. This possibility is already taken into account for by the FIPA ontology via the `supportsOntology` property and the `Ontology` class. For the switching optimization use case, it is assumed that each agent supports the complete switching optimization ontology, including all its domain and fragment ontologies.

### 7.2.1 Application-specific ontology instantiation

Figure 7.6 exemplifies how the automated switching optimization application-specific ontology is instantiated. It defines the `SA1_DfAgentDescription` and related information for Switch Agent 1. `SA1_DfAgentDescription` is an instance of the `DfAgentDescription` class. Its name is `SwitchAgent1` and it can be contacted via its address `asoip://sa1.auto.tuwien.ac.at`<sup>2</sup>.

`SwitchAgent1` offers a `SwitchingService`, which is further described using the `SA1_SwitchingServiceDescription` of type `ServiceDescription`. In the following sections, additional non-functional attributes are added to the `SA1_SwitchingServiceDescription` by making use of `hasProperty` and the rating domain ontology.

<sup>2</sup>asoip is short for automated switching optimization interaction protocol



For these reasons, it is not useful to discuss the applicability of each dependability attribute to each ontology domain exhaustively. Instead, examples of how the switching optimization use case may benefit from specifying dependability attributes for various ontology elements are given in the following. Thereby dependability attributes of a service also include the properties of the corresponding hardware. For example, if a SwitchingService is unavailable, this can either be caused by a software problem within the agent, a problem regarding the control interface, or a problem concerning the physical switch.

In the following, examples are provided of how agents may make use of their knowledge about dependability to improve the overall performance of the switching optimization use case. These examples cover availability, reliability, maintainability, integrity, and scalability. Confidentiality, safety, and privacy are not discussed because, according to the analysis provided in Table 5.1, no special measures have to be taken to ensure that the application's requirements regarding these attributes are met. Therefore, they will not influence the decision among multiple reconfiguration options and need not be modeled.

### Modeling service availability

Assuming there exist multiple reconfiguration possibilities providing roughly the same power savings, considering the availability of components may have several benefits. For example, if the Initiator agent possesses knowledge about the availability of each switch, it may choose to close the switch with the highest availability. This reduces the risk of getting stuck in a non-optimal configuration if loads shift again at a later point in time.

Furthermore, the overall availability of a specific configuration depends on the availability of multiple components, for example, the switches and transformers involved in supplying a specific area. If the LossEvaluationService of one of the switches or the transformer is unavailable, the system may again get stuck in a non-optimal configuration. Therefore, in a more sophisticated setup, the Initiator may apply techniques such as Markov models with probability matrices to determine a more complex but also more precise availability rating.

Figure 7.7 illustrates how the availability of the SwitchingService of SwitchAgent1 is modeled. The SA1\_SwitchingServiceAvailability individual is added via the hasProperty property. Availability is thereby expressed as a simple probability value. In the illustrated example, the SwitchingService of SwitchAgent1 is operational 99.8% of the time.

### Modeling service reliability

Similarly to availability, the Initiator agent may balance possible savings with varying reliability of multiple possible configurations. It may choose to close the switch with the highest reliability, thus, reducing the risk of a failure like the malfunction of a switch, which could cause a power outage within the corresponding area.

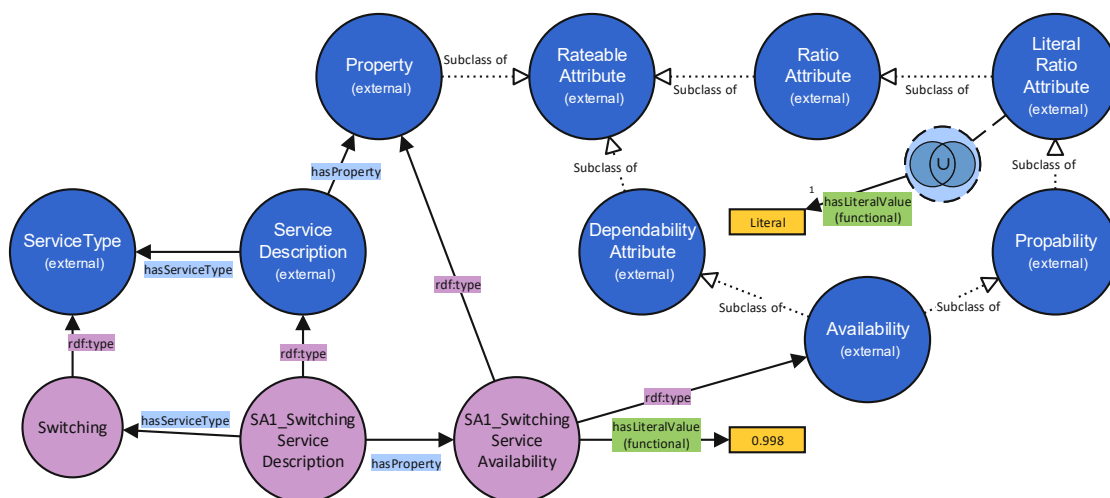


Figure 7.7: Modeling service availability

As reliability is modeled using a PDF (or its complementary CDF), determining the more reliable option among multiple configurations is not as trivial as in the case of availability. However, the Mean Time To Failure (MTTF) or the probability of a failure occurring before the next expected reconfiguration can easily be calculated from the PDF and used as a suitable rating. Furthermore, as in the case of availability, the overall reliability depends on the reliability of the involved components.

Figure 7.8 illustrates how the reliability of the SwitchingService of SwitchAgent1 is modeled. The SA1\_SwitchingServiceReliability is added as an individual of class Reliability and connected to the service description via hasProperty. Its value is defined by the SA1\_SwitchingServiceReliabilityPDF, which is an individual of type WeibullPDF. The parameters are set to  $\lambda = 16$ ,  $k = 0.7$ , and  $\Delta t = 2016-01-31$  in this example. If the argument  $t$  (cf. Equation 6.1) is given in years, this distribution specifies that the devices will have failed with 50 % probability after about 10 years, starting from 2016-01-31. The MTTF is approximately 20 years.

### Modeling service maintainability

Maintainability expresses the ability of a component to be repaired after a failure. Therefore, if the Initiator considers maintainability of components in its decision-making process and supplies as many loads as possible via well-maintainable equipment, the accumulated downtime across these loads can be reduced.

Modeling maintainability builds upon the same mathematical concepts as modeling reliability. Again, it can be expressed via a Weibull PDF or a similar probability distribution. In the example illustrated in Figure 7.9, parameter  $\lambda$  is set to 20 and  $k$  is set to 3. Parameter  $t$  is expected to be given in days, resulting in a 50 % chance of a

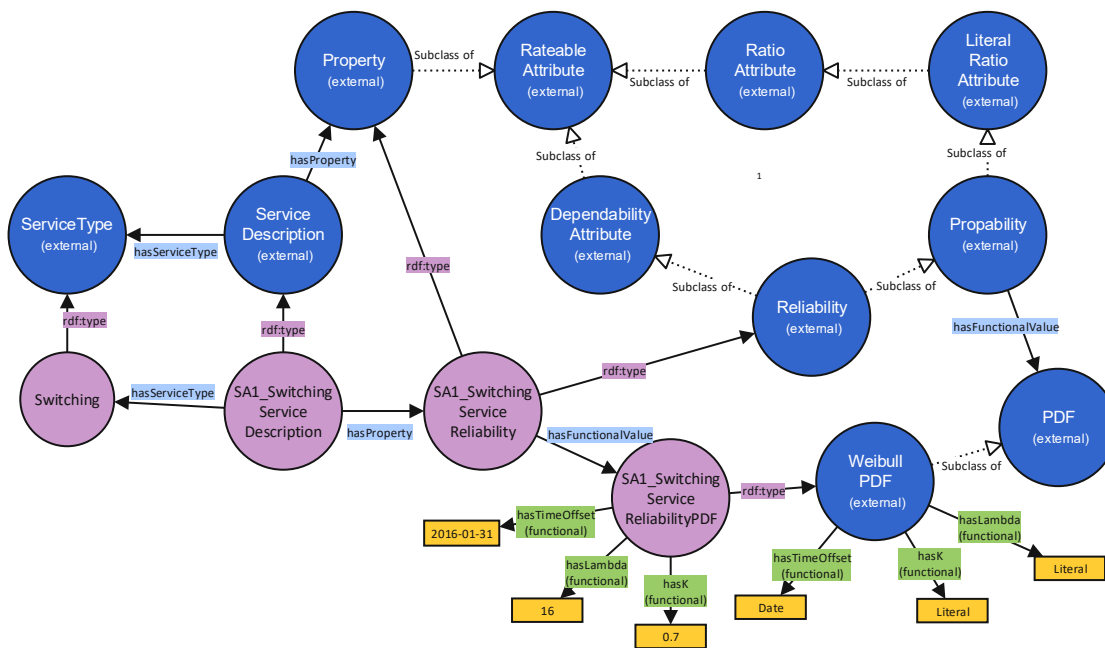


Figure 7.8: Modeling service reliability

device being repaired within the first 17 h, and a Mean Time To Repair (MTTR) of 20 h. In contrast to the Weibull PDF for reliability,  $\Delta t$  is set to 0 in the case of maintainability. Therefore,  $t = 0$  corresponds to the moment the service or the controlled device fails, and any time  $t > 0$  corresponds to the time passed since this event.

### Modeling communication protocol integrity

Agents may support multiple communication interfaces and communication protocols with varying integrity features, e.g., using no, basic, or very sophisticated encryption mechanisms. Knowledge about these features allows selecting an interface and protocol with an appropriate degree of integrity for a specific service or data request. For example, the content of a switching request message or its confirmation must be well protected against unauthorized modification as changing this information would immediately cause severe problems in the grid. However, if an agent does not provide appropriate integrity mechanisms or the corresponding communication link fails, this does not mean that the agent cannot at all contribute to the system. It may still be possible for this agent to offer the LossEvaluationService or other services with reduced integrity requirements.

Figure 7.10 illustrates how the integrity of a communication protocol is modeled. The integrity of the TLS protocol is defined via the `hasProperty` reference to `TLSIntegrity`. Furthermore, the `TLSIntegrity` is rated high by assigning the H ordinal categorical value to the `TLSIntegrity` individual via the `hasOrdinalValue` property. The SA1\_CII

communication interface supports the TLS standard. Finally, the SA1\_CI1 is offered as a communication interface by the SwitchingService of SA1.

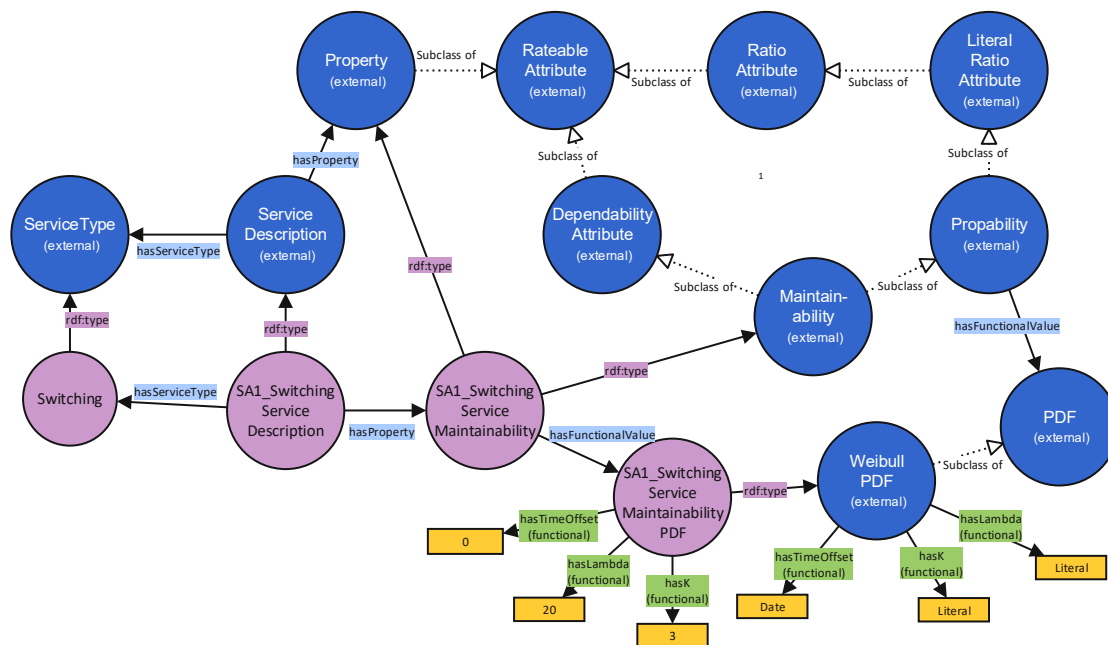


Figure 7.9: Modeling service maintainability

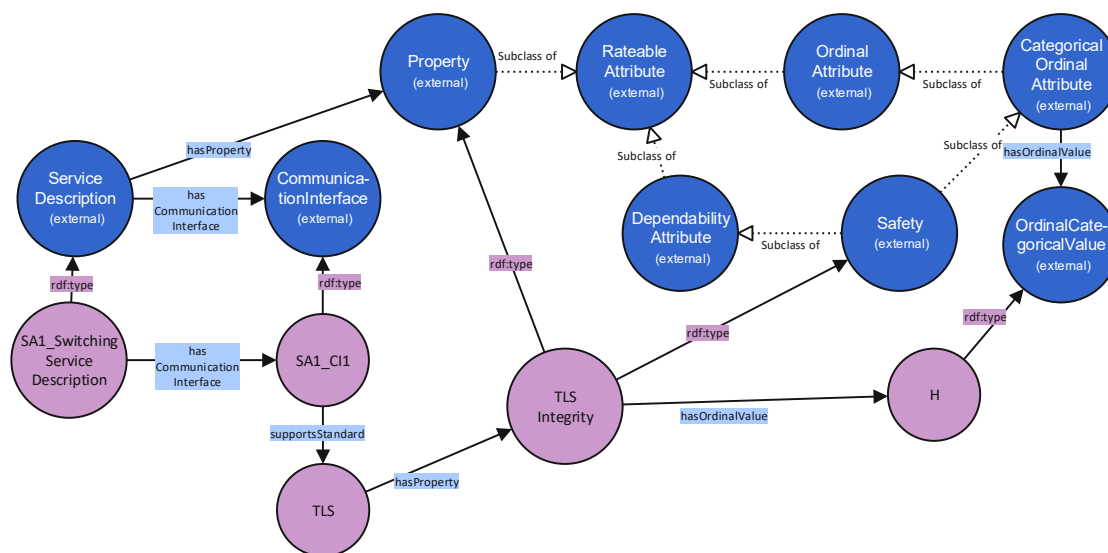


Figure 7.10: Modeling communication protocol integrity

### Modeling communication protocol scalability

Although most communication protocols are point-to-point/unicast protocols, more scalable alternatives exist in multicast protocols. These can either be broker-based like Message Queuing Telemetry Transport (MQTT) and Apache Kafka, or broker-less like Internet Protocol version 6 (IPv6) multicast. Multicast protocols allow reducing the communication loads that need to be handled by individual agents as the communication system takes care of replicating and delivering messages. Therefore, choosing a communication protocol that supports multicast over unicast alternatives may reduce the required processing power, memory consumption, and power consumption of agents. Additionally, multicast protocols may reduce the message load and power consumption of the communication network. It is also possible to address a large group of agents via a single multicast message while using unicast messages for the remaining agents that do not support the multicast protocol if this information is available in the triple store.

As illustrated in Figure 7.11, modeling communication protocol scalability builds upon the same concepts as modeling communication protocol integrity. The scalability of MQTT is rated medium (M) in this example, as MQTT requires a broker. This may constitute a communication bottleneck if many agents communicate simultaneously if no additional measures like a scalable MQTT broker are implemented.

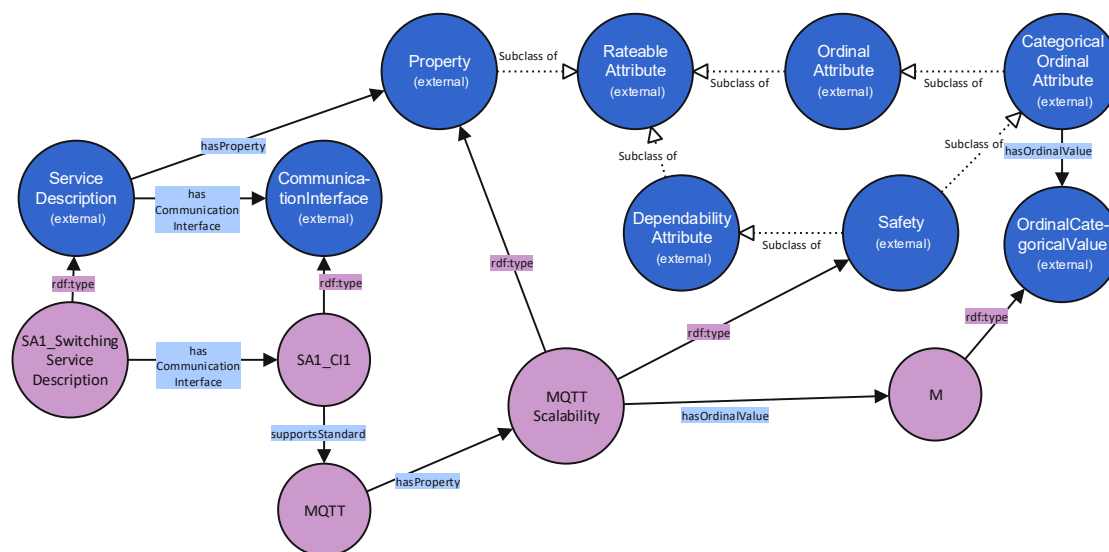


Figure 7.11: Modeling communication protocol scalability

This completes the implementation phase. The agents can now be implemented and equipped with the knowledge they require by storing the information in their individual triple stores. SG simulation is used as a means to evaluate the resulting MAS.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



# Evaluation

The evaluation phase is the last phase of the SELC to be discussed. Its activities and associated tools are depicted in Figure 8.1. Similarly to the phases before, the evaluation includes functional and non-functional aspects. The functional evaluation ensures that the agents, communication systems, interaction protocols, and other functional aspects perform as expected. This also includes the overall algorithms that implement the use case, i.e., the distributed switching optimization algorithm in this thesis. The non-functional evaluation elaborates to which extent considering non-functional attributes, such as dependability attributes, improves the solution regarding the KPIs that were identified during the IEC 62559 use case methodology in the planning phase (cf. Table 4.5).

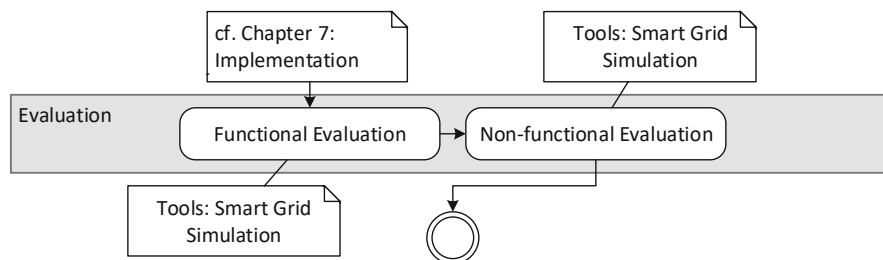


Figure 8.1: Evaluation phase: activities and tools

## 8.1 Smart grid simulation

The functional and non-functional evaluation of the MAS developed throughout this thesis is conducted using SG simulation. As SGs arise from the application of ICT in the power grid, a SG simulation framework needs to support both aspects. However, this requirement is a challenge, as simulation tools for the two domains are based on fundamentally different concepts. Power systems are electrical networks and, as such,

are continuous systems. In contrast, ICT systems are mostly discrete systems<sup>1</sup>, e.g., a processor operates on a discrete clock and communication networks exchange individual messages/packets. In [249], Palensky, Widl, and Elsheikh provide an overview of SG simulation approaches.

Sophisticated simulation tools exist for both types of systems. For example, PY-POWER [250], GridLAB-D [251], Siemens PSS@SINCAL [214], and DIgSILENT PowerFactory [252] target power systems simulation and load-flow calculations, while SimPy [253], NS3 [254], and OMNeT++ [255] were developed for discrete-event simulation. Therefore, instead of creating new tools to simulate SG applications, co-simulation frameworks allow to combine multiple simulation tools. Thereby, the individual simulation tools connect to a central component, e.g., a broker or coordinator. The broker then orchestrates the simulation steps and handles data exchange between the simulation tools. Examples for co-simulation frameworks are Simantics [256], Mosaik [257], and FNCS [258].

### 8.1.1 Framework for Network Co-Simulation (FNCS)

FNCS combines GridLAB-D for power grid simulation and Network Simulator 3 (NS3) for communication network simulation in a co-simulation framework. The individual power system components are already provided by GridLAB-D in the form of GridLAB-D objects. Agent functionality has to be added by implementing agents as additional GridLAB-D objects. This approach provides agents with easy access to the required data and functionality, e.g., voltage levels, transformer configurations, and switch control. The communication network is simulated using a simple Carrier Sense Multiple Access/Collision Detection (CSMA/CD) network with very low latency in NS3. FNCS acts as an orchestrator between GridLAB-D and NS3, as illustrated in Figure 8.2. Furthermore, it handles message exchange between the simulation tools. The software versions used for the functional and non-functional evaluations are summarized in Table 8.1. The following section discusses how agents can be introduced in GridLAB-D.

Table 8.1: Software and versions used for evaluation

<i>Software</i>	<i>Version</i>
FNCS	1.0
GridLAB-D	3.0.0-5269 (Hassayampa) 64-bit LINUX RELEASE
NS3	3.19

### 8.1.2 Agents in GridLAB-D

Out of the box, GridLAB-D supports a number of components typically found in power distribution networks, e.g., energy sources, transformers, switches, and loads.

<sup>1</sup>On a lower level, ICT systems build upon physical (and arguably continuous) phenomena, e.g., a crystal oscillator for clock generation and electromagnetic waves for wireless signal transmission. However, except for very detailed analyses, ICT systems are typically simulated on a more abstract, discrete level.

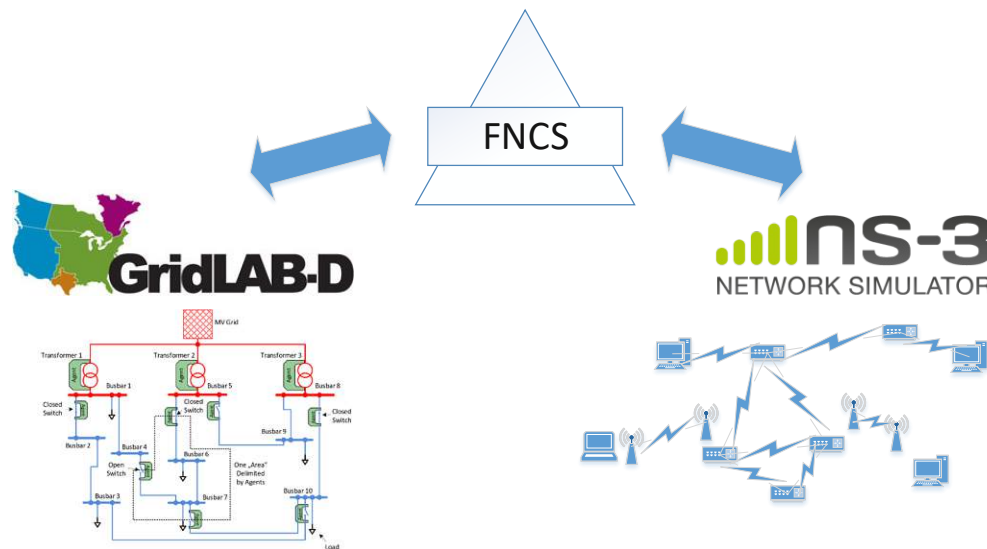


Figure 8.2: FNCS architecture with GridLAB-D [251] and NS3 [254]

Physical connections between the individual components are modeled via various types of transmission and distribution lines. Furthermore, GridLAB-D allows to specify objects that do not necessarily have a physical counterpart in a real world scenario, e.g., recorders to store simulation results as Comma-Separated Values (CSV) in files and players allowing to load time-dependent configurations (e.g., load values) from CSV files. The FNCS-compatible version of GridLAB-D includes the Network Interface object. It allows GridLAB-D objects to exchange string messages via FNCS and NS3. Furthermore, in the course of this work, the Switch Agent and the Transformer Agent objects have been developed and added to GridLAB-D. The behavior of each object in GridLAB-D is specified in C++. Therefore, agents can make full use of existing C++ libraries.

The realization of the Switch Agent (a) and the Transformer Agent (b) object as well as their interconnections to other GridLAB-D objects are illustrated in Figure 8.3. As mentioned, the Switch Agent directly controls a Switch object, i.e., it can open and close the switch and also determine the switch's current state. Additionally, the Switch Agent has access to metering data of Meter objects, which are placed at either side of each Switch object to be able to measure power consumption, voltage levels and electric currents. The Network Interface object allows the Switch Agent to communicate with other agents via FNCS and NS3. The current implementation of the Switch Agent includes a FIPA ACL library originally developed within the Mobile-C project [134]. It allows to easily encode/decode FIPA ACL messages. Non-functional attributes are stored in an owlcpp [259] triple store. Furthermore, a Switch Agent uses the Boost Graph Library (BGL) to store the grid topology of both areas it is connected to as weighted power grid graph (cf. Section 5.1.1).

The Transformer Agent follows the same basic structure. Unfortunately, the GridLAB-D version used does not model no-load losses of transformers directly. Therefore, additional Load objects (termed Shunt objects in this context to distinguish them from real loads) have been added at both sides of the transformer. These changes provide the necessary functionality to implement agents in GridLAB-D and to conduct the functional and non-functional evaluations.

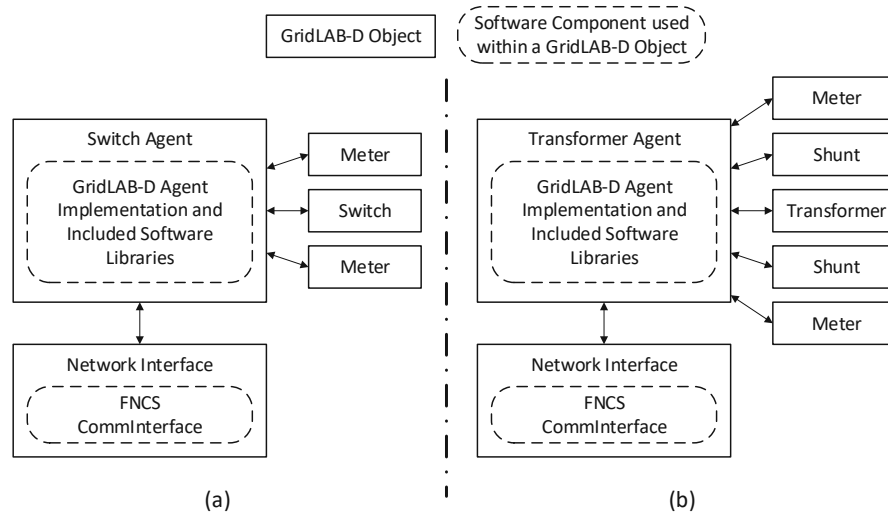


Figure 8.3: (a) Switch Agent and (b) Transformer Agent implementation in GridLAB-D

## 8.2 Functional evaluation

The functional evaluation ensures that the FNCS co-simulation environment performs as expected and that the individual agents act according to their specifications. Furthermore, it sets a baseline for energy savings achievable by implementing the switching optimization use case developed throughout this thesis.

### 8.2.1 Evaluation setup

Figure 8.4 depicts the GridLAB-D power system used as evaluation setup for the functional evaluation. Circled numbers and their corresponding arrows represent exchanged messages and are not relevant at this time, but are explained in the next section. The distribution network is supplied by two Transformer Agents and contains five Switch Agents, which enable dynamic restructuring of the grid topology. It consists of 6 areas (as they have been defined in Section 6.1.5), each containing a single busbar. The distribution network supplies one commercial load and three residential loads. It is assumed that the grid topology has already been optimized based on estimated loads when the distribution network was planned and built. The outcome of this optimization process is given in the figure. However, this might not be the optimum solution at any given time of the day.

Assuming people, following their daily routine, work at the office building during the day and spend their evenings at home, load values shift from residential loads 1-3 to the commercial load during office hours and back afterwards. Regarding the overall power losses, it is therefore beneficial to restructure the distribution network and try to equally distribute the loads among the available transformers multiple times a day. How this is achieved by the developed MAS is the topic of the next section.

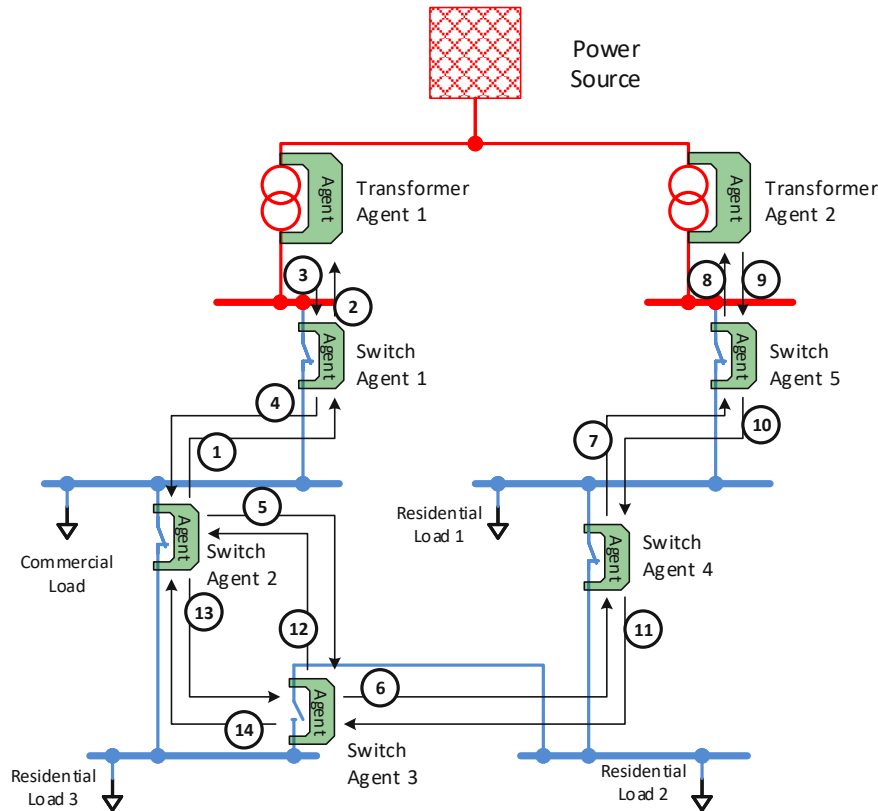


Figure 8.4: Functional evaluation setup and messages exchanged during optimization

Load values for the commercial and the three residential loads are illustrated in Figure 8.5. They were obtained from the Open Data Catalog [260] of the U.S. Department of Energy and from the Pecan Street project [261]. The commercial load represents a medium sized office building while the three residential loads aggregate load values of 25 arbitrarily chosen apartments, single-family homes, and town homes. All four datasets contain a 24 h load profile with one hour resolution on a normal working day in Texas, U.S.

### 8.2.2 Exemplary optimization run

The messages exchanged during a simple optimization run are illustrated in Figure 8.4. The optimization run takes place at 07:00. From there on, only the commercial load is supplied by transformer 1, while all residential loads are supplied by transformer 2.

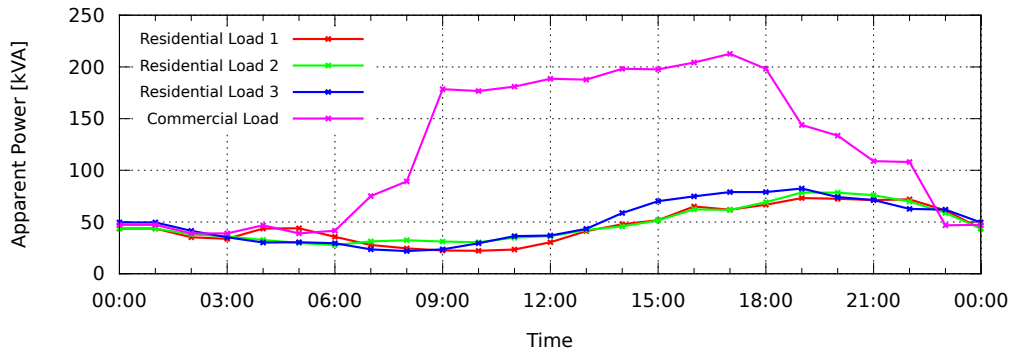


Figure 8.5: 24 h load profiles of the commercial and residential loads used for simulation

Another optimization run at 20:00 restores the topology to its original state but is not discussed in following. It is assumed that all agents are idle at the beginning, i.e., they do not participate in any other optimization run at the moment. The optimization run follows the interaction protocol specified in Figure 6.9.

① Switch Agent 2 sends the first *request* message to Switch Agent 1 upon recognizing that the power consumption dropped in the bottom left area due to the reasons outlined in the previous section. It asks Switch Agent 1 about the current losses that are caused by supplying residential load 3 and to not participate in any other optimization run meanwhile. ② Switch Agent 1, forwards the request in direction of the power source (i.e., the transmission network) to Transformer Agent 1. Transformer Agent 1 calculates the losses based on its electrical parameters and the current load situation. ③ It replies to Switch Agent 1 using an *inform-ref* message. ④ The information is forwarded to Switch Agent 2. ⑤ Switch Agent 2 now asks all neighboring agents in the area to be optimized (just Switch Agent 3 in this example) for proposals better than the current situation by using a *CFP* message that includes the current loss value. Again, two actions are specified in the *CFP* message: a request to offer a proposal for supplying the load, and a request to not participate in other optimization runs meanwhile. ⑥ ⑦ ⑧ In the same manner, Switch Agent 3, Switch Agent 4 and finally Switch Agent 5 themselves issue *CFP* messages until Transformer Agent 2 is reached. ⑨ ⑩ ⑪ ⑫ Transformer Agent 2 answers the *CFP* by issuing a *propose* message to Switch Agent 5, which is forwarded to Switch Agent 4, Switch Agent 3 and finally to Switch Agent 2. ⑬ Assuming the proposed configuration is better than the current one, Switch Agent 2 accepts the proposal and asks Switch Agent 3 to close its switch by sending an *accept proposal* message. ⑭ Switch Agent 3 closes its switch and confirms the request by sending an *inform* message back to Switch Agent 2. As residential load 3 is now supplied by Switch Agent 3, Switch Agent 2 opens its switch and the reconfiguration process has finished. A number of *inform* and *accept proposal* messages follow to inform all participants that the optimization run has been completed and they are free to participate in other optimization runs or start their own ones. For the sake of clarity, these messages are not included in Figure 8.4.

### 8.2.3 Evaluation results

The functional evaluation results are illustrated in Figure 8.6. Transformer losses with optimization disabled are approximately 58.8 kWh within the simulation period of 24 h. With optimization enabled, they reduce to approximately 54.5 kWh, thus being cut by 7.3%. Line losses have currently not been investigated but can be added to the simulation environment easily. However, using longer lines in favor of equal load balancing among transformers generally reduces the overall energy savings. If the switching optimization MAS is disabled, transformer 1 is at up to 110.85% of its rated power of 250 kVA in this scenario. A pleasant side-effect of the proposed system is that overload situations of the transformers are avoided if the switching optimization MAS is enabled.

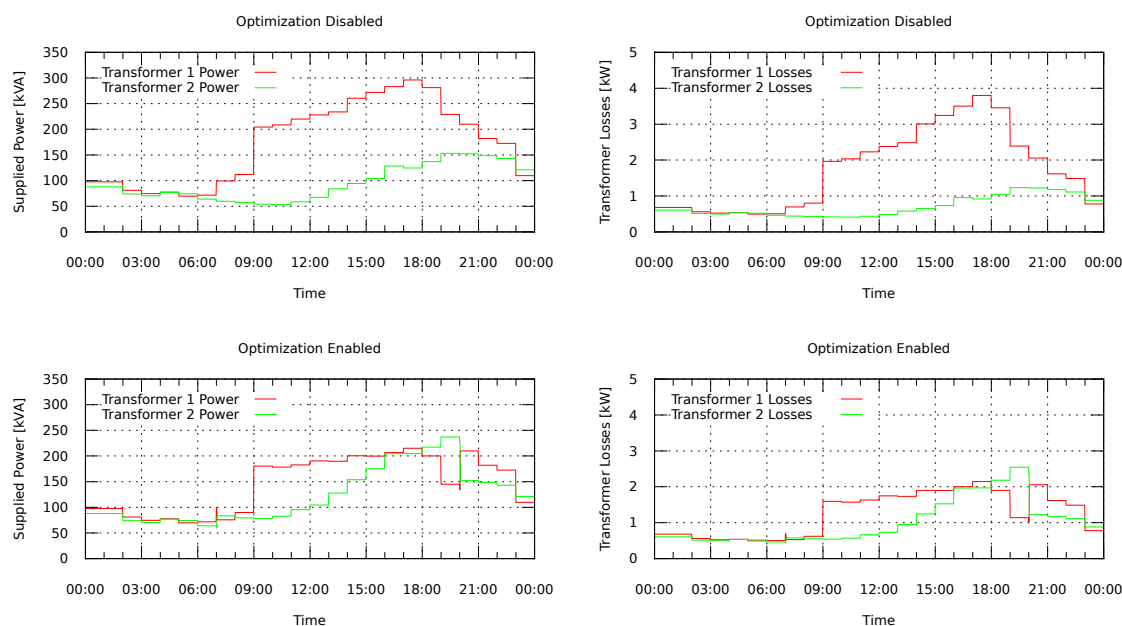


Figure 8.6: Supplied power and transformer losses of the functional evaluation scenario

## 8.3 Non-functional evaluation

In the functional evaluation performed so far, agents only consider possible savings during network reconfiguration. The non-functional evaluation shows that considering non-functional requirements, in particular dependability attributes, in the decision-making process can improve the overall performance of the MAS, which is specified by its KPI. This is exemplified based on the availability attribute of switching services in the following.

### 8.3.1 Evaluation setup

The evaluation setup used for the non-functional evaluation is depicted in Figure 8.7. It is an extension of the previously presented distribution network (cf. Figure 8.4). In

particular, Transformer Agent 3, Switch Agents 6-8, residential load 4+5, and three additional busbars have been added. Residential loads 4+5 use the same load profiles as residential loads 1+2 (cf. Figure 8.5). Furthermore, the scenario is similar as before: throughout the day, loads shift from the residential loads towards the commercial load, which triggers Switch Agent 2 to start an optimization procedure. However, the more complex distribution network provides two possible reconfigurations, either by closing Switch 3 as before, or by closing Switch 6. The transformer managed by Transformer Agent 3 is slightly more efficient than the transformer managed by Transformer Agent 2. This favors closing Switch Agent 6 over Switch Agent 3 to supply residential load 3 if no additional considerations are taken into account.

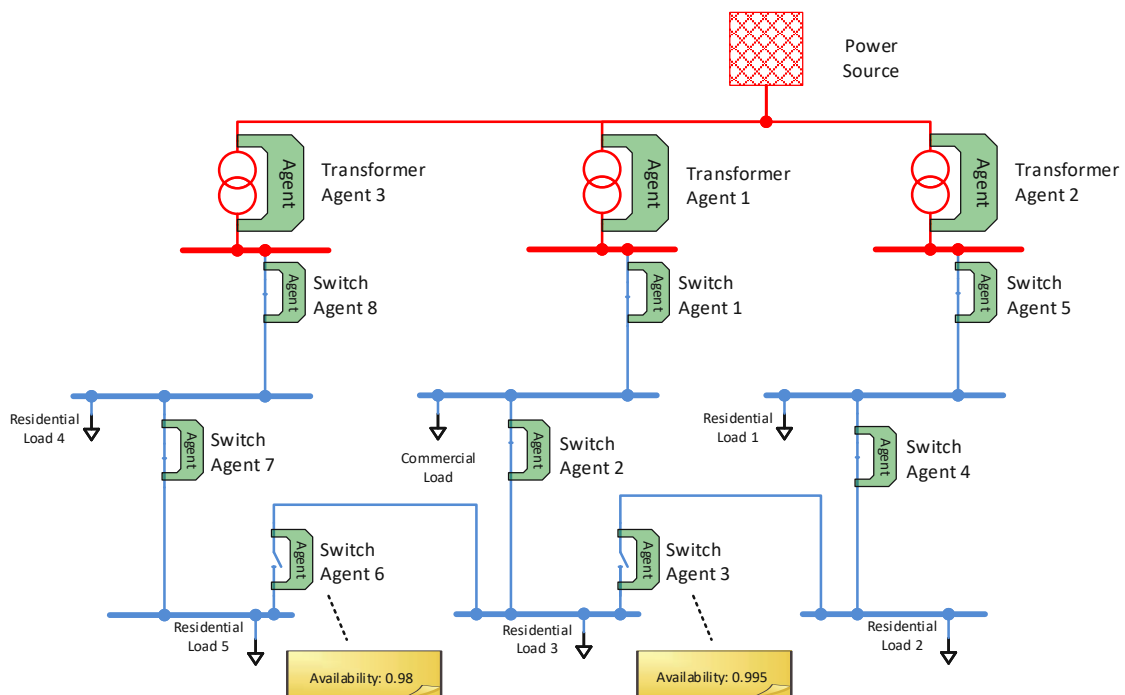


Figure 8.7: Non-functional evaluation setup

For evaluation purposes, the availability of each agent's switching service is configured using two parameters. The first parameter specifies the availability via an absolute value between 0.0 and 1.0, e.g., 0.995 for Switch Agent 3. This value corresponds to the definition of availability as specified in Section 1.2. Furthermore, the service remains unavailable for a specific duration, e.g., for 24 h, which reflects the downtime in Equation 1.1. If a switching service is unavailable, the Switch Agent does not trigger an optimization run. Therefore, the network cannot be reconfigured if the switching service of an agent supplying a particular area becomes unavailable. From these values (availability and downtime), the corresponding uptime required to satisfy Equation 1.1 can be calculated according to Equation 8.1. For the exemplary values of Availability



= 0.995 and downtime = 24 h, the uptime results to 4776 h. Therefore, the switching service of Switch Agent 3 is unavailable for 24 h every 200 days. The switching service of Switch Agent 6 has an availability of 0.98 and a downtime of 36 h in this scenario. It is unavailable for 36 h every 75 days. The availability of all remaining services, including the ones of Switch Agent 2, are not relevant in this evaluation scenario and set to 1.0.

$$uptime[s] = \frac{Availability * downtime[s]}{1.0 - Availability} \quad (8.1)$$

### 8.3.2 Considering availability

As mentioned, considering the evaluation setup of Figure 8.7, Switch Agent 2 can either select the switch controlled by Switch Agent 3 or Switch Agent 6 to temporarily supply the commercial load. Due to the slightly better efficiency of transformer 3, it favors Switch Agent 6 if availability is not considered in its decision. Otherwise, it selects Switch Agent 3 over Switch Agent 6 because this reduces the risk of temporarily getting stuck in a non-optimal configuration. As shown by the evaluation results in the following section, considering availability of individual services when selecting between multiple reconfiguration options improves the savings achievable by the switching optimization use case in this evaluation setup. In any case, either Switch Agent 3 or Switch Agent 6 select Switch Agent 2 to supply residential load 3 again after the power consumption shifts back to the residential loads.

The necessary information about the availability of the relevant services is modeled using the ontologies developed throughout the previous phases of the SELC. Figure 8.8 shows the corresponding parts of the instantiated application-specific ontology for Switch Agent 2. It preliminarily includes information to identify the agents it interacts with (SA3\_DfAgentIdentifier and SA6\_DfAgentIdentifier), the services they offer (SA3\_SwitchingService and SA6\_SwitchingService), the individuals for describing the availability of each service as instances of both Property and Availability (SA3\_SwitchingServiceAvailability and SA6\_SwitchingServiceAvailability), and their corresponding literal values (0.995 and 0.98). Furthermore, individuals of types DfAgent-Description and ServiceDescription are required to connect the agents to their services.

### 8.3.3 Evaluation results

To perform the non-functional evaluation, three different settings of the distribution network illustrated in Figure 8.7 are used. First, as a baseline for the evaluation, the optimization functionality is disabled, i.e., the switches do not perform any reconfigurations. Second, the switching optimization services are enabled, but the switches do not consider availability when choosing among possible reconfigurations. And third, the switches do consider availability by accessing the information stored in their ontologies. The simulation time is extended from 24 h (in the functional evaluation setup) to six months from the begin of April to the end of September. During this time frame, the

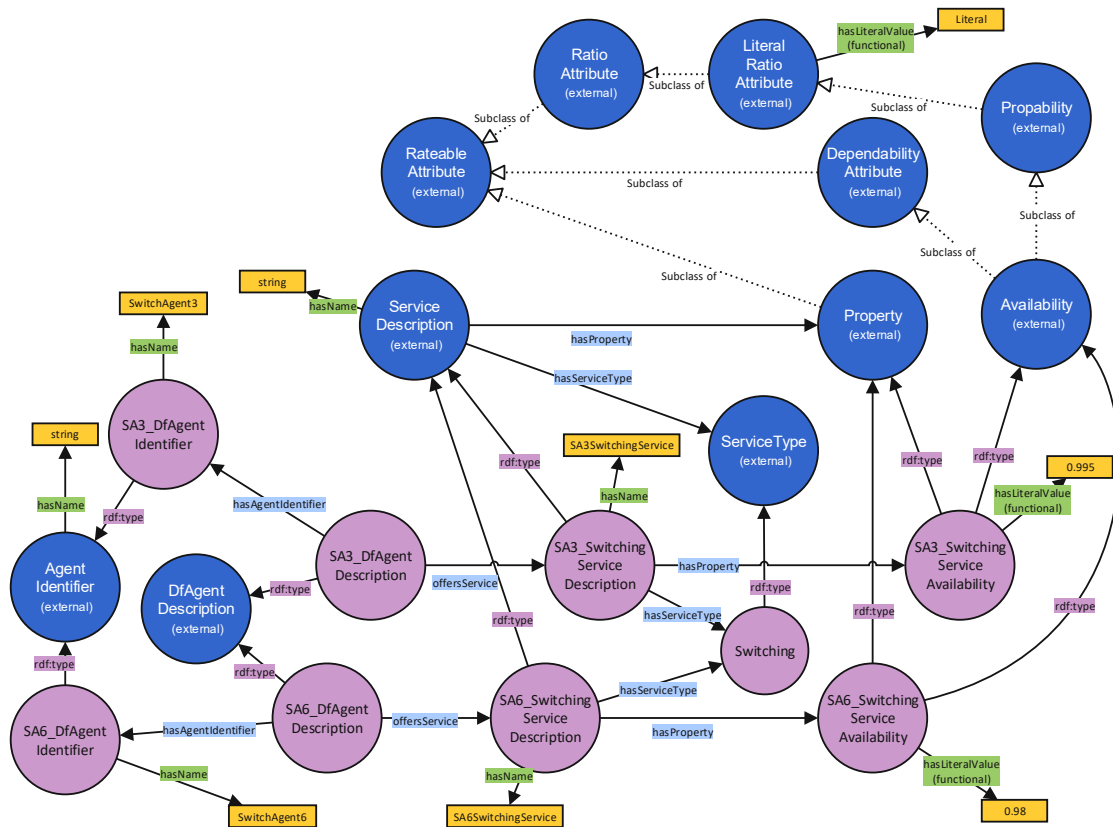


Figure 8.8: Switch Agent 2 ontology for the non-functional evaluation

switching services of both Switch Agents (Switch Agent 3 and Switch Agent 6) become unavailable at least once.

Figure 8.9 compares the load and loss values of the three transformers for each setting. Note that the supplied power and loss values for transformer 2 and transformer 3 are virtually identical for the first setting as they both supply loads with the same load profiles. The figure contains the time period from 12:00 on August 6 to 24:00 on August 7, which is particularly interesting because Switch Agent 6 becomes unavailable during August 6. At around 08:00 that day, the network is reconfigured such that Switch Agent 6 supplies residential load 3. It would return to its initial configuration at 21:00, when power consumption shifts back from the commercial to the residential loads. However, as the switching service of Switch Agent 6 is unavailable at this time, it is not possible to re-establish the initial configuration. The issue with Switch Agent 6 is resolved on August 7 and the initial configuration is restored at 21:00.

The depicted graphs show that the loads are very poorly balanced among the available transformers if the switching optimization algorithm is disabled entirely. If switching

optimization is enabled but availability is not considered, the system is stuck in a non-optimal configuration leading to additional losses between 21:00 on August 6 and 08:00 on August 7. Finally, if availability is considered, Switch Agent 3 is chosen over Switch Agent 6 to supply residential load 3. Therefore, the issue that the switching service of Switch Agent 6 becomes unavailable does not cause additional losses during the time period from 12:00 on August 6 to 24:00 on August 7. However, Switch Agent 3 is unavailable once for 24 h withing the 6 month simulation period, which is not covered in the depicted time period but reduces the achieved energy savings. In conclusion, enabling the optimization algorithm without considering availability reduced transformer losses from 13 844 kWh to 13 061 kWh during six months simulation period. Considering availability further reduced the transformer losses to 13 058 kWh.

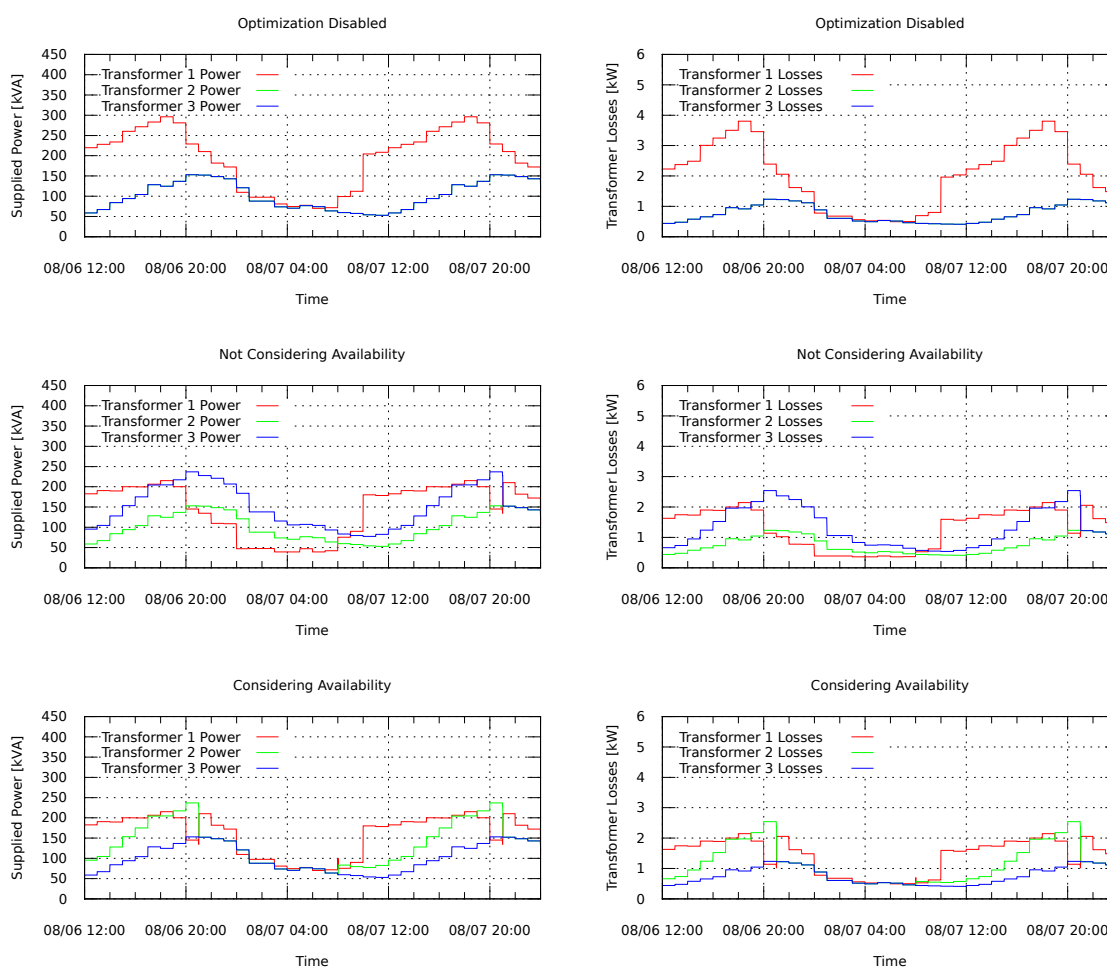


Figure 8.9: Supplied power and transformer losses of the non-functional evaluation



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

## Conclusion and future work

The work presented in this thesis aimed at answering the research question formulated in the first chapter of this document:

*RQ: Which activities need to be conducted during the Systems Engineering Life Cycle to incorporate dependability in Multi-Agent Systems for Smart Grid applications? How can state-of-the-art methodologies support this process?*

To address this question, a SELC was defined in Chapter 3. It combines the development of a MAS with ontology design to allow the description of functional and non-functional aspects of the system. The SELC consists of five consecutive phases: planning, analysis, design, implementation, and evaluation. Each phase was then discussed in a separate chapter (Chapters 4 – 8). The activities within each phase heavily rely on existing SG, MAS, and ontology design methodologies. The switching optimization use case served as a motivating example throughout this thesis. It aims at reducing losses within LV power distribution networks by utilizing their ability to be reconfigured by opening and closing electrical switches. However, aside from this motivating use case, the SELC is also applicable to other MASs in the SG domain.

The planning phase in Chapter 4 introduced general aspects of the use case independently of the system architecture. The IEC 62559 use case methodology was conducted during the *use case description* activity. During the *requirements description*, the dependability attributes defined by Avizienis, Laprie, Randell, et al. were introduced as non-functional requirements. Furthermore, these “classic” dependability attributes were complemented by scalability and privacy, as suggested in [7].

The analysis phase in Chapter 5 covered functional and non-functional aspects in more detail than the planning phase. Thereby, the *functional analysis* focused on the use case,

specifically existing algorithms and approaches to addressing the switching optimization use case. Dependability attributes were examined in more detail during the *non-functional analysis*, which focused on potential metrics for each dependability attribute. Furthermore, the importance of each dependability attribute for the present use case was investigated.

The design phase in Chapter 6 highlighted similarities to the industrial domain, which uses the concept of AASs to incorporate assets in Industry 4.0 systems. In the same manner, agents can be used to integrate power system components into the SG, resulting in the definition of a Component Administration Agent (CAA). The main software components of agents for SG applications were defined in a FIPA-compliant way in the *system architecture definition* activity. PASSI was identified as a suitable MAS design methodology to cover the functional aspects of the design phase and the relevant parts of the methodology were conducted during the *MAS design* activity. Ontologies provided the means to model various aspects of non-functional attributes and link them to other concepts of the SG domain like components of the distribution network. However, while state-of-the-art ontology design methodologies encourage incorporating existing ontologies, they themselves typically do not focus on creating reusable ontologies. Therefore, the reusable ontology design methodology was defined and conducted for the switching optimization use case during the *ontology design* activity.

The implementation phase in Chapter 7 proceeded with the PASSI MAS design methodology in the *agent implementation* activity. Based on the switching optimization use case, it provided examples of all models defined by the PASSI methodology. These models could then be implemented using existing OOP software development techniques. The *ontology instantiation* activity described and exemplified how each non-functional attribute of the previously defined ontology can be instantiated, assigned to other elements of the ontology, and rated. This enabled the individual agents to consider dependability attributes in their decision-making process.

Finally, the evaluation phase in Chapter 8 presented a number of distribution networks and load scenarios. The *functional evaluation* demonstrated that the MAS implementing the switching optimization algorithms can indeed reduce the distribution network's losses. It does so by dynamically redistributing loads among the available transformers when loads shift from one area of the distribution network to another throughout the day. Furthermore, the *non-functional evaluation* showed that considering dependability attributes in deciding how the distribution network shall be restructured may further improve the performance of the MAS.

### 9.1 Effects of the research objectives on the results

In addition to the research questions, the work presented in this thesis was guided by numerous research objectives. In the following, it is briefly summarized how each research objective influenced the results of this thesis.

*RO1: Whenever possible, existing methodologies shall be studied, compared, and the most suitable ones shall be applied during the systems engineering process.*

Numerous SG, MAS, and ontology design methodologies were examined and presented extensively in the state of the art analysis. Furthermore, existing SG standards were summarized and assigned to the various SGAM layers. Suitable methodologies were selected to support the various activities of the SELC. Furthermore, the reusable ontology design methodology referred to existing ontology design methodologies for modeling the individual concepts.

*RO2: Whenever possible, existing frameworks, technologies, tools, software, and protocols shall be studied, compared, and the most suitable ones shall be applied during the systems engineering process.*

The design of agents was inspired by FIPA and the concept of AASs, resulting in the concept of a CAA. CAAs were defined and implemented in a very modular approach. The various components (communication interfaces, DF, MTS, AMS, triple store, sensor/actuator interfaces, control interfaces, and other software libraries) were implemented based on existing open-source software. The agent interaction protocol defined for switching optimization heavily relies on the FIPA Query Interaction Protocol [129] and the FIPA Contract Net Interaction Protocol. Messages are encoded in the FIPA-ACL.

*RO3: The systems engineering process shall follow a “dependability by design” principle.*

The different aspects of the dependability tree were introduced at the very beginning of this thesis. Dependability attributes were selected as the main non-functional requirements, and the considerations and measures to address them were refined during each phase of the SELC.

*RO4: In addition to functional requirements, non-functional requirements shall be considered in decision-making processes.*

The functional aspects, which were covered by the algorithms incorporated in the MAS, were accompanied by non-functional requirements throughout the complete SELC. These non-functional requirements were analyzed, appropriate metrics were identified, and the required concepts were modeled in the rating domain ontology. Furthermore, the non-functional evaluation showed that considering non-functional requirements when selecting among multiple distribution network configurations improves the performance of the system.

*RO5: Existing ontologies shall be included in the ontology design process but, equally important, ontologies created during the design process shall themselves be reusable in other applications and domains.*

The reusable ontology design methodology was developed to cover these two aspects. During the top-down phase, the reusable ontology design methodology encouraged identifying existing ontologies that cover the concepts relevant for the particular level within the methodology. By subdividing ontologies into smaller parts (e.g., domains and fragments), the resulting ontologies – except for the top-level application-specific ontology – can also be reused in other applications and domains. The fragments were then re-combined into more complex ontologies during the bottom-up phase of the methodology.

### 9.2 Critical reflection

While the above discussions highlight the findings and contributions of this thesis, disclosing the various problems, challenges, and insights that arose while conducting the underlying research is equally important. Therefore, several issues that could not be sufficiently addressed in this work are critically discussed in the following.

#### 9.2.1 Distributed MAS as system architecture for the SG

A MAS was chosen as a system architecture for the SG application presented in this thesis. Furthermore, by examining the definitions formulated by Wooldridge, it was concluded that a MAS should be a distributed system with a very high degree of independence of its components. The individual agents are equal in the sense that no agent exercises direct control over any other agent. This type of MAS is less well researched compared to the more common centralized and multi-level hierarchical approaches. While a fully distributed approach provides benefits like inherent scalability and resilience in case of failures of individual agents, the algorithms each agent can execute are restricted due to its limited knowledge. The presented MAS can only perform localized optimization by switching a single area between transformers in each optimization run. Higher-level agents could be added to the system in a two-level or three-level hierarchical approach to enable optimization of larger portions of the distribution network.

#### 9.2.2 Switching optimization use case

The presented evaluation scenarios indicate that distributed MASs are a suitable paradigm to address the challenges of today's distribution networks. However, while switching optimization is a nice and illustrative use case, first simulation results suggest that the achievable energy savings alone will, in most cases, not justify the additional hardware and implementation effort. In particular, changes to the initial configuration of the distribution network will often increase the average line length between transformers and loads, which limits the potential savings. Just like transformer losses, these line losses could be modeled and considered by the agents when deciding if the distribution network shall be reconfigured. Additional simulations with larger distribution networks are required to provide better estimations of the achievable energy savings with a distributed MAS-based approach.



### 9.2.3 Improving dependability

The work presented in this thesis aims at incorporating non-functional attributes in MASs enable considering dependability in the decision-making-processes of agents. It was shown that this can increase the performance of the switching optimization use case, i.e., further increase the energy savings compared to a system that only considers functional aspects to reduce losses. However, while it is supported by the presented SELC, the thesis did not investigate in detail how modeling and considering availability could improve the dependability of the overall SG. Other use cases like Fault Localization, Isolation, and Supply Restoration (FLISR) would be more suitable to investigate this matter.

## 9.3 Future work

This thesis provides the groundwork to incorporate dependability considerations in MASs and showed that this approach is suitable to improve the performance in certain use cases. However, there still remain some open questions and challenges for the system to be applicable in real-world scenarios. The effects of the communication network such as delay (possibly triggering unexpected timeouts) and message loss have to be investigated, for which FNCS provides a profound basis. MASs in general require mechanisms for agent registration and discovery. Implementing them on a number of different or even on a single node, e.g., some sort of Discovery Agent, could quickly compromise the overall idea of a fully distributed approach.

Regarding the developed ontologies, an open topic to be discussed is their reusability during the SELC and beyond, including operation and decommission of the system. For this purpose, it may be beneficial to separate non-functional requirements among the various SELC phases while conducting the use case IEC 62559 methodology. Some attributes, e.g., scalability are more relevant during design time and influence how the system is designed and built, while others, e.g., availability, are more relevant during runtime and influence how the system operates. As a final note on the topic of ontologies, the reuse of existing information may be restricted by copyright and licensing issues. Further information about *legal aspects of linked data* can be found in [262].

Regarding the switching optimization use case, the various dependability attributes have to be balanced with the KPIs of the use case, e.g., the relative loss reduction. Whether or not to select a more available component at the cost of a less-optimal configuration is itself an optimization problem that quickly becomes increasingly complex if multiple dependability attributes shall be considered in the decision-making process.

As argued, the energy savings achievable with the switching optimization use case alone will likely not justify the additional hardware and implementation effort required to deploy the system in existing distribution networks. However, additional use cases may easily be implemented on the same devices following the methodology presented in this thesis. In particular if other opportunities like reducing the time to identify and

## 9. CONCLUSION AND FUTURE WORK

---

recover from faults or addressing the challenges arising from increasing energy demands in existing distribution networks are taken into account, adding communication and control capabilities to power system components may quickly become economically viable. Such applications are investigated in detail in the ongoing Power System Cognification (PoSyCo) project [29].

# Acronyms

- 6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks. 20
- AAL Ambient Assisted Living. 87
- AAS Asset Administration Shell. 101, 158, 159
- ABox Assertional Box. 9, 11, 137
- ACC Agent Communication Channel. 50
- ACL Agent Communication Language. 7, 46, 48, 49, 119, 147, 159
- AI Artificial Intelligence. 39
- AID Agent IDentifier. 47, 49
- AMI Advanced Metering Infrastructure. 20
- AMS Agent Management System. 46, 47, 50, 159
- ANSI American National Standards Institute. 17
- AP Agent Platform. 43–47, 49, 50, 60–62, 102, 106, 120, 132, 135
- API Application Programming Interface. 50, 60, 136
- ASM Agent Security Manager. 50
- ATL Alternating-time Temporal Logic. 58
- BGL Boost Graph Library. 147
- CA Communicative Act. 48
- CAA Component Administration Agent. 102, 135, 158, 159
- CB Category Bag. 120
- CDF Cumulative Distribution Function. 6, 87, 126, 140
- CENELEC Comité Européen de Normalisation Électrotechnique. 18

- CFP Call For Proposal. 106, 111, 112, 150
- CIA Confidentiality, Integrity, and Availability. 29–31, 93
- CIM Common Information Model. 20, 120
- CIRCA Cooperative Intelligent Real-Time Control Architecture. 59
- CMOS Complementary Metal-Oxide-Semiconductor. 87
- COSMO COMmon Semantic MOdel. 119
- CPS Cyber-Physical System. 1, 7
- CPU Central Processing Unit. 49
- CSMA/CD Carrier Sense Multiple Access/Collision Detection. 146
- CSV Comma-Separated Values. 147
- CTL Computation Tree Logic. 58
- CTR Common Technical Requirement. 31, 32, 95
- DAML DARPA Agent Markup Language. 52
- DE Domain Expert. 55
- DER Distributed Energy Resource. 1, 2, 4, 100, 120
- DF Directory Facilitator. 46, 47, 49, 50, 159
- DIAMOND Decentralized Iterative Multiagent Open Networks Design. 33, 60
- DIN Deutsches Institut für Normung. 18
- DL Description Logic. 9
- DMS Distribution Management System. 4
- DNO Distribution Network Operator. 77
- DNS Domain Name System. 62
- DUL DOLCE+DnS UltraLite. 119
- EA Enterprise Architect. 26, 27
- EMS Energy Management System. 4, 18
- EtherCAT Ethernet for Control Automation Technology. 19
- ETSI European Telecommunications Standards Institute. 18
- EV Electric Vehicle. 4, 162

- FAME Framework for Agent-oriented Method Engineering. 61
- FAML FAME Agent-oriented Modeling Language. 61
- FB Function Block. 21
- FHA Functional Hazard Assessment. 89
- FIPA Foundation for Intelligent Physical Agents. 39, 41–46, 48–50, 102, 106, 110, 119, 120, 122, 124, 127, 137, 147, 158, 159
- FIPS Federal Information Processing Standard. 86
- FLISR Fault Localization, Isolation, and Supply Restoration. 161
- FMEA Failure Mode and Effects Analysis. 89
- FNCS Framework for Network Co-Simulation. 136, 146–148, 161
- FTA Fault Tree Analysis. 89
- GRC Governance, Risk, and Compliance. 31, 32, 95
- GUI Graphical User Interface. 50, 58
- HTML HyperText Markup Language. 11
- HTTP HyperText Transfer Protocol. 48, 49
- HV High Voltage. 2, 123
- I/O Input/Output. 39, 103
- ICT Information and Communication Technology. 1, 3, 73, 75, 145, 146
- IE Industrial Ethernet. 103
- IEC International Electrotechnical Commission. 18, 20
- IED Intelligent Electronic Device. 20, 74, 75, 77, 80
- IEEE Institute of Electrical and Electronics Engineers. 17
- ILC Intelligent Load Controller. 57
- IoT Internet of Things. 1, 5, 6, 9, 20, 86, 118, 120, 121
- IP Internet Protocol. 120
- IPSL Interpreted Systems Programming Language. 58
- IPv6 Internet Protocol version 6. 143
- ISA International Society of Automation. 17, 20, 86

- ISO International Organization for Standardization. 17
- ITL Interval Temporal Logic. 58, 59
- ITS Intelligent Transportation System. 87
- JADE Java Agent DEvelopment framework. 43, 49, 50, 60, 135
- KE Knowledge Engineer. 55, 56
- KPI Key Performance Indicator. 23, 66, 145, 151, 161
- KR Knowledge Representation. 2, 9, 135
- LDAP Lightweight Directory Access Protocol. 49, 62
- LIC Logical Interface Category. 29–32, 93
- LV Low Voltage. 2, 3, 15, 84, 85, 104, 123, 157
- M-Bus Meter-Bus. 20
- MABD Multi-Agent Behavior Description. 43, 133, 134
- MAS Multi-Agent System. 1, 2, 7, 8, 14, 15, 17, 21, 32–34, 36, 38–44, 46, 56–63, 65, 67–69, 75, 83, 85, 99, 100, 104–107, 110, 117–121, 131, 132, 136, 137, 143, 145, 149, 151, 157–161
- MASD Multi-Agent Structure Definition. 42, 43, 132, 133
- MaSE Multiagent Systems Engineering. 36, 38, 39, 44, 45, 106
- MCMAS Model Checker for Multi-Agent Systems. 58
- MDE Model-Driven Engineering. 21, 26, 50
- MDE-CIM Computational Independent Model. 26
- MQTT Message Queuing Telemetry Transport. 143
- MTS Message Transport System. 46–48, 50, 159
- MTTF Mean Time To Failure. 140
- MTTR Mean Time To Repair. 141
- MV Medium Voltage. 2, 3, 123
- NILM Non-Intrusive Load Monitoring. 90
- NIST National Institute of Standards and Technology. 17, 28–31
- NISTIR National Institute of Standards and Technology Interagency Report. 28–31, 83, 86, 91, 92, 95

- NP Non-deterministic Polynomial-time. 84
- NS NameSpace. 9, 10
- NS3 Network Simulator 3. 146, 147
- OOD Object-Oriented Design. 36, 39, 51
- OOP Object-Oriented Programming. 33, 39, 43, 52, 133, 158
- OPAL Object, Process, Actor modelling Language. 56
- OPC UA OPC Unified Architecture. 20
- ORM Object Role Model. 38
- OS Operating System. 49, 136
- OWL Web Ontology Language. 9, 49, 56, 119, 122
- PASSI Process for Agent Societies Specification and Implementation. 14, 39–46, 69, 99, 106, 107, 109, 110, 131–133, 135, 158
- PDF Probability Density Function. 6, 87, 126, 140, 141
- PIM Platform Independent Model. 26
- PIN Personal Identification Number. 60
- PM Preventive Maintenance. 89
- PoSCo Power System Cognification. 162
- PROFIBUS Process Field Bus. 19
- ProSG Prosumer-Oriented Smart Grid. 120
- PSI Platform Specific Implementation. 26, 27
- PSM Platform Specific Model. 26, 27
- PTK PASSI Tool Kit. 43, 44, 135
- PV PhotoVoltaic. 4
- QoS Quality of Service. 49, 118, 120
- QoS-MO Quality of Service Modeling Ontology. 120
- RAMI 4.0 Reference Architectural Model for Industrie 4.0. 101
- RBAC Role-Based Access Control. 97, 98
- RDF Resource Description Framework. 9, 11, 20, 49, 56, 119

- RDFS Resource Description Framework Schema. 9, 20, 120
- RMP Risk Management Process. 29
- SABD Single-Agent Behavior Description. 43, 134
- SASD Single-Agent Structure Definition. 42, 132, 133
- SCADA Supervisory Control and Data Acquisition. 4, 93, 100
- SDLC Systems Development Life Cycle. 65
- SELC Systems Engineering Life Cycle. 1, 11–14, 44, 62, 65, 66, 68, 69, 71, 145, 153, 157, 159, 161
- SG Smart Grid. 1–4, 7, 11, 13–15, 17–22, 26, 28, 29, 31, 44, 50, 56, 57, 65, 66, 68, 69, 71, 75, 82, 85–87, 89, 95, 99–101, 103, 106, 118, 119, 136, 137, 143, 145, 146, 157–161
- SGAM Smart Grid Architecture Model. 3, 4, 17–19, 21, 22, 26, 27, 101, 159
- SIL Safety Integrity Level. 19, 88, 127
- SL Semantic Language. 49
- SOA Service-Oriented Architecture. 7, 100
- SPARQL SPARQL Protocol and RDF Query Language. 136
- SPoF Single Point of Failure. 85, 86, 100
- TBox Terminological Box. 9, 11, 137
- TCP Transmission Control Protocol. 120
- TLS Transport Layer Security. 120, 141, 142
- TSN Time-Sensitive Networking. 20
- UCMP Use Case Mapping Process. 26
- UCMR Use Case Management Repository. 22
- UML Unified Modeling Language. 23, 26, 39, 41, 44, 116
- UP Unified Software Development Process / Unified Process. 53
- UPON UP for ONtology. 51, 53–55
- URL Uniform Resource Locator. 24
- UTR Unique Technical Requirement. 31, 32, 95
- VOWL Visual Notation for OWL Ontologies. 9, 10



W3C World Wide Web Consortium. 9, 49, 120

Wireless M-Bus Wireless Meter-Bus. 20

wirelessHART wireless Highway Addressable Remote Transducer protocol. 20

WLAN Wireless Local Area Network. 20

XML eXtensible Markup Language. 11, 22, 48

XSD XML Schema Definition. 56



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.  
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

# Bibliography

- [1] V Cagri Gungor, Dilan Sahin, Taskin Kocak, Salih Ergut, Concettina Buccella, Carlo Cecati, and Gerhard P Hancke. “A survey on smart grid potential applications and communication requirements”. In: *IEEE Transactions on Industrial Informatics* 9.1 (2013), pp. 28–42.
- [2] A Muir and J Lopatto. *Final report on the August 14, 2003 blackout in the United States and Canada: causes and recommendations*. Accessed: March 22, 2021. 2004. URL: <https://www.energy.gov/sites/default/files/oeprod/DocumentsandMedia/BlackoutFinal-Web.pdf>.
- [3] International Electrotechnical Commission (IEC). *IEC 60038:2009 IEC standard voltages*. 2009.
- [4] Deepjyoti Deka, Scott Backhaus, and Michael Chertkov. “Structure learning and statistical estimation in distribution networks-part i”. In: *arXiv preprint arXiv:1501.04131* (2015).
- [5] Smart Grid Coordination CEN-CENELEC-ETSI. “Group.(2012)”. In: *Smart Grid Reference Architecture* (2012), pp. 1–107.
- [6] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, et al. *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.
- [7] Lukas Krammer. “Dependability in building automation networks”. PhD thesis. TU Wien, 2014.
- [8] D.R. Law. “Scalable means more than more: a unifying definition of simulation scalability”. In: *Simulation Conference Proceedings*. Vol. 1. Dec. 1998, 781–788 vol.1. DOI: 10.1109/WSC.1998.745064.
- [9] Paul Madsen, NTT Marco Cassasa Mont, and Robin Wilton. *A Privacy Policy Framework—A position paper for the W3C Workshop of Privacy Policy Negotiation*. 2006.

- [10] H. Kopetz. “On the Fault Hypothesis for a Safety-Critical Real-Time System”. English. In: *Automotive Software-Connected Services in Mobile Networks*. Ed. by Manfred Broy, Ingolf H. Krueger, and Michael Meisinger. Vol. 4147. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 31–42. ISBN: 978-3-540-37677-4. DOI: 10.1007/11823063\\_3. URL: [http://dx.doi.org/10.1007/11823063\\_3](http://dx.doi.org/10.1007/11823063_3).
- [11] Stephen DJ McArthur, Euan M Davidson, Victoria M Catterson, Aris L Dimeas, Nikos D Hatziargyriou, Ferdinanda Ponci, and Toshihisa Funabashi. “Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges”. In: *IEEE Transactions on Power systems* 22.4 (2007), pp. 1743–1752.
- [12] Stephen DJ McArthur, Euan M Davidson, Victoria M Catterson, Aris L Dimeas, Nikos D Hatziargyriou, Ferdinanda Ponci, and Toshihisa Funabashi. “Multi-agent systems for power engineering applications—Part II: Technologies, standards, and tools for building multi-agent systems”. In: *IEEE Transactions on Power Systems* 22.4 (2007), pp. 1753–1759.
- [13] M. Wooldridge and Ed. G. Weiss. “Intelligent Agents”. In: *Multi-agent Systems* 9 (1999), pp. 3–51.
- [14] Stan Franklin and Art Graesser. “Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents”. In: *International Workshop on Agent Theories, Architectures, and Languages*. Springer. 1996, pp. 21–35.
- [15] Michael Wooldridge. “Agent-based software engineering”. In: *IEE Proceedings-Software Engineering* 144.1 (1997), pp. 26–37.
- [16] Dayong Ye, Minjie Zhang, and Athanasios V Vasilakos. “A Survey of Self-Organization Mechanisms in Multiagent Systems”. In: *IEEE Trans. Systems, Man, and Cybernetics: Systems* 47.3 (2017), pp. 441–461.
- [17] Paul Baran. “On distributed communications networks”. In: *IEEE transactions on Communications Systems* 12.1 (1964), pp. 1–9.
- [18] Wei Wang, Suparna De, Ralf Toenjes, Eike Reetz, and Klaus Moessner. “A comprehensive ontology for knowledge representation in the internet of things”. In: *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE. 2012, pp. 1793–1798.
- [19] Dean Allemang and James Hendler. *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.
- [20] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. “Visualizing ontologies with VOWL”. In: *Semantic Web* 7.4 (2016), pp. 399–419.
- [21] Natalya F Noy, Deborah L McGuinness, et al. *Ontology development 101: A guide to creating your first ontology*. 2001.

- [22] Vojtech Svátek and Ondrej Šváb-Zamazal. “Entity naming in semantic web ontologies: Design patterns and empirical observations”. In: *University of Economics, Prague, Czech Republic* (2010), pp. 1–12.
- [23] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, and Chris Wroe. “A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.3”. In: *University of Manchester* (2011).
- [24] Filip Prössl Andrén. “Model-Driven Engineering for Smart Grid Automation”. PhD thesis. TU Wien, 2018.
- [25] T. Frühwirth, L. Krammer, and W. Kastner. “Dependability Demands and State of the Art in the Internet of Things”. In: *Proceedings of the 20th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Luxembourg, Sept. 2015, pp. 1–4.
- [26] T. Frühwirth, A. Einfalt, K. Diwold, and W. Kastner. “A distributed multi-agent system for switching optimization in low-voltage power grids”. In: *Proceedings of the 22nd IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Limassol, Cyprus, Sept. 2017, pp. 1–8.
- [27] T. Frühwirth, L. Krammer, and W. Kastner. “A methodology for creating reusable ontologies”. In: *Proceedings of the 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. Saint Petersburg, Russia, May 2018, pp. 65–70.
- [28] G. Steindl, T. Frühwirth, and W. Kastner. “Ontology-Based OPC UA Data Access via Custom Property Functions”. In: *Proceedings of the 24th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Zaragoza, Spain, Sept. 2019, pp. 95–101.
- [29] D. Herbst, M. Lager, R. Schürhuber, E. Schmautzer, L. Fickert, A. Einfalt, H. Brunner, D. Schultis, T. Frühwirth, and W. Prügler. “Zukünftige Anforderungen an NS-Netze und deren Lösungsansätze am Beispiel PoSyCo”. In: *16. Symposium Energieinnovation*. Graz, Austria, Feb. 2020, pp. 1–11.
- [30] International Electrotechnical Commission (IEC). *Smart Grid Standards Map*. Accessed: March 12, 2021. URL: <http://smartgridstandardsmap.com>.
- [31] International Electrotechnical Commission (IEC). *IEC 60059:1999+AMD1:2009 CSV Consolidated version IEC standard current ratings*. 1999.
- [32] International Electrotechnical Commission (IEC). *IEC 60196:2009 IEC standard frequencies*. 2009.
- [33] International Electrotechnical Commission (IEC). *IEC 60076-1:2011 Power transformers – Part 1: General*. 2011.
- [34] International Electrotechnical Commission (IEC). *IEC 60104:1987 Aluminium-magnesium-silicon alloy wire for overhead line conductors*. 1987.
- [35] International Electrotechnical Commission (IEC). *IEC 60889:1987 Hard-drawn aluminium wire for overhead line conductors*. 1987.

- [36] International Electrotechnical Commission (IEC). *IEC 61394:2011 Overhead lines – Requirements for greases for aluminium, aluminium alloy and steel bare conductors*. 2011.
- [37] International Electrotechnical Commission (IEC). *IEC 60305:1995 Insulators for overhead lines with a nominal voltage above 1000 V – Ceramic or glass insulator units for a.c. systems – Characteristics of insulator units of the cap and pin type*. 1995.
- [38] International Electrotechnical Commission (IEC). *IEC 60383-1:1993 Insulators for overhead lines with a nominal voltage above 1000 V – Part 1: Ceramic or glass insulator units for a.c. systems – Definitions, test methods and acceptance criteria*. 1993.
- [39] International Electrotechnical Commission (IEC). *IEC 61109:2008 Insulators for overhead lines – Composite suspension and tension insulators for a.c. systems with a nominal voltage greater than 1 000 V – Definitions, test methods and acceptance criteria*. 2008.
- [40] International Electrotechnical Commission (IEC). *IEC 61325:1995 Insulators for overhead lines with a nominal voltage above 1000 V – Ceramic or glass insulator units for d.c. systems – Definitions, test methods and acceptance criteria*. 1995.
- [41] International Electrotechnical Commission (IEC). *IEC 61466-1:2016 Composite string insulator units for overhead lines with a nominal voltage greater than 1 000 V – Part 1: Standard strength and end fittings*. 2016.
- [42] International Electrotechnical Commission (IEC). *IEC 61467:2008 Insulators for overhead lines – Insulator strings and sets for lines with a nominal voltage greater than 1 000 V – AC power arc tests*. 2008.
- [43] International Electrotechnical Commission (IEC). *IEC 61211:2004 Insulators of ceramic material or glass for overhead lines with a nominal voltage greater than 1 000 V – Impulse puncture testing in air*. 2004.
- [44] International Electrotechnical Commission (IEC). *IEC 61952:2008 Insulators for overhead lines – Composite line post insulators for A.C. systems with a nominal voltage greater than 1 000 V – Definitions, test methods and acceptance criteria*. 2008.
- [45] International Electrotechnical Commission (IEC). *IEC 60652:2002 Loading tests on overhead line structures*. 2002.
- [46] International Electrotechnical Commission (IEC). *IEC 60105:1958 Recommendation for commercial-purity aluminium busbar material*. 1958.
- [47] International Electrotechnical Commission (IEC). *IEC 60114:1959 Recommendation for heat-treated aluminium alloy busbar material of the aluminium-magnesium-silicon type*. 1959.

- [48] International Electrotechnical Commission (IEC). *IEC 61020-1:2009 Electromechanical switches for use in electrical and electronic equipment – Part 1: Generic specification*. 2009.
- [49] International Electrotechnical Commission (IEC). *IEC 61058-1:2016 Switches for appliances – Part 1: General requirements*. 2016.
- [50] International Electrotechnical Commission (IEC). *IEC TR 61508-0:2005 Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 0: Functional safety and IEC 61508*. 2005.
- [51] International Electrotechnical Commission (IEC). *IEC 60605-2:1994 Equipment reliability testing – Part 2: Design of test cycles*. 1994.
- [52] International Electrotechnical Commission (IEC). *IEC 61709:2017 RLV Redline version Electric components - Reliability - Reference conditions for failure rates and stress models for conversion*. 2017.
- [53] International Electrotechnical Commission (IEC). *IEC 61163-1:2006 Reliability stress screening – Part 1: Repairable assemblies manufactured in lots*. 2006.
- [54] International Electrotechnical Commission (IEC). *IEC 61070:1991 Compliance test procedures for steady-state availability*. 1991.
- [55] International Electrotechnical Commission (IEC). *IEC 60706-2:2006 Maintainability of equipment - Part 2: Maintainability requirements and studies during the design and development phase*. 2006.
- [56] International Electrotechnical Commission (IEC). *IEC 61703:2016 Mathematical expressions for reliability, availability, maintainability and maintenance support terms*. 2016.
- [57] International Electrotechnical Commission (IEC). *IEC 60793-1-1:2017 RLV Redline version Optical fibres – Part 1-1: Measurement methods and test procedures – General and guidance*. 2017.
- [58] International Electrotechnical Commission (IEC). *IEC 60794-1-1:2015 RLV Redline version Optical fibre cables - Part 1-1: Generic specification - General*. 2015.
- [59] International Electrotechnical Commission (IEC). *IEC 61158-1:2014 Industrial communication networks - Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*. 2014.
- [60] International Electrotechnical Commission (IEC). *IEC 61784-1:2014 Industrial communication networks – Profiles – Part 1: Fieldbus profiles*. 2014.
- [61] International Electrotechnical Commission (IEC). *IEC 61784-5-3:2018 Industrial communication networks – Profiles – Part 5-3: Installation of fieldbuses – Installation profiles for CPF 3*. 2018.
- [62] International Electrotechnical Commission (IEC). *IEC 61784-5-12:2018 Industrial communication networks – Profiles – Part 5-12: Installation of fieldbuses – Installation profiles for CPF 12*. 2018.

- [63] International Electrotechnical Commission (IEC). *IEC 61784-5-15:2010+AMD1: 2015 CSV Consolidated version Industrial communication networks – Profiles – Part 5-15: Installation of fieldbuses – Installation profiles for CPF 15*. 2010.
- [64] International Electrotechnical Commission (IEC). *IEC 62734 – Industrial networks – Wireless communication network and communication profiles – ISA 100.11a*. 2014.
- [65] Comité Européen de Normalisation (CEN). *EN 13757 Communication systems for meters – Part 2: Physical and link layer*. 2014.
- [66] Comité Européen de Normalisation (CEN). *EN 13757 Communication systems for meters – Part 3: Application layer*. 2014.
- [67] Comité Européen de Normalisation (CEN). *EN 13757 Communication systems for meters – Part 4: Wireless M-Bus communication*. 2014.
- [68] Institute of Electrical and Electronics Engineers (IEEE). *IEEE 802.3-2018 – IEEE Standard for Ethernet*. 2018.
- [69] Institute of Electrical and Electronics Engineers (IEEE). *IEEE 802.11-2016 – IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. 2016.
- [70] Institute of Electrical and Electronics Engineers (IEEE). *IEEE 802.15.4-2015 – IEEE Standard for Low-Rate Wireless Networks*. 2015.
- [71] ZigBee Alliance. “Zigbee specification. Technical Report Document 053474r20”. In: *Zigbee Alliance* (2012).
- [72] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. Internet Requests for Comments. RFC. Internet Engineering Task Force, Sept. 2007. URL: <http://www.rfc-editor.org/rfc/rfc4944.txt>.
- [73] Jianping Song, Song Han, Al Mok, Deji Chen, Mike Lucas, Mark Nixon, and Wally Pratt. “WirelessHART: Applying wireless technology in real-time industrial process control”. In: *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE. 2008, pp. 377–386.
- [74] International Electrotechnical Commission (IEC). *IEC TS 62443-1-1:2009 Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models*. 2009.
- [75] International Electrotechnical Commission (IEC). *IEC TS 62351-1:2007 Power systems management and associated information exchange - Data and communications security - Part 1: Communication network and system security - Introduction to security issues*. 2006.
- [76] Smart Grid Interoperability Panel Cyber Security Working Group et al. “Introduction to NISTIR 7628 guidelines for smart grid cyber security”. In: *NIST Special Publication 154* (2010).



- [77] Victoria Y Pillitteri and Tanya L Brewer. *Cybersecurity User's Guide to the Guidelines for Smart Grid Cybersecurity (NISTIR 7628 Vol. 1 2010)*. 2014.
- [78] International Electrotechnical Commission (IEC). *IEC TR 62541-1:2012 OPC unified architecture – Part 1: Overview and concepts*. 2012.
- [79] International Electrotechnical Commission (IEC). *IEC TR 61850-1:2013 Communication networks and systems for power utility automation – Part 1: Introduction and overview*. 2013.
- [80] International Electrotechnical Commission (IEC). *IEC 61970-1:2005 Energy management system application program interface (EMS-API) – Part 1: Guidelines and general requirements*. 2005.
- [81] International Electrotechnical Commission (IEC). *IEC 61970-501 Energy management system application program interface (EMS-API) – Part 501: Common Information Model Resource Description Framework (CIM RDF) schema*. Mar. 2006.
- [82] International Electrotechnical Commission (IEC). *IEC 61968-1:2012 Application integration at electric utilities – System interfaces for distribution management – Part 1: Interface architecture and general recommendations*. 2012.
- [83] International Electrotechnical Commission (IEC). *IEC 61986-13:2008 Application integration at electric utilities – System interfaces for distribution management – Part 13: CIM RDF Model exchange format for distribution*. 2008.
- [84] International Electrotechnical Commission (IEC). *IEC 60050-102:2007 International Electrotechnical Vocabulary – Part 102: Mathematics – General concepts and linear algebra*. 2007.
- [85] International Electrotechnical Commission (IEC). *IEC 60027-1:1992+AMD1:1997+AMD2:2005 CSV Consolidated version*. 1992.
- [86] International Electrotechnical Commission (IEC). *IEC 60617:2012 DB Graphical symbols for diagrams – 12-month subscription to regularly updated online database comprising parts 2 to 13 of IEC 60617*. 2012.
- [87] International Electrotechnical Commission (IEC). *IEC 61131-1:2003 Programmable controllers – Part 1: General information*. 2003.
- [88] International Electrotechnical Commission (IEC). *IEC 61131-3:2013 Programmable controllers – Part 3: Programming languages*. 2013.
- [89] International Electrotechnical Commission (IEC). *IEC 61499-1:2012 Function blocks – Part 1: Architecture*. 2012.
- [90] International Electrotechnical Commission (IEC). *IEC 62559-2:2015 Use case methodology – Part 2: Definition of the templates for use cases, actor list and requirements list*. 2015.
- [91] International Electrotechnical Commission (IEC). *IEC 62559-3:2017 Use case methodology – Part 3: Definition of use case template artefacts into an XML serialized format*. 2017.

- [92] Marion Gottschalk, Mathias Uslar, and Christina Delfs. *The Use Case and Smart Grid Architecture Model Approach: The IEC 62559-2 Use Case Template and the SGAM Applied in Various Domains*. Springer, 2017.
- [93] Marion Gottschalk and Mathias Uslar. “Ein Use Case Management Repository zur Unterstützung der Normungsarbeit”. In: *Mensch und Computer 2015–Workshopband* (2015).
- [94] Center for Secure Energy Informatics (EN-TRUST). *SGAM Toolbox*. Accessed: March 12, 2021. URL: <https://www.en-trust.at/downloads/sgam-toolbox/>.
- [95] Christian Dänekas, Christian Neureiter, Sebastian Rohjans, Mathias Uslar, and Dominik Engel. “Towards a model-driven-architecture process for smart grid projects”. In: *Digital enterprise design & management*. Springer, 2014, pp. 47–58.
- [96] Christian Neureiter. *A Domain-Specific, Model Driven Engineering Approach for Systems Engineering in the Smart Grid*. MBSE4U - Tim Weikiens, 2017. URL: <https://www.amazon.de/Domain-Specific-Driven-Engineering-Approach-Systems/dp/3981852923/>.
- [97] Sparx Systems. *Enterprise Architect*. Accessed: March 12, 2021. URL: <https://www.sparxsystems.de/uml/ea-function/>.
- [98] International Electrotechnical Commission (IEC). *IEC 15408-1:2009 Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model*. 2009.
- [99] International Electrotechnical Commission (IEC). *IEC 27001:2013 Information technology – Security techniques – Information security management systems – Requirements*. 2013.
- [100] PG Urbano, Th Wagner, and P Göhrner. “Softwareagenten–Einführung und Überblick über eine alternative Art der Softwareentwicklung. Teil 1: Agentenorientierte Softwareentwicklung”. In: *Automatisierungstechnische Praxis atp* 45 (2003), pp. 48–57.
- [101] Bernhard Bauer, Jörg P Müller, and James Odell. “Agent UML: A formalism for specifying multiagent software systems”. In: *International journal of software engineering and knowledge engineering* 11.03 (2001), pp. 207–230.
- [102] Jean-Paul Jamont and Michel Ocello. “Designing embedded collective systems: The DIAMOND multiagent method”. In: *IEEE International Conference on Tools with Artificial Intelligence-ICTAI 07*. IEEE Computer Society. 2007, pp. 91–94.
- [103] Michael Wooldridge, Nicholas R Jennings, and David Kinny. “The Gaia methodology for agent-oriented analysis and design”. In: *Autonomous Agents and multi-agent systems* 3.3 (2000), pp. 285–312.
- [104] James E Lovelock and Lynn Margulis. “Atmospheric homeostasis by and for the biosphere: the Gaia hypothesis”. In: *Tellus* 26.1-2 (1974), pp. 2–10.

- [105] Bowen Alpern and Fred B Schneider. “Defining liveness”. In: *Information processing letters* 21.4 (1985), pp. 181–185.
- [106] Mark F Wood and Scott A DeLoach. “An overview of the multiagent systems engineering methodology”. In: *International Workshop on Agent-Oriented Software Engineering*. Springer. 2000, pp. 207–221.
- [107] Mark F Wood. *Multiagent systems engineering: A methodology for analysis and design of multiagent systems*. 2000.
- [108] David J Robinson. *A component based approach to agent specification*. 2000.
- [109] Massimo Cossentino. “From requirements to code with PASSI methodology”. In: *Agent-oriented methodologies*. IGI Global, 2005, pp. 79–106.
- [110] Federico Bergenti and Agostino Poggi. “Exploiting UML in the design of multi-agent systems”. In: *International Workshop on Engineering Societies in the Agents World*. Springer. 2000, pp. 106–113.
- [111] Stephen Cranefield and Martin Kent Purvis. *UML as an ontology modelling language*. Department of Information Science, University of Otago New Zealand, 1999.
- [112] Massimo Cossentino, Luca Sabatucci, and Antonio Chella. “Patterns reuse in the PASSI methodology”. In: *International Workshop on Engineering Societies in the Agents World*. Springer. 2003, pp. 294–310.
- [113] Luca Sabatucci, Massimo Cossentino, and Salvatore Gaglio. “Pattern Repository and Reuse in the PASSI Methodology”. In: (2010).
- [114] Massimo Cossentino. *PASSI Toolkit*. Accessed: March 12, 2021. URL: <https://sourceforge.net/projects/ptk/>.
- [115] Massimo Cossentino. *PASSI Agent Factory*. Accessed: March 12, 2021. URL: <https://sourceforge.net/projects/ptk/files/AgentFactory/>.
- [116] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. “JADE – A FIPA-compliant agent framework”. In: *Proceedings of PAAM*. Vol. 99. London. 1999, p. 33.
- [117] Zina Mecibah and Fateh Boutekkouk. “Comparative study between Multi Agents Systems methodologies according to intelligent embedded systems requirements”. In: *The 4th International Conference on Automation, Control Engineering and Computer Science (ACECS-2017)*. 2017, pp. 28–30.
- [118] Quynh-Nhu Numi Tran, Graham Low, and Mary-Anne Williams. “A preliminary comparative feature analysis of multi-agent systems development methodologies”. In: *International Bi-Conference Workshop on Agent-Oriented Information Systems*. Springer. 2004, pp. 157–168.
- [119] Emilia Garcia, Adriana Giret, and Vicente Botti. “An evaluation tool for multiagent development techniques”. In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems. 2010, pp. 1625–1626.

- [120] Khanh Hoa Dam and Michael Winikoff. “Comparing agent-oriented methodologies”. In: *International Bi-Conference Workshop on Agent-Oriented Information Systems*. Springer. 2003, pp. 78–93.
- [121] FIPA TC Nomadic Application Support. *FIPA Nomadic Application Support Specification*. 2002.
- [122] IEEE FIPA DPDF Working Group. *Design Process Documentation Template*. 2012.
- [123] Foundation for Intelligent Physical Agents (FIPA). *Standard FIPA specifications*. Accessed: March 12, 2021. URL: <http://www.fipa.org/repository/standardspecs.html>.
- [124] FIPA TC Architecture. *FIPA Abstract Architecture Specification*. 2002.
- [125] FIPA TC Agent Management. *FIPA Agent Management Specification*. 2002.
- [126] FIPA TC Agent Management. *FIPA ACL Message Representation in String Specification*. 2002.
- [127] FIPA TC Communication. *FIPA ACL Message Structure Specification*. 2002.
- [128] FIPA TC Communication. *FIPA Communicative Act Library Specification*. 2002.
- [129] FIPA TC Communication. *FIPA Query Interaction Protocol Specification*. 2002.
- [130] FIPA TC Communication. *FIPA Contract Net Interaction Protocol Specification*. 2002.
- [131] FIPA TC Communication. *FIPA SL Content Language Specification*. 2002.
- [132] FIPA TC Gateways. *FIPA Device Ontology Specification*. 2002.
- [133] FIPA TC Nomadic Application Support. *FIPA Quality of Service Ontology Specification*. 2002.
- [134] Bo Chen, Harry H Cheng, and Joe Palen. “Mobile-C: a mobile agent platform for mobile C/C++ agents”. In: *Software: Practice and Experience* 36.15 (2006), pp. 1711–1733.
- [135] Danny B Lange and Oshima Mitsuru. *Programming and Deploying Java Mobile Agents Aglets*. Addison-Wesley Longman Publishing Co., Inc., 1998.
- [136] Robert S Gray. “Agent Tcl: A flexible and secure mobile-agent system”. In: *Tcl/Tk Workshop*. 1996.
- [137] David Wong, Noemi Paciorek, Tom Walsh, Joe DiCelie, Mike Young, and Bill Peet. “Concordia: An infrastructure for collaborating mobile agents”. In: *International workshop on mobile agents*. Springer. 1997, pp. 86–97.
- [138] Robert S Gray, George Cybenko, David Kotz, Ronald A Peterson, and Daniela Rus. “D’Agents: Applications and performance of a mobile-agent system”. In: *Software: Practice and Experience* 32.6 (2002), pp. 543–573.

- [139] Hyacinth S Nwana, Divine T Ndumu, Lyndon C Lee, et al. “ZEUS: An advanced tool-kit for engineering distributed multi-agent systems”. In: *Proceedings of PAAM*. Vol. 98. 1998, pp. 377–391.
- [140] Joachim Baumann, Fritz Hohl, Kurt Rothermel, Markus Strasser, and Wolfgang Theilmann. “MOLE: A mobile agent system”. In: *Software: Practice and Experience* 32.6 (2002), pp. 575–603.
- [141] Fabio Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. “JADE: A software framework for developing multi-agent applications. Lessons learned”. In: *Information and Software Technology* 50.1-2 (2008), pp. 10–21.
- [142] Yu-Cheng Chou, David Ko, and Harry H Cheng. “An embeddable mobile agent platform supporting runtime code mobility, interaction and coordination of mobile agents and host systems”. In: *Information and software technology* 52.2 (2010), pp. 185–196.
- [143] HH Cheng. *Ch—an Embeddable C/C++ Interpreter*. 2009.
- [144] Antonio De Nicola and Michele Missikoff. “A lightweight methodology for rapid ontology engineering”. In: *Communications of the ACM* 59.3 (2016), pp. 79–86.
- [145] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. “Methodology: from ontological art towards ontological engineering”. In: (1997).
- [146] York Sure, Steffen Staab, and Rudi Studer. “On-to-knowledge methodology (OTKM)”. In: *Handbook on ontologies*. Springer, 2004, pp. 117–132.
- [147] Antonio De Nicola, Michele Missikoff, and Roberto Navigli. “A software engineering approach to ontology building”. In: *Information systems* 34.2 (2009), pp. 258–275.
- [148] Matteo Cristani and Roberta Cuel. “A survey on ontology creation methodologies”. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 1.2 (2005), pp. 49–69.
- [149] Claudia Zanabria. “Adaptable engineering support framework for multi-functional battery energy storage systems”. PhD thesis. TU Wien, 2018.
- [150] Stanford University. *Protégé*. Accessed: March 12, 2021. URL: <https://protege.stanford.edu/>.
- [151] Michael Grüninger and Mark S Fox. “Methodology for the design and evaluation of ontologies”. In: (1995).
- [152] Stanford University. *Ontolingua*. Accessed: March 12, 2021. URL: <http://www.ksl.stanford.edu/software/ontolingua/>.
- [153] Defense Advanced Research Projects Agency (DARPA). *DAML Ontology Library*. Accessed: March 12, 2021. URL: <http://www.daml.org/ontologies/>.
- [154] Mike Uschold and Michael Gruninger. “Ontologies: Principles, methods and applications”. In: *The knowledge engineering review* 11.2 (1996), pp. 93–136.
- [155] Ivar Jacobson. *The unified software development process*. Pearson Education India, 1999.

- [156] Fulvio D’Antonio, Michele Missikoff, and Francesco Taglino. “Formalizing the OPAL eBusiness ontology design patterns with OWL”. In: *Enterprise Interoperability II*. Springer, 2007, pp. 345–356.
- [157] Abhilash Kantamneni, Laura E Brown, Gordon Parker, and Wayne W Weaver. “Survey of multi-agent systems for microgrid control”. In: *Engineering applications of artificial intelligence* 45 (2015), pp. 192–203.
- [158] Robin Roche, Benjamin Blunier, Abdellatif Miraoui, Vincent Hilaire, and Abder Koukam. “Multi-agent systems for grid energy management: A short review”. In: *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE. 2010, pp. 3341–3346.
- [159] AL Dimeas and ND Hatziargyriou. “Control agents for real microgrids”. In: *Intelligent System Applications to Power Systems, 2009. ISAP’09. 15th International Conference on*. IEEE. 2009, pp. 1–5.
- [160] Munir Merdan, Wilfried Lepuschitz, Thomas Strasser, and Filip Andren. “Multi-Agent system for self-optimizing power distribution grids”. In: *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*. IEEE. 2011, pp. 312–317.
- [161] Zhenhua Jiang. “Agent-based control framework for distributed energy resources microgrids”. In: *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*. IEEE Computer Society. 2006, pp. 646–652.
- [162] Mehdi Dastani, Koen V Hindriks, and John-Jules Meyer. *Specification and verification of multi-agent systems*. Springer Science & Business Media, 2010.
- [163] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. “MCMAS: A model checker for the verification of multi-agent systems”. In: *International conference on computer aided verification*. Springer. 2009, pp. 682–688.
- [164] Rajeev Alur, Thomas A Henzinger, and Orna Kupferman. “Alternating-time temporal logic”. In: *Journal of the ACM (JACM)* 49.5 (2002), pp. 672–713.
- [165] Robert S Boyer and J Strother Moore. *A computational logic handbook: Formerly notes and reports in computer science and applied mathematics*. Elsevier, 2014.
- [166] Wojciech Penczek and Alessio Lomuscio. “Verifying epistemic properties of multi-agent systems via bounded model checking”. In: *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM. 2003, pp. 209–216.
- [167] Michael Wooldridge, Michael Fisher, Marc-Philippe Huget, and Simon Parsons. “Model checking multi-agent systems with MABLE”. In: *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*. ACM. 2002, pp. 952–959.
- [168] Franco Raimondi and Alessio Lomuscio. “Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams”. In: *Journal of Applied Logic* 5.2 (2007), pp. 235–251.

- [169] Rafael H Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge. “Verifying multi-agent programs by model checking”. In: *Autonomous agents and multi-agent systems* 12.2 (2006), pp. 239–256.
- [170] Mika Cohen, Mads Dam, Alessio Lomuscio, and Francesco Russo. “Abstraction in model checking multi-agent systems”. In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems. 2009, pp. 945–952.
- [171] Stefania Costantini. “Self-checking logical agents”. In: *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2013, pp. 1329–1330.
- [172] Stefania Costantini and Giovanni De Gasperis. “Runtime Self-Checking via Temporal (Meta-) Axioms for Assurance of Logical Agent Systems”. In: *CILC*. Citeseer. 2014, pp. 241–255.
- [173] James F Allen and George Ferguson. “Actions and events in interval temporal logic”. In: *Journal of logic and computation* 4.5 (1994), pp. 531–579.
- [174] Staffan Haegg. “A sentinel approach to fault handling in multi-agent systems”. In: *Australian Workshop on Distributed Artificial Intelligence*. Springer. 1996, pp. 181–195.
- [175] Keinosuke Matsumoto, Tomoaki Maruo, and Akifumi Tanimoto. “A dependable multi-agent system with self-diagnostic function”. In: *Parallel and Distributed Computing, 2009. ISPDC’09. Eighth International Symposium on*. IEEE. 2009, pp. 213–217.
- [176] T Kohda, K Yoshida, Y Sujaku, et al. “Decentralized self-diagnosis in multi-agent systems”. In: *Proceeding of the 6th Multi-Agent and Cooperative Computing Workshop*. 1997.
- [177] Peter C Lockemann and Jens Nimis. “Agent dependability as an architectural issue”. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM. 2006, pp. 1101–1103.
- [178] Peter C Lockemann and Jens Nimis. “Dependable multi-agent systems: layered reference architecture and representative mechanisms”. In: *Safety and Security in Multiagent Systems*. Springer, 2009, pp. 27–48.
- [179] David J Musliner, Edmund H Durfee, and Kang G Shin. “CIRCA: A cooperative intelligent real-time control architecture”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 23.6 (1993), pp. 1561–1574.
- [180] Holger Giese, Sven Burmester, Florian Klein, Daniela Schilling, and Matthias Tichy. “Multi-agent system design for safety-critical self-optimizing mechatronic systems with UML”. In: *OOPSLA*. 2003, pp. 21–32.

- [181] Barbara Hayes-Roth. “Architectural foundations for real-time performance in intelligent agents”. In: *Real-Time Systems 2.1-2* (1990), pp. 99–125.
- [182] Jean-Paul Jamont, Clément Raievsky, and Michel Ocello. “Handling safety-related non-functional requirements in embedded multi-agent system design”. In: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer. 2014, pp. 159–170.
- [183] Niklas Borselius. “Security in multi-agent systems”. In: *Proceedings of the 2002 international conference on security and management (SAM’02)*. 2002, pp. 31–36.
- [184] MA Oey, M Warnier, and FMT Brazier. “Security in large-scale open distributed multi-agent systems”. In: *Autonomous Agents*. Intech. 2010.
- [185] Rodolfo Carneiro Cavalcante, Ig Ibert Bittencourt, Alan Pedro da Silva, Marlos Silva, Evandro Costa, and Robério Santos. “A survey of security in multi-agent systems”. In: *Expert Systems with Applications* 39.5 (2012), pp. 4835–4846.
- [186] Yaqin Hedin and Esmiralda Moradian. “Security in multi-agent systems”. In: *Procedia Computer Science* 60 (2015), pp. 1604–1612.
- [187] H Chi Wong and Katia Sycara. “Adding security and trust to multiagent systems”. In: *Applied Artificial Intelligence* 14.9 (2000), pp. 927–941.
- [188] Lars Braubach, Kai Jander, and Alexander Pokahr. “A Practical Security Infrastructure for Distributed Agent Applications”. In: *German Conference on Multiagent System Technologies*. Springer. 2013, pp. 29–43.
- [189] X Vila, A Schuster, and Adolfo Riera. “Security for a Multi-Agent System based on JADE”. In: *computers & security* 26.5 (2007), pp. 391–400.
- [190] Sarvapali D Ramchurn, Dong Huynh, and Nicholas R Jennings. “Trust in multi-agent systems”. In: *The Knowledge Engineering Review* 19.1 (2004), pp. 1–25.
- [191] Agostino Poggi, Giovanni Rimassa, and Michele Tomaiuolo. “Multi-user and security support for multi-agent systems”. In: *Proceedings of WOA 2001 (Dagli oggetti agli*. Citeseer. 2001.
- [192] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. “Role-Based Access Control in MAS using Agent Coordination Contexts”. In: *1st International Workshop “Agent Organizations: Theory and Practice”(AOTP’04)*. 2004, pp. 15–22.
- [193] Pongsin Pooankam et al. “Authentication and Access Control in Multi-agent Systems”. In: (2008).
- [194] Petr Novák, Milan Rollo, Jiří Hodík, and Tomáš Vlček. “Communication security in multi-agent systems”. In: *International Central and Eastern European Conference on Multi-Agent Systems*. Springer. 2003, pp. 454–463.
- [195] Praveen Paruchuri, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. “Security in multiagent systems by policy randomization”. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM. 2006, pp. 273–280.



- [196] Haralambos Mouratidis, Paolo Giorgini, and Gordon Manson. “Modelling secure multiagent systems”. In: *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM. 2003, pp. 859–866.
- [197] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. “Tropos: An agent-oriented software development methodology”. In: *Autonomous Agents and Multi-Agent Systems 8.3* (2004), pp. 203–236.
- [198] Niek Wijngaards, Maarten Van Steen, and Frances Brazier. “On MAS scalability”. In: *Proc. 2nd Int’l Workshop on Infrastructure for Agents, MAS and Scalable MAS*. 2001.
- [199] Phillip J Turner and Nicholas R Jennings. “Improving the scalability of multi-agent systems”. In: *Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents*. Springer. 2000, pp. 246–262.
- [200] Omer F Rana and Kate Stout. “What is scalability in multi-agent systems?” In: *Proceedings of the fourth international conference on Autonomous agents*. ACM. 2000, pp. 56–63.
- [201] Robert Shishko and Robert Aster. “NASA systems engineering handbook”. In: *NASA Special Publication 6105* (1995).
- [202] Kevin Forsberg, Hal Mooz, and Howard Cotterman. *Visualizing project management: models and frameworks for mastering complex systems*. John Wiley & Sons, 2005.
- [203] Union of the Electricity Industry - Eurelectric. *Power Statistics & Trends 2012*. 2012.
- [204] Giuliano Andrea Pagani and Marco Aiello. “Towards decentralization: A topological investigation of the medium and low voltage grids”. In: *IEEE Transactions on Smart Grid*. Vol. 2. IEEE, 2011, pp. 538–547.
- [205] V Glamocanin. “Optimal loss reduction of distributed networks”. In: *IEEE Transactions on Power Systems* 5.3 (1990), pp. 774–782.
- [206] Karsten Lehmann, Alban Grastien, and Pascal Van Hentenryck. “The complexity of DC-switching problems”. In: *arXiv preprint arXiv:1411.4369* (2014).
- [207] Karsten Lehmann, Alban Grastien, and Pascal Van Hentenryck. “The complexity of switching and FACTS maximum-potential-flow problems”. In: *arXiv preprint arXiv:1507.04820* (2015).
- [208] Burak Kocuk. “Global optimization methods for optimal power flow and transmission switching problems in electric power systems”. PhD thesis. Georgia Institute of Technology, 2016.
- [209] Giuliano Andrea Pagani and Marco Aiello. “The power grid as a complex network: a survey”. In: *Physica A: Statistical Mechanics and its Applications* 392.11 (2013), pp. 2688–2700.

- [210] Archana Khurana, Deepa Thirwani, and SR Arora. “An algorithm for solving fixed charge bi-criterion indefinite quadratic transportation problem with restricted flow”. In: *International Journal of Optimization: Theory, Methods and Applications* 1.4 (2009), pp. 367–380.
- [211] Archana Khurana and SR Arora. “Fixed charge bi-criterion indefinite quadratic transportation problem with enhanced flow”. In: *Investigación Operacional* 32.2 (2014), pp. 133–145.
- [212] Archana Khurana and Tripti Verma. “On a class of capacitated transshipment problems with bounds on rim conditions”. In: *International Journal of Mathematics in Operational Research* 7.3 (2015), pp. 251–280.
- [213] Archana Khurana. “Variants of transshipment problem”. In: *European Transport Research Review* 7.2 (2015), p. 11.
- [214] Siemens. *PSS®SINCAL*. Accessed: March 12, 2021. URL: <https://new.siemens.com/global/en/products/energy/energy-automation-and-smart-grid/pss-software/pss-sincal.html>.
- [215] Susan Zevin. *Standards for security categorization of federal information and information systems*. DIANE Publishing, 2009.
- [216] ISA99 ISA. *Industrial automation and control systems security*. 2018.
- [217] Govind S Mudholkar and Deo Kumar Srivastava. “Exponentiated Weibull family for analyzing bathtub failure-rate data”. In: *IEEE transactions on reliability* 42.2 (1993), pp. 299–302.
- [218] John R Clymer. *Systems analysis using simulation and markov models*. Prentice Hall Englewood Cliffs, NJ, 1990.
- [219] Daijiro Mizutani, Nam Lethanh, Bryan T Adey, and Kiyoyuki Kaito. “Improving the Estimation of Markov Transition Probabilities Using Mechanistic-Empirical Models”. In: *Frontiers in Built Environment* 3 (2017), p. 58.
- [220] Dennis J Wilkins. *The Bathtub Curve and Product Failure Behavior Part One-The Bathtub Curve*. Accessed: March 22, 2021. 2002. URL: <https://www.weibull.com/hotwire/issue21/hottopics21.htm>.
- [221] S Khoussi and A Mattas. “A Brief Introduction to Smart Grid Safety and Security”. In: *Handbook of System Safety and Security*. Elsevier, 2017, pp. 225–252.
- [222] Harold Ascher and Harry Feingold. *Repairable systems reliability: modeling, inference, misconceptions and their causes*. M. Dekker New York, 1984.
- [223] Québec (Province). Comité interministériel de régie pédagogique. *Dictionary of Production Engineering/Wörterbuch Der Fertigungstechnik/Dictionnaire Des Techniques de Production Mécanique Vol IV: Assembly/Montage/Assemblage*. Springer Berlin Heidelberg, 2011.

- [224] Khairy Ahmed Helmy Kobbacy and DN Prabhakar Murthy. *Complex system maintenance handbook*. Springer Science & Business Media, 2008.
- [225] Jean-Claude Laprie. “Dependable computing and fault-tolerance”. In: *Digest of Papers FTCS-15* (1985), pp. 2–11.
- [226] E Bordenabe Nicolás. “Measuring privacy with distinguishability metrics: Definitions, mechanisms and application to location privacy”. PhD thesis. PhD thesis, Ph. D. dissertation, École Polytechnique, Palaiseau, France, 2014.
- [227] Marek Jawurek. “Privacy in Smart Grids”. PhD thesis. Friedrich–Alexander University Erlangen–Nürnberg, 2013.
- [228] Pierangela Samarati. “Protecting respondents identities in microdata release”. In: *IEEE transactions on Knowledge and Data Engineering* 13.6 (2001), pp. 1010–1027.
- [229] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. “Privacy-friendly aggregation for the smart-grid”. In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2011, pp. 175–191.
- [230] Tracy Ann Kosa, Khalil El-Khatib, and Stephen Marsh. “Measuring Privacy”. In: *J. Internet Serv. Inf. Secur.* 1.4 (2011), pp. 60–73.
- [231] Pavel Vrba, Vladimír Mařík, Pierluigi Siano, Paulo Leitão, Gulnara Zhabelova, Valeriy Vyatkin, and Thomas Strasser. “A review of agent and service-oriented concepts applied to intelligent energy systems”. In: *IEEE transactions on industrial informatics* 10.3 (2014), pp. 1890–1903.
- [232] Luís Ribeiro, José Barata, and Pedro Mendes. “MAS and SOA: complementary automation paradigms”. In: *International Conference on Information Technology for Balanced Automation Systems*. Springer. 2008, pp. 259–268.
- [233] Michael Wooldridge and Nicholas R Jennings. “Intelligent agents: Theory and practice”. In: *The knowledge engineering review* 10.2 (1995), pp. 115–152.
- [234] Karsten Schweichhart. *Reference architectural model industrie 4.0 (RAMI 4.0)*. Accessed: March 22, 2021. 2016. URL: [https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference\\_architectural\\_model\\_industrie\\_4.0\\_rami\\_4.0.pdf](https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf).
- [235] Petr Marcon, Christian Diedrich, Frantisek Zezulka, Tizian Schröder, Alexander Belyaev, Jakub Arm, Tomas Benesl, Zdenek Bradac, and Ivo Vesely. “The Asset Administration Shell of Operator in the Platform of Industry 4.0”. In: *2018 18th International Conference on Mechatronics-Mechatronika (ME)*. IEEE. 2018, pp. 1–5.
- [236] German Feder Ministry for Economic Affairs and Energy, German Federal Ministry of Education and Research. *Definition of Administration shell in the Industrie 4.0 Glossary*. Accessed: March 17, 2021. URL: [https://www.plattform-i40.de/PI40/Redaktion/EN/Glossary/A/administration\\_shell\\_glossary.html](https://www.plattform-i40.de/PI40/Redaktion/EN/Glossary/A/administration_shell_glossary.html).

- [237] André Luís Andrade Menolli, Helena Sofia Pinto, Sheila S Reinehr, and Andreia Malucelli. “An Incremental and Iterative Process for Ontology Building.” In: *ONTOBRAS*. Citeseer. 2013, pp. 215–220.
- [238] Corentin Donzelli, Solomon Asres Kidanu, Richard Chbeir, and Yudith Cardinale. “Onto2MAS: An Ontology-Based Framework for Automatic Multi-Agent System Generation”. In: *12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE. 2016, pp. 381–388.
- [239] Patrick Cassidy. *COSMO ontology*. Accessed: March 12, 2021. URL: <http://micra.com/COSMO/COSMO.owl>.
- [240] Rachit Agarwal, David Gomez, Tarek Elsaleh, Jorge Lanza, Luis Sanchez, Amelie Gyrard. *FIESTA-IoT Ontology and Taxonomy*. Accessed: March 12, 2021. URL: <https://github.com/fiesta-iot/ontology>.
- [241] Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), Aldo Gangemi. *DOLCE+DnS Ultralite (DUL) ontology*. Accessed: March 12, 2021. URL: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>.
- [242] Syed Gillani, Frederique Laforest, and Gauthier Picard. “A Generic Ontology for Prosumer-Oriented Smart Grid”. In: *EDBT/ICDT Workshops*. 2014, pp. 134–139.
- [243] Mujahid Mohsin, Zahid Anwar, Farhat Zaman, and Ehab Al-Shaer. “IoTChecker: A data-driven framework for security analytics of Internet of Things configurations”. In: *Computers & Security* 70 (2017), pp. 199–223.
- [244] Stefan Jablonski, Ilia Petrov, Christian Meiler, and Udo Mayer. *Guide to web application and platform architectures*. Springer, 2004.
- [245] FIPA TC Nomadic Application Support. *FIPA Quality of Service Ontology Specification*. 2002.
- [246] Gustavo Fortes Tondello and Frank Siqueira. “The QoS-MO ontology for semantic QoS modeling”. In: *Proceedings of the 2008 ACM symposium on Applied computing*. ACM. 2008, pp. 2336–2340.
- [247] Lars Torsten Berger, Andreas Schwager, and J. Joaquin Escudero-Garzas. “Power line communications for smart grid applications”. In: *Journal of Electrical and Computer Engineering* 2013 (2013).
- [248] Michel Christiaan Alexander Klein. *Change management for distributed ontologies*. 2004.
- [249] Peter Palensky, Edmund Widl, and Atiyah Elsheikh. “Simulating cyber-physical energy systems: Challenges, tools and methods”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44.3 (2013), pp. 318–326.
- [250] Richard Lincoln. *Pypower*. Accessed: March 12, 2021. URL: <https://pypi.org/project/PYPOWER/>.

- [251] David P Chassin, K Schneider, and C Gerkenmeyer. “GridLAB-D: An open-source power systems modeling and simulation environment”. In: *Transmission and distribution conference and exposition, 2008. t&d. IEEE/PES*. IEEE. 2008, pp. 1–5.
- [252] DIgSILENT. *PowerFactory*. Accessed: March 12, 2021. URL: <https://www.digsilent.de/de/powerfactory.html>.
- [253] Stefan Scherfke. “Discrete-event simulation with SimPy”. In: *OFFIS–Institute for Information Technologie, USA* (2014).
- [254] Gustavo Carneiro. “NS-3: Network simulator 3”. In: *UTM Lab Meeting April*. Vol. 20. 2010, pp. 4–5.
- [255] András Varga and Rudolf Hornig. “An overview of the OMNeT++ simulation environment”. In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. 2008, pp. 1–10.
- [256] Tommi Karhela, Antti Villberg, and Hannu Niemistö. “Open ontology-based integration platform for modeling and simulation in engineering”. In: *International Journal of Modeling, Simulation, and Scientific Computing* 3.02 (2012), p. 1250004.
- [257] Steffen Schütte, Stefan Scherfke, and Martin Tröschel. “Mosaik: A framework for modular simulation of active components in smart grids”. In: *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*. IEEE. 2011, pp. 55–60.
- [258] Selim Ciraci, Jeff Daily, Jason Fuller, Andrew Fisher, Laurentiu Marinovici, and Khushbu Agarwal. “FNCS: a framework for power system and communication networks co-simulation”. In: *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative*. Society for Computer Simulation International. 2014, p. 36.
- [259] Mikhail K. Levin. *owlcpp C++ library for parsing, querying, and reasoning with OWL 2 ontologies*. Accessed: March 22, 2021. URL: <http://owl-cpp.sourceforge.net/>.
- [260] U.S. Department of Energy. *OpenEI*. Accessed: March 12, 2021. URL: <https://openei.org/doe-opendata/dataset/>.
- [261] Pecan Street Inc. *Pecan Street Dataport*. Accessed: March 12, 2021. URL: <https://dataport.pecanstreet.org/>.
- [262] Víctor Rodríguez-Doncel, Cristiana Santos, Pompeu Casanovas, and Asunción Gómez-Pérez. “Legal aspects of linked data–The European framework”. In: *Computer Law & Security Review* 32.6 (2016), pp. 799–813.

# Thomas Frühwirth

## Curriculum Vitae

### Education

- 2014–Present **Doctoral programme in Engineering Sciences**, *Technical University of Vienna*.
- 2012–2014 **Master of Computer Engineering**, *Technical University of Vienna*.
- 2009–2012 **Bachelor of Computer Engineering**, *Technical University of Vienna*.

### Work Experience

- 2017–Present **Research Engineer**, *CDP Center for Digital Production GmbH*, Vienna.
- 2014–Present **Project Assistant**, *Institute of Computer Engineering, Automation Systems Group, TU Wien*, Vienna.
- 2012/13 **Tutor for Distributed Automation**, *4 months at the Department for Computer Aided Automation*, *Technical University of Vienna*.
- 2011 **Tutor for Theory and Logic**, *4 months at the Department for Computer Languages*, *Technical University of Vienna*.

### Research Projects

- 2019–Present **PoSyCo**, *Power System Cognification*, Call: Energy Research 4th call, Project Number 867276.  
<https://www.ascr.at/power-system-cognification-posyco/>
- 2019–Present **Safety over TSN**, via the Young Researchers Program of the Austrian Competence Center for Digital Production, Project Number 854 187.  
<https://acd.p.at/de/>
- 2017–2021 **Demonstrator Project 2.4**, *Real-Time Factory Network*, via the Austrian Competence Center for Digital Production, Project Number 854 187.  
<https://acd.p.at/de/>
- 2019 **Digital Transformation Manager**, *Executive Course*, TU Wien.
- 2016–2018 **DigiTrans 4.0**, *Innovationslehrgang zur Gestaltung der Digitalen Transformation in der Produktentwicklung und Produktion*, 2. Ausschreibung Innovationslehrgänge, Project Number 854 157.  
<https://www.digitrans.at/>
- 2015–2017 **ABS4SmartGrids**, *Agent-based System for Smart Grids*, TU Wien in cooperation with Siemens AG.
- 2014–2016 **OPC4Factory**, *OPC UA based Communication for Flexible Automated Manufacturing Cells*, Call: Produktion der Zukunft, Project Number 843 613.  
<https://www.ift.at/forschungsbereiche/forschungsprojekte/opc4factory/>

## Publications

- T. Trautner, I. Ayatollahi, D. Strutzenberger, T. Frühwirth, F. Pauker, and B. Kittl. "Behavioral modeling of manufacturing skills in OPC UA for automated execution by an independent cell controller". In: *Procedia CIRP* 99 (2021), pp. 633–638
- D. Strutzenberger, R. Hinterbichler, F. Pauker, and T. Frühwirth. "Information model for human-machine (tool) interaction". In: *Procedia CIRP* 99 (2021), pp. 98–103
- D. Etz, T. Frühwirth, and W. Kastner. "Flexible Safety Systems for Smart Manufacturing". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. vol. 1. IEEE. 2020, pp. 1123–1126
- S. Gent, P. Gutiérrez Peón, T. Frühwirth, and D. Etz. "Hosting functional safety applications in factory networks through Time-Sensitive Networking". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. vol. 1. IEEE. 2020, pp. 230–237
- D. Herbst, M. Lager, R. Schürhuber, E. Schmautzer, L. Fickert, A. Einfalt, H. Brunner, D. Schultis, T. Frühwirth, and W. Prügler. "Zukünftige Anforderungen an NS-Netze und deren Lösungsansätze am Beispiel PoSyCo". In: *16. Symposium Energieinnovation*. Graz, Austria, Feb. 2020
- D. Strutzenberger, T. Frühwirth, T. Trautner, R. Hinterbichler, and F. Pauker. "Communication interface specification in OPC UA". in: *Proceedings of the 24th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Zaragoza, Spain, Sept. 2019
- G. Steindl, T. Frühwirth, and W. Kastner. "Ontology-Based OPC UA Data Access via Custom Property Functions". In: *Proceedings of the 24th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Zaragoza, Spain, Sept. 2019, pp. 95–101
- M. Messner, F. Pauker, G. Mauthner, T. Frühwirth, and J. Mangler. "Closed Loop Cycle Time Feedback to Optimize High-Mix / Low-Volume Production Planning". In: *Proceedings of the 52th CIRP Conference on Manufacturing Systems*. Ljubljana, Slovenia, June 2019, pp. 689–694
- D. Etz, T. Frühwirth, A. Ismail, and W. Kastner. "Self-Configuring Safety Networks". In: *Kommunikation in der Automation - KommA*. Lemgo, Germany, Nov. 2018, pp. 1–9
- D. Etz, T. Frühwirth, A. Ismail, and W. Kastner. "Simplifying functional safety communication in modular, heterogeneous production lines". In: *Proceedings of the 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. Imperia, Italy, June 2018, pp. 1–4. DOI: 10.1109/WFCS.2018.8402371
- T. Frühwirth, L. Krammer, and W. Kastner. "A methodology for creating reusable ontologies". In: *Proceedings of the 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. Saint Petersburg, Russia, May 2018, pp. 65–70
- T. Frühwirth, A. Einfalt, K. Diwold, and W. Kastner. "A distributed multi-agent system for switching optimization in low-voltage power grids". In: *Proceedings of the 22nd IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Limassol, Cyprus, Sept. 2017, pp. 1–8
- F. Pauker, T. Frühwirth, B. Kittl, and W. Kastner. "A systematic approach to OPC UA information model design". In: *Proceedings of the 49th CIRP Conference on Manufacturing Systems*. Stuttgart, Germany, May 2016, pp. 321–326

T. Frühwirth, F. Pauker, A. Fernbach, I. Ayatollahi, W. Kastner, and B. Kittl. “Guarded state machines in OPC UA”. in: *Proceedings of the 41st Annual Conference of the IEEE Industrial Electronics Society (IECON)*. Yokohama, Japan, Nov. 2015, pp. 4187–4192

T. Frühwirth, L. Krammer, and W. Kastner. “Dependability Demands and State of the Art in the Internet of Things”. In: *Proceedings of the 20th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. Luxembourg, Sept. 2015, pp. 1–4. DOI: 10.1109/ETFA.2015.7301592

T. Frühwirth, W. Steiner, and B. Stangl. “TTEthernet SW-based End System for AUTOSAR”. in: *Proceedings of the 10th IEEE International Symposium on Industrial Embedded Systems (SIES)*. Siegen, Germany, June 2015, pp. 1–8. DOI: 10.1109/SIES.2015.7185037