Combining Goals and SysML for Traceability and Decision-Making in the Development of Adaptive Socio-Cyber-Physical Systems

by

Amal Ahmed Anda

Thesis submitted to the
Faculty of Engineering
In partial fulfillment of the requirements
for the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Socio-cyber-physical systems (SCPSs) are cyber-physical systems (CPSs) with a socio-technical system (STS) aspect. Several SCPSs need to adapt dynamically to changing situations in order to reach an optimal symbiosis with users in their contexts. Tailoring requirements engineering activities and modeling techniques is needed for developing SCPSs and supporting their runtime adaptability. Goal models are used to support early requirements engineering activities by capturing system and stakeholder objectives and their links, and by enabling what-if and trade-off analysis in a decision-making context. They are also increasingly used in system monitoring and self-adaptation contexts. Yet, linking system models to goals and automatically converting goal models to code for supporting analysis and adaptation activities remain important issues.

This thesis combines a common way of modeling STSs (i.e., goal modeling with the Goal-oriented Requirement Language – GRL) with a conventional way of modeling CPSs (i.e., SysML, optionally with feature models) in order to integrate social concepts early in SCPS requirements, design, simulation, optimization, and implementation activities. To help guarantee system quality and compliance during both design time and runtime adaptations, the thesis translates goal and feature models to mathematical functions used to validate possible design and adaptation alternatives during simulations at design time and adaptations at runtime. Feature model functions help constrain candidate goal model functions to valid solutions. These functions are used outside goal and feature modeling tools and are combined with SysML models, simulations, constraint solvers, and implementation tools. Furthermore, an integration between GRL and SysML models via a third-party requirements management system is achieved in order to strengthen system traceability and help ensure that stakeholder goals are considered properly during the SCPS development process.

The main contributions of this thesis are:

1. An approach to integrate social concerns with CPSs, called *CGS4Adaptation,* composed of three independent methods supporting:
   (a) System design in SysML with a companion goal view, to support the documentation and consideration of goal-based rationales behind requirements and design elements, traceability and change management, impact analysis, and consistency and completeness checks (between the elements of goal and SysML models).
   (b) System optimization and simulation, to select and validate system designs, as well as adaptation strategies predefined at design time for common contexts and generated at runtime for unforeseen contexts.
   (c) System implementation, to support systems in choosing between adaptation alternatives at runtime while monitoring their quality and compliance.
2. A new and integrated arithmetic semantics for standard GRL, optionally combined with feature models.

3. An automated transformation of GRL models, and optionally feature models, to mathematical functions in multiple programming languages. Such code allows for quantitative GRL and FM model evaluations to be performed outside of modeling tools, including in running systems.

Acknowledgements

I would like to express my deep appreciation to Professor Daniel Amyot, my research supervisor, for his patient guidance, constructive suggestions, and valuable advice. I have been extremely lucky to have a supervisor who cares so much about his students' research work. Prof. Amyot has created an outstanding work environment which fosters collaboration between students, promoting professional development with our experts in the same field of study and encourages innovative thinking. He involves us in social and scientific activities to strengthen our knowledge and build trust within. He takes it upon himself to provide exceptional support with his patience, encouraging us during our study to review our work, take time to study, and to correct anything that we are not content with. He is there for us! He discerns quickly our individual needs and moves efficiently to provide key information, recommendations, or direction. He encouraged me to be confident and achieve what I wanted to do.

I would like to thank the Libyan Ministry of Education for its financial support. I would also like to extend my thanks to the University of Ottawa for providing me the funding to attend conferences and the opportunity to meet people that have much relevant experience in my research area.

I am thankful to Yuxuan Fan for having prototyped a SymPy formula generator for GRL models in the jUCMNav tool. I am also thankful to Professor Jean-Michel Bruel for providing me with an Internet of Things (IoT)-based smart home case study. Finally, I thank my examiners, namely Professor Marc Frappier, Professor Timothy Lethbridge, Professor Chung-Horng Lung, and Professor Mehrdad Sabetzadeh, for their comments and constructive feedback.

Dedication

This is dedicated to my loving and sweet family, my husband, my daughter, and my son, for their patience and understanding.

I also dedicate this work and give special thanks to my parents for their love, support, encouragements, and prayers.

In addition, I dedicate this work to my sisters and brothers, who have never left my side and who are very special to me.

Table of Contents

List of Tables

List of Figures

List of Listings

List of Acronyms

BDD	Block Definition Diagram
BDI	Belief-Desire-Intention
CGS4Adaptation	Combining Goals and SysML for Traceability and Decision Making in the Development of Adaptive SCPSs
CPS	Cyber-Physical System
DOORS	Dynamic Object Oriented Requirements System
DSL	Domain-Specific Language
DSM	Design Science Methodology
DXL	DOORS eXtension Language
FR	Functional Requirement
FG	Functional Goal
FLAGS	Fuzzy Live Adaptive Goals for Self-adaptive systems
GORE	Goal-oriented Requirements Engineering
GRL	Goal-oriented Requirement Language
INCOSE	International Council on Systems Engineering
IoT	Internet of Things
IS	Information System
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
KAOS	Knowledge Acquisition in autOmated specification or Keep All Objects Satisfied
KPI	Key Performance Indicator

LTL	Linear Temporal Logic
MAPE	Monitor, Analyze, Plan, and Execute
MDE	Model-Driven Engineering
MI-DSL	Model Import Domain-Specific Language
NFR	Non-Functional Requirement
NFG	Non-Functional Goal
OCL	Object Constraint Language
OMG	Object Management Group
OSLC	Open Services Lifecycle Collaboration
RE	Requirements Engineering
RMS	Requirements Management System
SAS	Self-Adaptive System
SCPS	Socio-Cyber-Physical System
SoS	System of System
SPL	Software Product Line
STS	Socio-Technical System
SymPy	Symbolic Python
SysML	Systems Modeling Language
TIM	Traceability Information Model
UCM	Use Case Map
UML	Unified Modeling Notation
URN	User Requirements Notation
VTL	Velocity Template Language

Chapter 1

Introduction

This thesis introduces a modeling approach that integrates the Goal-oriented Requirement Language (GRL), the Systems Modeling Language (SysML), and feature models for the development of socio-cyber-physical systems able to adapt to users' concerns through monitoring quality and compliance. This approach consists of two complementary parts supporting traceability and self-adaptation. The first integrates GRL and SysML models using a third-party traceability system (e.g., a requirements management system) while the second transforms the goal and feature models to mathematical functions for effective and efficient goal-based reasoning in SysML models and in implementations. This first chapter introduces the problem context and motivation, and then highlights the problem statement, the research methodology, as well as the contributions.

1.1 Problem Context

Cyber-physical systems (CPSs) are composed of hybrid components such as hardware (e.g., sensors, devices, and networks) and software, which can even be integrated at runtime. Horváth [79] observes that the complexity, emergent properties, and adaptability of CPSs have increased substantially in the past decade in order for CPSs to be compatible with different components and changes in their surrounding environment. Moreover, CPSs are characterized by a high level of uncertainty, which is difficult to address with current design methods [79, 91, 107].

Socio-cyber-physical systems (SCPSs) have been developed as a type of CPSs where human concerns are considered during the development process (i.e., at design time) and during execution (i.e., at runtime). Many SCPSs should ideally be able to adapt to changing conditions in order to reach an optimal symbiosis with users and their contexts [79]. Examples include existing systems such as air traffic control systems, and emerging ones such as smart homes/cities [148], human-oriented services exploiting the Internet of Things (IoT) [161], adaptive Systems of Systems (SoS) [46], and intelligent production networks [63].

In some contexts, a SCPS may no longer accomplish what it was intended to do or meet its goals. *Self-adapting systems* are capable of detecting such situations and change their own behaviour accordingly. Kephart and Chess [92] divided the adaptation process into four different activities (Monitor, Analyze, Plan, and Execute), collectively called *MAPE-cycle* and illustrated in Figure 1.1. The general functionality of each activity is as follows:

- *Monitor*: Gathers information about monitored features and the context.
- *Analyze*: Analyzes the features and determines whether to activate the planning process and what information should be passed on to it.
- *Plan*: Selects the most suitable adaptation strategy depending on the information provided by the analysis activity.
- *Execute*: Executes the selected adaptation strategy.

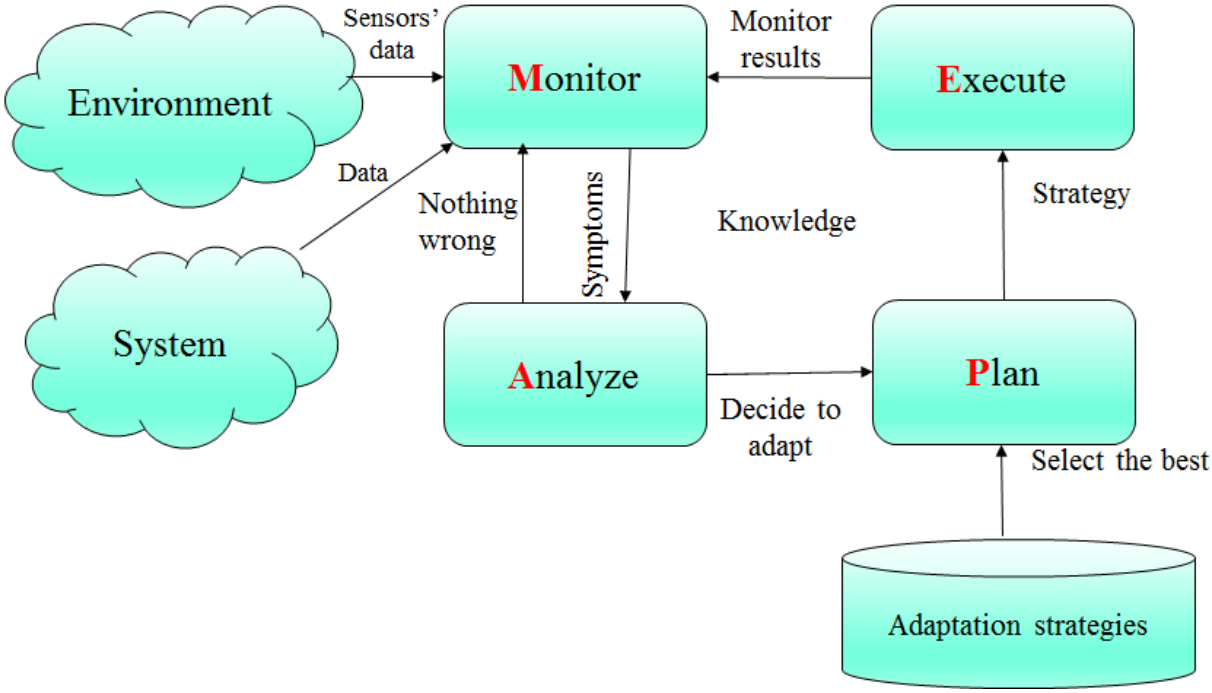

Figure 1.1: Self-adaptation activities: MAPE-cycle (adapted from [92])

Several challenges related to these kind of systems were addressed in the literature [79, 91, 117] where model-driven engineering (MDE) approaches and goal-oriented requirements engineering (GORE) were both presented as solutions. Traceability should be also managed between many types of artifacts such as goals, requirements, design, and code [157]. Traceability is important to support system consistency/completeness assessment [11], validation, verification, change management, and impact analysis. Moreover, in order to help

guarantee system quality and compliance, system and stakeholder goals should be considered during system evolution at design time but also at runtime while adapting to new purposes.

Kephart and Walsh [93] as well as Botanegra et al. [38] observed that current requirements engineering (RE) activities and languages need improvements in order to satisfy the emerging needs of SCPSs and adaptive systems. The integration between MDE and GORE shows potential as a solution here. Such integration could be particularly realized with two standard languages: 1) the *Systems Modeling Language* (SysML) for software and hardware elements [64], and 2) the *Goal-oriented Requirement Language* (GRL) for stakeholder goals and adaptation support [14].

SysML is a profile of UML that supports modeling systems with hardware and software components, such as CPSs and systems of systems. SysML reuses part of UML, including use case, sequence, activity, and state machine diagrams, and refines class diagrams with block and internal block diagrams. Moreover, SysML adds two new types of diagrams (parametric and requirements), which facilitate the connection between system components and their requirements [64]. SysML aims to enable the modeling of software and hardware components, together with their relationships, within a single environment while simplifying their design [62] and reducing their complexity [8, 9]. Although SysML modelers can connect requirements to different design elements such as use cases, test cases, and blocks, SysML lacks important "social" modeling concepts for SCPS such as stakeholders and their goals or measurable objectives.

GRL, part of the *User Requirements Notation* (URN) standard [17, 90], is used to model and analyze stakeholder and system objectives as well as their relationships and related processes. GRL supports quantitative and qualitative trade-off and what-if analyses [14] usable at design time but also in an adaptation context [16, 17]. Note that GRL also supports *Key Performance Indicators* (KPIs) to monitor contextual information [16] and support evidence-based decision making. A KPI assesses a current observable value against target, threshold, and worst value parameters, and outputs a satisfaction level that can be propagated to tasks, goals, and softgoals in the model. KPIs can be used 1) at design time via evaluation strategies or external data sources (database, web server, Excel sheet, etc.) and 2) at runtime using monitoring sensors and real-time data [137]. GRL is an excellent candidate for integration with SysML because it supports not only goals but also indicators (for the monitoring of contexts in an adaptation situation), it is a standardized language, and it has relevant types of tool-supported analyzes.

A GRL model can be used to evaluate different *strategies* composed of initial satisfaction values attached to goals and other elements. Strategies can correspond to configurations of alternatives during design and adaptation [137]. However, the validity of these configurations and strategies should be determined by the system features' variability, possibly formalized as a *feature model* where features and qualities (e.g., GRL softgoals) can be monitored [54, 135].

In order to develop a SCPS able to adapt to its users' concerns while monitoring quality and compliance, we propose a modeling approach that integrates GRL, SysML, and feature models. By linking stakeholders' goals and social concerns to a SCPS's design and imple-

mentation, we aim to support developers in producing rational system architectures and adaptation solutions. However, this support is not offered by traditional RE processes and further improvements are required [3, 42, 72, 100]. Hence, the main objective of this thesis is to improve requirements engineering activities through several integrations between GRL, feature models, and SysML during the design, simulation, and implementation of adaptive SCPSs.

1.2 Motivation

The development process of SCPSs faces several challenges. These challenges have been identified in the literature with a particular focus on SCPSs [79, 100]:

- Modeling user goals in addition to hardware and software elements [79, 100, 146].
- Managing traceability between goals, requirements, design, and implementation artifacts [38, 52, 100, 146]
- Dealing with emergent properties [79, 100].
- Managing a high level of uncertainty [38, 52, 79, 100]
- Reducing complexity [79, 100].
- Adapting behaviour according to external/internal changes [79, 100].
- Getting access to suitable tool support [72].
- Defining a well-defined RE process [38, 52, 100] as many ad-hoc steps have been used when developing modern SCPSs using traditional RE processes [42].
- Dealing with a large number of requirements [72, 100].
- Continuously monitoring system functionality and quality at design time and runtime (e.g., along the MAPE steps) [38, 52, 72, 79, 100].
- Specifying the design space that describes possible design alternatives, their interdependencies, and the decisions designers should make [38, 52].
- Using precise and quantitative analysis [100] to assist and speed up the product line process, system reuse [127], system verification, and simulation at design time [49], as well as to support self-adaptation and quality monitoring at runtime [47, 146].

Requirements engineering for this kind of systems should deal with uncertainty by combining the possible alternatives and related decisions with system development and operation phases because the environment is expected to change frequently over time [38, 47, 49, 48, 52, 100]. In order to adjust requirements engineering and modeling techniques to be suitable for modeling SCPSs, and especially adaptive ones, it is important to consider

the integration between SysML (which supports system specification) and a goal modeling language that supports social concepts and comprehensive decision making [14, 37, 91]. Furthermore, continuous integration of goal models with all system development activities provides unprecedented opportunities for solving the above-mentioned challenges as well as helping guarantee system functionality, behaviour, and quality together throughout the development and operation phases. In addition, a quantitative goal-based analysis is required to offer effective, efficient, and precise reasoning for system simulation and design tools as well as for further adaptability at operation time [47]. For instance, in order to design adaptive systems that satisfy stakeholders and system goals (SCPSs), SysML 2.0's Request for proposals suggests integrating goals and their evaluation criteria with SysML tools:

> *"Proposals for SysML v2 shall include a capability to represent goals, objectives, and evaluation criteria."* [131]

1.3 Problem Statement

SysML and goal models can be integrated in many different ways to support the development of SCPSs. For example, Amyot et al. [14] highlighted different types of integrations at the language level (e.g., by bringing goals into SysML, integrating tools, or using third-party tools for traceability or requirements management), and the literature review in Chapter 3 surveys many existing approaches, including those related to adaptation. Yet, existing integration solutions show severe limitations in terms of traceability, usability, scalability, analysis, and support for adaptation at design time and runtime. In this context, our research questions are:

1. How should the integration between goal, feature, and SysML models be done in early stages to support effective SCPS requirements definition, modeling, and analysis?
2. What information should be carried from goal and feature models to SysML models and implementations in order to support SCPSs at runtime effectively, especially in a self-adaptation context?

1.4 Methodology

As this research aims to produce a method artifact with partial tool support in a context where new requirements are discovered as the research evolves, the Design Science Methodology (DSM) [74] was selected. DSM proposes iterative steps to reach feasible and efficient solutions for research needs leading to artifacts that increase the abilities of humans and organizations. According to DSM, the identified artifacts are developed and evaluated based on rigorous methods following seven guidelines described by Hevner et al. [74] and shown in Table 1.1.

Table 1.1: The seven guidelines of Design Science Research [74]

Guidelines	Description
Guideline 1 Design as an artifact	Design science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2 Problem relevance	The objective of design science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3 Design evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4 Research contributions	Effective design science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5 Research rigor	Design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6 Design as a search process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7 Communication of research	Design science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

In DSM, the developed artifacts (i.e., tools, methods, models, and/or their instantiations) are built iteratively using the feedback from the evaluation process. DSM is an appropriate methodology since we develop two complementary artifacts (method and tool) where changes of one artifact affect the evolution of the other. Figure 1.2 shows the framework of DSM. As shown in the figure, the environment helps identify the problem space which consists of people, organization, and the existing technology while business needs represent the identified problem itself. The relevance cycle determines the requirements and the assessment criteria for the final evaluation of the results whereas the rigor cycle represents the knowledge as a foundation to build and evaluate the artifacts. Information System (IS) research, which is the central activity of that framework, iterates the development and justification processes until the artifacts satisfy the research requirements.

This thesis instantiates DSM to suit the presented research context and artifacts. We follow the DSM steps in general but, because we are designing two related artifacts (an integration and development *method* for SCPSs and a *tool* to transform GRL models and feature models to functions in different programming languages), DSM's iteration cycle is adapted to include 4 steps: further develop the method, further develop the tool, evaluate the method, and evaluate the tool. This allows many iterations across the specified artifacts. To develop a practical solution, we also explore existing work and possible solutions in the problem's environment. Figure 1.3 illustrates the main steps of the thesis

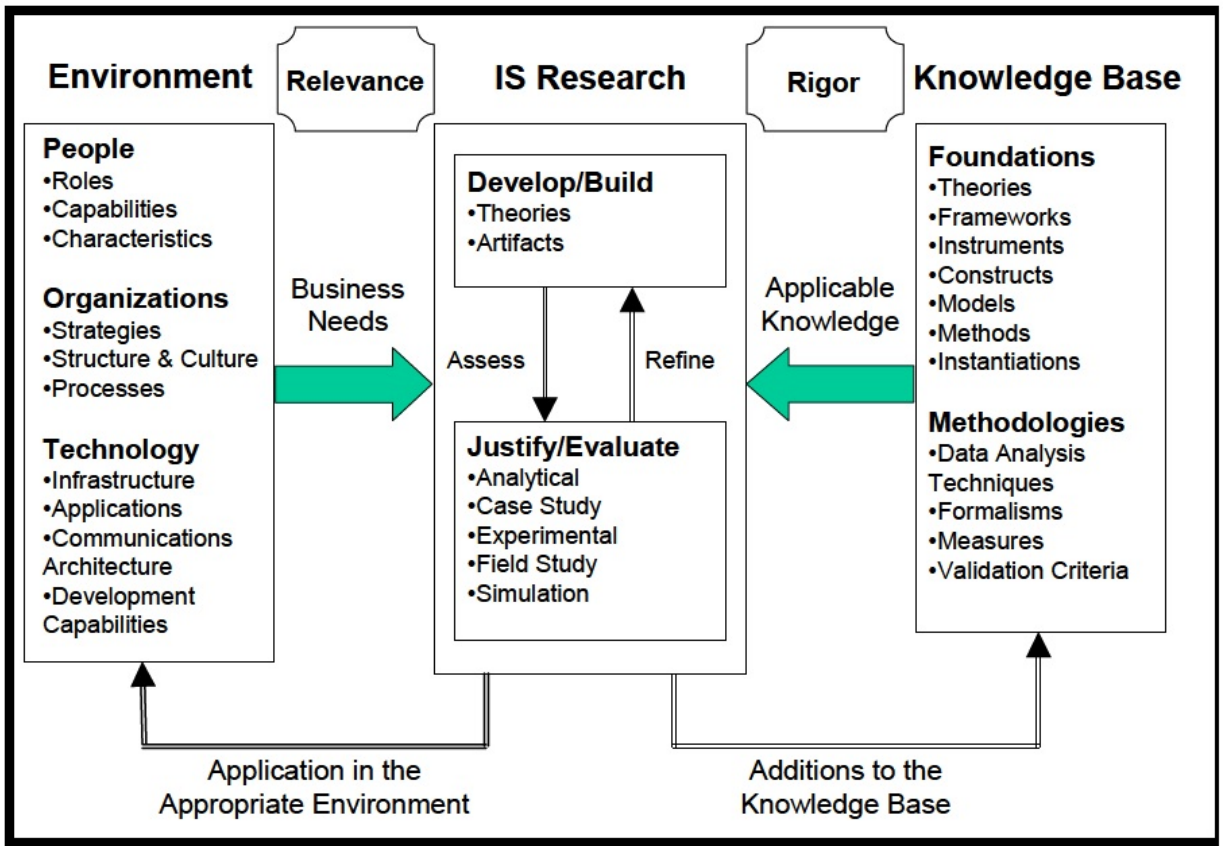

Figure 1.2: Conceptual framework of the Design Science Methodology [74]

methodology. The study follows the DSM guidelines by constructing the artifacts on a rigid knowledge base and providing a technology-oriented and management-oriented solution. The following are the steps for the given methodology:

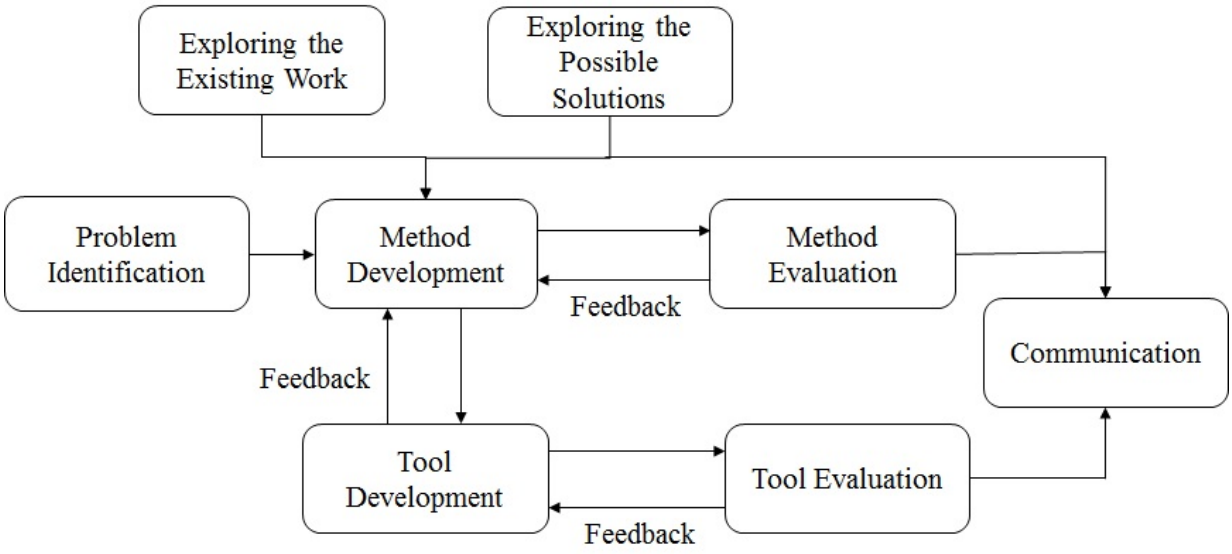

Figure 1.3: Thesis methodology (adaptation of DSM)

- **Identify the problem**: The problem was identified in this chapter and Chapter 3 as the lack of RE approaches appropriate for the development and operation of adaptive SCPSs that consider stakeholders' goals in their design and self-adaptation decisions.
- **Explore possible solutions**: In our early work [14], we explored possible and practical methods to integrate stakeholders' goals with a SysML model and provide assessment criteria for such integration. This set of criteria is further extended in Chapter 3.
- **Explore the existing work**: In order to develop a technology-based solution and identify the problems related to the self-adaptation ability of SCPSs, we conducted literature reviews of 1) approaches integrating SysML and goal models together with-/out self-adaptation support, and 2) SysML modeling and goal modeling (in isolation) for self-adaptation support. The problems identified are: 1) the lack of manageable traceability between goals, requirements, design, and implementation artifacts and, 2) the lack of a complete and consistent quantitative goal model reasoning that is employed in a system architecture selection process at design time, but also in relevant adaptation activities (Monitor, Analyze and Plan) at runtime.
- **Design as artifact**: Tailoring some RE activities is required to support the emergent needs of SCPSs. In particular, we identified two artifacts: 1) a method for modeling, managing traceability, and supporting the development of SCPSs called *Combining*

Goals and SysML for Traceability and Decision-Making in the Development of Adaptive SCPSs (CGS4Adaptation), and 2) a tool for automatic function generation from GRL and feature models, targeting multiple programming languages.

- **Design evaluation**: The above artifacts are evaluated iteratively first using an ongoing illustrative example (a hybrid car) and then case study (smart home management system). The design phase of the artifacts is an iterative phase used to evaluate the identified artifacts. Method and tool artifacts are enhanced incrementally using the feedback supplied by the two evaluation processes (method evaluation and tool evaluation). Table 1.2 and Table 1.3 show the iterative evaluation processes for the CGS4Adaptation method and for the code generation tool, respectively. The developed artifacts are evaluated in terms of correctness of the tool (i.e., the generated functions reflect the input models), consistency between the proposed method and tool (i.e., the models and functions that are employed by the method represent the actual inputs and outputs of the tool), and feasibility of the developed artifacts (i.e., the method and the tool are applicable using existing standard technologies and support useful solutions).

Table 1.2: Iterative evaluation process for the CGS4Adaptation method.

Iteration	**Evaluation method**	**Result**
Iteration 1	Illustrative case study: Hybrid car engines system.	Assess the feasibility of the method and gain understanding about challenges and opportunities.
Iteration 2	Case study: An Internet of Things (IoT)-based smart home, provided by researchers in Toulouse, France.	Modifications and improvement to the method and the tool.

- **Design as search process**: In order to implement the tool that translates GRL and feature models to mathematical functions, we exploit the *jUCMNav* tool [18] to transform those models to mathematical expressions that can be further simplified and converted to several programming languages (to provide flexibility in incorporating GRL models in many simulation and execution environments) using *SymPy* [59, 151]. To manage traceability between stakeholders' goals and SysML models, we exploit the jUCMNav tool and a commercial Requirements Management System (RMS), namely *IBM DOORS* [83]. Moreover, the Xtext-based *Model Import Domain Specific Language* (MI-DSL) editor for the Eclipse environment [139] and No Magic's Cameo Systems Modeler tool [126] are used to describe the meta-model of the SysML tool (using MI-DSL) in order to be imported by the DOORS RMS, where traceability links between goals, features, and SysML model elements will be managed.

- **Communication**: The results of this research were shared in several venues, listed in Section 1.6.

Table 1.3: Iterative evaluation process for the function generation tool (and the method)

Iteration	**Evaluation method**	**Result**
Iteration 1 (for GRL models only)	Integration of the tool within the business environment technology: implementation as a plugin for the jUCMNav tool and as a SymPy extension to generate the functions in several programming languages.	First version of the conversion tool, for GRL.
Iteration 2 (for GRL and feature models)	Illustrative case study, and test cases for covering different patterns.	Generalization of the tool for GRL+FM.
Iteration 3 (for function integration)	Integration of the functions within business environment technology: Implementation of the functions' integration as an optimization and a simulation models using IBM CPLEX [82] and Cameo systems modeler tool respectively for testing system behaviours using all possible scenarios.	First version of the method with SysML and CPLEX integration, with tool improvements.
Iteration 4 (for RMS integration)	Integration of the *Model Import Domain Specific Language* (MI-DSL) [139] classes and the DOORS eXecutable Language (DXL) [85] library within business environment technology. Import of the DXL library and the classes using IBM DOORS [83] and testing of their correctness	Improvement to the method, with RMS integration.
Iteration 5 (for GRL and feature models)	Interpretation of the case study's models.	Improvements to the method and the tool.
Iteration 6 (for function integration)	Integration of the functions of the case study within business environment technology. Implementation of the functions' integration as an optimization model with IBM CPLEX to test their behaviours using specific and all possible scenarios	Improvements to the method and the tool.
Iteration 7 (for RMS integration)	Integration of the MI-DSL classes of the case study and the DXL library within business environment technology. Execution of the DXL library and the classes using the DOORS tool, traceability link management between the models, and consistency and completeness checking	Improvements to the method and the tool.
Iteration 8 (for performance at run time)	Integration of the generated functions of the case study within business environment technology. Implementation of run-time integration as an optimization model using Java and a CPLEX library, with recording and assessment of relevant metrics	Improvements to the method and the tool.

1.5 Thesis Contributions

Answering the research questions will contribute to the tailoring of several requirements engineering activities to become suitable for emergent SCPSs using existing modeling notations supported with existing and new tools. The contributions of the thesis include:

1. An approach to integrate social concerns with CPSs (CGS4Adaptation), composed of three independent methods supporting:

   (a) System design in SysML with a companion goal view, to support the documentation and consideration of goal-based rationale behind requirements and design elements, traceability and change management, impact analysis, and consistency and completeness checks (between the elements of goal and SysML models).

   (b) System optimization and simulation, to select and validate system designs, as well as adaptation strategies predefined at design time for common contexts and generated at runtime for unforeseen contexts.

   (c) System implementation, to minimize the risks of introducing translation issues (i.e, information loss) between goal models and code, and to support choosing between adaptation alternatives at runtime while monitoring their quality and compliance.

   This is done practically by taking advantage of standard notations (SysML and GRL) and existing tools (e.g., jUCMNav for GRL/feature modeling & analysis [18], Cameo Systems Modeler for SysML modeling simulation, and IBM DOORS for traceability management).

2. A new and integrated arithmetic semantics for standard GRL, optionally combined with feature models. These functions provide a way to support runtime system adaptation, but also research development on goal-based reasoning/analysis outside goal modeling tools (including in SysML simulation tools).

3. An automated transformation of GRL and feature models to mathematical functions in multiple programming languages, via an existing intermediate representation (SymPy). Without mapping or remodeling systems' goals and features, these functions can be transformed to several languages such as Java, Matlab, and C++, ready to be integrated in operational models and running adaptation code. The mathematical functions of feature models can be used with goal models as constraints to control system quality and compliance while designing, executing, and adapting the system dynamically. Such code allows for quantitative GRL and FM model evaluations to be performed outside of modeling tools, including in running systems.

Note that although CGS4Adaptation is defined for GRL, SysML, and feature models in the first contribution, the use of feature models is actually optional, at the risk of generating invalid solutions in some contexts. In addition, the second and third contributions are independent from the presence of SysML models, and again feature models are optional. This hence provides more opportunities for the above contributions to be used outside the SCPS context targeted by CGS4Adaptation.

1.6 Publications

This thesis has led to the following publications:

1. Amyot, D., Anda, A. A., Baslyman, M., Lessard, L., Bruel, J.-M. Towards Improved Requirements Engineering with SysML and the User Requirements Notation. In *2016 IEEE 24th International Requirements Engineering Conference(RE)*, pp. 329–334. IEEE CS (2016) [14]

2. Akhigbe, O., Amyot, A., Anda, A. A., Lessard, L., Xiao, L. Consistency Analysis for User Requirements Notation Models. In *Proceedings of the Ninth International i* Workshop (iStar 2016)*, CEUR Workshop Proceedings 1674, pp. 43–48 (2016) [11]

3. Anda A. Modeling Adaptive Socio-Cyber-Physical Systems with Goals and SysML. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, Doctoral Symposium, pp. 442–447. IEEE CS (2018) [19]

4. Fan, Y., Anda, A.A., Amyot, D. An Arithmetic Semantics for GRL Goal Models with Function Generation. In *System Analysis and Modeling. Languages, Methods, and Tools for Systems Engineering - 10th International Conference (SAM 2018)*, LNCS 11150, pp. 144–162. Springer (2018) [58]

5. Anda, A. A., Amyot, D. Arithmetic Semantics of Feature and Goal Models for Adaptive Cyber-Physical Systems. In *27th IEEE International Requirements Engineering Conference (RE'19)*, pp. 245–256. IEEE CS (2019) [20]

6. Anda, A., Amyot, D. An Optimization Modeling Method for Adaptive Systems Based on Goal and Feature Models. In *Tenth International Model-Driven Requirements Engineering Workshop (MoDRE)*. IEEE CS, pp. 11–20 (2020) [21]

7. Anda, A. A., Amyot, D. Traceability Management of GRL and SysML Models. In *SAM'20: 12th System Analysis and Modeling Conference Proceedings*, ACM *(to appear)*. [22]

In papers 1 and 2, I have equal participation with the other authors while in paper No. 4, I was the co-supervisor of the work implemented by a master's student, Yuxuan Fan. I am the main author of the other papers. The papers are distributed through the thesis as follows:

- Paper 1 is distributed over Chapters 3, 4, and 10.
- Paper 2 is used in Chapter 8
- Paper 3 is distributed over Chapters 1, 5, and 3
- Paper 4 represents Subsections 4.3.1 and 6.2.6

- Paper 5 represents Subsections 4.3.2 and 6.2.6
- Paper 6 is introduced in Subsections 4.3.4 and 6.2.5, and
- Paper 7 is presented in Sections 4.2 and 5.1 and Subsection 8.1

1.7 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2 highlights important background technologies and tools supporting this thesis.
- Chapter 3 provides a systematic literature review about integrating goal and SysML models with/without adaptation support, and a non-systematic literature review about quantitative goal-based reasoning.
- Chapter 4 describes the CGS4Adaptation approach. In addition to the new arithmetic semantics of GRL and feature models, the elements of block definition diagram and requirement diagram from SysML models are selected to be traced in a RMS. Consequently, several integrations of goal, feature, and SysML models are presented in this chapter.
- Chapter 5 discusses the implementation of methods and tools supporting various aspects of CGS4Adaptation.
- Chapter 6 provides a first evaluation of CGS4Adaptation. The automatic generation of the mathematical functions of GRL and feature models, the MI-DSL classes and the import library of the SysML model, and their integrations for different purposes (optimization, simulation, and implementation) are explained using an illustrative example.
- Chapter 7 further evaluates CGS4Adaptation with a Smart Home Management System case study and its validation via an optimization model.
- Chapter 8 explores how to manage traceability links through GRL, requirements, and SysML block definition diagrams of the smart home case study using a commercial RMS (DOORS). Also, automated rule-based consistency and completeness checks are conducted on these models, with implicit links discovery checks.
- Chapter 9 describes an experiment conducted to evaluate the scalability and performance of CGS4Adaptation's optimization method for runtime adaptation.
- Chapter 10 compares the thesis results with related work, and discusses threats to the validity of the thesis.
- Chapter 11 recalls the main contributions, answers the research questions, and provides future work opportunities.

Chapter 2

Background

This chapter provides important background information on existing technologies and tools that are exploited by the method developed in this thesis. The relevant languages include SysML, GRL, and Feature Diagrams, the relevant tools include jUCMNav, SymPy, IBM Rational DOORS, and Cameo Systems Modeler, and finally the relevant technologies include Model Import Domain-Specific Language (MI-DSL), which exploits the DOORS eXtension Language (DXL), and optimization models.

2.1 Modeling Languages

2.1.1 SysML and its Relevant Diagrams

Overview

The *Systems Modeling Language* [64, 132] is a language standardized by the Object Management Group (OMG) to support systems engineering methods. SysML is a UML profile that reuses part of UML (e.g., use case, sequence, and state machine diagrams) and modifies other types of diagrams to produce block diagrams, internal block diagram, and activity diagram. Moreover, SysML includes parametric and requirements diagrams, which facilitate the connection between system components and their requirements [64]. SysML enables the modeling of software and hardware components as well as their relationships, in a way that simplifies their design [149] and reduces their complexity [8, 9].

Figure 2.1 illustrates four SysML diagrams (activity, parametric, block definition, and requirement) and a requirements table (a different presentation of requirement diagrams) for a simplified hybrid car's engine.

SysML modelers can connect requirements to different design elements such as use cases, test cases, and blocks using a few diagram types. Yet, SysML lacks important "social" modeling concepts for SCPSs such as goals and stakeholders' objectives. Flexible model integration is known to be challenging [119], but a multi-level modeling approach

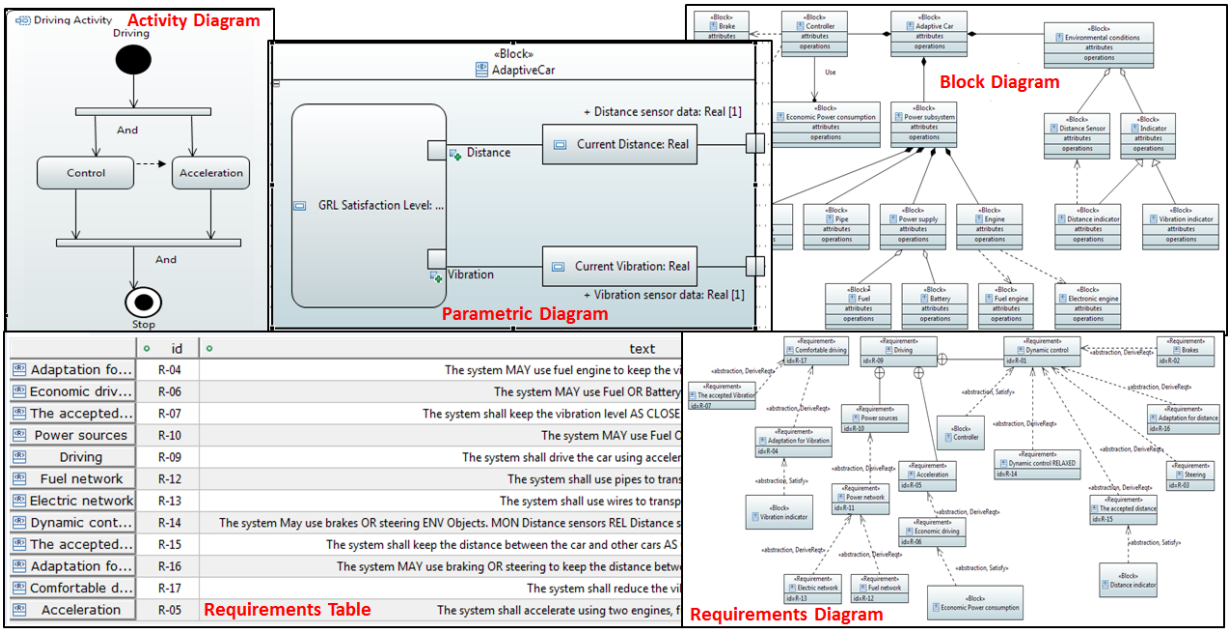

Figure 2.1: Some SysML diagrams of a simplified hybrid car's engine.

involving SysML is still a promising avenue in contexts such as Systems of Systems and adaptive SCPS [102, 121].

We select requirement diagrams and block definition diagrams from the eight SysML diagrams to be created and traced in a requirements management system. These diagrams often capture operationalizations and structure that refine goals in goal models and are hence very relevant in this thesis. The next subsections illustrate these diagrams and why they are selected.

Requirement Diagrams

According to the literature review in the next chapter, requirement diagrams are the most often used diagram type when integrating SysML model with goal models. They describe the main functional and non-functional requirements, as well as their relationships, including links from test cases. Figure 2.2 highlights part of the SysML metamodel describing important concepts of such diagrams.

From this metamodel, standard relationships in requirement diagram are defined:

1. Between two requirements, such as *Copy* (a requirement is a copy of other requirement), *Derive* (a requirement is derived from other requirement), and *Containment* (a requirement is composed of other requirements).

2. Between requirements and other model elements, such as *Verify* relationships between requirement and test cases, *Satisfy* relationships between requirement and blocks or use cases, as well as *Trace* and *Refine* relationships between requirements and any

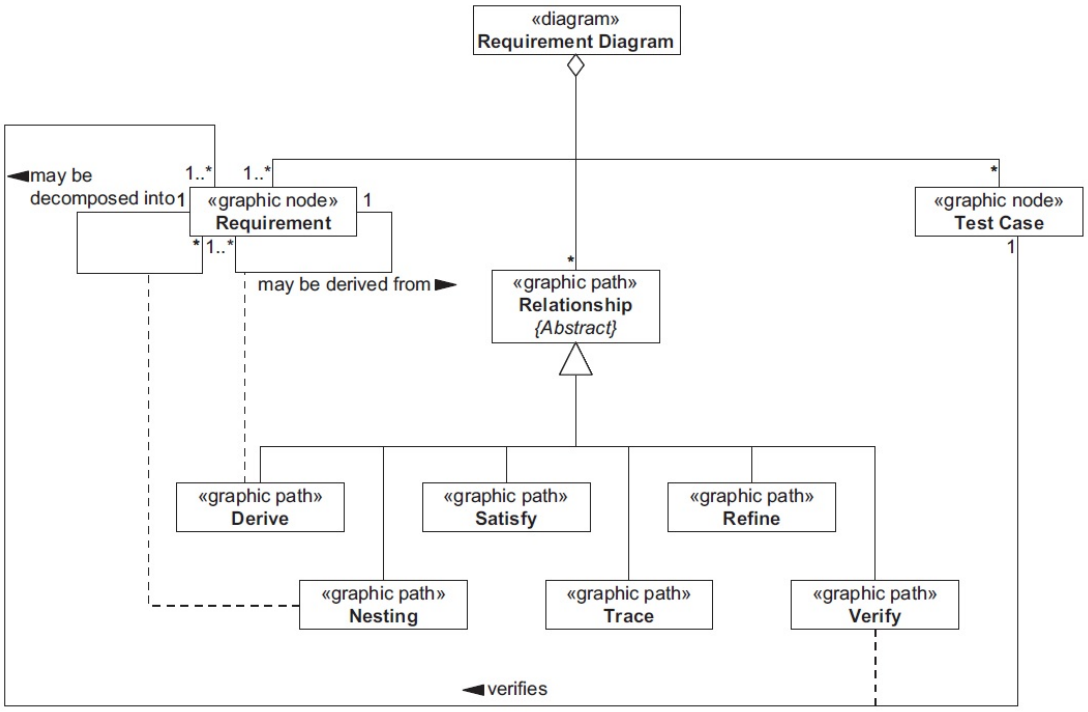

Figure 2.2: Part of the requirement diagram metamodel (from [76]).

model element for traceability purposes. *Containment* relationships can be defined between requirements and packages or any name-space.

Block Definition Diagrams (BDD)

Block definition diagrams are the most commonly used SysML diagram type when designing systems; they are also the backbone of a system design artifact. A BDD describes system structure through its conceptual parts (software, hardware, material, people, and process) and their relationships, such as generalization. The core element of this diagram is a *Block*, which is a reusable element describing a group of similar objects. Figure 2.3 shows part of the BDD metamodel.

As seen in this metamodel, a block has constraints, operations, and properties, where a property has one of many types: part, reference, flow, or value:

- Part property: lists the owned blocks (composition relationships) or the parts of a block that are shared with other blocks (aggregation relationships).
- Reference property: describes other blocks that are referenced by the block (association relationships).
- Flow property: describes the items that flow between blocks.
- Value property: describes a feature of the block such as a specific weight, width, or speed.

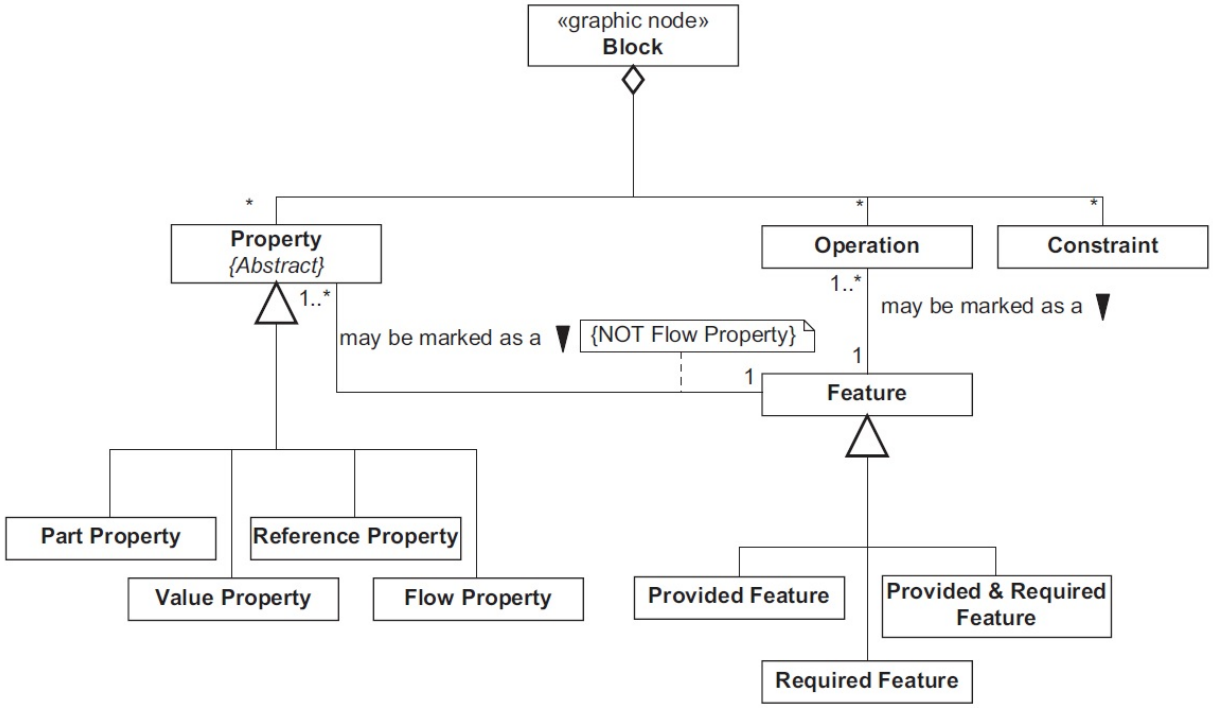

Figure 2.3: Part of the block definition diagram metamodel (from [76]).

- Operation: describes how the block responds to actions/events (required or provided operations).
- Constraint: is a Boolean expression that prescribes the main usage of constraint blocks.

2.1.2 Goal-oriented Requirement Language (GRL)

There exist many goal modeling languages that can be combined with SysML for adaptive SCPSs, including KAOS [154, 159], *i** [160], and the Goal-oriented Requirement Language [17], part of the User Requirements Notation [90]. GRL is discussed further here because this is the only internationally standardized goal modeling language so far and one of the few that supports indicators, which enable monitoring in an adaptive context.

GRL helps capturing stakeholders (roles, organizations, systems, etc., collectively named actors), their intentions (goals, softgoals, or tasks), their relationships (AND/OR decomposition, positive/negative contributions, dependencies), and indicators to measure intention satisfaction based on external evidence. Figure 2.4 illustrates a simplified GRL model of a hybrid car's engine system and its related user's goals. GRL *actors* (illustrated as ellipses ⬭) are used to capture the the system itself as well as its users and other stakeholders. Their *goals* (⬭) should be fulfilled, while their *softgoals* (⬭) point out the non-functional or quality aspects desired. *Tasks* (⬭) capture the alternatives that the system has to chose from in the plan activity. *Indicators* (⬭) are used to monitor internal/external conditions

and to convert this information into satisfaction levels. Intentions can also be AND/OR-decomposed (⟶), whereas an arrow (⟶) with a negative/positive weight (between −100 and +100) represents the contribution of some element to another one. The color coding (the greener, the better) and the numbers above intentions (between −100 and +100 in the standard, or between 0 and +100 in an alternative scale) indicate the current (initial) level of satisfaction in a given context called a *strategy*, whose initial values (shown on a GRL model with the * annotation) are propagated to the other elements and to the actors based on an automated propagation algorithm [16]. A detailed illustration of this model is provided in Section 6.2.

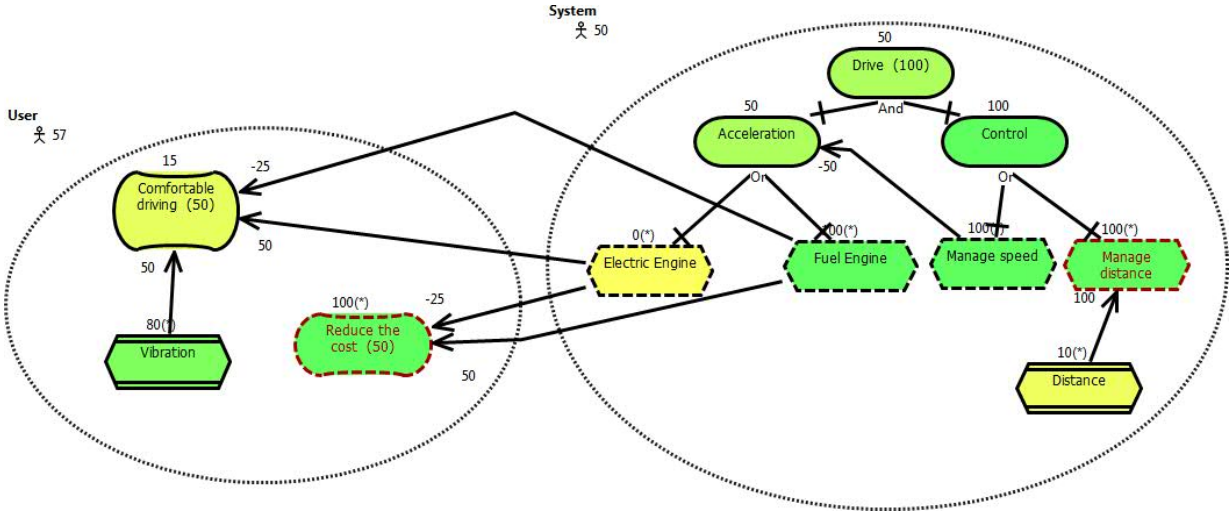

Figure 2.4: Simplified goal model (in GRL) of a hybrid car's engine.

GRL was initially created to support requirements engineering activities at design time; however, it can also be used in an adaptation context [15]. GRL supports a system's dynamic adaptation by connecting goals with requirements, feeding indicators from external sources of information, and providing comprehensive alternative strategies/tasks supporting trade-off analysis.

2.1.3 Feature Models and Diagrams

Product *features* are the characteristics of a product visible to its stakeholders. *Feature models* describe such features and their relationships in terms of commonality and variability, which facilitate product reuse. A *feature diagram* is a graphical representation of a feature model that organizes product features in a hierarchical structure, with additional cross-tree constraints [103]. Features may be decomposed with:

1. *Mandatory* features (—●), which are features that must be included.
2. *Optional* features (—○), which are features that could be included if their parent is included, but not necessarily.

3. Conventional decompositions: exactly one feature (*XOR*), at least one feature (*OR* or *IOR*), or all features *AND* must be included if their parent is included.

In addition to links, feature diagrams include *well-formedness constraints* used to describe that one features requires/includes another one or excludes another one.

A *feature configuration* is a selection of features in a feature diagram/model. Such configuration is *valid* if all the decomposition and well-formedness constraints are satisfied. A feature diagram is valid if it offers at least one valid configuration. *Core features* are features that belong to all valid configurations, whereas *dead features* are features from the model that do not belong to any valid configuration (akin to dead code that cannot be reached in a program).

Feature diagrams have been used widely in architecture selection process, software product line, and system reconfiguration contexts [25]. Figure 2.5 illustrates a sample feature model with different relationships. The green features are part of the selected configuration.

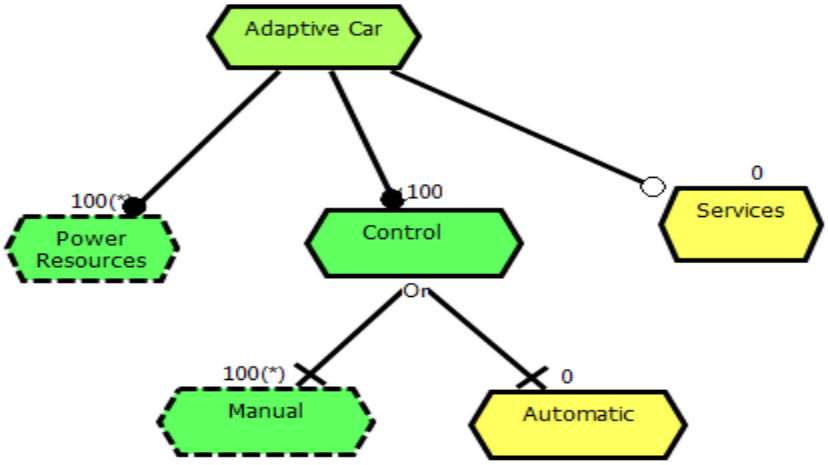

Figure 2.5: Sample feature diagram with different relationships.

2.2 Existing Tools and Technologies

In order to create GRL models and feature models and transform them into arithmetic functions in different programming languages, this thesis exploits two existing tools: jUCMNav and SymPy. Cameo Systems Modeler is used for SysML modeling whereas IBM Rational DOORS is the requirements management system used for traceability management. MI-DSL is used to import models into DOORS whereas optimization models and the IBM CPLEX tool are use to solve models.

2.2.1 jUCMNav

jUCMNav is a free, Eclipse-based graphical editor and analysis tool for URN and feature models [18]. jUCMNav supports the bottom-up evaluation of GRL strategies (i.e., initial values associated with some of the goals/indicators) using different quantitative and qualitative algorithms [16]. Figure 2.6 shows the main screen of the tool, with an evaluated GRL model containing actors, goals, tasks, and indicators. jUCMNav also offers an extensible architecture where model transformations are implemented as plug-ins. Such a transformation was previously defined for exporting a constraint-based representation of GRL models [108], enabling tools to solve models instead of just propagating satisfaction values to higher-level goals and to actors. However, a constraint-based approach is often much too computationally expensive for usage by running systems in an adaptation context.

In this thesis, jUCMNav is used to support goal and feature modeling and analysis.

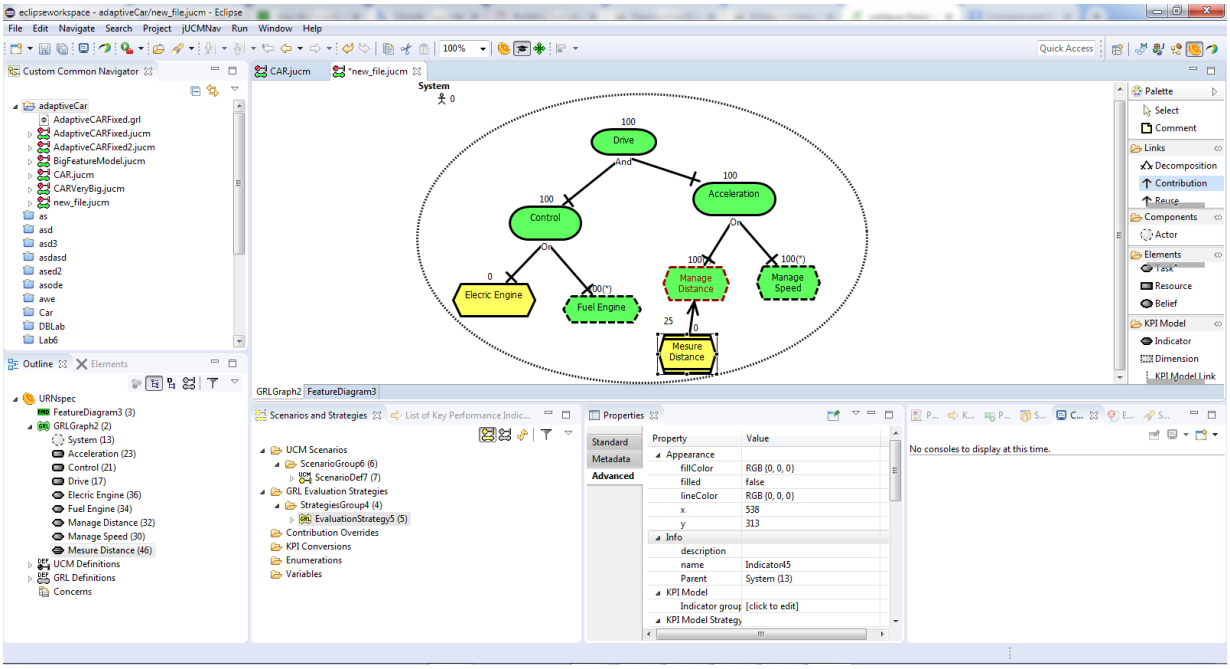

Figure 2.6: jUCMNav tool with an evaluated GRL model.

2.2.2 SymPy

SymPy (Symbolic Python) is a free, lightweight Python library for symbolic mathematics [151]. This library is used to symbolically simplify mathematical expressions (e.g., $10 \times Y/2.5 + 2 * 25$ is simplified to $4 \times Y + 50$), and it includes *code printers* that convert SymPy expressions into target languages such as Java, JavaScript, and C.

SymPy is used in this thesis as the target language for our transformation from GRL and feature models, and then code printers (existing ones, as well as new ones) enable efficient and maintainable code generation to multiple target programming languages.

2.2.3 Cameo Systems Modeler

Cameo Systems Modeler [126] is a Model-Based Systems Engineering (MBSE) modeling tool from No Magic Inc., based on the MagicDraw Modeling Platform. It supports the SysML language including package, requirement, block definition, internal block, activity, use case, and parametric diagrams. It also supports complex system specifications, analysis, and design, as well as system verification and verification. Cameo Systems Modeler can perform simulations and trade-off analysis across SysML parametric diagrams, and constraint blocks that describe mathematical expressions. It also supports model-based system engineering by supporting Domain Specific Languages (DSL), and requirements management by integration with Rational DOORS (Figure 2.7).

In this thesis, Cameo Systems Modeler is used for tracing system design (in SysML) and exploring design alternatives.

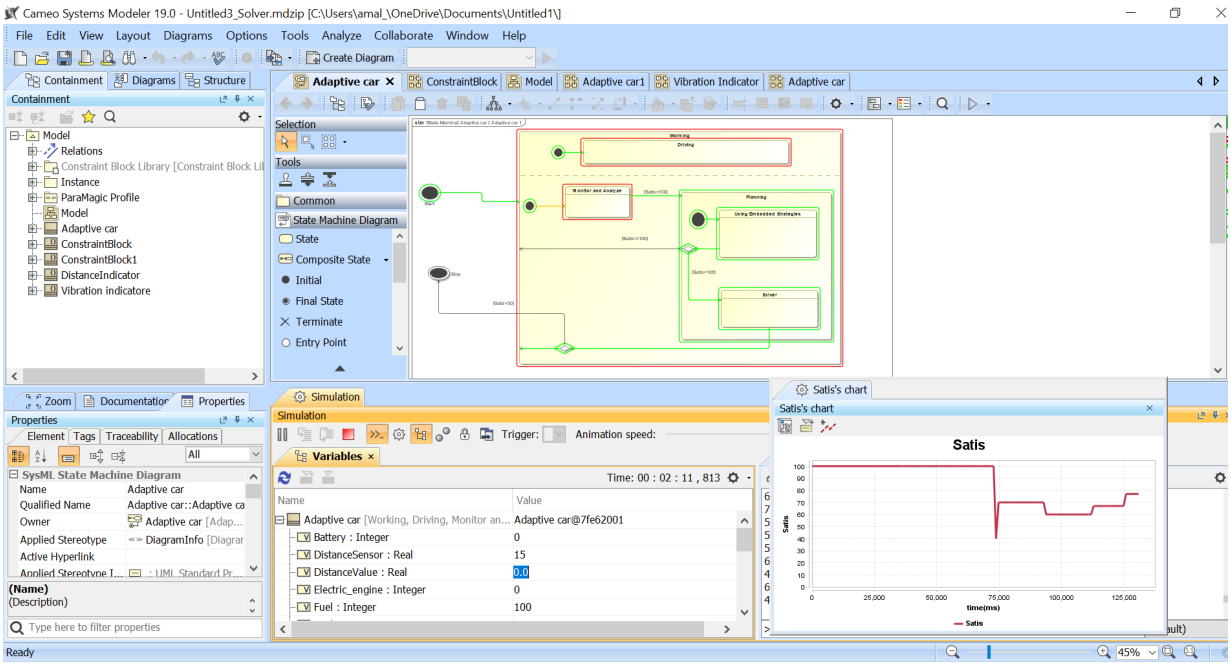

Figure 2.7: Cameo Systems Modeler, for SysML modeling.

2.2.4 Requirements Management System: IBM DOORS

IBM's Dynamic Object Oriented Requirements System (DOORS) is a popular requirements management tool used by CPS developers. DOORS manages extensive information to control system compliance with its requirements and facilitate the communication and cooperation between cross-functional teams in the same project. In addition to capturing product requirements and links between them, DOORS traces and analyzes changes to requirements, providing several views of their impacts on the product as a whole. This thesis exploits and takes advantage of this tool to trace system artifacts (goals, requirements, and

designs) and their changes across development phases, thus helping to ensure that system design is in compliance with system requirements, goals, and quality requirements.

In order to manage traceability between artifacts, they should be first stored in the DOORS database [86]; they can then manipulated using a domain specific language for task automation.

DOORS Database

A DOORS database contains folders, projects, and modules. A *project* is a type of folder that may hold folders and modules. An *object* is the core element of the database and represents the concept that we want to trace (e.g., a requirement). Each object has a type, and it may contain additional attributes defined by users (e.g., status, author, or priority). In the database, a *formal module* is a module that contains the objects in a specific order and forms a particular unit (e.g., a system requirements specification, a software requirements specification, or test cases) whereas a *link module* holds the links between these objects. Link modules essentially define link types (e.g., refines, satisfies, or tests). Link instances (visualized in formal modules with triangles) are typically used for navigation between objects, impact analysis, coverage analysis, change propagation, and document generation.

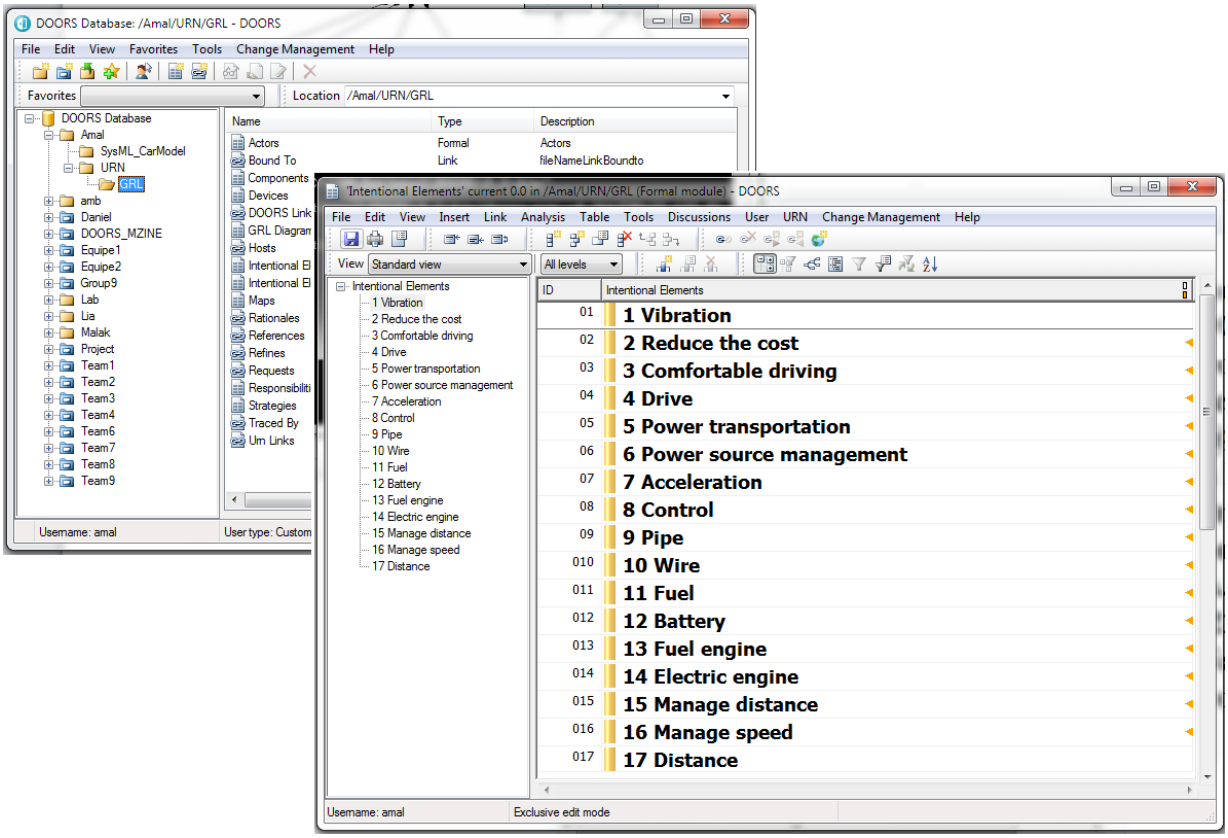

Figure 2.8: A DOORS database and the content of one formal module.

Figure 2.8 illustrates a DOORS database with its projects, as well as the contents of a formal module.

DOORS eXtension Language (DXL)

DXL language is used by DOORS to extend its features and connect with other tools. This is a scripting language whose syntax has characteristics of the C language, with additional database operators. Using DXL, users can:

- Connect DOORS with other tools through an Application Programming Interface (API) for importing and exporting features;
- Create, delete, and modify objects, links, and attributes;
- Create and extend DOORS functions;
- Automate the manipulation process of objects, links, and attributes.

This thesis will use DOORS as an RMS storing and linking SysML and GRL models.

2.2.5 Model Import Domain-Specific Language (MI-DSL)

MI-DSL [139] is a textual DSL that can be used to describe a subset of a modeling language whose models need to be traced and managed in DOORS. MI-DSL facilitates models import and synchronization (through re-imports) by generating a DXL library automatically from a modeling language description. This library can then be used to "execute" models exported to DXL, and import them (with links) to DOORS. Rahman and Amyot [139] identified a metamodel to describe the subset of a modeling language that needs to be traced. They supported this metamodel with an Eclipse-based editor (implemented using the Xtext language development framework [55]) that generates the DXL library automatically. Figure 2.9 describes the metamodel of this domain-specific language.

The MI-DSL metamodel includes the following concepts:

- *Classes*: Describe the modeling language terms that are selected for import in the DOORS database. Classes have any number of typed attributes and are linked together through any number of typed associations.
- *Attribute*: Belongs to a class and specify the name, type, and default value of an attribute.
- *Association*: Belongs to a class and refers to another class. Associations are used to create link sets in DOORS.
- *DataType*: Describes attribute types (Boolean, int, string, text, and diagram).
- *AssociationType*: Describes association types used to create link modules in DOORS.

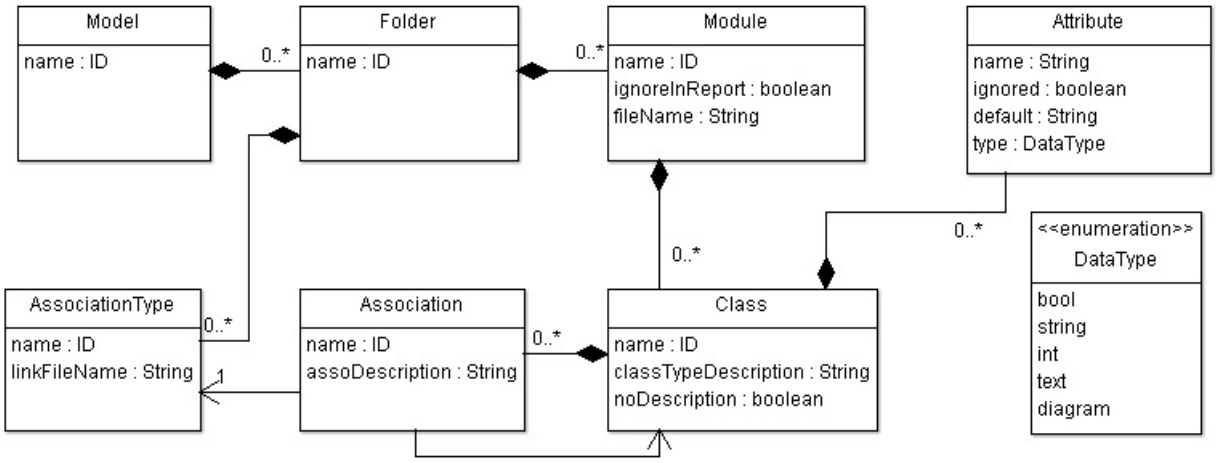

Figure 2.9: The MI-DSL metamodel

- *Module*: Is a collection of classes that have similar or compatible attributes.
- *Folder*: Contains modules.
- *Model*: Contains folders. For each generated library, we must have only exactly one model.

MI-DSL will be used to facilitate the import of SysML, and GRL models into the DOORS database.

2.2.6 Optimization Models

An *optimization model* is a mathematical decision model that contains a set of mathematical functions (objective functions) and variables (decision variables). The *objective functions* are related to real-world problems with quantitative values that need to be minimized or maximized by selecting values for the decision variables. The *decision variables* are the free alternatives of the model and represent its inputs. In addition, if the decision variables are described by functions of other variables, these variables are called *ancillary variables*, whose values are also inputs of the model. A mathematical decision model usually includes constraints that restrict the issued solutions. A set of values of the decision variables satisfying these constraints is called a *feasible solution* and when this set of values minimizes or maximizes the objective function, they represent an *optimal solution* [118].

For example, a tailor shop has workers that make clothes using materials. The owner needs to know how many workers he should hire and what the total price of the produced clothes is, in order to maximize his benefits while offering a competitive price. The salary of a worker is between 1000 and 2500 while the total price should be greater than his expenses and less than or equal to the market price (20000). We define the benefit function to be:

$$Benefit = Total\_price - Cost \tag{2.1}$$

Where:

$$Cost = Materials\_price + Number\_of\_workers \times Salary \tag{2.2}$$

Then from Equations 2.1 and 2.2, the objective function $Max(Benefit)$ becomes:

$$Max(Total\_price - (Materials\_price + Number\_of\_workers \times Salary))$$

The constraints are:

$$(Materials\_price + Number\_of\_workers \times Salary) > Total\_price \leq 20000$$

$$1000 \geq Salary \leq 2500$$

$$1 \geq Number\_of\_workers \leq 9$$

In this example, the *decision variables* are the Number_of_workers, and the Total_price, whereas the *ancillary variables* are the Materials_price and the Salary.

This thesis will use optimization models for finding valid configurations that optimize goal satisfaction during system adaptation. IBM CPLEX [82] is a commercial optimization tool that will be used to solve these models. CPLEX also offer a constraint programming solver (CP Optimizer) that can be embedded as a library in executable systems [84].

2.3 Chapter Summary

In this chapter, we introduced tools and technologies that are exploited by the method developed in this thesis. Cameo Systems Modeler is used for SysML modeling and analysis whereas jUCMNav is used for GRL and feature models. SymPy is used as an intermediate format and simplification engine for converting goal/feature models to mathematical expressions in multiple programming languages. IBM Rational DOORS is used to store, trace, and analyze models imported via MI-DSL and DXL. IBM CPLEX is used for solving optimization models.

The next chapter reviews existing literature related to SysML and goal modeling for adaptive systems.

Chapter 3

Literature Review

This chapter reports on existing work in two relevant areas. The first literature review explores how SysML and goal models were integrated and how this integration supports SCPSs and self-adaptation. This first literature review (Sections 3.1 to 3.4) explores three relevant and complementary topics:

- A systematic review of SysML and goal models integration with and without self-adaptation support;
- A systematic review of the roles SysML models (without goals) play in the development of adaptive systems;
- A review, based on expert suggestions and forward references searches, of the roles goal models (without SysML) play in self-adaptation support.

The collected methods of these three searches are gathered and compared against specific criteria to assess the integration approaches between the two types of models as well as their support for self-adaptation.

The second review (Section 3.5), which is non-systematic, explores the use of quantitative goal-based reasoning in system design and implementation. This work is related to the generation of mathematical functions from goal models.

3.1 First Review: Methodology

Based on Kitchenham's systematic literature review approach [95], three common steps are followed: planning, conducting, and reporting the review. Figure 3.1 shows the three steps and their related activities.

3.1.1 Planning the Review

This step includes setting research questions, identifying the search scope and strategy, as well as formulating quality assessment criteria and data extraction items.

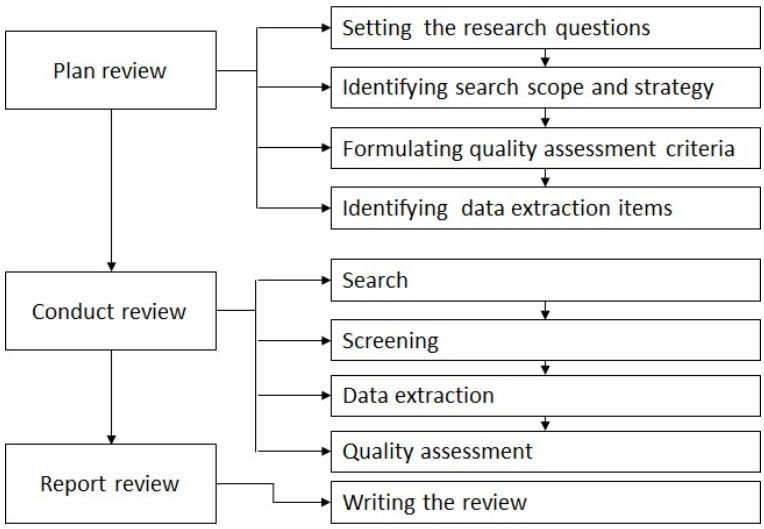

Figure 3.1: Overview of the systematic literature review steps (adapted from [95]).

Setting the Study Goal and Research Questions

SCPSs combine stakeholder goals, software, and hardware components. Some SCPSs may be also be self-adaptive. In this context, the objective of this work is to investigate the possible modeling methods that 1) integrate goal and SysML models, or 2) support self-adaption via SysML only or via the integration of SysML and goal models. The research questions for this objective include two main questions, each of which containing secondary questions.

RQ1 What are the existing methods that integrate goal-oriented models with SysML models?

- SQ1.1 Why have these integrations been proposed?
- SQ1.2 How do the methods integrate the two types of models?

RQ2 What are the collected methods that support self-adaptation?

- SQ2.1 How do the methods support self-adaptive systems?
- SQ2.2 What are the roles that each model plays in this adaptation support?

Identifying the Search Scope and Strategy

The search scope combines three areas: 1) the studies that are relevant to goal models and SysML models together, independently of support for self-adaptive systems; 2) the principal studies that use SysML models to support adaptive systems; and 3) important studies (selected manually) that use goal models to support adaptive systems, as a comparison point outside the SysML world.

The searches were more exhaustive for the first two areas (involving SysML) than for the last one. The main strategy for the first two areas is based on automatic searches performed on popular databases. As the topic of the last step (goal models for adaptive systems) is quite wide and already well covered in the literature, a selection based on a domain expert's opinions and on forward citation searches (i.e., recent papers citing previously selected papers, including from the same authors) was used to highlight the main trends and contributions without being exhaustive.

Data Sources Five important electronic databases were used to discover scientific papers related explicitly to the research questions: Elsevier's *Scopus* and Clarivate Analytics' *Web of Science* are two wide-scope search engines, *IEEE Xplore* and the *ACM Digital Library* are covering the two main societies publishing on systems modeling, and finally *Google Scholar* is a catch-all academic search engine. Note that Google Scholar discriminates less than the other databases in terms of paper quality, and its query language is less powerful than the ones of the other engines. Together, these databases provide a very high coverage of the literature related to SysML and goal-oriented modeling.

Search Queries Many synonyms of goal models were used in order to cover the most common goal modeling languages (*i**, GRL, URN, and KAOS). Adaptive, adaptation, socio-technical, and socio cyber were also considered as potential synonyms. The automatic search was conducted in two phases: 1) SysML/goal integration and 2) SysML model and self-adaptation. Table 3.1 specifies each search conducted with the related query. Because Google Scholar retrieved thousands of papers (with much pollution), we eliminated adaptation/social terms from the second query to ensure papers only integrating goal and SysML models would be included in our dataset, an we excluded goal models from the fourth query to focus on non-goal-oriented SysML adaptation.

These abstract queries were transformed to concrete queries for the different languages used by the databases. With Google Scholar, as we were mainly interested in using its results as a complement to the other (and more reliable) databases while minimizing the effort needed to prune out irrelevant papers, only the first 60 papers returned by each query were further inspected.

Inclusion and Exclusion Criteria The thesis author used inclusion and exclusion criteria to select which papers to keep. She asked for a second opinion (from her supervisor) when the decision was not obvious. The inclusion criteria were:

1. The article is peer reviewed (no book, patent, tutorial, magazine, or gray literature).
2. The articles is written in English.
3. For queries 1 and 2 in Table 3.1, the article provides or clarifies methods about Goal/SysML integration.

Table 3.1: Queries used for Goal/SysML integration (1 and 2) and self-adaptation with SysML (3 and 4).

No.	Search	Query
1	SysML and goal models.	`TITLE-ABS-KEY( SysML AND ( "goal oriented" OR "goal model" OR "i star" OR istar OR KAOS OR "user requirements notation" OR URN OR adaptation OR adaptive OR "Socio cyber" OR "Socio technical" ) )`
2	SysML and goal models, using Google Scholar	`( SysML AND ( "goal oriented" OR "goal model" OR "i star" OR istar OR KAOS OR "user requirements notation" OR URN ) )`
3	SysML models and self-adaption.	`TITLE-ABS-KEY( SysML AND ( adaptation OR adaptive ) )`
4	SysML models and self-adaption, using Google Scholar	`( SysML adaptation OR adaptive ) -"goal model"`

4. For queries 3 and 4 in Table 3.1, the article includes methods using SysML for self-adaptation support.

The exclusion criteria are:

1. The article duplicates (or is a subset of) another paper in terms of the Goal/SysML integration or self-adaptation methods.
2. The article does not provide any information related to our research questions.

The work satisfying one of the exclusion criteria or not satisfying the inclusion criteria (1, 2, and 3 or 4) was excluded.

Some papers did discuss a combination of goal modeling with SysML modeling, but not their integration. For example, Tueno Fotso et al. [153] used models that already integrate KAOS-like AND/OR goal models presented by Laleau et al. [101] with a small subset of SysML for the generation of Event-B specifications (mainly for the formalization and verification of domain models). However, such papers were not included because they do not discuss the integration itself, and in this particular case that integration was already covered by Laleau et al. [101] (exclusion criterion #1) .

Quality Assessment Criteria

We used the checklist in Table 3.2 to provide a qualitative assessment of each study.

Table 3.2: Quality assessment criteria and possible values.

Code	Quality	Qualitative Score
C1	Is the problem specified clearly?	Yes, No, Partially
C2	Is a method provided?	Yes, No, Partially
C3	Is the presented method original?	Yes, No, Partially
C4	Is the method detailed?	Yes, No, Partially
C5	Is the method complete?	Yes, No, Partially
C6	Is a case study provided?	Yes, No
C7	Does the case study clearly illustrate the method?	Yes, No, Partially
C8	Is self-adaptation handled?	Yes, No, Partially
C9	Is self-adaptation specified in detail?	Yes, No, Partially

Identifying data extraction items

Table 3.3 details the extracted data and their related research questions from the planning stage.

For data documentation, from each article, we collected the title, the publication year, the publisher, the authors' names, and the database engine used to retrieve the article.

Table 3.3: Data extracted from each paper.

Questions	**Data item**
Documentation	Title, Year, Publisher, Authors, Database engine
RQ1	Goal model, Automation, Integrated diagrams, Method realization
RQ2	Quality attribute, Realization dimension, Adapted object, Temporal features, Modeling dimension (Goal), Why SysML model.
RQ1, RQ2	Goal model, Goal concepts, Goal analysis, Objective, Development phase, Environment of the method, Realization type

To answer research questions RQ1 and RQ2, the following information was collected by posing the following questions:

1. Are the goals integrated with SysML as a model, or as text/requirement?
2. Are common goal modeling concepts considered in the integration? These include goal types, qualitative/quantitative contributions between different types of goals, and goal dependencies.
3. Is goal analysis considered in the integration?
4. What is the purpose of the integration?
5. Is the integration fully automatic, semi-automatic, or manual?
6. What are the diagram types that were used in the integration?
7. What method was utilized when the integration was done?
8. What are the non-functional requirements (NFRs) that were the focus of attention?
9. For which development phase was the integration done?
10. How are the methods realized? This includes the adaptation type and approach, if any.
    (a) How is the adaptation type explained from these three different perspectives?
        i. When are the alternatives handled? (*closed*: at development phase; or *open*: at runtime).
        ii. Is the method created for a particular field (specific) or not (generic)?
        iii. Is the method model-based or not?
    (b) How is the adaptation approach realized? This is grouped into the following:
        i. The decision-making process decides the adaptation and choses between alternatives (analysis and selection processes) [98]. Is it *static* and created at development time as rules, or *dynamic* using an equation or algorithm?

ii. The adaptation approach is based on the phase of the system in which the adaptation approach was included [91]. Is *Making adaptation* included at development time or *Achieving adaptation* included at runtime using learning approaches?

11. What is the object affected by the adaptation process? Three different sets of information related to this object are:
    (a) The layer in which the object is located (application, middleware, network, hardware, etc.);
    (b) The impacted object (architecture, subsystem, service, component, parameter, etc.); and
    (c) The adaptation action, which could be *weak* or *strong* depending on the effect and cost of adaptation. For example, strong adaptation includes adaptations that add or change the system architecture or components behaviours at runtime. This result exists because much system time and effort is consumed to achieve the adaptation goals. A weak adaptation is related to any inexpensive change.
12. When does the adaptation happen?
13. Does the system monitor specific features or does it monitor its environment continually using sensors?
14. Is human intervention involved in the adaptation process?
15. How is the adaptation done? For example, using a specific language or algorithm.
16. Goal:
    (a) Does the number of goals change at runtime?
    (b) Do system goals remain unchanged, change within constraints, or change without constraints?
    (c) Is the duration of a goal persistent or temporary?
    (d) How many goals are considered in the adaptation process? (Multiplicity)
    (e) Are the goals dependent or independent of each other? (Dependency)
17. What is the reason for the adaptation?
    (a) Is the source of the change *external* (environmental) or *internal* (system)?
    (b) Is the change due to functional requirements, to non-functional requirements, or to a technical reason?
18. Was the time spent for adaptation process guaranteed or not?
19. What was the reason for using a SysML model to specify self-adaptive systems?

3.1.2 Conducting the Review

After having identified the queries and databases engines, the study was conducted (in 2017, and again in December 2018) based on four steps: search, screening, data extraction, and quality assessment.

Search Method

The retrieval of papers to satisfy the conditions we identified included two search methods : 1) goals/SysML integration and 2) SysML and self-adaptation support. The search method for **goals/SysML integration** consisted of the following steps:

1. The first query was used to capture the papers from the Scopus, IEEE, Web of Science and ACM database engines.
2. Because Google scholar retrieves many articles, we executed the following queries using this engine:
   (a) The first query (Table 3.1) was applied to retrieve papers using the "any field" and 'title field" options. The latter retrieved only 10 papers while the former (using any field) retrieved 4,200 papers. We considered only the first 60 papers due to the large number of results, and their decreasing relevance.
   (b) To improve the chances of including most of the relevant papers, we conducted the second query using the "any field" option. This returned 660 papers, and again only the first 60 papers were considered after sorting the list by relevance.

The search method for **SysML and self-adaptive systems** consisted of the following steps:

1. The third query was used to capture the papers from the Scopus, IEEE, Web of Science and ACM database engines.
2. Using Google Scholar, the fourth query was used with the two options, "any field" and "title field" separately. The number of the retrieved papers using the "any field" option was 3,860 papers. As before, only the first 60 papers were included in our dataset. The same query with the "title field" option retrieved only three papers.

To get a non-exhaustive overview of the role that goals have played in supporting self-adaptation features, our dataset was supplied with 12 primary articles using goals in self-adaptive systems by an expert and two more papers using forward search.

Screening

The retrieved papers from the last step were also the input of this process. The exclusion and inclusion criteria were applied on each paper while reading the abstract and conclusion. If the information was insufficient to decide if it was relevant or not, its introduction and method were read. If the information still insufficient to decide, the full-text of the article also was read. The help of another domain expert was required for nine papers because it was difficult to decide whether they were relevant or not.

Result

From the Goal/SysML search, 26 papers were deemed relevant whereas 204 were rejected[1]. When compared with the accepted papers from the first search result, the number of accepted papers from the SysML and adaptation search was small with only seven articles because of the intersection between the two searches. In addition to the last 33 papers, nine papers on goal modeling for adaptive systems were included at the suggestion of an expert and with a simple forward search. This resulted in 42 papers that were eligible for the study. Figure 3.2 illustrates the results of the screening process. Next, the relevant 42 full-text articles were read and summarized.

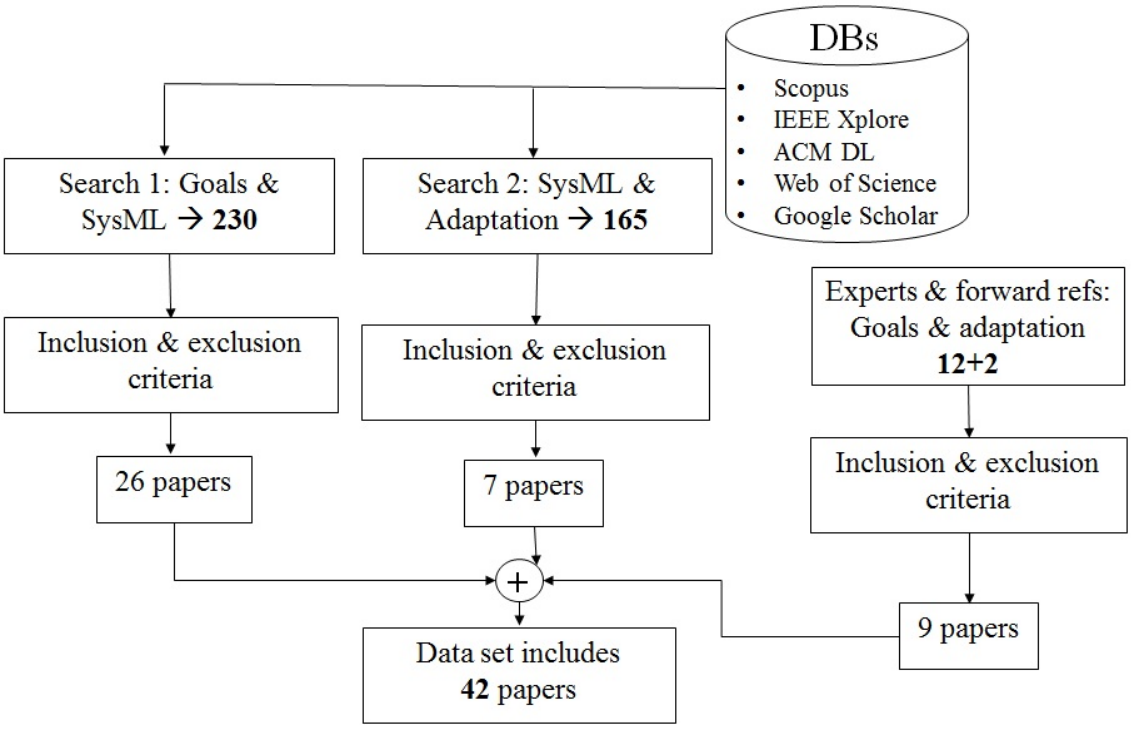

Figure 3.2: Result of the screening process

[1]A table with the accepted and rejected papers is available online at http://site.uottawa.ca/~damyot/pub/GoalSysMLslr/

Data Extraction

For each study, we extracted the data items mentioned in Table 3.3. Extracting this data was done iteratively from the selected studies to accumulate information concerning our research questions.

Quality Assessment Process

The studies collected through the 3 approaches were compared against the criteria listed in Table 3.2. We did not evaluate how good the articles were, but we did evaluate how useful each would be to the study. All papers met at least 3 criteria and were kept. The result of the quality assessment process is detailed in Tables 3.4 and 3.5.

Table 3.4: Assessment of the studies on Goal/SysML against the identified quality criteria. (Y=Yes, N=No, P=Partially, –= Not provided)

Research study	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
Amyot et al. 2016 [14]	Y	Y	Y	N	N	Y	Y	Y	N
Ahmad 2010 [1]	Y	Y	Y	Y	N	N	–	Y	P
Ahmad et al. 2012 [6]	Y	Y	N	Y	Y	Y	Y	Y	Y
Ahmad et al. 2013 [2]	Y	–	–	–	–	Y	Y	P	N
Ahmad et al. 2013 [7]	Y	Y	N	Y	Y	Y	Y	P	P
Ahmad et al. 2015 [3]	Y	Y	N	Y	Y	Y	–	Y	Y
Ahmad and Bruel 2013 [4]	Y	Y	N	Y	Y	Y	Y	P	P
Ahmad and Bruel 2014 [5]	Y	–	–	–	–	N	–	Y	Y
Apvrille and Roudier 2013 [26]	Y	Y	Y	Y	Y	N	–	N	N
Apvrille and Roudier 2015 [27]	Y	Y	N	Y	Y	Y	Y	N	N
Badreddin et al. 2016 [30]	Y	Y	Y	P	P	Y	Y	Y	N
Belloir et al. 2014 [35]	Y	Y	N	Y	Y	Y	Y	Y	P
Bousse 2012 [41]	Y	P	P	N	N	N	–	N	N
Cui and Paige 2012 [51]	Y	Y	Y	Y	Y	Y	P	N	N
Fan et al. 2016 [60]	Y	Y	Y	Y	Y	Y	Y	N	N
Gnaho et al. 2013 [68]	Y	Y	N	Y	Y	Y	Y	N	N
Gnaho et al. 2013 [69]	Y	Y	P	Y	Y	N	–	N	N
Ingram et al. 2014 [88]	Y	Y	Y	Y	Y	Y	Y	Y	Y
Laleau et al. 2014 [101]	Y	Y	Y	Y	Y	Y	Y	N	N
Mammar and Laleau 2016 [111]	Y	Y	N	Y	Y	Y	P	N	N
Matoussi et al. 2011 [112]	Y	Y	N	Y	Y	N	–	N	N
Ozkaya 2007 [134]	Y	P	Y	N	N	N	–	N	N
Roudier and Apvrille 2015 [143]	Y	Y	N	Y	Y	Y	Y	N	N
Spyropoulos and Baras [150]	Y	Y	Y	Y	Y	Y	Y	P	N
Tsadimas et al. 2012 [152]	Y	Y	Y	Y	Y	Y	Y	N	N
Vanderperren and Dehaene 2005 [155]	Y	P	Y	P	N	N	–	P	N

Table 3.5: Assessment of the adaptation studies on Goal *or* SysML searches against the identified quality criteria. (Y=Yes, N=No, P=Partially, –= Not provided)

Research study	C1	C2	C3	C4	C5	C6	C7	C8	C9
Goals and Adaptation									
Baresi et al. 2010 [34]	Y	Y	Y	P	Y	Y	Y	Y	P
Baresi and Pasquale 2010 [32]	Y	Y	P	Y	Y	Y	P	Y	Y
Baresi and Pasquale 2010 [33]	Y	Y	P	P	Y	Y	Y	Y	P
Horkoff et al. 2014 [77]	Y	Y	Y	Y	Y	N	–	N	N
Morandini et al. 2008 [115]	Y	Y	N	N	Y	Y	Y	P	N
Morandini et al. 2017 [116]	Y	Y	Y	Y	Y	Y	Y	Y	Y
Qian et al. 2015 [138]	Y	Y	Y	Y	Y	N	–	Y	Y
Ramnath et al. 2008 [141]	Y	Y	Y	Y	Y	Y	Y	Y	P
Welsh et al. 2014 [157]	Y	Y	Y	Y	Y	Y	Y	N	N
SysML and Adaptation									
Akbas and Karwowski 2013 [8]	Y	Y	N	Y	Y	Y	Y	N	N
Akbas et al. 2014 [9]	Y	Y	Y	Y	Y	Y	Y	N	N
Baresi et al. 2016 [31]	Y	Y	Y	Y	Y	Y	Y	Y	Y
Lopes et al. 2011 [106]	Y	Y	Y	P	P	Y	P	P	P
Meacham 2017 [113]	Y	Y	Y	Y	Y	Y	Y	P	N
Ribeiro et al. 2013 [142]	Y	Y	Y	Y	Y	Y	Y	P	N
Soyler and Sala-Diakanda 2010 [149]	Y	Y	Y	P	P	Y	Y	P	P

3.2 First Review: Discussion

We classify and present the selected studies in ways that will enable answering our questions accurately. In particular, we split the discussion into four subsections: integration methods, adaptation support methods, adaptation assessment, and traceability and integration assessment.

3.2.1 Integration Methods

To answer the first research question (RQ1: What are the existing methods that integrate goal-oriented models with SysML models?), we used the articles retrieved by the Goal/SysML search (Table 3.4). To get a deeper understanding of the integration itself, we classified the studies according to the applied methods and current objectives. A total of 13 methods, named M1 to M13 in Table 3.6, are presented amongst these 26 studies. The types of goal modeling languages and SysML diagrams used in each method are also listed in Table 3.6, and the main paper in each collection is highlighted in **bold**.

Table 3.6: Selected studies and their methods used (Functional goals = FG; Non-functional goals = NFG)

Research study	**Code**	**Goal language**	**SysML diagram**
Ingram et al. [88]	M1	Fault tolerance strategies	Dependency relationship, Block diagram
Amyot et al. [14]	M2	GRL	Requirement diagram Block diagram, others
Vanderperren and Dehaene [155]	M3	NFRs	Use cases, behavioural diagrams
Ozkaya [134]	M4	GRL	Requirement diagram
Matoussi et al. [112], Laleau et al. [101]	M5	KAOS (FG)	Requirement diagram
Cui and Paige [51]	M6	Business Motivations	Requirement diagram
Gnaho et al. [68, 69], Mammar and Laleau [111], Bousse [41]	M7	KAOS (FG, NFG)	Requirement diagram
Ahmad et al. [3], Ahmad et al. [6, 2, 7], Ahmad [1], Ahmad and Bruel [4, 5], Belloir et al. [35]	M8	KAOS (FG, NFG), RELAX	Requirement diagram
Apvrille and Roudier [26, 27], Roudier and Apvrille [143]	M9	NFR, Mechanisms	Requirements, Block, state machine and Parametric diagrams
Tsadimas et al. [152]	M10	Quantitative NFR	Requirements
Spyropoulos and Baras [150]	M11	Trade-off analysis	Block definition, Parametric diagrams
Badreddin et al. [30]	M12	Goals	Block, Requirements, Users, Use case
Fan et al. [60]	M13	Stakeholders goals	Block, Requirements, Activity

Languages and Diagrams Involved

For each method, we extracted the goal modeling language and SysML diagrams used (Table 3.6). Any additional model was considered out of the scope of the study.

The most commonly used SysML diagrams are requirement diagrams and block diagrams, in that order. All presented methods but three (M1, M3, M11) connected goal models with requirement diagrams, while nearly half the methods (M1, M2, M9, M11, M12, M13) used block diagrams in their integration. From a goal model perspective, many different languages were used. The most popular language is KAOS [154] (M5, M7, M8) while GRL [90] ranks second (M2, M4). Three methods connected only security/safety NFRs with SysML diagrams while two methods used goals as a general term. Instead of integrating goal models themselves, Ingram et al. [88] used goal model analysis results in their integration to increase the confidence of system designers when defining the system architecture.

Objectives of the Integration

In our investigation, extracting information about the adaptation objective is different from extracting information for the integration itself. To answer the secondary question SQ1.1 (Why have these integrations been proposed?), we clustered the studies according to their objectives to figure out which ones were most frequently used in the literature to justify a goal/SysML integration.

Table 3.7 reports on seven main objectives, together with their related methods and articles. The uncertainty and adaptation objective has attracted the highest number of studies (10), with three different methods. However, the architecture selection and modeling objective has a more varied set of methods (5). These two important objectives are followed by formal validation and verification, traceability, process improvement, requirements visualization, and impact of non-functional requirements on functional requirements.

Method Characteristics

The integration of goal models with SysML models has different dimensions depending on the objective of the study and the researchers' vision for a specific problem and its solutions. To answer the secondary question SQ1.2 (How do the methods integrate the two types of models?), Table 3.8 includes information about each study and the related data that explains the following:

1. Whether the environment was specific or generic,
2. Whether the method was automated,
3. Whether the method integrated goals as a model,
4. Whether the main goal concepts are used in the integration,

Table 3.7: Objectives of Goal/SysML integration and related methods

Objectives	Methods	Articles
Uncertainty & adaptation	M1, M2, M8	Ingram et al. [88], Amyot et al. [14], Ahmad [1], Ahmad et al. [3, 6, 2, 7], Gnaho et al. [68], Ahmad and Bruel [4, 5], Belloir et al. [35]
Architecture selection & modeling	M1, M9, M10, M11, M13	Ingram et al. [88], Apvrille and Roudier [26, 27], Roudier and Apvrille [143], Tsadimas et al. [152], Spyropoulos and Baras [150], Fan et al. [60]
Formal V&V	M5, M7, M8	Matoussi et al. [112], Laleau et al. [101], Bousse [41], Ahmad et al. [7], Mammar and Laleau [111]
Traceability	M6, M12	Cui and Paige [51], Badreddin et al. [30]
Development process improvement	M3	Vanderperren and Dehaene [155]
Requirements visualization	M4	Ozkaya [134]
Impact of NFRs on FRs	M7	Gnaho et al. [69]

5. Whether goal analysis was supported, and

6. The method realization (usually through a profile).

Note that some methods were still under development or did not provide sufficient details about their process, so the level of automation was difficult to assess at times. Most of the studies did not mention how goals or requirements are transferred to extended SysML profiles. Some of them developed specific editors for their methods but did not explain if the goals or requirements were translated automatically or re-entered manually.

Goals Model Mapping

To measure how far the methods go in their integration, Table 3.8 includes columns that are further explained below.

MDE Approach Does the method support an MDE (automated) approach? Several studies [38] have addressed the advantages of an MDE approach, including information traceability, holistic validation and verification, as well as code generation. These features are not only important to support self-adaptability, but also to improve productivity and system quality [38, 155]. As seen in Table 3.8, most selected methods used goals (partially) as a model with SysML requirement diagram. In contrast, Badreddin et al. [30] proposed the only method that does not support a graphical MDE approach and presented a new

Table 3.8: Extracted data on the integration dimensions (Y=Yes, P=Partially, N=No, ?=Unknown)

Research study	**Environment**	**Auto**	**Goal model**	**Goal concepts**	**Goal analysis**	**Method realization**
Amyot et al. [14]	Specific	P	Y	P	Y	Investigating
Ahmad [1]	Generic	P	P	P	N	Profile
Ahmad et al. [6]	Specific	P	P	P	N	Profile
Ahmad et al. [2]	Generic	P	P	P	N	Profile
Ahmad et al. [7]	Specific	P	P	P	N	Profile
Ahmad et al. [3]	Generic	P	P	P	N	Profile
Ahmad and Bruel [4]	Generic	P	P	P	N	Profile
Ahmad and Bruel [5]	Generic	P	P	P	N	Profile
Apvrille and Roudier [26]	Specific	P	N	P	P	Profile
Apvrille and Roudier [27]	Specific	P	N	P	P	Profile
Badreddin et al. [30]	Generic	?	N	P	N	Textual manner
Belloir et al. [35]	Specific	P	P	P	N	Profile
Bousse [41]	Specific	?	P	P	N	Profile
Cui and Paige [51]	Specific	?	P	P	N	Profile
Fan et al. [60]	Specific	P	N	N	N	Profile
Gnaho et al. [68]	Generic	?	P	P	N	Profile
Gnaho et al. [69]	Specific	?	P	P	N	Profile
Ingram et al. [88]	Specific	?	N	N	P	Profile
Laleau et al. [101]	Specific	?	P	P	N	Profile
Mammar and Laleau [111]	Specific	?	P	P	N	Profile
Matoussi et al. [112]	Specific	?	P	P	N	Profile
Ozkaya [134]	Generic	?	?	?	N	Investigating
Roudier and Apvrille [143]	Specific	P	N	P	P	Profile
Spyropoulos and Baras [150]	Specific	P	N	P	P	Profile
Tsadimas et al. [152]	Specific	P	N	P	N	Profile
Vanderperren and Dehaene [155]	Specific	?	N	P	N	Profile

language that combines the models using a textual syntax. In many studies, goals have actually been translated to a textual, hierarchical structure using a requirement diagram and profiles. Some studies used SysML block and parametric diagrams with some goal model analysis such as trade-off analysis.

Goal Modeling Concepts Were the main goals modeling concepts (goals, softgoals, decompositions, actor importance, contribution weights, indicators, etc.) part of the integration with the SysML model? Goals were integrated with SysML requirement diagrams in most methods but not all goal modeling concepts were mapped. These methods extended requirement diagrams with goal types (functional and/or non-functional) and some goal relationships (mainly AND/OR decomposition). However, contributions between functional goals, quantitative/qualitative contribution weights, and indicator parameters (lowest, highest, and threshold) are seldom covered. This prevents modelers from 1) performing goal analysis to guide the selection of alternatives (at design time) and 2) supporting dynamic adaptation at runtime according to user preferences [109]. For example, Cui and Paige [51] integrated goals model without considering the quantitative values of the contribution relationships between goals or indicator parameters, whereas Ahmad et al. [3] integrated all types of goals and their relationships except for contribution weights and importance levels.

When dealing with non-functional requirements, the methods presented by Apvrille and Roudier [26, 27], Tsadimas et al. [152], and Spyropoulos and Baras [150] focused on the important role of goal-oriented techniques in system architecture and design selection. However, none of them actually transformed or linked goals to the design phase. Instead, they broke down systems goals into non-functional requirements and linked them to design elements of SysML requirement diagrams.

In summary, the methods generally support some level of goal-orientation, but none practically integrates all goal concepts with SysML models.

Goal Analysis Trade-off analysis can be conducted through positive and negative contributions between goals during the decision-making process. Some methods were used to analyze fault tolerance and security mechanisms using quantitative values in their goal/SysML integration, but mainly to select the best architecture/design [26, 27, 3] or to include possible choices in the system implementation phase [88, 150]. However, these analyses are just exploiting the goal models as a whole and are addressing a specific feature or behaviour.

Method Realization As seen in Table 3.8, all but three studies used a profile to map goal concepts to SysML. Badreddin et al. [30] proposed integrating both views through a new textual language, whereas two other studies were still investigating this aspect.

Method Environment

Eight studies out of 26 were conducted for generic environments. The generic methods were presented by Ozkaya [134], Gnaho et al. [69], Ahmad et al. [3], and Badreddin et

al. [30]. The 18 remaining studies were produced for specific fields, as shown in Table 3.9, where we included the related research field or project name. The TACOS project and EVITA embedded systems are the target of the largest number of studies.

Table 3.9: Distribution of the studies over fields of study and projects

Research study	**Related field/project**
Amyot et al. [14]	Socio-cyber-physical systems
Ahmad et al. [6, 7]	AAL system
Apvrille and Roudier [26, 27], Roudier and Apvrille [143]	**EVITA embedded systems**
Belloir et al. [35]	Security of Systems of Systems
Bousse [41]	Mitsubishi Electric R&D Center Europe (MERCE)
Cui and Paige [51]	Business Motivation and Alignment
Fan et al. [60]	Ship Command and Control Systems (SCCSs)
Ingram et al. [88]	COMPASS project
Mammar and Laleau [111]	FORMOSE project
Matoussi et al. [112], Laleau et al. [101], Gnaho et al. [69]	**TACOS project (component localization)**
Spyropoulos and Baras [150]	System Modeling Hub at the University of Maryland
Tsadimas et al. [152]	Enterprise Information Systems
Vanderperren and Dehaene [155]	Systems-on-a-chip

3.2.2 Adaptation Support Methods

To answer research question RQ2 (What are the collected methods that support self-adaptation?), we selected additional articles coming from digital libraries and provided by experts. The sub-questions SQ2.1 and SQ2.2 are answered using adaptation concepts and dimensions.

In order to find the methods that support self-adaptation characteristics in a context where adaptation objectives are not explicitly mentioned in some of the studies, we classified the methods using two criteria: self-adaptation properties and adaptation type. These two criteria are respectively based on two classifications: 1) the non-functional requirements that guide a particular system architecture design, and 2) the phase used to realize the adaptation.

Self-Adaptation Properties

We classified the studies based on the four common *self-** properties of self-adaption [92, 37], namely self-healing, self-configuration, self-optimization, and self-protection. This

classification was done with the help of related quality attributes, as suggested by Mistrik et al. [114] and Salehie and Tahvildari [145]. We extracted the non-functional requirements (NFRs) cited in the 42 eligible studies before we related them to the self-management properties.

For the studies where adaption rationale was available, we established a mapping to self-management types via NFRs. Table 3.10 details the results. Fourteen methods support systems in adapting themselves while running by responding to changes that could be external (environmental) or internal (the system itself) [23]. Only three of them ([88, 3, 150]) integrate goal and SysML models for system design and self-adaptation.

Table 3.10: Distribution of self-* properties and non-functional requirements among the studies.

Self-management	**NF-requirements**	**Research studies**
Self-Healing	Fault diagnosing tolerance	Ingramet al.[88], Morandini et al. [115, 116], Baresi et al. [31]
Self-Configuration	Adaptability, Integrity and Availability	Ahmad [1], Ahmad et al. [6, 2, 7, 3], Ahmad and Bruel [4, 5]
	Adaptability	Meacham [113], Baresi et al. [34], Baresi and Pasquale [32, 33]
	Reliability	Ribeiro et al. [142]
	Usability	Welsh et al. [157]
Self-Optimization	Resource utilization	Spyropoulos and Baras [150], Lopes et al. [106]
	Time behaviour	Qian et al. [138]
Self-Protection	Security	Belloir et al. [35], Ramnath et al. [141]

In terms of adaptation approaches that do not combine goals with SysML, several methods such as Morandini et al. [115, 116], Qian et al. [138], Ramnath et al. [141], Welsh et al. [157], Baresi et al. [34], and Baresi and Pasquale [32, 33] use goal models only to support self-adaptation while the rest of the methods use SysML. Additional diversity is brought by pattern-based and case-based approaches [113, 138].

Adaptation Phase

In order to cover methods that support adaptation without clarifying NFRs that guide system design, we classified the studies according to their adaptation type: design time or runtime adaptation. In design time adaptation, the situations triggering adaptation, the adaptation mechanisms, and the strategies for decision making are already known and applied in the system at design time. Systems that apply runtime adaptation are distinguished by the ability to deal with unpredictable environmental changes while running [113, 145].

Table 3.11 shows that most of the studies that integrate goal and SysML models target the design phase for different reasons but adaptation, while most of the methods that target

self-adaptation at runtime trough SysML models or goal models separately implement their adaption strategies, mechanisms, and decision at design time (design time adaptation). Runtime adaption in SysML is currently lacking contributions.

Table 3.11: Distribution of the methods and studies among adaptation types

Data Source	**Without Adaptation**	**Adaptation at Runtime**	
	Analysis & Design	Adaptation Type	
		Design Time Adaptation	Runtime Adaptation
Goal/SysML	Vanderperren and Dehaene [155], Ozkaya [134], Matoussi et al. [112], Laleau et al. [101], Cui and Paige [51], Gnaho et al. [68, 69], Mammar and Laleau [111], Bousse [41], Apvrille and Roudier [26, 27], Roudier and Apvrille [143], Tsadimas et al. [152], Badreddin et al. [30], Fan et al. [60]	Amyot et al. [14], Ingram et al. [88], Spyropoulos and Baras [150], Ahmad [1], Ahmad et al. [2, 3, 6, 7], Ahmad and Bruel [4, 5], Belloir et al. [35]	
SysML and adaptation	Akbas et al. [9], Akbas and Karwowski [8]	Meacham [113], Ribeiro et al. [142], Baresi et al. [31], Lopes et al. [106], Soyler and Sala-Diakanda [149]	
Goals and Adaptation	Horkoff et al. [77]	Morandini et al. [115, 116], Ramnath et al. [141], Baresi et al. [34], Baresi and Pasquale [32, 33], Welsh et al. [157]	Qian et al. [138]

Method Characteristics

To get an overview and fair assessment of each method that used goal models or SysML models separately, we extracted the information illustrated in Table 3.12. The fully-automated approach of Morandini et al. [116] needs four tools, which negatively affects its usability, while the other studies did not explicitly state the number of tools used. Only four methods out of 14 were generic while the rest presented their approaches within a specific field of study. It is noteworthy that all the generic methods are based on goal modeling.

Table 3.12: Extracted data from studies that used SysML models or goal models separately

Research Study	**Environment**	**Field of study**	Auto.
Morandini et al. [115, 116]	Generic	–	Yes
Qian et al. [138]	Generic	–	Unknown
Ramnath et al. [141]	Specific	Enterprise architecture goals	No
Meacham [113]	Specific	IoT applications	Yes
Ribeiro et al. [142]	Specific	Real-time systems	Partial
Baresi et al. [31]	Specific	Automated production systems (APS)	Yes
Lopes et al. [106]	Specific	Energy Smart Grids	Unknown
Soyler and Sala-Diakanda [149]	Specific	Disaster management systems	Unknown
Akbas and Karwowski [8] and Akbas et al. [9]	Specific	Technology Training Management System TTMS	Unknown
Horkoff et al. [77]	Generic	–	Partial
Welsh et al. [157]	Specific	GridStix system	Yes
Baresi et al. [34]	Generic	–	No
Baresi and Pasquale [32, 33]	Specific	Service compositions	Partial

Adaptation Approaches

Tables 3.13 and 3.14 summarize the approaches each method applies to meet its objectives. Three methods (in four articles) used the i^* goal modeling language while four methods used the KAOS language (in 11 articles). RELAX [158] was used in a few instances. Several integrations were also done with goal models or SysML models separately, hence answering the sub-question SQ2.2.

Table 3.13: Methods using combined Goal/SysML models to represent self-adaptive systems.

Study	**Approach**
Ingram et al. [88]	Employed conditions and roles of a fault tolerance study to choose the best strategy for managing traffic problems.
Ahmad et al. [3]	Used SysML, KAOS and RELAX to manage uncertainty at runtime.
Spyropoulos and Baras [150]	Used trade-off analysis to optimize resource distribution of an Electrical Microgrid system using mathematical algorithms applied in a SysML model. The last model was integrated with the Consol-Optcad optimization tool for early cost and performance estimation.

Goal Model Integrations

1. In order to support dynamic adaptive systems, Morandini et al. [115, 116] integrated models of goals, failures, and the environment.

Table 3.14: Methods using SysML models or goal models separately to represent self-adaptive systems.

Study	Approach
Morandini et al. [115, 116]	Unified goal model (i^*), failure model and environmental model to support self-adaptation.
Qian et al. [138]	Combined strategies selection and case-based reasoning self-adaptation approaches. In order to determine the embedded strategies, the lowest level of parameterized goal models was linked with the highest level of softgoals via weighted contribution relationships.
Ramnath et al. [141]	Linked strategies for attack and protection at the design layers of the proposed architecture.
Meacham [113]	Combined pattern-based with case-based reasoning approaches where repeated falls were collected and analyzed to identify their patterns, leading to solutions as plans.
Ribeiro et al. [142]	Modeled real-time requirements and managed traceability through extending SysML requirement diagram with relationships and properties. Synchronized relationships were used to represent parallel real-time requirements.
Baresi et al. [31]	Modified the SysML meta-model to create a SysML4Pack profile that combines SysML model, OCL [156] and the state machines of OMAC Baresi et al. [31] to represent predictable faults of automatic production systems.
Lopes et al. [106]	Integrated SysML models with trade-off analysis and techno-economical cost-benefit analysis to optimize electricity management, generation, and distribution among customers.
Soyler and Sala-Diakanda [149]	Included disaster management strategies in the system architecture with continuous feedback from the last disaster data.
Akbas and Karwowski [8]	Combined dynamic models with agent-based models that were extracted from system design using SysML models.
Horkoff et al. [77]	Integrated goal models i^* with the MAVO framework of [144] to iterate over the analysis process for early uncertainty reduction.
Welsh et al. [157]	Integrated alternative goals with system behaviour history using claim refinement models Welsh et al. [157] and i^* goal models.
Baresi et al. [34]	Modified the KAOS language with fuzzy goals (i.e., non-functional goals with uncertainty) leading to a new language called FLAGS, which supports functional models (crisp goals) and adaptive models (fuzzy goals). The crisp goals were formalized through Linear Temporal Logic language (LTL) [136] plus fuzzy temporal operations such as $<$, $>$, $<=$, and *approximately* to express the fuzzy goals.

Table 3.15: Methods using SysML models or goal models separately to represent self-adaptive systems (continued).

Study	Approach
Baresi and Pasquale [33]	Used service composition based on the Business Process Execution Language (BPEL) [56] to transform the FLAGS/KAOS model in Baresi et al. [34] to membership functions and abstract processes, semi-automatically. These functions trigger the adaptation strategies using Boolean conditions.
Baresi and Pasquale [32]	Added operators from RELAX Language to the FLAGS language in Baresi and Pasquale [33] to represent the fuzzy goals. Member functions are used in the monitoring process but the adaptation strategies are triggered by conditions associated with the operational model.

2. To deal with unpredicted changes at runtime, Qian et al. [138] integrated goal models with case-based reasoning.

3. Horkoff et al. [77] integrated goal models with the MAVO framework to reduce uncertainty early.

4. Baresi et al. [34] and Baresi and Pasquale [32, 33] described goals using a formal linear temporal logic (LTL) language and the RELAX language for usage at runtime.

SysML Model Integrations

1. Meacham [113] did an integration with UML to specify cases of presented patterns, while Soyler and Sala-Diakanda [149] also supported an integration with UML, but this time to represent structure and behaviour of systems in one single environment.

2. Additional relationships and properties were added to SysML requirement diagrams by Ribeiro et al. [142] for representing runtime requirements in a hierarchical way and for managing requirements traceability for system validation and verification.

3. Baresi et al. [31] used an integration with OMAC state machines, ISA-88 physical models and OCL constraints for transforming models to code.

4. Lopes et al. [106] provided an integration supporting trade-off analysis and techno-economical cost-benefit analysis when modeling detailed system architectures.

5. System dynamic models and agent-based simulation were integrated by Akbas et al. [9] and Akbas and Karwowski [8] for minimizing system complexity and specifying system agents in a hierarchical structure.

3.2.3 Adaptation Assessment

In order to answer research questions SQ2.1 and SQ2.2 on adaptation methods, we extracted information that identifies terms inspired from the adaptation taxonomies [98, 145] and modeling dimensions of self-adaptation [23, 48]. Using these terms was helpful in inferring correct indicators that specify how each method supports self-adaptation and what the roles that each model plays in this adaptation support are.

Among the papers collected, some articles were eliminated from this assessment because their adaptation methods were redundant or not described in sufficient detail. In particular, Akbas and Karwowski [8] and Horkoff et al. [77] designed self-adaptive systems for reducing uncertainty and system complexity while Welsh et al. [157] explained self-adaptive system behaviour via goals and the history of system behaviour. Thus, they supported the use of self-adaptive systems but not the use of adaptation where the system reconfigures itself to become more usable. In addition, Baresi et al. [34] and Baresi and Pasquale [32, 33] expressed the same methods with different emphases, so we considered them as one method represented by the most detailed paper [32]. Finally, Spyropoulos and Baras [150] provided information about their dynamic decision-making process but not about the adaptation strategies and properties, so this paper was excluded from the adaptation properties and dimensions assessment, except for the decision-making criterion. As a result, 11 methods are discussed in this assessment.

Adaptation Terms

The selected terms were defined in Section 3.3 on data extraction. Table 3.12 highlighted the specific and generic methods for self-adaptation that use goal models or SysML models separately. Table 3.16 illustrates the assessment of each adaptation term against the related methods and studies.

Adaptation Modeling Dimensions

Three types of modeling dimensions (goal, change, and mechanisms) are used to specify self-adaptive properties [23]. However, some of these properties are overlapping with the adaptation taxonomy previously mentioned. Some of the collected methods, such as those presented by Ahmad et al. [3] and Baresi and Pasquale [32], are generic and can be applied to different applications; we estimated their values based on the provided information. We extracted the methods' information related to the chosen modeling dimensions, which is summarized in Table 3.17.

Assessment Results

From Tables 3.16 and 3.17, several observations can be made:

- Most of the collected methods realize a *closed* approach of adaptation by including their strategies with system design. Only two were clearly *open*.

Table 3.16: Adaptation terms related to each selected method.
(C=Closed, O=Open, ?=Not provided, Y=Yes, N=No, P=Partially, Dy=Dynamic, Sta=Static, M=Making, A=Achieving, Md=Middleware, Ap=Application, Sr=Service, St=Structure, W=Weak, Rt=Reactive, Pt=Proactive, Co=Continuous, Ad=Adaptive)

Adaptation terms	Ingram et al. [88]	Ahmad et al. [3]	Morandini et al. [116]	Qian et al. [138]	Ramnath et al. [141]	Meacham [113]	Ribeiro et al. [142]	Baresi et al. [31]	Lopes et al. [106]	Soyler and Sala-Diakanda [149]	Baresi and Pasquale [32]
Adaptation Type	C	N	C	O	C	C	C	C	?	O	C
Model-based	Y	Y	Y	P	Y	Y	Y	Y	Y	Y	Y
Decision (Analyze / Selection activities)	?	Sta	Sta	Dy	Sta	Sta	?	Sta	?	?	Dy/ Sta
Adaptation approach	M	M	M	M	M	M	M	M	M	M	M
Layer	Md	Ap	Ap	Ap	Ap	Ap	?	?	Md	Ap	Ap
Artifact	Sr/St	Sr	Sr	Sr	Sr	Sr	?	St	Sr/St	Sr	Sr
Cost-impact	W	W	W	W	W	W	?	W	W	W	W
Temporal adaptation	Rt	Rt	Pt/Rt	Rt	Rt	Rt	?	Rt	Rt	Rt	Rt
Temporal monitoring	Ad	Co	Co	Co	?	Co	?	Co	Co	Co	Co
Human intervention	N	N	P	N	N	N	?	P	?	?	P

Table 3.17: Modeling dimension of the selected methods.
(Sta=Static, Dy=Dynamic, Rgd=Rigid, Cns=Constrained, Ncn=Unconstrained, Tmp=Temporary, Prs=Persistent, Mlti= Multiple, S=Single, E=External, I=Internal, NFR=Non-functional requirement, Gt=guaranteed, NGt=Not guaranteed, ?=Unknown)

Adaptation terms	Ingram et al. [88]	Ahmad et al. [3]	Morandini et al. [116]	Qian et al. [138]	Ramnath et al. [141]	Meacham [113]	Ribeiro et al. [142]	Baresi et al. [31]	Lopes et al. [106]	Soyler and Sala-Diakanda [149]	Baresi and Pasquale [32]
Goal											
Evolution	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Dy
Flexibility	Rgd	Cns	Cns	Ncn	Rgd	Rgd	Rgd	Rgd	Rgd	Rgd	Cns
Duration	Prs	Prs	Prs	Tmp	Prs	Prs	Prs	Prs	Prs	Prs	Prs
Multiplicity	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	S	Mlti	Mlti	Mlti
Dependency	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	S	Mlti	Mlti	Mlti
Change											
Source	E	E	E& I	I	E	E	?	I	E& I	E	E&I
Type	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	-	NFR
Mechanisms											
Timeliness	Gt	None	Gt	NGt	Gt	Gt	?	Gt	Gt	?	NGt

- All of the collected methods support their adaptation approach at design time, and they do not enhance or change them at runtime using a (machine) learning technique.
- According to their effected layers and artifacts, they support only a weak adaptation.
- The closed approaches affect the goals flexibility feature negatively and consequently lead to different ways of managing adaptivity via fixed goals or flexible goals with constraints, as shown in Table 3.17.
- Most methods that used goals as a model manage flexible goals with constraints because of the conditions that were used to trigger system plans and strategies during the strategy selection process (closed approach and design time adaptation).
- The collected methods do not support unconstrained goals except for Qian et al. [138], which uses two methods to generate solutions: 1) goal-reasoning to generate a new solution when the current cases did not match the conditions of the stored cases, and 2) using the average of the similar cases to generate new solutions. However, the new solution could be unsuitable for the current problem and consequently lead to non-guaranteed adaptation timeliness, as shown in Table 3.17.
- Baresi and Pasquale [32] presented the only method that changes the number of system goals during adaptation by adding and deleting goals. As a consequence, the time needed for adaptation is not guaranteed even if the conditions and related plans are already known and embedded in the system at design time.
- All but one method (Ahmad et al. [3]) included work on mechanisms.

To conclude, using goal models in adaptation methods strengthens their flexibility and ability to deal with unknown conditions at runtime. However, this can also lead to the generation of infeasible solutions or unguaranteed adaptation timeliness due to insufficient validity checking of the generated solutions and the new alternatives.

The surveyed methods handled self-adaptation from several perspectives: early management of uncertainty, the use of different languages to deal with adaptation, frameworks for developing self-adaptive systems, adaptation strategies, decision-making and strategy selection processes, and self-explanation approaches, which are all discussed next.

Early Management of Uncertainty Reducing or eliminating uncertainty before having to manage it is one way to analyze and design self-adaptive systems. To support the decision-making process in analysis and design phases, early in the requirement engineering process, Horkoff et al. [77] presented a formal iterative goal analysis process with a tool that integrated i^* goal models with the MAVO framework [144] to remove unnecessary requirements alternatives.

Language Usage Self-adaptive systems offer an opportunity for more relaxed language to be used to better specify their requirements, because common patterns such as "the system shall do this" are not sufficient in that context. As requirements may change at runtime, "the system may do this or that" could be a substitute specification. This need was addressed in different works [3, 109, 145, 52]. Ahmad et al. [3] also used the RELAX language [158] as a more formal representation of this idea for monitoring environmental conditions and detecting violations. In addition, the formal language called FLAGS [32, 33] formalizes the KAOS goal language through LTL. In order to represent fuzzy goals with uncertainty, LTL is accompanied by fuzzy temporal operators based on RELAX [32]. This language was used to keep tracking and using the goal model from requirements elicitation to the implementation phase.

Frameworks for Designing Self-Adaptive Systems Several approaches and frameworks were presented to design and select an appropriate architecture for self-adaptive systems (SAS). To design adaptable systems, Morandini et al. [116] extended the Tropos framework [43] for Adaptive Systems (Tropos4AS). This framework helps analyzing requirements of SAS from early requirements to the implementation by mapping the goal model of particular actors to architecture agents and by mapping the plan (tasks) to activity diagrams. This framework uses goal, failure, and environmental models. The Tropos goal modeling language, itself based on *i**, was extended to add goal types (achieve, maintain, perform), relationships (sequence, inhibition) and conditions. With the support of four tools, code is generated automatically from the models by mapping Tropos4AS terms to Belief-Desire-Intention (BDI) agents, which enable SAS validation and verification via simulation.

Security strategies can also affect user privacy and cost. For this reason, Ramnath et al. [141] proposed a non-functional framework to deal with adaptive security analysis. The goal model is linked from and to dynamic behaviour of the organization via a transaction-based mechanism. Such goal model is used to support trade-off analysis between cost and privacy in order to help with the definition of a secure architecture.

To reduce the complexity of SAS and manage traceability between their components, Soyler and Sala-Diakanda [149] presented a model-based framework exploiting SysML. This framework was selected to capture a Disaster Management System in one single environment using feedback to adapt the embedded strategies, plans, and policies.

Finally, similar to the framework of Morandini et al. [116], Akbas and Karwowski [8] proposed an agent-based framework that uses a hybrid simulation model to support system design, validation, and verification, as well as to provide quick feedback about the chosen design.

Adaptation Strategies The collected methods dealt with possible adaptation strategies or configurations through open and closed adaptation.

- **Closed adaptation approaches:** In a closed approach, possible alternatives, strategies, and configurations are embedded in the system during the development phase.

Assuming environmental conditions and changes are well-known at design time, the closed methods (Table 3.16) manage uncertainty through rigid or constrained goals. From the Goal/SysML integration methods, Ingram et al. [88] used fault tolerance analysis and rules to deal with errors. Without considering goal models, Ribeiro et al. [142], Baresi et al. [31], and Soyler and Sala-Diakanda [149] triggered their embedded strategies and plans to respond to internal or environmental changes. Similarly, Morandini et al. [115, 116] represented the goal model in an agent structure while embedding the environmental and failure conditions, alternatives, and plans in agent beliefs and system design. Designing self-adaptive systems with predicted or predictable change management is a common characteristic of these types of methods. One issue here is that they cannot deal with unpredictable changes that could emerge at runtime. On the other hand, they guarantee that the selected adaptation strategy is suitable and timely for the predefined changes (see Tables 3.16 and 3.17).

- **Open adaptation approaches:** Open approaches do not solely rely on predetermined adaption strategies and conditions. Feedback can be used to update the embedded strategies, as suggested by Soyler and Sala-Diakanda [149] (although they give little explanation on how to do so). Case-based reasoning is an approach that uses previously stored solutions in solving current similar problems. To deal with unexpected environmental changes, case-based reasoning can be employed to update embedded configurations and strategies. Based on such feedback loop, Qian et al. [138] create new solutions or configurations from the average of the parameters' values of two or more stored cases or from goal reasoning (such as label propagation algorithms [67]). In contrast, Meacham [113] used case-based reasoning to manage fall cases of elderly people and infer their patterns in order to determine the related system reactions. She used stored cases and patterns only while the feedback technique was not applied, in order to continue enhancing the stored cases, as Qian et al. [138] did.

Decision-Making and Strategy Selection Processes The collected methods have not provided much diversity in the decision-making process that triggers the adaptation and the selection of the most suitable strategy (see Figure 1.1) . All the decision processes were encoded inside the system (i.e., static decision-making) and no adaptation was performed on these processes using learning techniques. However, the decision-making process can use different policies: action (static decision), goal and utility (dynamic decision), as well as hybrid policies [93, 98]. The collected methods realized their decision-making processes as follows.

- **Action policies.** These policies apply to the process that decides when the adaptation should be done and what the system should do based on the current state, conditions, and actions (if-then logic) [93]. Meacham [113] used a pattern analysis algorithm to trigger the adaptation while Morandini et al. [116] used a goal modeling approach and several types of conditions that trigger the adaptation process. In order to select a suitable recovery strategy, Baresi et al. [31] used a diagnosis model

that compared the current system state with the pre- and post-conditions of each operation state. The if-condition-then-plan technique is used here because it is a simple way for humans to express a rational logic in the systems. However, action policies become complex in real-world conditions, and additional techniques (e.g., prioritization) are needed to solve policy conflicts [93, 117].

- **Utility and goal policies.** In order to select the optimal adaptation strategy, experts are needed to identify the control variables and state space required by the utility policy approach. This approach has been used in the decision-making process providing a flexible way to trigger the optimal adaptation's strategy by exploiting designer knowledge at design time and real data at runtime [93]. Ramnath et al. [141] used utility functions for cost and benefit of the involved stakeholders and trade-off analysis to select a suitable design. From there, related strategies were connected to the security layers of this architecture to be executed at runtime. Similarly, Spyropoulos and Baras [150] used trade-off analysis to get the optimal solution for power allocation in their Microgrid system. In their approach, Lopes et al. [106] added an enterprise service management plan using utility functions to select the best strategy based on the techno-economical costs/benefits and trade-off analysis at design time. However, none of the previous studies used these policies at *runtime* with the real data. Baresi and Pasquale [32] used satisfaction equations and goal reasoning in analyzing system state. However, they were not used in their strategy selection process triggering the possible solutions depending on several conditions attached to the system operations as rules. Similarly, Qian et al. [138] used case-based reasoning in all MAPE activities (see Figure 1.1) except in the planning process, the latter being supported by goal-based reasoning when it failed. However, they applied goal-based reasoning in generating new configurations only by increasing the weights of the violated goals to get solutions related to these specific goals, but the new solutions could still be unsuitable for the current problem. Hence, although using such a utility function leads to an optimal solution without strategy conflicts, its usability is affected significantly and experts are required [93, 117].

Self-Explanation. A self-adaptive system may exhibit unexpected behaviour in a well-known situation or a well-known behaviour in an unexpected situation. To increase trust and understanding of system behaviour, goal models and claim refinement models were integrated into the method of Welsh et al. [157] to provide an explanation about system adaptation's choices due to predictable and predicted changes. The resulting domain-specific language was used to transfer the reason of the adaptation from goal alternatives embedded in the system to natural language that human operators can understand.

3.2.4 Traceability and Integration Assessment Criteria

We compare the presented methods against the criteria proposed by Amyot et al. [14] for the integrated goal-SysML modeling of SCPSs. Two new adaptation criteria are also added to the initial set, and the methods that did not provide enough information about their

adaptation approaches are eliminated from this comparison. Table 3.18 provides the result of this assessment against the following ten criteria:

- AC1: Elicitation of and general communication about goals and/or SysML
- AC2: Consistency and completeness analysis involving goals and/or SysML models
- AC3: Traceability management and analysis
- AC4: Scalability, for handling large models
- AC5: Change management supporting goals/SysML and textual requirements
- AC6: Trade-off analysis and possible solutions
- AC7: Concurrent modeling supporting multiple modelers, with avoidance of duplication and inconsistencies
- AC8: Usability of integration, through the minimization of the number of tools used and of necessary imports/exports
- AC9: Goal-based reasoning at runtime and adaptation activities support
- AC10: Adaptation

The best methods according to these criteria are the ones that map or link goal models with (SysML) system design. For example, Ahmad et al. [3] provided such mapping using profiles. By using goal models only, Morandini et al. [115] and Ramnath et al. [141] proposed the best approaches where the goal model is transferred and mapped semi-automatically to system architecture and implementation (without SysML artifacts). However, the studies have faced several challenges while employing their methods, and they are explored below. Table 3.18 and this assessment suggest that there is still room for much improvement here.

Usability of Integration The integration processes are characterized by the development of new tools in addition to two or more existing tools, which impact the usability of these processes negatively. Moreover, remodeling goals with design tools (duplication of work) not only causes risks of information loss and inconsistencies, but also consumes much development effort and time.

Goal-based Reasoning at Runtime The use of goal-based reasoning at runtime differed from one study to another, and it was affected negatively by several factors: 1) transferring only part of a goal model to the design and/or runtime phases (e.g., not transferring contribution weights, high-level goals, or contribution relationships between goals) and 2) handling the reasoning process in several ways (i.e., considering softgoals and tasks only, or violated softgoals only). As a result, the methods' ability to use goal reasoning at runtime for selecting the best (or even just one) suitable solution during the analysis

Table 3.18: Comparison of adaptation methods against the integration criteria from Amyot et al. [14]. (Y=Yes, N=No, P=partially, –=No integration, W=Weak)

Study	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	AC10
Goals and SysML										
Cui and Paige [51]	P	P	P	N	P	N	N	N	N	N
Tsadimas et al. [152]	N	P	P	N	P	N	N	N	N	N
Apvrille and Roudier [26]	N	P	P	N	P	N	N	N	N	N
Gnaho et al. [69]	P	P	P	N	P	N	N	N	N	N
Spyropoulos and Baras [150]	P	P	P	N	P	N	N	N	N	W
Ingram et al. [88]	N	P	P	N	P	N	N	N	N	W
Ahmad et al. [3]	P	P	P	N	P	N	N	N	N	W
Fan et al. [60]	P	P	P	N	P	N	N	N	N	N
Goals only										
Morandini et al. [115, 116]	N	Y	Y	N	Y	N	N	–	N	W
Ramnath et al. [141]	N	P	P	N	P	N	N	–	N	W
Baresi and Pasquale [32]	N	N	N	N	N	N	N	–	P	W
Qian et al. [138]	N	N	N	N	N	Y	N	–	P	W
SysML only										
Soyler and Sala-Diakanda [149]	N	P	P	N	P	N	N	–	N	W
Lopes et al. [106]	N	N	P	N	P	N	N	–	N	W
Ribeiro et al. [142]	N	P	P	N	P	N	N	–	N	W
Baresi et al. [31]	N	N	N	N	N	N	N	–	N	W
Meacham [113]	N	N	N	N	N	N	N	–	N	W

and strategy selection processes was limited, and unsuitable solutions could be generated along the way. Furthermore, goal analysis and trade-off analysis cannot be done at runtime accurately using those methods, which consequently limits the ability to self-adapt in the developed systems by using conditions and implementing inflexible methods that cannot deal with unpredictable changes.

Goal Models and MAPE Activities The MAPE activities (monitor, analyze, plan the adaptation, and execute it, see Figure 1.1) are not all supported at the same level by the collected methods. Managing and changing system goals at runtime is one suggested solution for conducting trade-off analysis and selecting the best adaptation strategy using real-time variables. From the adaptation assessment results (Section 3.2.3), Ahmad et al. [3] translated variation requirements using RELAX to conditions during the monitoring only. Qian et al. [138] used goal model reasoning only when their method failed in finding a solution during the planning. Morandini et al. [115, 116] used goals as agents with conditions to trigger the first successful adaptation plan. On the other hand, Baresi and Pasquale [32] employed goal-based reasoning in the monitoring and analysis phases, but then switched conditions attached to system operations in the planning phase. Although some of the articles translated goal models to parts of system implementations, they did not employ goal-based reasoning in all of the MAPE activities.

Unmanageable Traceability It is difficult to manage traceability between models with the current lack of model synchronization, concurrent modeling, model import/export, and low scalability and usability. Consistency and completeness analysis as well as change management processes were degraded in the all methods due to unmanageable traceability that resulted from 1) the lack of relationships between the textual requirements, stakeholder goals, and system design artifacts, 2) the lack of synchronization during model evolution, and 3) low scalability, which is a salient feature of the current SysML tools and requirement diagrams [14].

Overall, although establishing links between goals/requirements, system design, and the implementation was an objective of most selected methods, none of them actually offered sufficient and practical traceability support. One reason is that each of these artifacts has a specific environment and tools that deal with its needs. Representing a goal model using another tool with a different purpose (such as SysML/design tools) was a major obstacle faced by most methods. Furthermore, trade-off analysis as well as runtime adaptation selection are other affected features within all these methods because the goal model is not mapped completely and used effectively in the MAPE activities.

High-level goals are usually more stable than low-level ones, and they help guide the evolution of requirements from elicitation to runtime adaptation [5, 38]. However, to truly unlock the benefits of goal-orientation (including consistency/completeness, conflict, trade-off, and impact analyses), system design components should be linked to goals at all levels [41, 3, 38].

3.3 First Review: Limitations and Threats to Validity

The validity of any study focuses on the degree of correctness of its conclusions [61], including threats related to bias and over-generalization. We applied some strategies to mitigate common threats to validity, but several remain, as discussed below.

3.3.1 Internal Validity

Only one researcher (the thesis author, with supervision and informal checks from her supervisor) selected and reviewed the papers, and extracted the data, which increases the risk of bias. To mitigate this threat, we consulted experts, including the authors of some of the selected papers, to increase the level of confidence about their contributions. However, the authors have not been rigorously surveyed, and hence there is a remaining risk that some of their contributions were classified or assessed incorrectly.

There is also a risk that important and relevant papers have been missed or incorrectly excluded in this literature review. To mitigate this risk, we used different and recognized scientific databases in the areas of systems modeling, with fairly permissive queries (refined over many iterations based on previous results). We also used Google Scholar with different queries and choices to increase our confidence that relevant studies from different sources were included. Precise inclusion and exclusion criteria were defined and used, and the supervisor was involved in the selection in cases where the thesis author was unsure about relevance. Note however that given the date at which the queries were performed, the papers from the last year were not included

We tried to be exhaustive for papers combining goal and SysML modeling, as well as for papers about SysML for self-adaptation. However, due to a lack of articles using both goal and SysML modeling for self-adaptive systems, we manually added primary articles (proposed by experts based on citations and reputation, as there were too many such papers) that support adaptation using goal models (Figure 3.2). One threat here is that many papers related to goals and self-adaptation have not been considered. Yet, the sample we have selected was useful to understand what is being done outside the SysML world.

3.3.2 External Validity

This type of validity is related to the generalization of the results outside of the study's scope [61]. The number of studies that focus on the integration of goal models with SysML models is very small. If we consider the method granularity, only 13 methods were presented and three of them were for adaptation. What is published in peer-reviewed venues also may not be representative of what practitioners actually use in industry. Generalizing the results of these methods is a threat due to the relative immaturity of the field. However, we tried to mitigate this threat by systematically including papers on SysML for self-adaptive systems, and manually including primary papers on goal models for self-adaptation.

3.4 First Review: Conclusion

The number, complexity, and importance of socio-cyber-physical systems (SCPSs), which consider the goals of their stakeholders at design time and runtime, is increasing in our societies [121]. In some SCPSs, the need for adaptability driven by stakeholder goals was partially addressed in the peer-reviewed scientific literature. This first part of the chapter reviewed 42 publications and assessed methods that integrate goal models with SysML models (or use them separately) to support runtime self-adaption, with a consideration for the SCPS context. The review answers many questions of broad interest both to researchers and to practitioners who are considering the use of goal models, SysML models, or both in SCPSs or self-adaptive systems contexts. The research questions were answered through this review as follows:

RQ1 *What are the existing methods that integrate goal-oriented models with SysML models?* This was answered by Table 3.6 where a total of 13 methods, named M1 to M13, are presented amongst 26 studies.

- SQ1.1 *Why have these integrations been proposed?* The objective of each study was presented in Table 3.7 where the common objectives are system architecture selection and modeling, uncertainty and adaptation, as well as traceability and formal validation and verification, in that order.
- SQ1.2 How do the methods integrate the two types of models? The answer was provided in Table 3.8 and its explanation in Section 3.2.1, which concluded that mapping part of goal models to SysML requirement diagram through profiles is the most often used approach through all 13 methods.

RQ2 *What are the collected methods that support self-adaptation?* By classifying the collected methods using NFRs, self-adaptation properties (Table 3.10), and adaptation phase (Table 3.11), methods that support self-adaptation are listed and described in Tables 3.13 and 3.14.

- SQ2.1 *How do the methods support self-adaptive systems?* This question was answered by Tables 3.16 and 3.17, which identify terms inspired from the adaptation taxonomies and modeling dimensions of self-adaptation. The discussion around these tables (Section 3.2.3) provides insight into how the assessed methods support the activities of self-adaptive systems.
- SQ2.2 *What are the roles that each model plays in this adaptation support?* This was answered by exploring the reasons for using each model in each integration in Section 3.2.2, and by discussing the adaptation assessment criteria in Section 3.2.3.

Although there was much improvement in the last decade, the main results show that complete and practical traceability has not yet been achieved by the collected methods due to a lack of clearly documented links between stakeholder goals, system goals, system requirements, system design, and system implementation artifacts. Mapping goals at

design time is common among the collected methods to support traceability, architecture selection, system validation and verification, as well as self-adaptation. However, existing mappings usually suffer from a loss of important information (e.g., contribution links and weights) that plays a key role in runtime goal analysis and flexible self-adaptation. Goal modeling is actually used sparsely and differently in MAPE activities of adaptive systems. Thus, in addition to consuming time and efforts, the proposed methods were unable to implement goal-based reasoning in all activities. This consequently leads to situations were incorrect adaptation solutions are produced and used, and in time constraints that cannot be guaranteed.

This review hence highlights the challenges current methods face in offering comprehensive traceability and self-adaptation solutions. These challenges indicate the research areas that need the attention of the requirements engineering and systems engineering research/practitioner communities. In particular, we identify important questions refining this thesis' first research question (Section 1.3):

1. How can we produce manageable and usable traceability between goals, requirements, and system elements related to self-adaptation without information loss, without duplicating work (e.g., by re-creating goal models in SysML tools), and with high scalability?

2. How can we reason about goal models accurately and efficiently, both at design time and at runtime?

3. What factors affect the validity of solutions that are generated by goal models in adaptive SCPSs?

The above questions can be addressed in many ways. However, this thesis will tackle them in two ways. Firstly, it aims to integrate textual requirements, GRL models, and SysML models using a third-party requirements management system as a repository, for instance using the approach of Rahman and Amyot [139]. This will help support concurrent modeling in a scalable and manageable way. Secondly, we will formalize GRL goal models and feature models (to restrict combinations of features and alternatives to valid ones only) as arithmetic functions. These functions will be used: a) at design time in a simulation context (e.g., within SysML models) to select suitable designs considering stakeholder goals and b) at runtime to support self-adaptation through goal-oriented MAPE activities. Existing work related to such functions is explored in the next section.

In supporting industrial needs of practitioners, these steps will facilitate and strengthen the integration between the requirements engineering phase and the design, help take advantage of existing tools, and produce executable models for design time and runtime usage.

3.5 Second Review: Quantitative Goal-Based Analysis in System Design and Implementation

To support the design selection process through simulation and runtime adaptation, some existing methods have attempted to transform goal models and/or feature models into mathematical functions. An informal search complemented with suggestions from reviewers of one of my publications [20] was used to get the primary studies related to this area, relevant to the second research question of this thesis (Section 1.3). This section assesses these methods using the following criteria:

C1 Goal-based analysis: Is the reasoning based on the satisfaction level of a goal model?

C2 Quantitative: Does the goal-based analysis use numerical values for its arguments?

C3 Goal model: Are all the elements of the goal model involved in the analysis?

C4 Feature model: Is a feature model involved in the analysis?

C5 Quantitative feature model: Is the involved feature model represented numerically?

C6 Trade-off analysis: Are trade-off analysis and alternative selection part of the proposed reasoning?

C7 Reasoning phase: What is the targeted phase (design time or runtime) of the proposed reasoning?

The next two subsections summarize the existing methods investigated and assess them against the above criteria.

3.5.1 Methods for Quantitative Goal-Based Reasoning

Quantitative goal models are usually constructed iteratively with stakeholders, or based on evidence. Different consensus mechanisms can be used to find contribution weights based on Delphi, the Analytical Hierarchy Process, and others [104, 10], with various additional validation steps, e.g., based on stakeholder-specific questionnaires [73] or simulations with stakeholders. GRL offers additional mechanisms such as contribution overrides and value ranges, which can be used to produce model variants in case consensus cannot be reached [90, 18].

Goal-based reasoning was used at *runtime* to analyze environmental conditions and to decide whether a system should adapt or not. Ramirez and Cheng [140] proposed the *Athena* method, which uses the KAOS [154] and RELAX [158] goal languages to monitor environmental conditions and determine whether requirements are violated or not. In order to monitor the surrounding environment, Athena generates fuzzy functions automatically for softgoals while using templates that return Boolean values for functional goals. The values of the generated functions are propagated to calculate the overall satisfaction level

and determine whether the goals are violated. However, the method does not cover contribution relationships, contribution weights, and the relative importance of each goal in the analysis process. Moreover, the proposed method does not support trade-off analysis or selection between alternatives to provide the most-suited adaptations. In addition, limiting the satisfaction of goals to Boolean values is at times perceived as too restrictive for run-time adaptation [47].

Similarly, in order to support adaptive and complex systems in their evolution while running, Chatzikonstantinou and Kontogiannis [47] presented a framework that is based on conditional goal models to express extra information about system goals and their dependencies as well as to automate the reasoning process. They propose the ReqRV method, which is an adapted requirement-based view used to verify systems while running. In order to decide whether the requirements are fulfilled or not, conditions that include domain assumptions as well as current satisfaction values are assigned to each requirement. A fuzzy approach is used to monitor the environment and determine if the related goals are violated or not using quantitative weights. Although the proposed method can monitor its environment and verify its requirements, it is unable to deal with unpredictable and unknown situations at runtime because of the predetermined conditions. The method does not support trade-off analysis or selection between alternatives to provide the most-suited adaptations. Moreover, assigning the conditions to each goal depending on the related contexts is not simple especially when dealing with large and complex models.

At *design time*, Nguyen et al. [125, 123] proposed an automatic reasoning through the *Constrained Goal Model (CGM)* approach. Conventional goal models are extended with preconditions and numerical variables (e.g., cost and performance) to create multi-objective functions that generate optimal system configurations. The reasoning process however does not exploit all types goal model elements and relationships found in standard GRL, and the objective functions are generated manually. The preconditions can ensure some level of validity in the selection of configurations, but at a cost of mixing the concerns of goal modeling and product configuration. The absence of generated functions makes difficult the use of such non-standard models in design simulations (e.g., in SysML) or running implementations.

Because of the difficulty of conducting goal-based reasoning on a large goal model to select a suitable solution at design time, Chitra et al. [50, 49] proposed a multi-objective optimization based on the satisfaction levels of the goal model. They identify the quantitative weights of the leaf softgoals automatically without the analyst's intervention. Next, these weights are propagated to calculate the satisfaction level of the top softgoal. The satisfaction level of each softgoal is used as argument of the main multi-objective function to select the solution that maximizes the satisfaction level of the softgoals. The score and the weight of the leaf softgoals are the constraints of this function. However, this function cannot solve goal models whose alternatives related by "And" relationships. Moreover, functional goals, relationships between goals, indicators and importance values of model elements are not involved in the analysis.

Goal models were also translated to mathematical functions usable in *fields other than self-adaption*. In order to support software product lines (SPL) and product reuse, Noo-

rian et al. [127] combined goals, softgoals, and features in building feature models and selecting a product using an optimization model. Conventional feature models determine valid configurations while the goals determine which valid configurations maximize softgoal satisfaction. The required goals and softgoals of the new product are selected by a user and an objective function is then built by the *summation* of the impact of each feature on the selected softgoals and goals, separately. Three types of constraints (features, goals, and their integration) are used as rules with the proposed utility function to eliminate invalid configurations. However, only part of the goal model is involved in the optimization model and the utility function does not represent goal-based reasoning in which the softgoals, the goals, and their relationships are involved.

Asadi et al. [28] combine feature models (specifying all potential configurations) and *i** goal models (for prioritizing configurations based on intentions) by explicitly annotating feature models with Boolean Presence Conditions that represent corresponding goals/tasks, whereas our approach simply reuses tasks in the goal model as features in the feature model. Asadi et al.'s approach is however limited to qualitative contributions and satisfaction levels, and does not provide a mechanism to collect runtime information (e.g., KPIs).

As assessed by Olaechea et al. [129], many approaches add *quality annotations* to features in feature models, or to grammars [94], to indicate their contributions to desirable goals or qualities, hence bypassing the need for a separate goal model for optimizing valid configurations. These annotations can take various forms, from binary indicators [162], to simple/multiple quantitative ($Cost = 25$) or qualitative ($Security = Good$) assignments. Different multi-objective optimization techniques exploit these models, from metaheuristic searches to genetic algorithms and other evolutionary techniques, with various objective functions such as weighted sums and Pareto-optimality, and observed performance and scalability trade-offs. Many of these approaches target design-time product configurations, without consideration for environmental conditions from monitored KPIs. Others build on such work to support adaptation (e.g., [147]), but generally with objective functions that are manually written.

3.5.2 Reasoning Methods Assessment

Table 3.19 summarizes the assessment of the reasoning and analysis capabilities of the six methods related to goal modeling presented in Section 3.5.1 using the comparison criteria.

As seen in table 3.19, although goal and/or feature models have been part of proposed transformations into mathematical functions to deal effectively with the complexity and scalability of large models, their reasoning processes have often been: 1) incomplete (i.e., using only part of the source model), 2) imprecise (i.e., by using qualitative values, or by using numerical and Boolean values for softgoals and goals, respectively, and not including dependencies among the models' elements), and 3) not supporting goal-based analysis (e.g., using conditions and utility functions). To support software product lines and product reuse, system simulation (selecting a suitable system architecture) at design time, and self-adaptation at runtime, transforming a complete and quantitative goal model, ideally supporting monitoring and restricted to valid feature configurations, into mathematical

Table 3.19: Assessment of the collected quantitative reasoning methods.

Method	C1	C2	C3	C4	C5	C6	C7
Ramirez and Cheng [140]	Yes	Partially	Partially	No	No	No	Runtime
Chatzikonstantinou and Kontogiannis [47]	Partially	Partially	Partially	No	No	No	Runtime
Nguyen et al. [125]	No	Partially	Partially	No	No	Yes	Design
Chitra et al. [50, 49]	Yes	Yes	Partially	No	No	Yes	Design
Noorian et al. [127]	No	Yes	Partially	Yes	No	Yes	Design
Asadi et al. [28]	Partially	No	No	Yes	No	Yes	Design

functions is needed. Such transformation promises effective, efficient, and precise reasoning for optimization, simulation tools, and implementations, especially in an adaptation context.

This thesis proposes a transformation of complete GRL models, combined to feature models, into simplified mathematical functions in numerous programming languages.

3.6 Chapter Summary

This chapter first reviewed and assessed methods involving SysML models, goal models, or their integration for supporting adaptive systems, especially in a SCPS context. A second review assessed current methods for generating functions from goal and feature models for adaptive systems. The results of these assessments identified two major issues: 1) existing integration solutions show severe limitations in terms of traceability, usability, and scalability, and 2) support for goal-based reasoning (both at design time and runtime) is incomplete, imprecise, and not used in all relevant activities (monitoring, analysis and planning) of an adaptation process. The identified issues and gaps further motivate the research questions and the approach proposed in this thesis.

The following chapters will propose a new approach with its contributions addressing some of the above issues.

Chapter 4

Combining Goals and SysML for Traceability and Decision-Making in the Development of Adaptive SCPSs (CGS4Adaptation)

This chapter presents a novel approach, called CGS4Adaptation, with a high-level description. Two important parts of CGS4Adaptation are introduced: the traceability management of goal and SysML models, as well as the automatic transformation of goal and feature models to mathematical functions usable in design, simulation, adaptation, and implementation contexts.

4.1 Overview of CGS4Adaptation

The successful development of SCPSs requires engineers to strengthen the link between stakeholder goals and system artifacts at design time and runtime while minimizing development effort and information loss. This was the inspiration for this thesis, which focuses on improving requirement engineering activities by injecting social concerns into the SCPS design, simulation, and implementation activities, as shown in Fig. 4.1. The CGS4Adaptation approach consists of two main parts: traceability management between goal model and design elements, as well as the generation of mathematical functions from goal and feature models, usable for system optimization, simulation, and implementation, including adaptation activities.

- Traceability management: We aim to minimize the hard-to-manage traceability issues that were mentioned in Chapter 1 by integrating GRL and SysML models using a third-party traceability system (e.g., a requirements management system such as IBM DOORS), through the use of MI-DSL transformations [139]. The elements of the GRL model and of the SysML model, together with system requirements, will be stored in the RMS repository, as shown in the left part of Fig. 4.1.

- Mathematical functions generation and integration: In order to support self-adaptation, as shown on the right side of Fig. 4.1, the goal and feature models will be transformed to arithmetic functions for effective and efficient reasoning in SysML models via simulation, and in implementations for runtime adaptation. The feature model will be used to constrain design and adaptation alternatives in the goal model to configurations that are globally valid. The feature model will hence influence the evaluation of the arithmetic functions. With a particular focus on SCPSs, goal-based functions will be transferred and employed in numerous places (optimization model, simulation at design time, as well as MAPE's Monitor, Analyze, and Plan activities at runtime) to strengthen the effectiveness and benefits of goal-based reasoning in supporting the development of self-adaptive systems.

This leads to three main areas of interest: traceability, design time adaptation through optimization, and simulation and runtime adaptation.

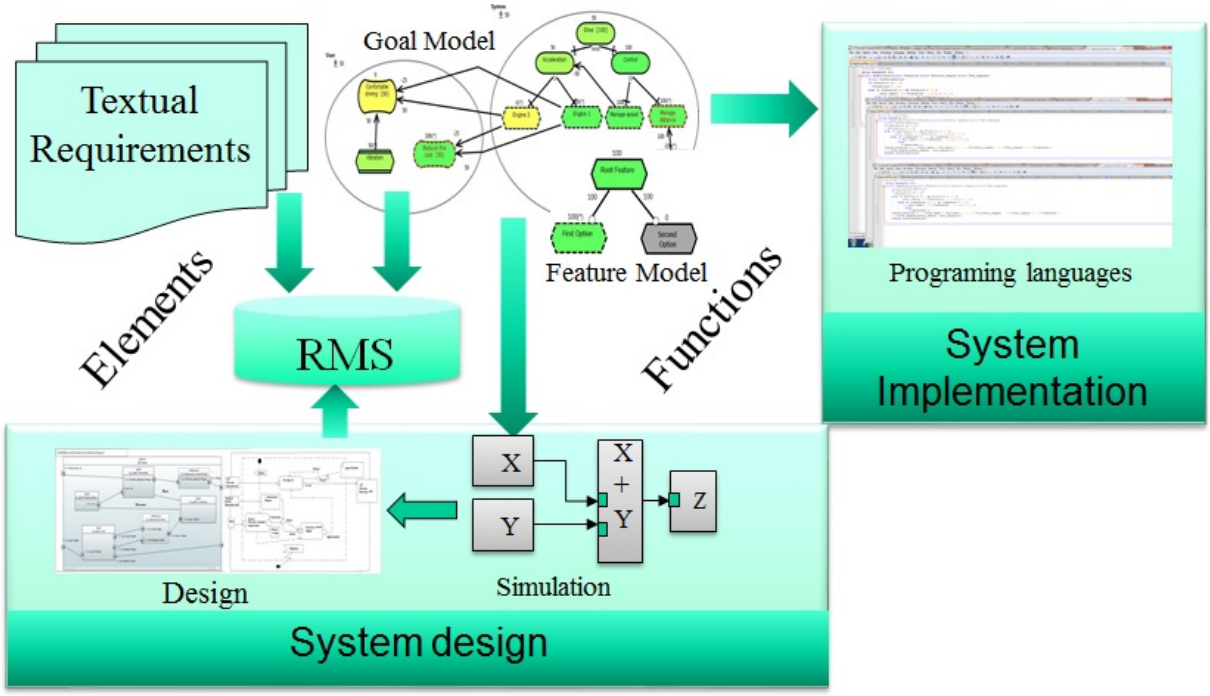

Figure 4.1: Approach overview

The input of our approach is composed of SysML, GRL, and feature models, as well as connections between them. Note that common approaches for the construction and validation of such models can be used [10, 73, 105, 122] and are hence not the focus on this thesis. The overall CGS4Adaptation approach is described in Fig. 4.2, where the involved RE activities are: 1) traceability management between goal model elements and SysML model elements (including requirements), 2) optimization and simulation supporting GRL and feature models as mathematical functions, and 3) system code implementation injected by GRL and feature models as mathematical functions. The description of the modeling process is given in the following sections.

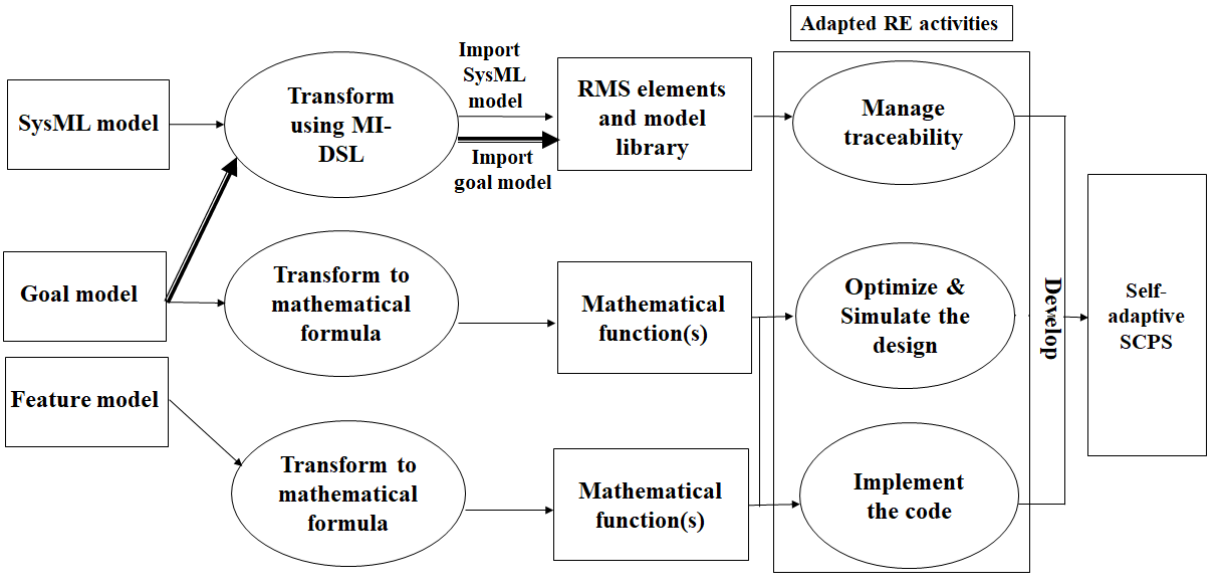

Figure 4.2: CGS4Adaptation approach integrating SysML, GRL, and feature models for adaptive SCPS development.

4.2 Managing Traceability at Design Time

A SCPS development process particularly needs comprehensive traceability that connects requirements, stakeholders' goals, and system design and implementation [157]. This approach supports system developers in managing the problem space as well as enabling the validation and verification of the selected architecture. This kind of traceability is needed for SCPS development just like it is also required in software product lines (SPL) and system reuse [135]. Based on the result of Amyot et al. [14], which explored the efficiency of means to manage traceability between goal and SysML models and based on the results of the comparison of existing approaches integrating goal and SysML models for self-adaptation and other reasons (previous chapter), our approach integrates SysML models with GRL models through a third-party repository (i.e., a commercial RMS, namely DOORS to begin with). As this method promotes the synchronization of models and their consistency, and as the RMSs are mature products used in many companies that model with SysML, this approach should also minimize adoption and integration efforts. Also, traceability links can be managed easily via these systems to support consistency and completeness checks, change management, and impact analysis.

4.2.1 Traceability Links

In order to support change management, derivation analysis, and impact analysis, consistency between the involved models needs to be checked [36]. Thus, traceability links are required between the imported models to assess completeness and consistency as well as to detect violations, especially after modifying or deleting linked elements. Such activities

can be automated using RMSs. Goal models provide rationales behind requirements and system design, but they also provide possible design alternatives useful for systems with socio-technical concerns [36]. In this context, the initial set of possible traceability links considered in this work are:

1. *Trace* links between goal model elements (intentional elements in a GRL model) and requirements.
2. *Satisfy* links between design elements (blocks in a BDD) and goal model elements (intentional elements in a GRL model).
3. *Satisfy* links between design elements (blocks in a BDD) and requirements (these links already exist in SysML).

These links can be established in both directions, as shown in Fig. 4.3. However, not all elements need to be traced. Developers can select the important elements based on their organizational roles and the nature of the developed project itself [36]. Exploring the benefits of these links and the potential need for other types of links cal help answer part of our first research question (How should the integration between goal and SysML models be done in early stages to support effective SCPS requirements definition, modeling, and analysis?).

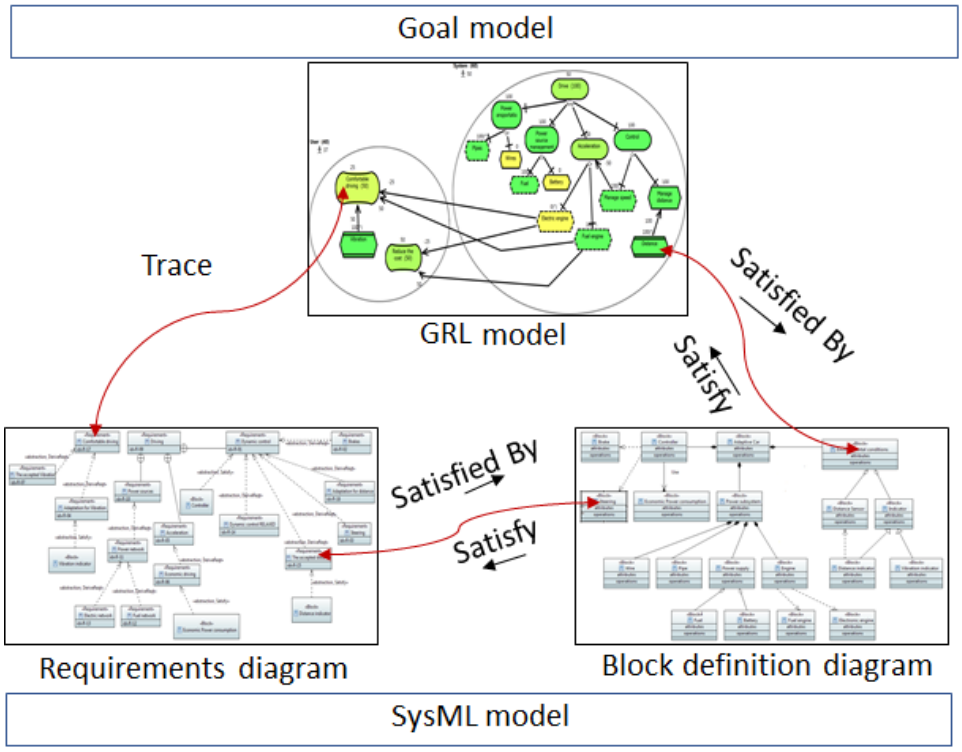

Figure 4.3: Overview of traceability in CGS4Adaptation

4.2.2 Imported Elements from SysML models

Supporting SysML models in RMSs requires importing relevant model elements only, to avoid degrading the performance of importing, tracing, and analyzing the models in these systems. Hence, we initially select the core elements of SysML diagrams that represent the main elements of goal models conceptually in the design space, and we keep the internal relationships between the imported elements themselves to support implicit traceability [36]. The selected elements are from SysML requirement diagrams and block definition diagrams (BDD). Table 4.1 shows these elements, the rationale for choosing them, the expected related elements in goal models, and related attributes.

4.2.3 Setting a Traceability Information Model and Consistency Rules

To facilitate tracing the links between models, we use a *Traceability Information Model* – TIM [70] (sometimes referred to as a link schema) between the involved models (Fig. 4.3). This TIM/schema (Fig. 4.4) describes the direction and type of each link connecting two objects in different modules. From this TIM, the traceability between the modules are established by exploiting two types of links: 1) links that are imported from models themselves and represented by red/bold arrows, and 2) links that can be added by the developers, represented by blue arrows.

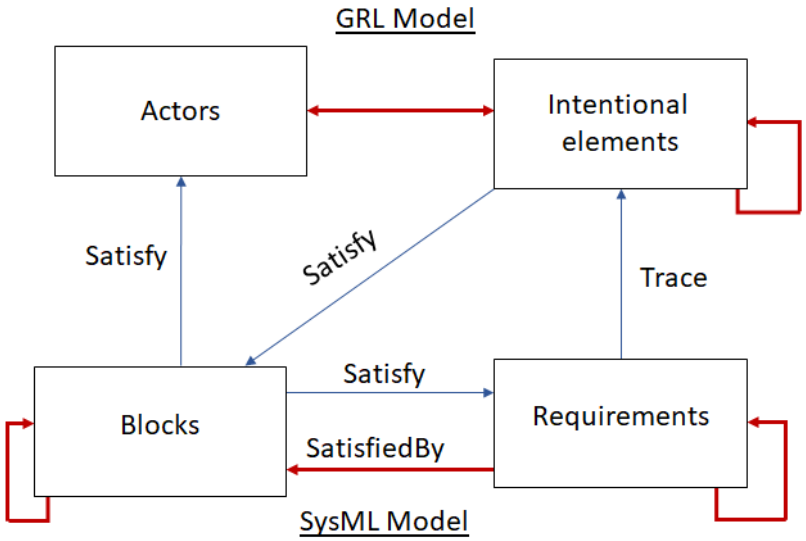

Figure 4.4: Traceability information model for managing traceability between goal and SysML elements in a RMS.

The new *Satisfy* and *Trace* links between GRL model elements and SysML model elements in Fig. 4.4 need to be populated manually in the RMS. Other such links (possibly involving other model element types) could be created should the need arise, but this thesis is only exploring these particular links, to keep the manual effort to a minimum.

Table 4.1: Exported elements from a SysML model to the RMS.

Diagram	**Exported element**	**Rationale**	**Linked to**	**Related attributes**
Requirement Diagram	Requirement	Describes requirements textually	Functional requirements are linked to goals, and nonfunctional requirements are linked to softgoals or to related requirements stored in the RMS	ID, Name, and Text
	Relationships: *Containment*, *Derive*, *Satisfy*	Describe the SysML relationships between requirements as well as between requirements and blocks		ID, Name, Source-ID and Destination-ID
Block Definition Diagram	Block	Core element representing software, hardware, material, or a process	Goals, softgoals, indicators, and tasks	Name and Type
	Block's Parts	Describe the *Composed-of* relationships between blocks		Source and Destination Names
	Aggregation	Describes the shared blocks		Source and Destination Names
	Dependency	Describes dependency relationships between blocks		Source and Destination Names
	Reference	Describes the association relationships between blocks		Source and Destination Names
	Generalization Relationship	Describes the type-of relationships between blocks		Source and Destination Names

According to the specified TIM (and with inspiration from [11]), consistency and completeness rules that can help uncover issues between and within models in a SCPS context include the following:

1. Each GRL element (intentional element and actor) shall be satisfied by one or more blocks in the blocks module.
2. Each block shall satisfy one or more intentional elements.
3. Each intentional element shall be traced to one or more requirements in the requirements module.
4. Each requirement shall trace one or more intentional elements.
5. Each block shall satisfy one or more requirements.
6. Each requirement shall be satisfied by one or more blocks.
7. If the parent/owner of objects is linked to another module, all its associated children/parts shall be implicitly considered linked to the same module (with the same link type).
    - In a SysML model, all objects linked to another object in the same module via ContainedBy, PartOf, or ChildOf relationships are considered children of that object.
    - In a GRL model, all objects linked to another object in the same module via (AND, OR, XOR) composition relationships are considered parts of that object.

The last rule will help minimize the number of required links to be manually provided. Note that some model elements might be considered exceptions to the above rules (e.g., a requirement about the color of a system element may not need to be linked to an intentional element). In order to enable modelers to identify false positives explicitly in such cases (especially when the consistency and completeness analysis is repeated), objects that are marked/labeled with an "ignoring note" will be ignored during analysis (and hence these objects will not violate the rules).

4.3 Methods for Design, Simulations, and Runtime Adaptation

Self-adaptation support should be considered early while selecting SCPS design and architecture, and later while implementing the system itself. Therefore, reusing a design-level goal model (or part thereof) in a simulation context and at runtime is desirable, especially if we want to minimize the distance between the original goal model and its implementation (and minimize the risks of translation errors). We aim to support designers when

choosing between design alternatives by transforming goal and feature models to mathematical functions that can be executed outside goal modeling tools through SysML-based simulation tools or optimization/solver tools such as JaCoP [99] and IBM CPLEX [82].

This approach enables designers to consider user preferences and objectives as well as conflicts when choosing architectures and selecting among alternative designs. Similarity, having the goal model runnable in a SysML-based simulation can help explore design-time adaptation by considering and prioritizing the embedded adaptation strategies according to user preferences and contexts. In addition to simulation support, we are interested in solving goal models (in GRL) at run-time while a system is running. This method is of particular interest to adaptive systems, where goals are often used to react/adapt to a monitored context through indicators, sensors, and other information sources. The mathematical functions generated from goal and feature models can be employed to support most activities of runtime adaptation (Monitor, Analyze and Plan/Select the best strategy).

- The *Monitor* activity detects symptoms of insufficient system quality or goal violation through indicators and the satisfaction level of the goal function.
- The *Analyze* activity decides whether to trigger the plan activity or not using the evaluated goal function. Also, the feature function evaluation can be used to prevent activating the plan activity when infeasible strategies are activated by users or contexts.
- The *Plan* activity is responsible for assessing the strategies using the functions of goal and feature models as well as the current context in order to select the best one. This aims to enhance system performance, degraded by specific contexts, or solve emergent conflicts that lead to infeasible configurations. However, the sources of assessed strategies are different and based on the time available to select the "best" adaptation.
  - Predefined embedded strategies, tested, and prioritized at design time, are used when facing well-known situations.
  - Otherwise, new strategies are generated on the fly by a solver (possibly with a given time budget) using the optimization model based on the functions of goal and feature models, with the values of the functions' variables supplied by real data from the current context. This makes the proposed approach an *open adaptation* approach (as defined in Section 3.2.3 on page 53).

Before applying the new strategies, the cost of adaptation can be measured by, for example, the effort and time needed to change from the current configuration to the best one, which depends on the nature of the changed tasks themselves. This can be done trough different techniques which are out of the scope of this thesis.

Before discussing the details of the simulation, optimization, and adaptation methods, new arithmetic semantics need to be provided to goal and feature models.

4.3.1 Arithmetic Semantics for GRL

The complete goal model (including actors, resources, goals, softgoals, tasks, indicators, relationships, weights, importance, etc.) needs to be translated to a mathematical function encoded in languages compatible with (SysML) simulation tool, optimization tools, and system implementations.

To do so, a new arithmetic semantics for standard GRL, based on preliminary work by Luo and Amyot [108], is used in this transformation. There are three main categories of concepts in GRL:

1. **Intentional elements**, including goals, softgoals, tasks, resources, and indicators. Intentional elements have a satisfaction value (v) computed at runtime based on a selected GRL strategy or on external inputs. They also have an importance value representing their weight in the computation of the satisfaction of their containing actor, if any. An indicator additionally includes four parameters (current value, target value, threshold value, and worst value) used to transform an external input (current value) into a satisfaction value.

2. **Intentional links**, including AND/OR decomposition links, (weighted) contribution links, and dependency links.

3. **Actors**, which also have a satisfaction value, as well as an importance value used in the computation of the satisfaction of the entire model.

Figures 4.5 to 4.8 show the different types of links between source intentional elements (S) and destination intentional elements (D_x).

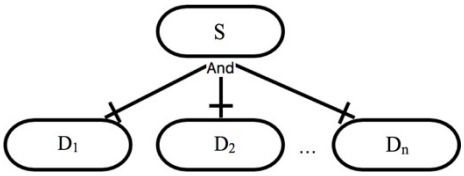

Figure 4.5: AND-decomposition link

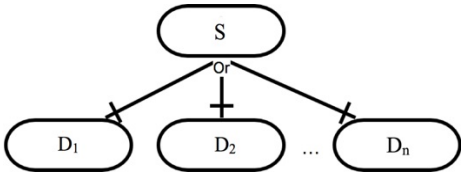

Figure 4.6: OR-decomposition link

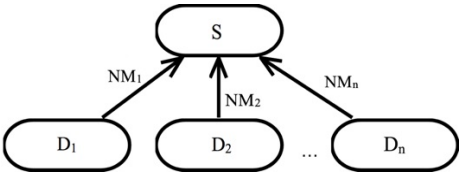

Figure 4.7: Contribution link

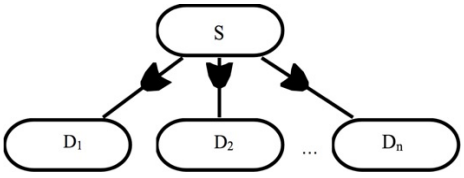

Figure 4.8: Dependency link

The quantitative satisfaction value of a leaf **intentional element** is provided as explicit input in a strategy. For the other intentional elements, the satisfaction value of the source ($v(S)$) is a function of the destination ($v(D_x)$) of its **intentional links**. Satisfaction is evaluated over [0..100]. The following functions are compatible with GRL's informal semantics defined in the URN standard [90].

1. For the AND-decomposition (Fig. 4.5), the minimum is propagated.

$$v(S) = Min(v(D_1), v(D_2), ..., v(D_n)) \tag{4.1}$$

2. For OR-decomposition (Fig. 4.6), the maximum is propagated.

$$v(S) = Max(v(D_1), v(D_2), ..., v(D_n)) \tag{4.2}$$

3. For contribution links (Fig. 4.7), NM_x represents the quantitative contribution (weight, over $[-100..100]$) of destination intentional element D_x to the source intentional element S. A weighted sum truncated to the range of allowed values is propagated.

$$v(S) = Max\Big(0, Min\big(100, \frac{\sum_{x=1}^{n}(v(D_x) \times NM_x)}{100}\big)\Big) \tag{4.3}$$

4. For dependency links (Fig. 4.8), the current element's satisfaction (defaulted to 0) is truncated to the minimum satisfaction value of its dependees.

$$v(S) = Min(v(S), v(D_1), v(D_2), ..., v(D_n)) \tag{4.4}$$

In GRL, an intentional element can have decomposition (one type only), contribution, and dependency links simultaneously. In such cases, first the satisfaction from decomposition links is computed, then the contribution satisfactions are added, and finally the dependency links are used to truncate the result (as specified in the URN standard [90]). For example, the arithmetic interpretation of Fig. 4.9 is:

$$v(S_{decomp}) = Min(v(D_1), v(D_2))$$

$$v(S_{contrib}) = Max\Big(0, Min\big(100, \frac{25 \times v(D_3) + 100 \times v(S_{decomp})}{100}\big)\Big)$$

$$v(S) = Min(v(S_{contrib}), v(D_4))$$

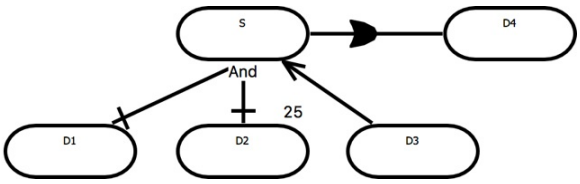

Figure 4.9: Multiple types of links

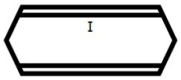

Figure 4.10: Indicator

Figure 4.10 shows an **indicator** element I that can also be linked to other intentional elements (except that an indicator cannot be decomposed or receive contributions). The CGS4Adaptation approach support three types of indicators, where the last two represent a minor contribution of this thesis to enable more flexibility in monitoring SCPS contexts than what is offered by standard URN:

1. *Standard indicator*, where the satisfaction value of an indicator is computed by comparing the indicator's current (observed) value against its target, threshold, and worst parameter values. The threshold value is always between the target and the worst values, and the target, threshold, and worst values cannot be simultaneously equal. The satisfaction is: 100 if the current value reaches the target value (or does better), 50 if it equals the threshold value, and 0 if it reaches the worst value (or is even worse). Linear interpolations are used in the other cases.

   In this context, when the target value of an indicator is higher than its worst value, the satisfaction value of this indicator (where C represents the current value, T is the target value, TH is the threshold, and W is the worst value) becomes:

$$v(I) = \begin{cases} 100 & \text{if } C \geq T \\ 0 & \text{if } C \leq W \\ Abs\left(\frac{C-TH}{T-TH}\right) \times 50 + 50 & \text{if } TH \leq C < T \\ -Abs\left(\frac{C-TH}{W-TH}\right) \times 50 + 50 & \text{if } W < C < TH \end{cases} \tag{4.5}$$

   From the Hybrid car engine example (Section 6.2.3), the values of the Distance indicator's parameters are: Target=25, Worst=5, and Threshold=10. Since the target value here is higher than the worst value, the indicator's evaluation formula becomes:

$$v(Distance) = \begin{cases} 100 & \text{if } C \geq 25 \\ 0 & \text{if } C \leq 5 \\ Abs(\frac{C-10}{25-10}) \times 50 + 50 & \text{if } 10 \leq C < 25 \\ -Abs(\frac{C-10}{5-10}) \times 50 + 50 & \text{if } 5 < C < 10 \end{cases} \tag{4.6}$$

   If the Current distance, C, equals 11, the satisfaction value of the Distance indicator becomes:

$$v(Distance) = Abs(\frac{C-10}{25-10}) \times 50 + 50$$

$$v(Distance) = Abs(\frac{11-10}{25-10}) \times 50 + 50 = 53.3$$

   When its target value is inferior to the worst value (which happens, for example, when an indicator represents a wait time where the smaller the current value, the better), the satisfaction value of the indicator becomes:

$$v(I) = \begin{cases} 100 & \text{if } C \leq T \\ 0 & \text{if } C \geq W \\ Abs\left(\frac{C-TH}{TH-T}\right) \times 50 + 50 & \text{if } T < C \leq TH \\ -Abs\left(\frac{C-TH}{TH-W}\right) \times 50 + 50 & \text{if } TH < C < W \end{cases} \tag{4.7}$$

   From the Hybrid car engine example (Section 6.2.3), the values of the Vibration indicator's parameters are: Target=0, Worst=20, and Threshold=10. Because the

target value here is less than the worst value, the indicator's evaluation formula becomes:

$$v(Vibration) = \begin{cases} 100 & \text{if } C \leq 0 \\ 0 & \text{if } C \geq 20 \\ Abs(\frac{C-10}{10-0}) \times 50 + 50 & \text{if } 0 < C \leq 10 \\ -Abs(\frac{C-10}{10-20}) \times 50 + 50 & \text{if } 10 < C < 20 \end{cases} \tag{4.8}$$

As a result, if the Current value of the indicator, C, equals 9, the satisfaction value of the Vibration indicator becomes:

$$v(Vibration) = Abs(\frac{C-10}{10-0}) \times 50 + 50$$

$$v(Vibration) = Abs(\frac{9-10}{10-0}) \times 50 + 50 = 55$$

2. *Boolean indicators* use logical conditions on their current values. Their values are: 100 if the value of the related condition is true, and 0 if the condition is false. Developers can use any logical conditions on the current value of the indicator. The satisfaction semantics of this kind of indicators is:

$$v(I) = \begin{cases} 100 & \text{if } Condition = True \\ 0 & \text{Otherwise} \end{cases} \tag{4.9}$$

For example, in a smart home, a Thermostat indicator can be set to operate the heater if the temperature is less than 15 degrees Celsius. The evaluation formula is:

$$v(Thermostat) = \begin{cases} 100 & \text{if } C < 15 \\ 0 & \text{Otherwise} \end{cases}$$

If the current temperature C equals 10, the satisfaction value of the Thermostat indicator equals 100.

3. *User-defined indicators* use equations assigned explicitly by modelers. Their satisfaction value is calculated by evaluating the provided function.

The satisfaction of a GRL **actor** depends on the satisfaction of its contained intentional elements with non-null importance values (the latter being specified over [0..100]). A weighted average is used if the sum of the weights is greater than 100, otherwise a weighted sum is used, as prescribed by the URN standard [90]. Figure 4.11 illustrates an actor A containing n intentional elements E_x with corresponding importance weights W_x. The arithmetic semantics of an actor's satisfaction is:

$$v(A) = Max\Big(0, Min\big(100, \frac{\sum_{x=1}^{n}(v(E_x) \times W_x)}{Max(100, \sum_{x=1}^{n} W_x)}\big)\Big) \tag{4.10}$$

If none of the top-level (root) intentional elements of an actor has a strictly positive weight, then these top-level elements are considered to be weighted equally, with the weights summing up to 100.

Finally, actors themselves can also be weighted in order for the satisfaction of the entire GRL **model** to be computed. The satisfaction of the model has semantics similar to an actor's (a weighted sum or average). In Fig. 4.12, A_x is one of n actors in a GRL model and AW_x is the weight of actor A_x. The quantitative evaluation value of a GRL *Model* is:

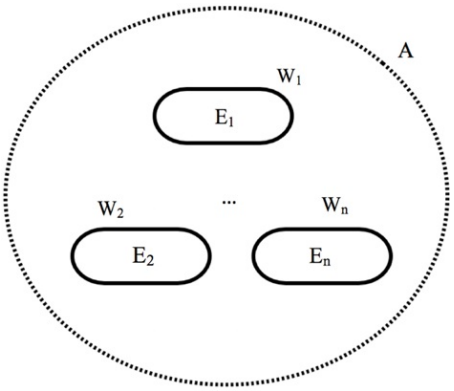

Figure 4.11: Actor containing intentional elements with importance weights

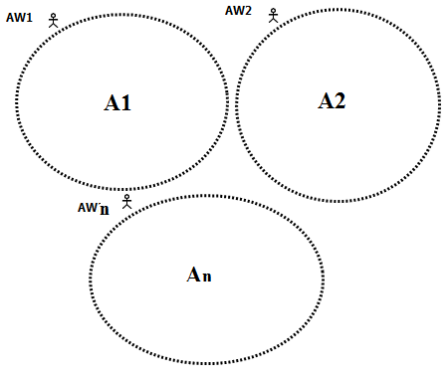

Figure 4.12: Actors with importance weights in a model

$$v(Model) = Max\Big(0, Min\big(100, \frac{\sum_{x=1}^{n}(v(A_x) * AW_x)}{Max(100, \sum_{x=1}^{n} AW_x)}\big)\Big) \tag{4.11}$$

For example, the GRL model of the Hybrid car engine example that has two actors, the User and the System, with different weights, respectively set to 40 and 60 (Section 6.2). The quantitative evaluation value of that GRL model becomes:

$$v(AdaptiveCar) = \frac{(v(User) \times 40 + v(System) \times 60)}{100} \tag{4.12}$$

Additional semantic rules for GRL models include:

1. If there is no actor present in the model, then a default actor containing all intentional elements and that has an importance weight of 100 is assumed to exist.

2. If there are actors but they have no weight explicitly declared, then these actors are considered to be weighted equally, with the weights summing up to 100.

This arithmetic semantics for GRL enables the generation of executable mathematical functions for any valid GRL model. The implementation details of a transformation from GRL models to executable arithmetic functions are provided in Chapter 5.

4.3.2 Arithmetic Semantics for Feature Models

Typically, a goal model alone may not encode all the restrictions needed to ensure the validity of global solutions [54]. The validity of these configurations (and corresponding GRL strategies) can be determined by a companion feature model where the features correspond to the leaves of a GRL model (typically GRL tasks). Arithmetic semantics for feature models are defined and are exploited to realize the validity evaluation.

New arithmetic semantics for standard feature models are used in this approach to transform theses models into computable mathematical functions. There are two main concepts in feature models: *features* capture system characteristics of importance to stakeholders, and *links* are used to compose features together. In order to include a feature in the satisfaction calculation process, two answers need to be considered: Is the feature selected (represented by the $s(f)$ function, returning a value between 0 and 100)? If yes, is its related sub-tree valid (represented by the $v(f)$ function)? Features and links influence the computations of the two functions:

- $s(f)$ is a **selection function** that returns 100 when a feature f is selected explicitly or when a feature in a sub-tree linked to f is selected, and 0 otherwise. Figure 4.13 shows selected feature ($F1$) and unselected feature ($F2$). The selection of features can be based on a GRL strategy (initializing selected tasks) or external inputs (from indicators). The computation of $s(f)$ is simply the maximum value among the sub-features f_x it depends on (e.g., via decompositions):

$$s(f) = Max(s(f_1), s(f_2), ..., s(f_n)) \tag{4.13}$$

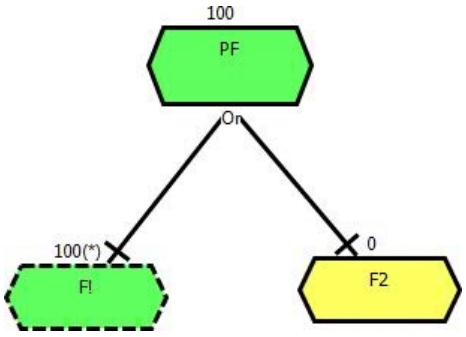

Figure 4.13: Selected and unselected features

- $v(f)$ is a **value function** that returns the satisfaction value, between 0 and 100 inclusively, of feature f. Satisfaction values are calculated by validating each feature using the related features and the relationships linking them. Any invalid selection (e.g., two selected features linked together by an exclusion relationship) must be detected and its value (less than 100) should be propagated. The computation of $v(f)$ is more complicated than that of $s(f)$ and depends on the types of links involved.

The overall satisfaction value of a feature model corresponds to the satisfaction value of its root feature and must be exactly 100 for a configuration (i.e., a design or a strategy) to be considered valid. Anything less than 100 means the configuration is invalid and should not be considered.

Note also that given the acyclic structure of feature models, we also conjecture that $s(f) \geq v(f)$ for any feature f in a model.

Definition of *v(f)* Based on Links Different link types represent mandatory and optional relationships, as well as AND, OR, and XOR decompositions. *Exclusion* constraints, which specify that two features conflict and cannot be selected simultaneously, and *inclusion* constraints, which indicate that one feature requires another one, are also links (although many feature diagram concrete syntaxes show them textually rather than graphically).

Figures 4.14 to 4.20 show the different types of links between parent features (PF) and children features (f_x) or between two features.

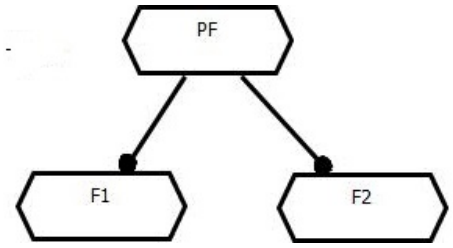

Figure 4.14: Mandatory link

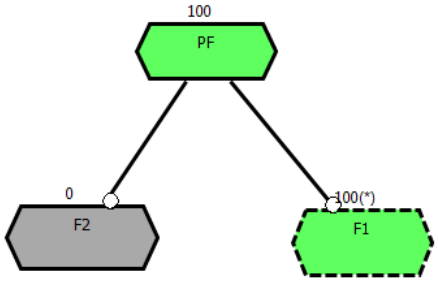

Figure 4.15: Optional link

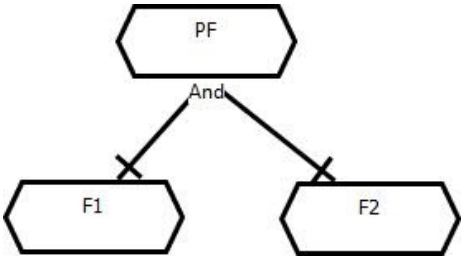

Figure 4.16: AND decomposition link

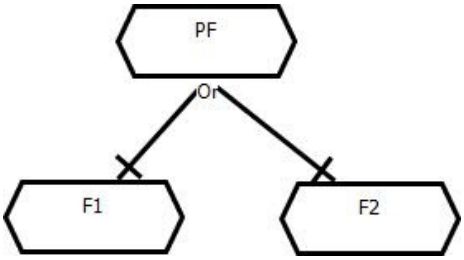

Figure 4.17: OR decomposition link

As the selected and unselected features translate to numerical values, it becomes possible to transform the links between features to mathematical expressions that are used to propagate the values $v(f)$ of lower-level features to the higher-level and parent features. The include and exclude links, the decomposition links (AND, OR, XOR and mandatory), and the optional links are handled for each feature starting from the bottom of the model to the root feature as follows:

1. For leaf features without include or exclude features, $s(f)$ is propagated as is.

$$v(f) = s(f) = f \tag{4.14}$$

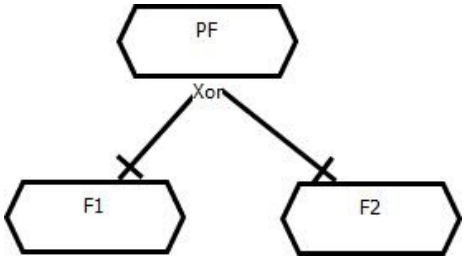

Figure 4.18: XOR decomposition link

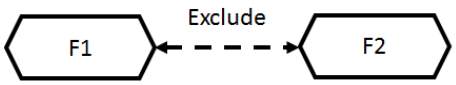

Figure 4.19: Exclude link

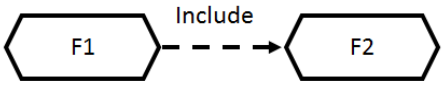

Figure 4.20: Include link

Where values of $s(f)$ are 0 for an unselected feature and 100 for a selected feature.

2. For exclude links (Fig. 4.19), the difference between the maximum value of the $s(f_x)$ of the excluded features and the $s(f)$ of the feature itself is calculated. $v(f)$ can only be 100 if all of the $s(f_x)$ are 0.

$$v(f) = Max\Big(0, \big(s(f) - Max(s(f_1), s(f_2), ..., s(f_n))\big)\Big) \tag{4.15}$$

For example, the value of feature $F1$ in Fig. 4.19 is:

$$v(F_1) = Max\big(0, (s(F_1) - s(F_2))\big)$$

3. For include links (Fig. 4.20), where the direction matters, the sum of the minimum values of the included features $s(f_x)$ with the $s(f)$ of the feature itself, divided by 2, is calculated.

$$v(f) = \frac{s(f) + Min(s(f_1), ..., (f_n))}{200} \times 100 = \frac{s(f) + Min(s(f_1), ..., (f_n))}{2} \tag{4.16}$$

Where n is the number of included features, whether they are selected or not. For example, the value of the feature $F1$ in Fig. 4.20 is:

$$v(F_1) = \frac{s(F_1) + s(F_2)}{2}$$

4. For the AND-decomposition and mandatory links (Figs. 4.16 and 4.14), the minimum value of the involved features' values ($v(f_x)$) is propagated, with n being the number of child features.

$$v(f) = Min(v(f_1), v(f_2), ..., v(f_n)) \tag{4.17}$$

For example, the value of the parent feature in Fig. 4.16 is:

$$v(PF) = Min(v(F_1), v(F_2))$$

5. For the OR-decomposition (Fig. 4.17), the sum of the child features' satisfaction values $v(f_x)$ divided by the sum of their selection values $s(f_x)$ is propagated. Note that divisions by zero are also avoided by using $Max(1, ...)$.

$$v(f) = Min(100, \frac{\sum_{x=1}^{n} v(f_x)}{Max\big(1, \sum_{x=1}^{n} s(f_x)\big)} \times 100) \tag{4.18}$$

For example, the value of the parent feature in Fig. 4.17:

$$v(PF) = Min(100, \frac{v(F_1) + v(F_2)}{Max\big(1, s(F_1) + s(F_2)\big)} \times 100)$$

6. For XOR-decomposition (Fig. 4.18), the maximum value of the child features $v(f_x)$ divided by the sum of their selection values $s(f_x)$ is propagated.

$$v(f) = \frac{Max\big(v(f_1), v(f_2), v(f_x)\big)}{Max\big(1, \sum_{x=1}^{n} s(f_x)\big)} \times 100 \tag{4.19}$$

For example, the value of the parent feature in Fig. 4.18:

$$v(PF) = \frac{Max\big(v(F_1), v(F_2)\big)}{Max\big(1, s(F_1) + s(F_2)\big)} \times 100$$

7. For optional links (Fig. 4.15), the sum of the $v(f_x)$ of the optional features divided by the sum of their $s(f_x)$ is propagated if at least one the $s(f_x)$ is 100. If nothing is selected, as the links are optional, the constraint is still satisfied. The only moment where $PF \neq 100$ is when one satisfaction value from a linked subtrees is less than 100.

$$v(f) = \begin{cases} 100 & \text{if } Max(s(f_1), ..., s(f_n)) = 0 \\ Min(100, \frac{\sum_{x=1}^{n} v(f_x)}{\sum_{x=1}^{n} s(f_x)} \times 100) & \text{Otherwise} \end{cases} \tag{4.20}$$

For example, the value of the parent feature in Fig. 4.15, without considering the initialization, is:

$$v(f) = \begin{cases} 100 & \text{if } Max(s(f_1), ..., s(f_n)) = 0 \\ Min(100, \frac{v(F1)+v(F2)}{s(F1)+s(F2)} \times 100) & \text{Otherwise} \end{cases}$$

Assuming that $F1$ is selected (i.e., $s(F1) = 100$), as highlighted in green in Fig. 4.15, then we get:

$$v(PF) = Min(100, \frac{100 + 0}{100 + 0} \times 100) = 100$$

8. Finally, for situations where multiple include, exclude, and other link types are mixed together (e.g., see features *PF1* and *PF2* in Fig. 4.21), the minimum value of the involved link types previously computed $v_x(f)$ is propagated.

$$v(f) = Min(v_1(f), v_2(f), ..., v_x(f)) \tag{4.21}$$

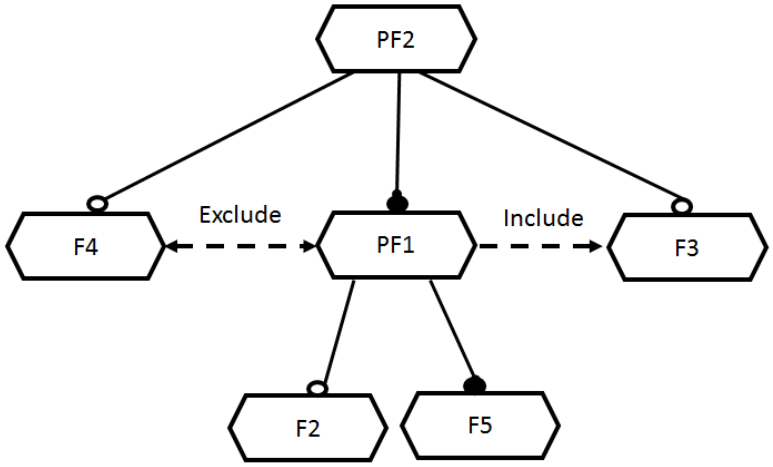

Figure 4.21: Feature model with different links

These values are propagated from the lower level to the upper level until the root feature of the model is reached. A *constraint violation* is detected when the value of the root feature is less than 100, which means the chosen configuration is invalid.

4.3.3 Multi-Language Transformations for Simulation and Adaptation

A secondary problem is that as the number of target languages for specifying functions can be high, so we would like to minimize the effort required for supporting many languages. Ideally, an intermediate representation for mathematical functions can be used as a target for GRL and feature models transformations, with additional (but small, and hopefully already existing) transformations to specific programming languages. We have selected the SymPy environment, briefly introduced in Section 2.2.2, to transform GRL and FM functions into common implementation languages (Java, JavaScript, Python, C, and C++) as well as simulation languages (R and Matlab), some of which can be used embedded in other modeling languages (e.g., SysML). The implementation of transformation is detailed in Chapter 5.

Using this approach, we can support effective and consistent goal-based reasoning during design time simulation activity and runtime adaptation activities (monitoring, analysis, and planning). Moreover, this transformation translates *all* goal model elements (including contributions, indicators, and actors, unlike most other approaches seen in the literature review) for additional reuse in activities related to system design and implementation. The strategies generated via goal model functions can be validated by the related feature model, also translated to a mathematical function, to speed up and facilitate the analysis of large and complex models. As these functions are used as black boxes whose input is the current situation and the outputs are the feasible solutions or configurations, they may be adapted according to the required usage (multi-criteria reasoning, optimization functions, goal reasoning, etc.).

4.3.4 Optimization and Simulation Models

In order to select the "best" configurations, goal-based reasoning is needed to assess the overall satisfaction of the system and its stakeholders using valid configurations [97] constrained by the feature model. Fig 4.22 shows the intersection area between these two sets. This area represents valid (feasible) configurations (i.e., strategies in GRL) that satisfy the stakeholders' goals, system qualities, and the current environmental conditions. The solutions in this intersection are also candidates for various optimizations. For example, one could optimize the satisfaction of one given actor, or the weighted sum of all actors. GRL offers such weights out of the box with *importance* values for goals to its containing actor, and for actor themselves for the model [90].

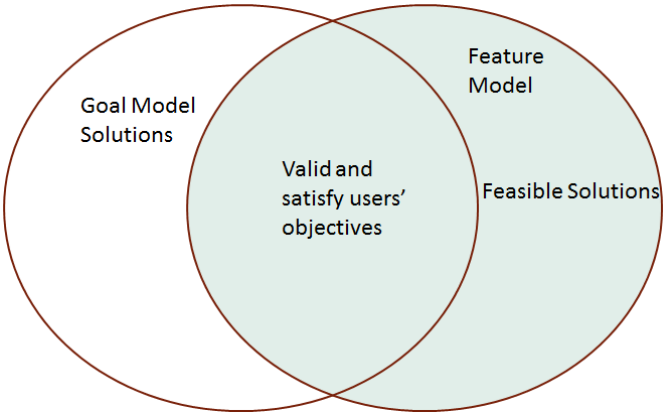

Figure 4.22: Sets of configurations

In order to get the optimal strategies that consider stakeholders' preferences and comply with system quality (optimal strategies from the intersection area in Fig. 4.22), we aim to use the mathematical functions generated from GRL/feature models as objective functions of our mathematical **optimization model** (Section 2.2.6). The alternatives in the GRL model represent the decision variables while the mathematical function of the feature model is a filter that forces only valid solutions to be optimized [45]. The functions share common variables because the *leaf features* in the feature model correspond to *GRL tasks* in the goal model. The other parts of the two models are built independently and can lead to different structures.

In CGS4Adaptation, we use the current value of the GRL indicators (e.g., sensor data) as *ancillary variables* that describe the environmental context. Then, an optimizer can use the generated values of the *decision variables* (alternatives) to generate the optimal solution for the given context. The value of a decision variable must be 100 (selected feature) or 0 (unselected feature). The goal model function contains two type of variables $FG(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m)$ where n is the number of the alternatives (decision variables x_i) while m is the number of ancillary variables (indicators' current value y_j). As for the feature model function, it focuses on the alternatives only: $FFM(x_1, x_2, ..., x_n)$. Based on these mappings, our optimization model maximizes the value of the goal satisfaction function for valid strategies only and the objective function is shown in Equation 4.22:

Maximize

$$FG(x_1, ..., x_n, y_1, ..., y_m) \tag{4.22}$$

Subject To:

$$FFM(x_1, ..., x_n, y_1, ..., y_m) == 100$$
$$(x_{1,...,n} \in \{0, 100\})$$

In a **simulation context**, we collect quantitative data (e.g., feeding the GRL model's KPIs) to evaluate the effectiveness and the correctness of the strategies chosen in the previous optimization process. The different strategies can be ranked (according to their overall model satisfaction and coverage of contexts) and stored to be used during runtime adaptation. This integration of goal and feature model functions:

1. Uses a feature model function to determine *feasible* solutions.
2. Uses goal satisfaction levels to find *optimal* solutions.
3. Reduces risks by assessing system quality to obtain *appropriate* designs.
4. Uses data analysis based on KPIs to reach the *useful* solutions in a given context.
5. Considers stakeholders' preferences (e.g., via the importance of goal model's elements and their weighted contributions) to finally get the *correct* solutions.

Figure 4.23 shows a Venn diagram illustrating the different sets of solutions. During design time, stakeholders' preferences and opinions about a specific context help in going from candidate designs to the correct design. This process is also applicable during runtime adaptation where contextual data can be obtained using indicators and sensors. The mathematical functions of goal and feature models can be turned into code embedded in system applications to generate correct solutions for the current environmental context.

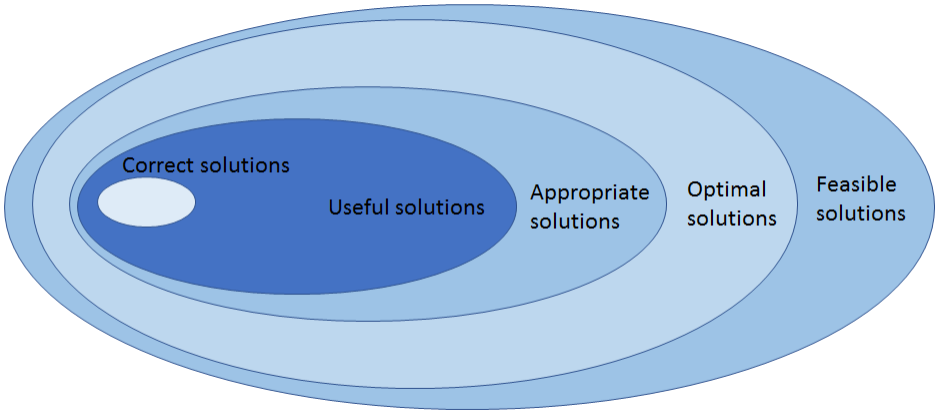

Figure 4.23: Set of the generated solutions, adapted from [45]

Also, this model is used to validate goal and feature models during design time to ensure that these models meet the requirements of the developed system and there is at least one suitable solution for each context.

4.4 Chapter Summary

This chapter described a new CGS4Adaptation approach to support traceability and decision making for developing SCPSs with models. For traceability support, the elements exported from goal and SysML models to RMS tools were described, together with rationales behind their selection. Also, arithmetic semantics for standard GRL and for feature models were introduced in order to enable the automated generation of functions to be embedded in both system design (through simulation and optimization) and system operation (through code). The next chapter focuses on the details of the implementation of several parts of CGS4Adaptation and the tools involved.

Chapter 5

Approach Implementation

This chapter provides one specific implementation of the CGS4Adaptation approach, which exploits the tools and techniques mentioned in Chapter 2. The import of GRL models (produced with jUCMNav) and of SysML models (produced using Cameo Systems Modeler) into the third-party requirements management system (IBM DOORS) is done using DXL scripts, MI-DSL classes, and related libraries. In addition, a new jUCMNav plug-in has been implemented to generate mathematical functions for GRL and feature models, while an existing SymPy library is reused and enhanced to convert these functions into 7 programming languages. Optimization model are built and solved using the IBM ILOG CPLEX optimizer.

5.1 Traceability Management at Design Time

The integration of modeling tools can be done using technologies such as the *Open Services Lifecycle Collaboration* (OSLC) standard [133], which manages a loose integration between modeling tools and an RMS with changes synchronization. However, not all tools support OSLC, especially in the Eclipse world. For that reason, Rahman and Amyot [139] produced the MI-DSL environment (Section 2.2.5) as a way to import models from any language into the DOORS RMS (Section 2.2.4). They demonstrated its applicability to URN models, for traceability management between textual requirements, GRL models, and UCM models. This thesis builds on this work and reuses an existing jUCMNav export plug-in that generates DXL descriptions of GRL models [65], with an existing DXL library for importing GRL models into DOORS. This work is visualized with bold lines in Figures 4.2 and 5.1.

In the same context, SysML models can also be integrated with DOORS by extending a tool (e.g., Papyrus) with a simple export mechanism invoking the MI-DSL library [139] that is produced once only using the MI-DSL Xtext editor. An alternative is to use a commercial (SysML) product such as No Magic's *MagicDraw* [126], with additional commercial modules for DOORS integration. As a result, the elements of both the GRL model and the SysML model will be represented in the DOORS repository, as shown in Fig. 5.1. At this point, we can take advantage of the RMS to define and exploit traceability *between* models as well as conduct different types of analysis.

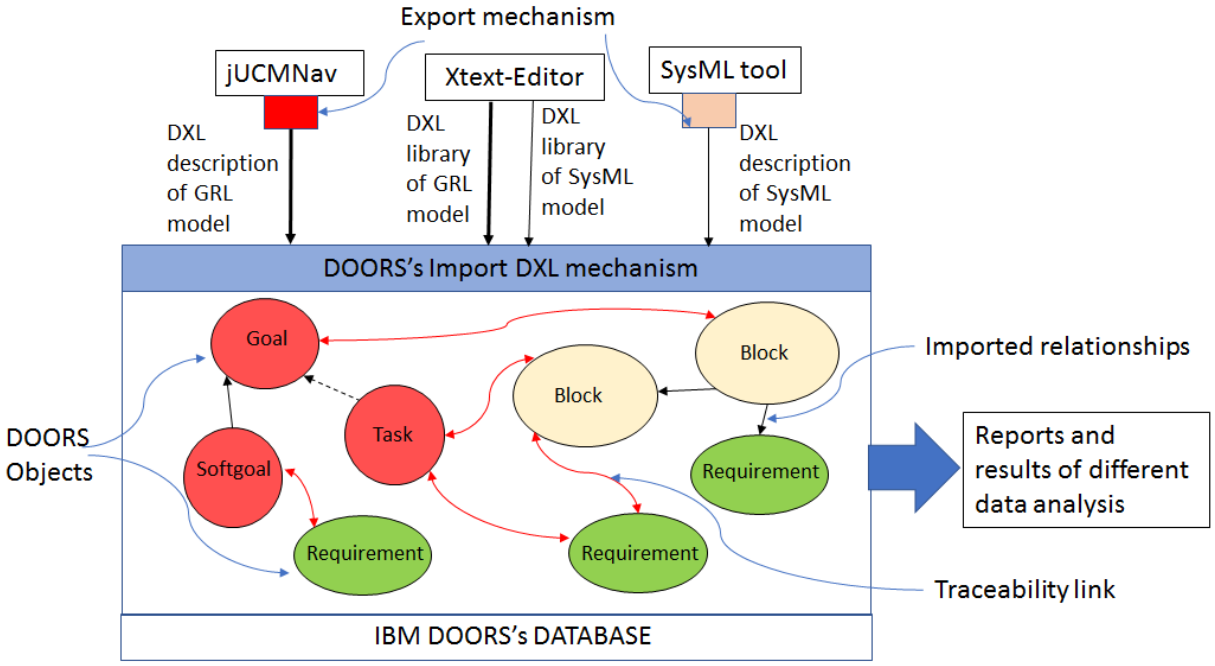

Figure 5.1: Traceability management at design time

In this thesis, we use MI-DSL and its Xtext-based editor [139] to enable importing and managing in DOORS relevant SysML model elements such as block definition diagrams and requirement diagrams. MI-DSL enables the specification of what SysML elements (classes) and links (associations) to import and then generates a corresponding DXL library of functions that can then be invoked by a model description (DXL script invoking the library's functions) corresponding to a specific SysML model. In our case, we used the Report Wizard of NoMagic's *Cameo Systems Modeler* tool to convert these elements of a SysML model into a DXL script runnable by DOORS against the library produced from our MI-DSL traceability information model.

5.1.1 MI-DSL's DXL Library for SysML Model Import

The MI-DSL code in Listing 5.1 defines the elements (classes) of the SysML language that we selected to be imported and tracked in DOORS database. MI-DSL then generates a DXL library with functions used to manipulate (create, modify, and delete) these elements in the DOORS database. DOORS uses *formal modules* to store the information about objects (and their attributes) and *link modules* to store the relationships between them. The selected elements from the SysML language, listed in Table 4.1, are mapped to modules and objects in DOORS via MI-DSL classes. As shown in Listing 5.1, two SysML diagrams are described: Requirement diagram and Block definition diagram. Each class in this MI-DSL description has default attributes: Name, ID, and Description (and hence they are not explicitly mentioned in this specification). To avoid information loss and duplication of work, the relationships between the model elements in both diagrams are also imported.

These relationships are very useful in realizing different types of analysis such as impact analysis and change management analysis (Chapter 8).

Requirement diagrams The MI-DSL classes for requirement diagrams (requirementsDiagram, requirement, and requirementlink) translate the requirement diagram elements and their relationships into DOORS objects and links as follows:

1. Formal modules: hold the selected elements of requirement diagrams. The relevant MI-DSL classes (DOORS modules) are:
   (a) requirementsDiagram module: includes the instances of the requirement diagrams.
   (b) requirement module: includes the instances of requirements.
   (c) requirementlink module: includes all the relationships within a particular requirement diagram, as well as the relationships between requirements and blocks.
2. Linksets: used to classify the links between objects in sets or groups to facilitate the traceability analysis process. The DOORS linksets are described by all the associations in the requirement diagram's classes such as derivedFrom (links between requirements), satisfiedBy (links between requirements and blocks) and containedBy (links between requirements or between an individual requirement and its related requirement diagram).
3. Link modules: represent the link types between the created objects; this is represented by the related associationTypes: Derive, Satisfy, and ContainedBy.

Block definition diagrams The MI-DSL classes (blockDefinitionDiagram, block, and blocklink) translate the block definition diagrams and their relationships into DOORS objects and links as follows.

1. Formal modules: hold the selected elements of block definition diagrams. The classes describing these elements are:
   (a) blockDefinitionDiagram module: includes the instances of block definition diagrams.
   (b) block module: includes block instances
   (c) blocklink : includes the relationships in a particular block definition diagram.
2. Linksets: include all the associations in the classes related to block definition diagrams, such as partOf, childOf, refersTo, Dependency, and IsSharedBy.
3. Link modules: represent the links between the created blocks, including all the related associationTypes: Composition, Generalization, Aggregation, Dependency, and Association.

We used the MI-DSL editor to generate DXL library files automatically from the MI-DSL traceability model for SysML in Listing 5.1. The generated DXL files and the related functionalities are described in Table 5.1. These files are responsible for supporting the import and re-import of the selected language concepts of SysML models into DOORS, while keeping the consistency between the imported models. Rahman [139] provides more information about the editor and the generation of DXL library files from MI-DSL descriptions.

Table 5.1: Generated DXL library files for SysML models (adapted from [139]).

File	Description
Utility.dxl	This file contains the list of import statements to import all other library files
Global.dxl	This file declares global variables used in all DXL files in the library.
ModulesUtility.dxl	This file includes the helper DXL functions that are invoked during the model import process in DOORS.
Links.dxl	This file contains DXL library code for the links described in the modules.
InitExist.dxl	This library file contains all the DXL functions to initialize and finalize the import process (including GUI interactions).
Import.dxl	This DXL library file provides the utility method that would be invoked to start the model import process in RMS. This has the DXL function beginImport to start the import process.
Report.dxl	This DXL library file contains generated DXL code for creating a report at the end of the import process.
RequirementsDiagram.dxl	DXL file for module requirementsDiagram
Requirement.dxl	DXL file for module requirement
Requirementlink.dxl	DXL file for module requirementlink
BlockDefinitionDiagram.dxl	DXL file for module blockDefinitionDiagram
Block.dxl	DXL file for module block
Blocklink.dxl	DXL file for module blocklink

Listing 5.1: MI-DSL classes of the selected SysML diagrams and elements

```
model SysMLModel{
 folder SysML{

  // Requirement diagrams
  module requirementsDiagram{
    // Default name, id, description attributes are already provided.
    class requirementsDiagram{
      diagram "graphFileName" shows as "Diagram File Name"
      string "title" shows as "Diagram Title"
```

```
  }
}

module requirement{
  // Default name, id, description are already provided,
  // and the description could contain the Text string.
  class requirement{
    string "ReqID" shows as "ReqID"
  }
}

// SysML relationships involving requirements
module requirementlink{
  fileName "Requirements Relationships"

  class requirementlink{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association derAsso1 : derivedFrom to  "requirement"."requirement"
       "source ID"
    association derAsso2 : derivedFrom to  "requirement"."requirement"
       "destination ID"
  }

  class requirementToBlock{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association SatisfyAsso1 : satisfiedBy to  "requirement"."
       requirement" "source ID"
    association SatisfyAsso2 : satisfiedBy to  "block"."block" "
       destination ID"
  }

  class subrequirement{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association contAsso1 : containedBy to "requirement"."requirement"
       "source ID"
    association contAsso2 : containedBy to "requirement"."requirement"
       "destination ID"
  }

  class requirementToDiagram{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association contAsso1 : containedBy to "requirement"."requirement"
       "source ID"
    association contAsso2 : containedBy to "requirementsDiagram"."
       requirementsDiagram" "destination ID"
  }
```

```
}

// Block definition diagrams
module blockDefinitionDiagram{
  // Default name, id, description attributes are already provided.
  // The description could contain the documentation of the block at
      design time.
  class blockDefinitionDiagram{
    diagram "graphFileName" shows as "Diagram File Name"
    string "title" shows as "Diagram Title"
  }
}

module block{
  class block {
    string "Type" shows as "block Type"
  }
}

// Additional SysML relationships involving blocks only
module blocklink{
  fileName "Block Relationships"

  class blocklink{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association Asso3 : refersTo to  "block"."block" "source ID"
    association Asso4 : refersTo to  "block"."block" "destination ID"
  }

  class blockparts{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association CompAsso1 : partOf to  "block"."block" "source ID"
    association CompAsso2 : partOf to  "block"."block" "destination ID"

  }

  class blockToDiagram{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association CompAsso1 : containedBy to  "block"."block" "source ID"
    association CompAsso2 : containedBy to  "blockDefinitionDiagram"."
       blockDefinitionDiagram" "destination ID"
  }

  class Blockparent{
    string "sourceID" shows as "Source ID"
    string "destinationID" shows as "Destination ID"

    association GenAsso1 : childOf to  "block"."block" "source ID"
```

```
        association GenAsso2 : childOf to  "block"."block" "destination ID"

      }

      class Blockaggregation{
        string "sourceID" shows as "Source ID"
        string "destinationID" shows as "Destination ID"

        association aggAsso1 : IsSharedBy to  "block"."block" "source ID"
        association aggAsso2 : IsSharedBy to  "block"."block" "destination
           ID"
      }

      class BlockDependency{
        string "sourceID" shows as "Source ID"
        string "destinationID" shows as "Destination ID"

        association DepAsso1 : Dependency to  "block"."block" "source ID"
        association DepAsso2 : Dependency to  "block"."block" "destination
           ID"
      }
    }

    // Association types
    associationType derivedFrom "Derive"
    associationType satisfiedBy "Satisfy"
    associationType partOf  "Composition"
    associationType IsSharedBy "Aggregation"
    associationType Dependency "Dependency"
    associationType refersTo  "Association"
    associationType childOf  "Generalization"
    associationType containedBy "ContainedBy"
  }
}
```

5.1.2 SysML Model Descriptions as DXL Scripts

Developers can create DXL scripts (ideally generated from modeling tools) to describe the imported models and invoke the created library. Listing 5.2 highlights the structure of such a script. The first line of the script is used to include the DXL library for SysML models, generated from MI-DSL classes in the previous subsection and stored in the specified path in Rational DOORS' folder. The two next lines and the last line are used to start to import the model and diagrams from a specific folder "FolderName" and to end the import process, respectively. In order to import a particular model into DOORS, the functions in the generated DXL library can be called and element attributes can be passed via their parameters. The first three parameters are the default parameters: ID, Name, and Description. Then, the user-defined parameters are passed in the same order as defined in the related class.

Listing 5.2: DXL script describing a specific SysML model

```
#include "addins/DSL/lib/Utilities.dxl"
pragma runLim, 0
beginImport( "FolderName" )
// Call statements for the created functions corresponding to the MI-DSL
    classes and their parameters
requirement("R-01","Driving","The system shall drive the car using
    acceleration, control and power" ,"")
endImport
```

In order to generate these DXL scripts automatically for the elements and links of interest in SysML models, the Report Wizard tool in Cameo Systems Modeler was used to create a custom report that navigates through a SysML model and generates DXL as output. Reports are generated using Apache's Velocity Template Language (VTL) [24]. We have created a new report template to access, read, and save the properties of the SysML models' elements in a user-defined format. In this template, all the elements of the BDD and requirement diagrams from Listing 5.1 are read and converted into a DXL script that invokes the SysML DXL library produced in the previous section. In addition to the name and identifier of each diagram, requirement, block, part, and relationship, we export the source and destination of each relationship and the image of each diagram.

Listing C.1 in Appendix C is the VTL report template that traverses a SysML model and produces a corresponding DXL script. Listing C.2 gives an example of a DXL script produced by this template for a SysML model to be presented in Chapter 7. This kind of DXL script will be used to import and re-import (in case of model modification) SysML models into DOORS, and manage the consistency between old and new versions of a model, including their links to DOORS objects outside the SysML model (e.g., textual requirements or goals from a goal model). Detailed illustrations will be provided in the next chapters using an adaptive car example and a Smart Home case study.

Having both the SysML and GRL models in DOORS enables the seven consistency and completeness rules described in Section 4.2.3 to be automatically checked. Listing C.3 implements these rules (and others) in DXL. This script reports consistency and completeness issues between the GRL and SysML models using additional attributes (violation flags) that can be filtered by DOORS users, enabling simple violation assessment and fixes.

5.2 Goal and Feature Models for Simulations and Runtime Adaptation

Goal models can be developed to describe how systems should adapt to their environment. Once a goal model is developed and validated (e.g., through the techniques described by Horkoff et al. [78] or Hassine and Amyot [73]), it needs to be transformed into a format executable by various environments, including simulation engines, optimization solvers, and running systems. In addition, the candidate strategies for satisfying a goal model may be

subject to additional constraints, coming for instance from a related feature model. This section focuses on the automated generation of executable functions from goal/feature models.

Figure 5.2 illustrates the main steps involved in transforming a combined GRL/feature model into arithmetic functions in multiple programming languages. A new jUCMNav plug-in was developed to traverse the GRL and feature models and generate a SymPy file conforming to the arithmetic semantics introduced in Chapter 4. This intermediate representation exploits the SymPy Python library (Section 2.2.2) to symbolically simplify the generated functions. For example, $10/2\times(v+v+10)$ would be simplified to $100\times v+50$ (i.e., leading to fewer computations). Then, a new Python script we produced invokes existing or new SymPy code printers to generate executable mathematical expressions in predetermined target programming languages, as separate files.

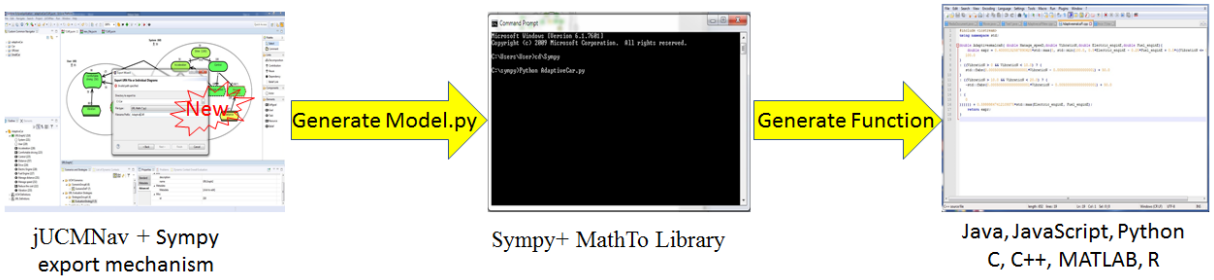

Figure 5.2: Overview of the transformation from goal/feature models to SymPy and then to executable functions

5.2.1 Transformation from GRL and Feature Models to SymPy Mathematical Functions

In order to transform GRL and feature models to mathematical functions according to our new arithmetic semantics, we developed a new export plug-in for jUCMNav. As systems and simulations are implemented using many different languages, it quickly becomes impractical to define one transformation for each target language. One way to reduce this complexity is to use an intermediate representation supporting mathematical functions for which there are transformations to common programming languages already existing. Then, only one transformation from GRL/feature models to that intermediate language would be needed.

We selected SymPy [151] as the intermediate representation. SymPy is an open-source Python-compatible language and library that already comes with a variety of code printers (generators). In addition, it symbolically simplifies functions, which leads to improved performance during evaluations.

The SymPy export plug-in for jUCMNav is freely available online [58]. It supports the entire semantics discussed in the previous chapter. It currently generates two types of functions: one global function and simplified combined functions.

One global function export mechanism: This generates one *global function* for the entire GRL model (which composes all functions from its actors and other elements) as well as one global function for the feature model (which composes all functions from its features and links). Moreover, the export also generates *individual* mathematical functions for the model's intentional elements and actors, for future use (e.g., to support the implementation of the integration between the GRL and feature functions, or to enable testing parts separately).

Simplified combined functions export mechanism: To facilitate the integration between the GRL and feature functions through the leaf features (in a feature model) associated with indicators (in a GRL model), as shown in Fig. 5.3, we have enhanced the initial SymPy export mechanism in jUCMNav to export additional types of mathematical functions. This mechanism defines a *variable* to evaluate each indicator formula and another variable to evaluate the related feature. Then, each indicator/feature formula in the model formula is replaced by the related variable. In fact, we separate these formulas from the models' Formulas to be used individually to facilitate the integration of the two models and improve their performance. Figure 5.4 shows the global and simplified separated functions generated from the GRL model, whereas Fig. 5.5 shows the generated global and split functions generated from the feature model. Note that any formula that does not include decision variables (free features) will be evaluated before optimizing the model. This, for example, includes formulas for features F4 and F7, as well as formulas for the related indicators, LighT and SpeeD in Fig. 5.4. As a result, the generated functions become easier to integrate through those variables, as we explain in Section 5.3, and their complexity as well as their size are smaller than the complexity and size of the first type of functions, which increases the performance of the generated functions (see Chapter 9).

5.2.2 Transformation to Programming Languages

SymPy mathematical expressions can be transformed to many target languages. We selected seven common implementation and simulation languages: Java, JavaScript, Python, C, and C++, R, and Matlab.

We had to extend existing SymPy code printers of these languages to support generic $Max()$ and $Min()$ functions, heavily used in our semantics. We also (surprisingly) had to add a simple Java code printer as none was available for that popular language. Each Java code printer generates a model class with one static method corresponding to the global model function. We also had to fix several small bugs discovered in the Matlab code printer.

For convenience, mathematical functions in different target languages are exported in different files with the appropriate extensions (e.g., Java arithmetic functions in `.java` files). We created a Python module library named MathTo, which can be imported into SymPy files. This MathTo module enables the selection of target languages to export and the generation of corresponding files.

5.2.3 Optimization and Simulation Models

In order to generate optimal strategies that consider stakeholders' preferences and comply with system quality while narrowing the sets of solution (Fig. 4.23), we can use an **optimization model** (Section 2.2.6) that exploits the arithmetic function obtained from the goal model as an objective function, and the arithmetic function from the feature model as a constraint. Tool-wise, we use *IBM ILOG CPLEX Optimization Studio* [82] to implement and solve our optimization model. This optimizer applies mathematical and constraint programming to analyze candidate solutions and select the optimal one(s). It has two optimization engines, CPLEX for linear programming (LR), and Constraint Programming (CP) for integer programming (IP).

We use the *current value* parameter of each GRL indicator (e.g., sensor data) as ancillary variables that describe the environmental context. Then, the optimizer uses the generated values of the decision variables (alternatives) to generate the optimal solution for the given context. The value of the decision variables must be either 100 (selected feature) or 0 (unselected feature). Listing 5.3 highlights the structure of an optimization model. A detailed example will be illustrated in Chapter 6.

Listing 5.3: Structure of a CPLEX mathematical decision model

```
dvar int decision_variables; // Their values are generated by the
    optimizer
float ancillary_variables=..; // Provided as input

// Objective function
maximize Function of GRL model

// Constraints
subject to
{
    Function of Feature model == 100
    decision_variable == 0 || decision_variable == 100
}
```

5.3 Rationale Behind the Combination of Functions

A GRL function selects part of the desired strategies/configurations by disabling or enabling some features according to the given context via indicators (e.g., features F4 and F7 in Fig. 5.3). The rest of the features (not associated with indicators, such as F6 and F5 in Fig. 5.3) are *free decision variables* that the optimizer/program can select/unselect to reach an optimal satisfaction level. As the feature function carries the validation function for each feature/branch, it is used as a constraint on the candidate solutions. Therefore, to integrate the GRL and feature functions, we need to synchronize the values of both free features and features linked to indicators from the GRL functions with the same features

in the feature functions. Indeed, having the same variable names in GRL and feature functions propagates the same feature values to both functions. However, some features (non-free features) are linked to indicators in GRL models and their variables are replaced by the functions/formulas of theses indicators (features linked by the red lines in Fig. 5.3).

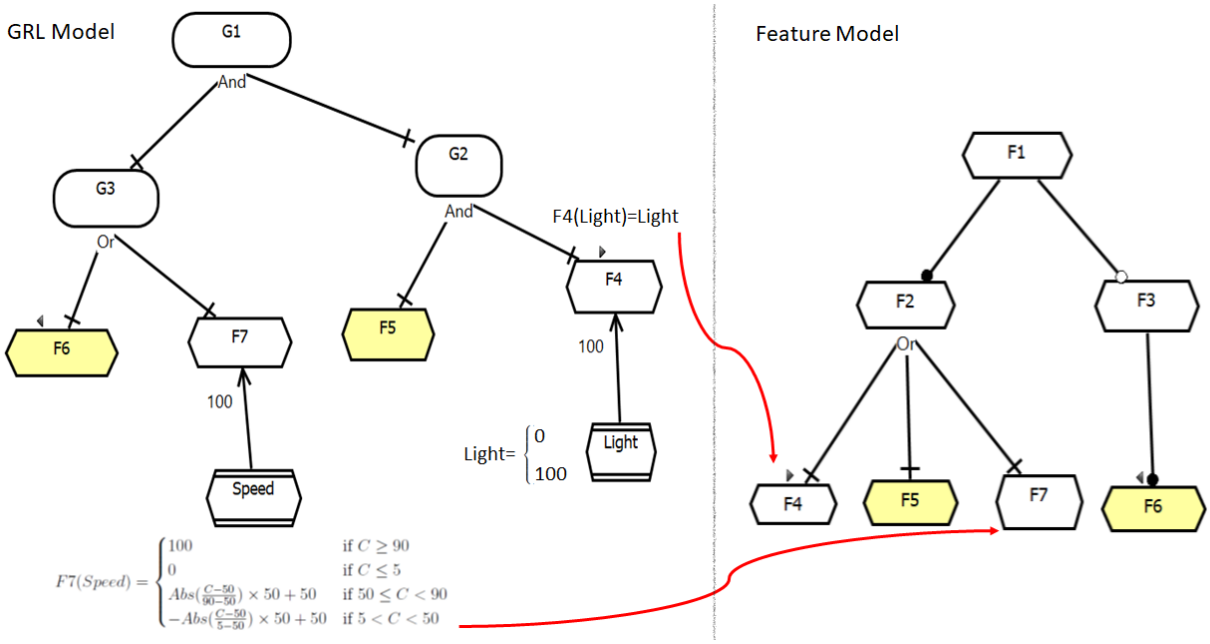

Figure 5.3: Feature integration between GRL and feature models

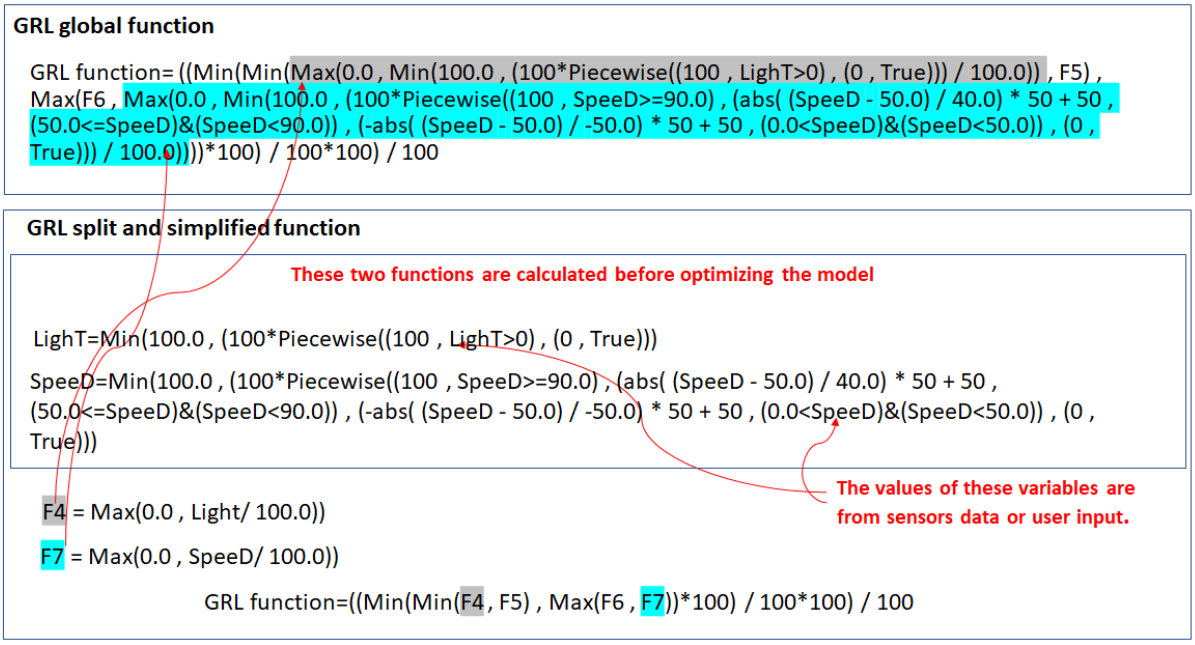

Figure 5.4: Global and simplified functions of the GRL model in Fig. 5.3

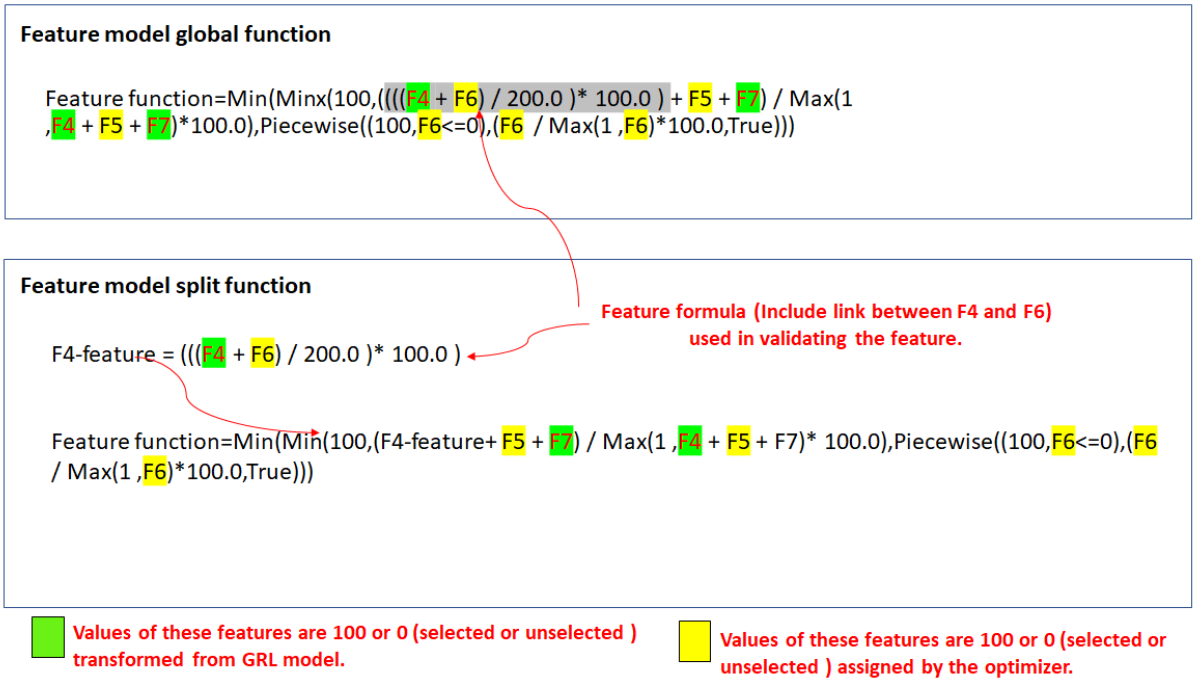

Figure 5.5: Global and split functions of the feature model in Fig. 5.3

5.3.1 Synchronization of the Non-Free Features

In our approach, non-free features are always associated with GRL indicators that determine their values (can be any number) according to the current context at runtime. GRL functions can be evaluated using any parameter values while only 100 or 0 are accepted by feature functions as parameters. To integrate these features, we have to convert their values in the GRL functions to the acceptable values (100 or 0) in the feature functions. This conversion is practicable only by evaluating the GRL formulas of these features before evaluating the two functions as described in the following.

The situation when the two functions are integrated determines how this integration can be done. At **design time**, when covering as many contexts as possible, and when discovering suitable solutions for these contexts, the integration is as follows:

1. If the formulas of the leaf features are not separated from the GRL function, or if they are separated but their values are only 100 or 0, the following formula is recommended:

$$Leaf\_feature = \begin{cases} 100 & \text{If } GRL\_formula() > 0 \\ 0 & \text{Otherwise} \end{cases} \tag{5.1}$$

   where the Leaf_feature variable represents a specific task in the GRL model and the corresponding leaf feature in the feature model. This formula synchronizes the features selected and unselected by the GRL function with the feature function.

Figure 5.4 shows the formula of feature F7 in the GRL model. The following formula is used to convert the same value to feature F7 in the feature model.

$$F7 = \begin{cases} 100 & \text{If } Max(0.0, SpeeD/100.0)) > 0 \\ 0 & \text{Otherwise} \end{cases}$$

The simpler formula $Leaf\_feature = GRL\_formula()$ can instead be used when the values can only be 0 or 100. In Fig. 5.4, the value of feature F4 in GRL model could be 0 or 100 so the formula of feature F4 in feature model is:

$$F4 = Max(0.0, Light/100.0))$$

where the value of the Light indicator is either 0 or 100.

2. If formulas of the features linked to indicators are separated from the main GRL function, and if their values could be any number in [0..100], then different variables for the GRL formula and feature function formula must be used, as follows:

   (a) The following formula is used to represent the leaf features linked to indicators in the GRL function to evaluate the satisfaction value using the real values of indicators.

   $$Leaf\_feature = GRL\_formula() \tag{5.2}$$

   For example, feature F7 in the GRL model (Fig. 5.4) equals:

   $$F7 = Max(0.0, SpeeD/100.0))$$

   (b) The following formula is used to translate the real values of these features to the corresponding values in the feature function. In this case, the names of these features must be changed in the feature function as well when using the optimizer only (the last value will be sent to the optimizer).

   $$Leaf\_featureF = \begin{cases} 100 & \text{If } Leaf\_feature > 0 \\ 0 & \text{Otherwise} \end{cases} \tag{5.3}$$

   If the optimizer is not targeted, the same names can be used but the functions must be reordered. For example, the GRL formulas must be evaluated before the GRL function, and Equation 5.3 must be evaluated before evaluating the main feature function. In the example, F7 in the GRL model (Fig. 5.4) can have any value between 0 and 100 while F7 in feature model (Fig. 5.5) can be only 100 or 0; we hence need to define another variable (F7F) to hold the value of the feature F7 to be used in the feature function:

   $$F7F = \begin{cases} 100 & \text{If } F7 > 0 \\ 0 & \text{Otherwise} \end{cases}$$

The second type of integration is when high reliability at **runtime** is the main objective. In this case, we can calculate the GRL formulas for these features first to check if they are selected or not (100, 0). Then, we validate them before the optimization process. To validate the selected features associated with indicators and resolve any conflicts caused by unforeseen conditions, the corresponding formulas of these features from the feature model have to be calculated before optimizing the model. The integration form is as follows:

1. The most suitable integration from the last mentioned integrations must be used.
2. To validate any feature, its formulas in the feature function must be used before evaluating the two main functions as follows, in order to enable and disable the unwanted features (as Subsection 7.7.1 will further illustrate) between the conflicted features:

$$Leaf\_feature = \begin{cases} 0 & \text{If } Feature\_formula(GRL\_formula()) < 100 \\ 100 & \text{Otherwise} \end{cases} \tag{5.4}$$

Equation 5.4 turns off the invalid features whose values are less than 100. Then, feature and GRL functions are ready to be optimized to get the better solutions using the valid choices of the current context. During the optimization process, the optimizer looks for valid and optimal solutions that include the selected features and exclude the unselected features by selecting and unselecting the features not associated with indicators (free decision variables via GRL and feature functions' values), and then outputs the valid (feature function=100) and optimal configurations (highest GRL function value).

For example, variable F4-feature in Fig. 5.5 validates feature F4 using the related features and the relationships linking them in the feature model ($v(f)$ in Section 4.3.2) while variable F4 equals F4 in the GRL model, which decides whether F4 is selected or not. However, if F4-feature is less than 100, this means the feature F4 is *invalid* (in this case, F6 is not selected while F4 is selected). Instead of returning no solution (feature function < 100), we choose to unselect the invalid feature F4 (F4=0) and issue an optimal strategy suitable for the current context. The formula is:

$$F4 = \begin{cases} 0 & \text{If } F4\_feature < 100 \\ 100 & \text{Otherwise} \end{cases}$$

Finally, by providing the ability to create a single GRL function, separated GRL functions, individual GRL and feature formats for each intentional element, the proposed approach provides a flexible integration method that can be customized by developers according to their needs and creativity.

To get the most benefit from the proposed approach, see the modeling tips in the appendix D.

5.4 Chapter Summary

This chapter presented how the proposed CGS4Adaptation approach is realized using a combination of existing and new tools and techniques.

From a goal-SysML traceability angle, DOORS is used as the RMS where all traceability links are documented and analyzed. jUCMNav, used for GRL modeling, has an existing plug-in that generates the goal model information needed to be imported in the DOORS repository. MI-DSL was used to generate a DXL library so relevant SysML model concepts (from requirements and block definition diagrams) can be exported from a SysML modeling tool and then imported into DOORS.

From an adaptation perspective, a complete transformation from GRL and feature models to SymPy and then to seven programming languages is now available. These functions can be used embedded in simulation models (e.g., in SysML), in running implementations, and in optimization models solved using optimizer tools such as IBM CPLEX.

The next chapter presents several methods for the evaluation of the CGS4Adaptation approach, illustrated with one example, case study and experiment.

Chapter 6

Approach Evaluation: Illustrative Example

This chapter explains, in Section 6.1, the evaluation phases used to assess the effectiveness of the CGS4Adaptation approach. It also examines, in Section 6.2, the feasibility of the approach with an artificial but illustrative proof-of-concept example of a simplified hybrid car system. This example is the first phase of the thesis evaluation.

6.1 Evaluation Phases

We used an iterative process to develop, evaluate, and improve the CGS4Adaptation approach and its supporting tools.

6.1.1 Illustrative Proof-of-Concept Example

We have first used an illustrative example, developed in Section 6.2, as well as test cases to evaluate the feasibility of CGS4Adaptation. These lead, for example, to small modifications in the arithmetic rules describing the semantics of goal and feature models, whose final definition was presented in the previous chapter. Implementing these rules is an additional way to evaluate, justify, and enhance the approach.

The SymPy export plug-in for jUCMNav, whose first version for GRL was developed in collaboration with Yuxuan Fan as part of a Master's project, is freely available online [58]. It now supports the entire GRL/FM semantics discussed in Sections 4.3.1 and 4.3.2. A test suite that covers all GRL model elements and their transformation to the seven target languages is also available online [58]. Similarly, the generated function of the feature model of the illustrative example was validated using test cases that cover all the semantics discussed in Section 4.3.2. Listing A.14 in Appendix A is a Java program that was used to test exhaustively the functions generated from these GRL and feature models.

We have also implemented the optimization and simulation models of the illustrative example and evaluated their feasibility and correctness by testing exhaustively all possible situations according to the monitored conditions, as shown in Subsections 6.2.5 and 6.2.6.

In addition, we implemented and validated the automatic generation of DXL scripts from SysML models (Block definition and Requirement diagrams) for one SysML tool (Cameo Systems Modeler [126]) and one MI-DSL specification. The feasibility and correctness of this part are evaluated using the illustrative example (Subsection 6.2.2).

6.1.2 Case Study

We used a realistic case study, developed in Chapter 7, to evaluate the correctness and usefulness of the all parts of the CGS4Adaptation approach, including functions generation and integration, optimization and simulation models, as well as traceability management based on DXL scripts generation, import mechanisms, and consistency and completeness checks.

We have developed an *IoT-based smart home* case study, based on an initial description from the SM@RT Team [87] at the Informatics Research Institute of Toulouse. A smart home is considering the in-home support services that support elderly and disabled people with the care needed to live independently in the home environment. A smart home is an interesting type of adaptive SCPS often built from scratch and often resulting in repeated design errors [40]. The design and implementation of such a system are indeed complex activities where defining an appropriate architecture that includes software and hardware to satisfy stakeholder goals is a challenge [40, 44].

To meet this challenge, we reflect the requirements provided by the Sma@RT Team in the models that manage a variety of hardware/smart devices whose features satisfy inhabitants' objectives and preferences. However, the design phase is complicated by the fact that each of these devices has its own context and that any change can lead to many actions to choose from. Also, system reliability can be negatively affected by conflicts that may arise due to these multiple possible actions.

Consequently, we have provided an iterative verification method that includes modeling the solutions and optimizing the models using specific scenarios (and also all possible scenarios, as a comparison point) so that the submitted design meets the stakeholders' goals (Chapter 7). The case study demonstrates the usefulness of the automatic generation of arithmetic functions and their conversion to programming languages, as well as the integration of these functions in an optimization model to facilitate the iterative process, reduce design errors, and increase system reliability. Also, this case study suggests the correctness and the usefulness of the automatic DXL generation and import processes to support the traceability management process through complex models in one environment, while saving time and effort for developers (Chapter 8).

6.1.3 Consistency and Completeness Checks

Chapter 8 implements and evaluates the import of GRL+FM and SysML models in DOORS described in Section 5.1, as well as the rules for checking consistency and completeness between the models that were proposed in Section 4.2. This contributes to the evaluation of the feasibility, correctness, and usefulness of these mechanisms in supporting traceability management at design time.

6.1.4 Performance and Scalability of Optimizations

The optimization and simulation models are used to validate the created GRL, FM, and SysML models, and to find useful architectures and predefined adaptation solutions for a given set of contexts. Chapter 9 further evaluates the performance and scalability of the optimization method, as optimizations are known to be computationally expensive.

6.1.5 Other Research Areas

Our method for generating arithmetic functions from GRL models was recently used in other contexts by other researchers, which demonstrates the relevance, usefulness, and generality of this part the CGS4Adaptation approach. For example:

- Alwidian [13] generated arithmetic functions for more than 200 GRL models, and generalized the conversion method to support groups of merged GRL models called *union models*. The generated functions were used in an optimization model to show the effectiveness of her approach.
- In a *process mining* context, Ghasemi and Amyot [66] used goal models to guide the selection of useful traces in event logs that satisfy predefined goals. In that context, they used arithmetic functions generated from goal models in calculating satisfaction values to be used in this selection process.

6.2 Proof-of-Concept Example: Simplified Hybrid Car System

To illustrate our traceability management and function generation approaches, we use the GRL, FM, and SysML models of a simplified hybrid car example system initially described in the SysML standard [132], with its related user goals. This example was selected because the conflicts between its stakeholder goals (e.g., comfortable driving for the user, and acceleration for the system) require the software to control and manage the engine's overall performance [89]. We select the following models for illustrating the main aspects of the CGS4Adaptation approach: traceability management, automatic transformations (into mathematical functions and code), optimization, simulation, and runtime adaptation.

6.2.1 Goal, Feature, and SysML Models

GRL Model Figure 6.1 is a GRL model describing the concerns of the System and its User (with short names, for simplicity). The overall goal of the system is to support car driving (Drive), which is decomposed into four sub-goals that aim to provide Power transportation, Power source management, Acceleration, and Control. In this simplified model, each of these four goals can be realized by two tasks, which are not mutually exclusive here:

- The power transportation goal uses Pipes, Wires, or both.
- The power source management goal manages the two sources Fuel and Battery.
- The acceleration goal can be satisfied by the use of an Electric engine, a Fuel engine, or a combination of the two.
- The control goal aims to regulate the car's speed (Manage speed) and the distance between the car and surrounding objects (Manage distance, informed by the Distance indicator).

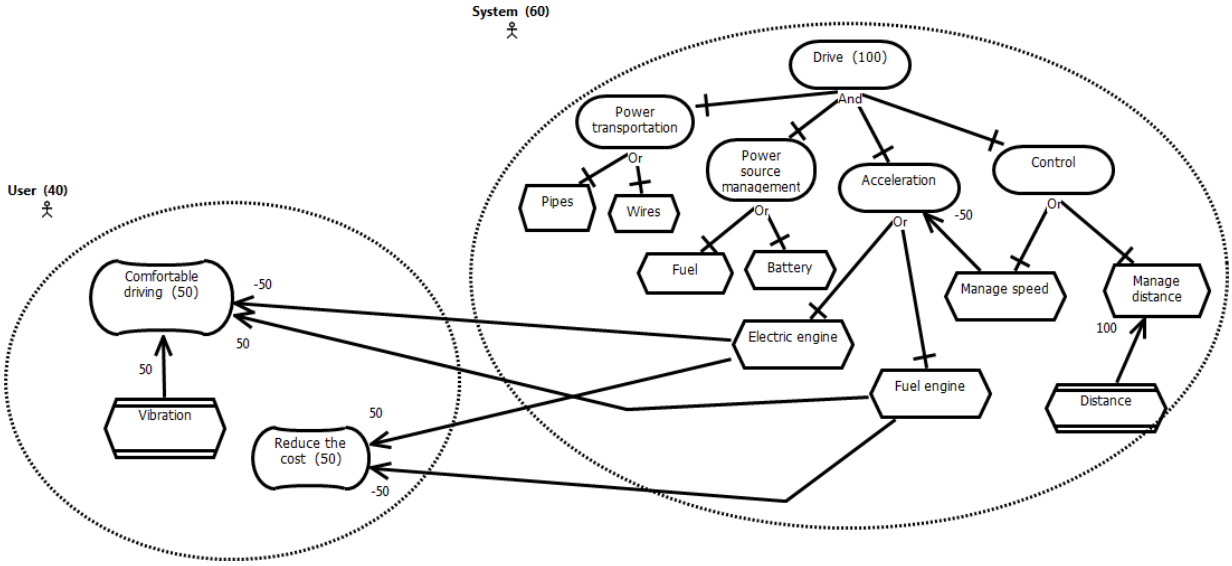

Figure 6.1: GRL model of the simplified hybrid car system example.

User concerns are represented by softgoals (i.e., Comfortable driving and Reduce the cost), of equal importance. The system monitors its environment using sensors, modeled as indicators, to measure Distance and car Vibration. These sensors identify problem symptoms using target, threshold, and worst value parameters (see Table 6.1). Other indicators could be defined (e.g., vibration can also be measured according to frequency and phase) but these two are sufficient to illustrate the approach. Positive and negative contributions of different weights were identified between some of the intentional elements; for this illustrative example, these weights are assumed to be correct (but could be different in practice).

Table 6.1: Indicator parameter values

Indicator	Target	Threshold	Worst	Unit
Distance	25	10	5	Meter
Vibration	0	10	20	Decimetre per second squared

Calculating the overall model satisfaction enables deciding whether to change the task selection in the system (i.e., to *adapt*). If yes, the system selects the combination of tasks that satisfies the driving goal while maximizing driving comfort and minimizing costs, depending on observed environmental conditions.

Feature Model Figure 6.2 illustrates the feature view of the model of the hybrid car example. As can be seen from Figures 6.1 and 6.2, the goal model and the feature model provide complementary views, in the sense that the former provides common features (Pipes, Wires, Fuel, Battery, Fuel engine, Electric engine, Manage distance) with different relationships (e.g, OR between Pipes and Wires in the goal model), which are not provided by the feature model, as well as the contribution relationships of the tasks that hold the impact of each feature on the overall system performance. The leaf features in the feature model represent the alternatives from which the system can select tasks, and their relationships in both the GRL model and the feature model represent the constraints over the selection process.

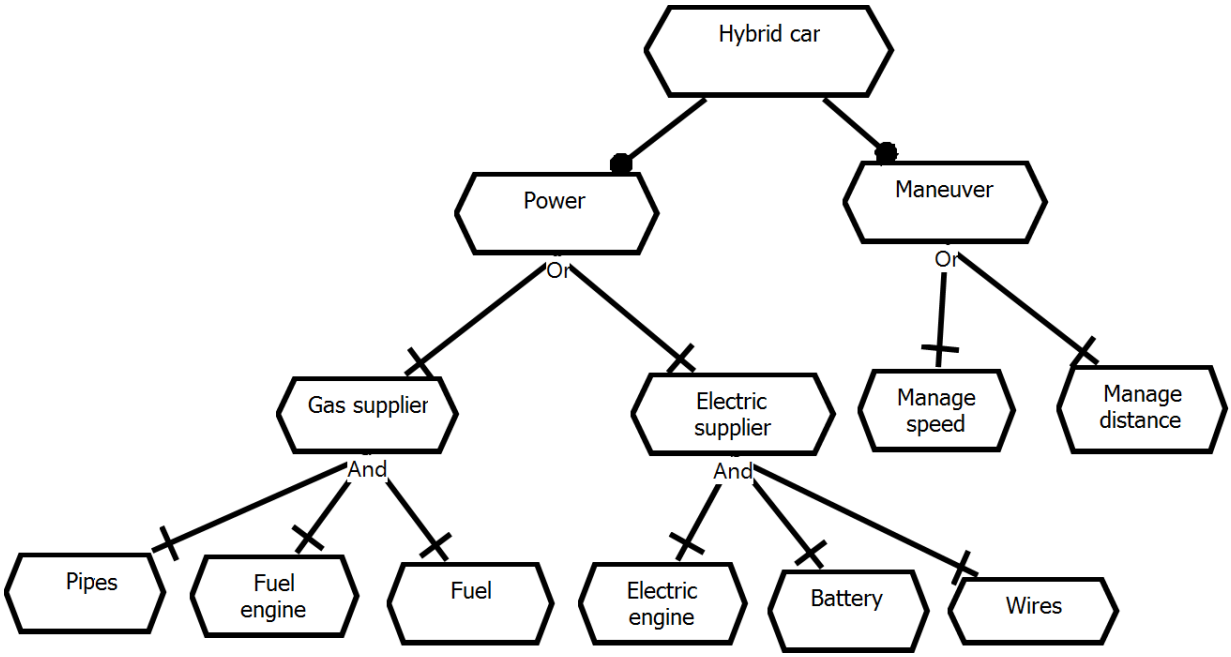

Figure 6.2: Feature model of a simple hybrid car system example.

Our approach also aims to manage traceability between the elements of the GRL model and those of the SysML model, the latter being represented by block definition and requirement diagrams.

SysML Requirement Diagram Figure 6.3 illustrates the requirement diagram of the example. The diagram shows a compound requirement (Driving) and the related subrequirements (Acceleration, Dynamic control, Power sources). The diagram also shows the derived requirements and design elements (blocks) that satisfy them, such as Vibration indicator, Electric power consumption, and Distance indicator. Table 6.2 displays the corresponding requirements table where the identifier, name, and body text of each requirement are described.

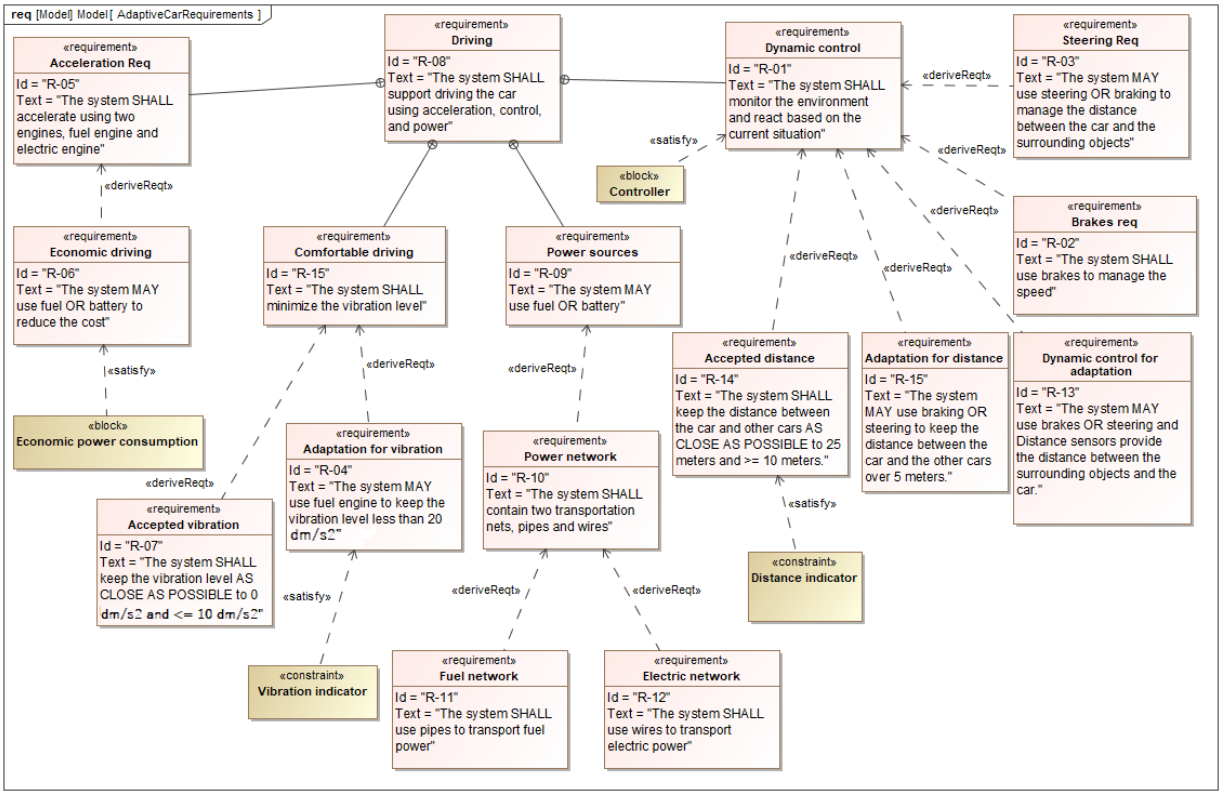

Figure 6.3: Requirement diagram of the simplified hybrid car example.

SysML Block Definition Diagram The structure of the hybrid car system is shown in a block definition diagram (Figure 6.4), where the block hierarchy, associations, dependencies, and decompositions are described. The diagram shows different levels of decomposition (e.g., AdaptiveCar contains a PowerSubsystem) and of generalization (Distance indicator and Vibration indicator are Indicators).

6.2.2 Managing Traceability at Design Time

In order to import the SysML model of the hybrid car system into the DOORS database, the DXL script shown in Listing A.1 was generated automatically using the Report Wizard of the Cameo System Modeler tool and the report template we developed in Listing C.1. This script is then run in Rational DOORS to import the SysML model. The DXL library

Table 6.2: Requirements table of the simplified hybrid car example.

Id	Name	Text
R-01	Dynamic control	The system SHALL monitor the environment and react based on the current situation
R-02	Brakes	The system SHALL use brakes to manage the speed
R-03	Steering	The system MAY use steering OR braking to manage the distance between the car and the surrounding objects
R-04	Adaptation for vibration	The system MAY use fuel engine to keep the vibration level less than 20 dm/s^2
R-05	Acceleration	The system SHALL accelerate using two engines, fuel engine and electric engine
R-06	Economic driving	The system MAY use fuel OR battery to reduce the cost
R-07	Accepted vibration	The system SHALL keep the vibration level AS CLOSE AS POSSIBLE to 0 dm/s^2 and <= 10 dm/s^2
R-08	Driving	The system SHALL support driving the car using acceleration, control, and power.
R-9	Power sources	The system MAY use fuel OR battery
R-10	Power network	The system SHALL contain two transportation nets, pipes and wires
R-11	Fuel network	The system SHALL use pipes to transport fuel power
R-12	Electric network	The system SHALL use wires to transport electric power
R-13	Dynamic control for adaptation	The system MAY use brakes OR steering and Distance sensors provide the distance between the surrounding objects and the car.
R-14	Accepted distance	The system SHALL keep the distance between the car and other cars AS CLOSE AS POSSIBLE to 25 meters and >= 10 meters.
R-15	Adaptation for distance	The system MAY use braking OR steering to keep the distance between the car and the other cars over 5 meters.
R-16	Comfortable driving	The system SHALL minimize the vibration level

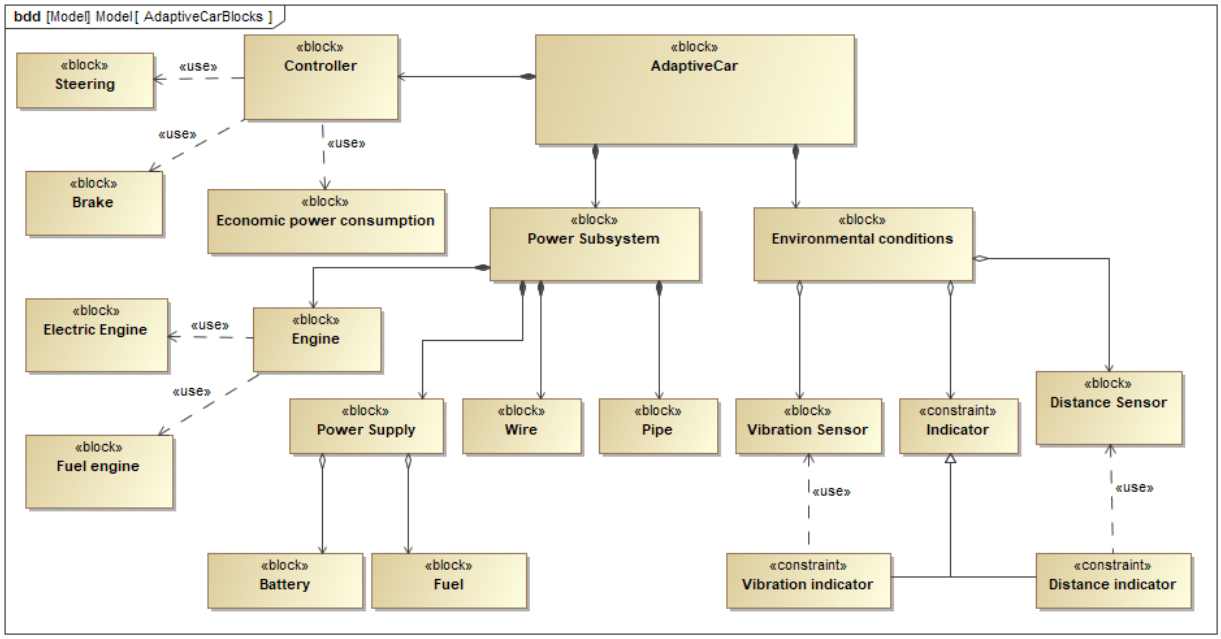

Figure 6.4: Block definition diagram of the simplified hybrid car system example.

for SysML (Section 5.1.1) is invoked by the first line in the script. This script describes two SysML diagrams: requirement and block definition diagrams. Each of them has its ID, Name, Description, Picture file and Title. The requirement diagram has 16 requirements and 18 relationships while The block definition diagram has 20 blocks with 22 relationships.

Similarly, a DXL script was produced for the GRL model using jUCMNav, and was run against the existing DXL library that was developed by Rahman [139]. As this only reuses existing tools to import GRL models into the DOORS database, the scripts are not included in this thesis.

Results

Six formal DOORS modules (RequirementsDiagrams, Requirements, Requirements Relationships, BlockDefinitionDiagram, Blocks, and Block Relationships) and eight link modules (Derived, ContainedBy, Satisfy, Association, Aggregation, Dependency, Composition, and Generalization) result from importing the two diagrams in the DOORS database, as shown in Fig. 6.5. This indeed corresponds to the traceability information model specified in MI-DSL in Listing 5.1. Figures 6.6 to 6.10 further detail the results.

Figure 6.6 shows the imported GRL diagram of the hybrid car example.

Figure 6.7 shows the SysML imported model in DOORS and its diagrams in the BlockDefinitionDiagrams and RequirementsDiagrams modules.

The contents of the Blocks and Requirements modules are displayed in Fig. 6.8. The Blocks module contains 20 block objects with attributes while the Requirements module contains 16 requirement objects with attributes. The small yellow triangles show the

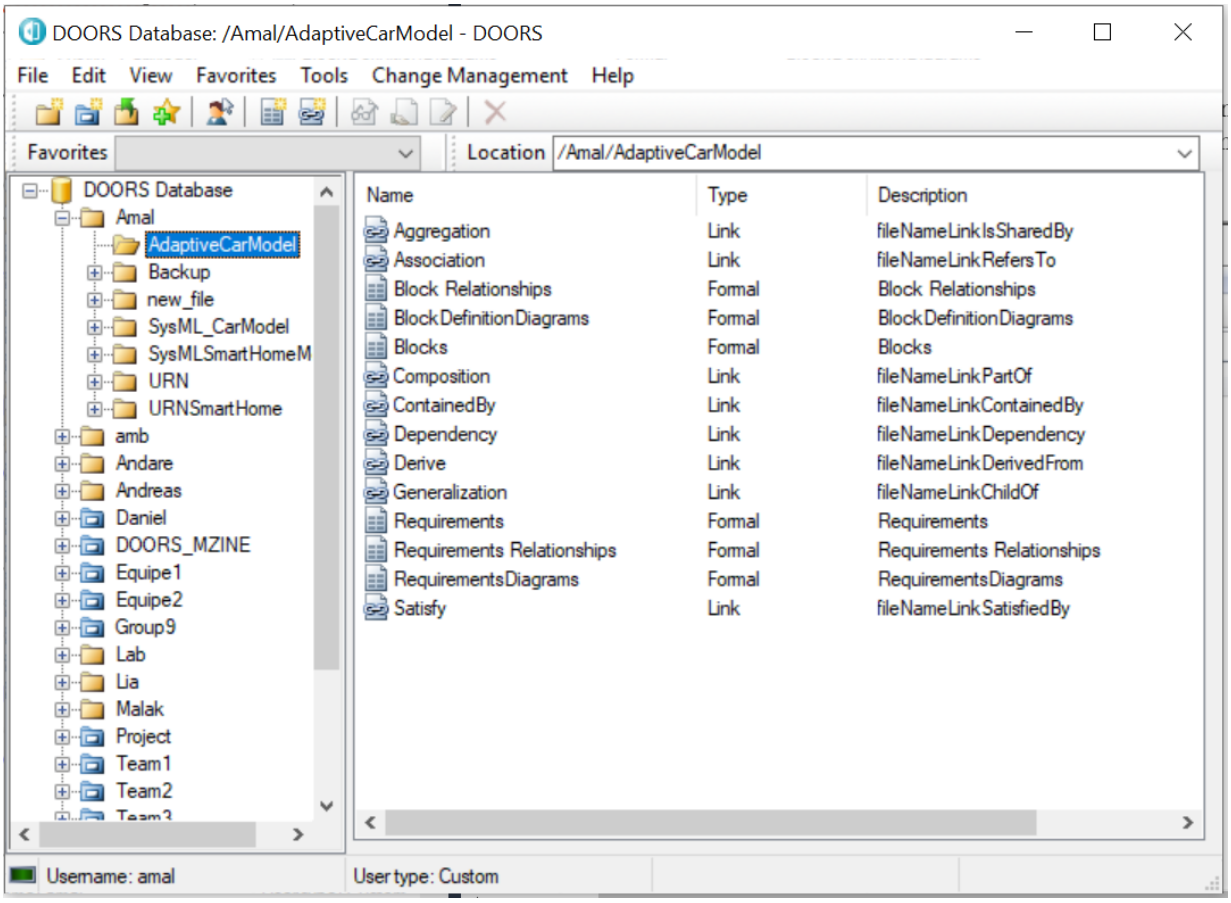

Figure 6.5: SysML model imported in DOORS.

presence of typed traceability links that are coming in or going out of each requirement or block.

Figure 6.9 shows the Satisfy link module, with a focus on the "From requirements relationships to blocks" linkset, which appears in the window's top bar. The screenshot shows the *Controller* block and the *Dynamic control* requirement satisfied by this block.

At this point, only links between GRL model elements and between SysML model elements, which come from the models themselves, exist in the DOORS database. In order to assess consistency and support several analyses involving the goal model and the SysML design model, links between the GRL model elements and the SysML model elements must be created manually. These links are of types defined with linksets in DOORS (e.g., Satisfy between GRL intentional elements and SysML blocks), as shown in Fig. 6.10. As shown in the figure, a new *Satisfy* link was created between the Control goal and the Controller block.

This example demonstrates the technical feasibility of the first part of the CGS4Adaptation approach: traceability management between GRL and SysML models using existing tools and techniques.

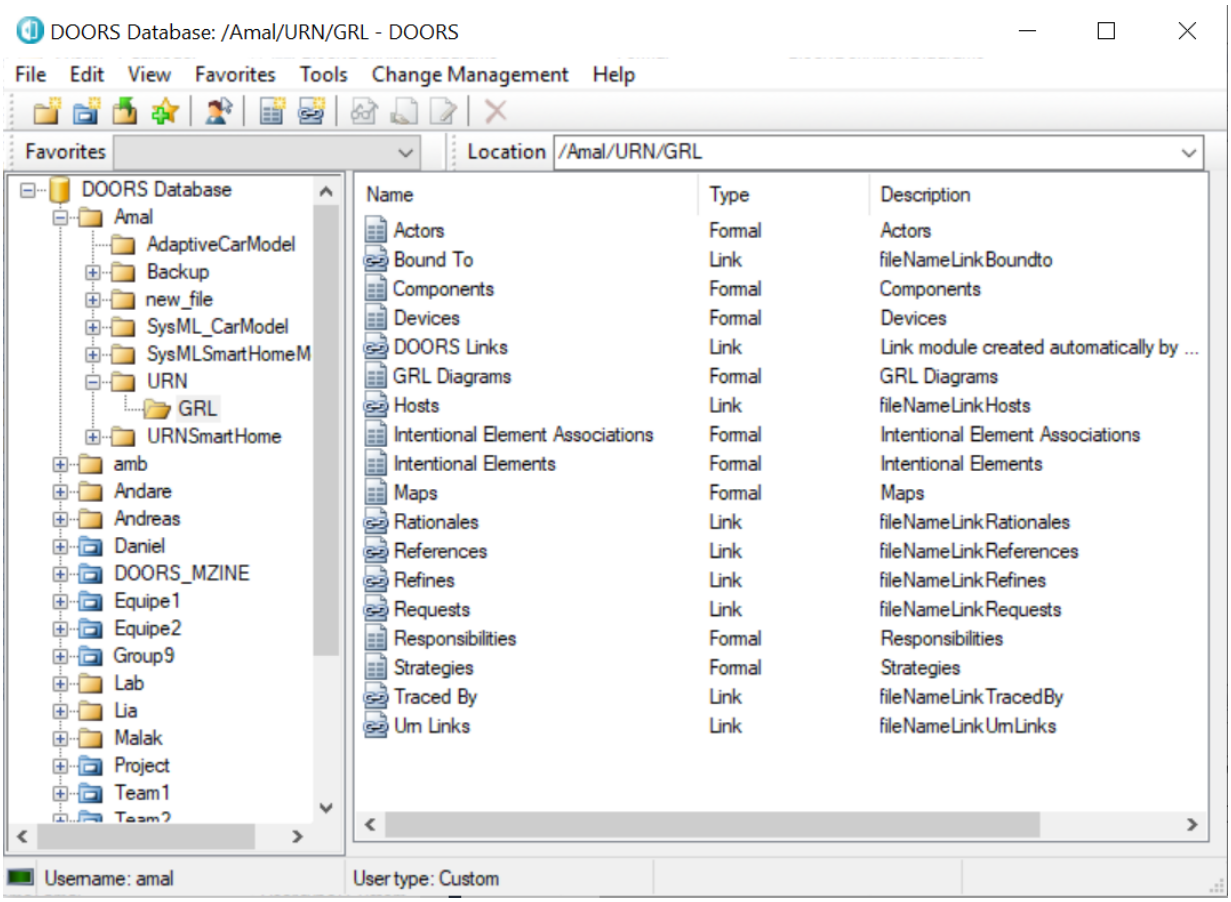

Figure 6.6: GRL model imported in DOORS.

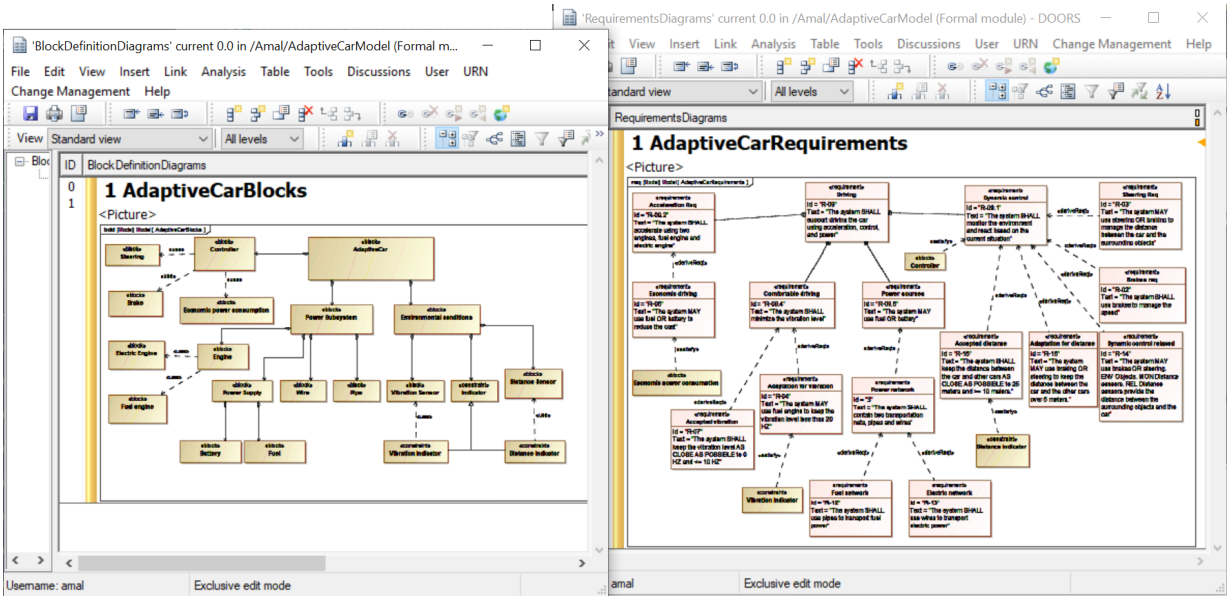

Figure 6.7: Requirement and Block diagram names, titles, and diagrams imported in DOORS.

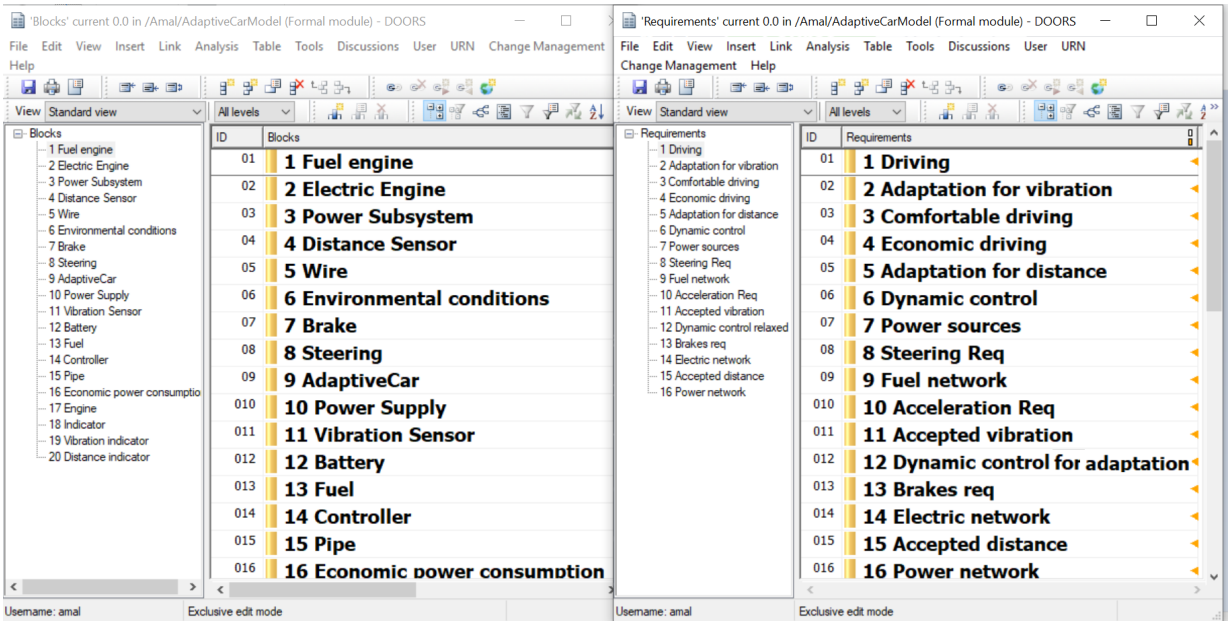

Figure 6.8: Blocks and Requirements modules imported in DOORS.

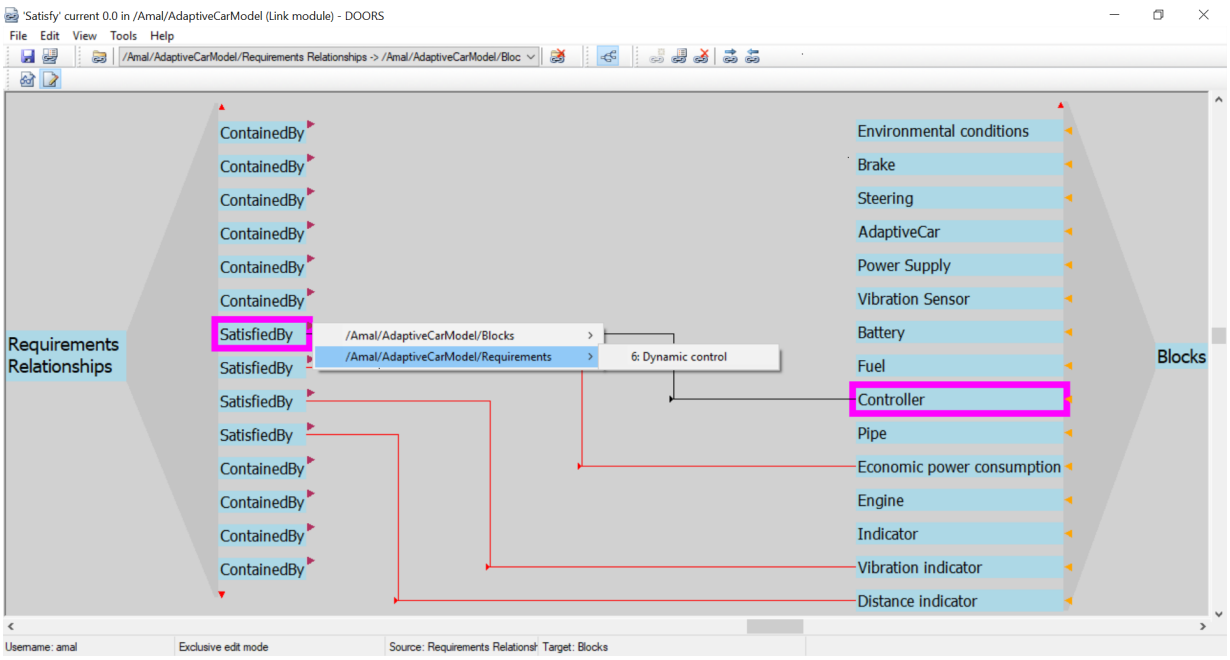

Figure 6.9: *Satisfy* links between SysML blocks and requirements.

6.2.3 Automatic Generation of Sympy Functions for the GRL/Feature Model

To transform the GRL/feature models of the simplified hybrid car example to arithmetic functions, we invoked the SymPy export mechanism in jUCMNav, newly developed in this thesis. A file named SimplifiedadaptiveCar.py was generated automatically. Listing 6.1

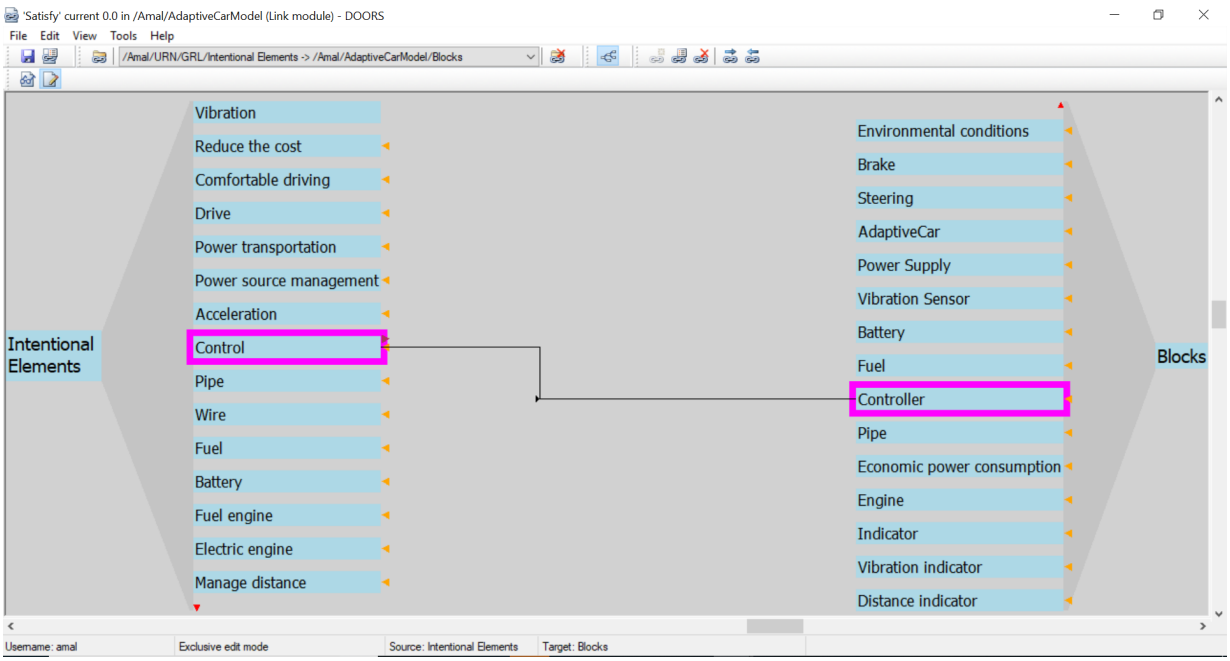

Figure 6.10: New *Satisfy* link between a SysML block and a GRL goal in the DOORS database.

displays the Python/SymPy code of the models. Note that the tool renames the variables to avoid name clashes. It also embeds translation code that can be used to invoke language-specific SymPy code printers. When operations on known constants are involved (e.g., $T - TH$ in Eq. 4.5), then the pre-computed result is directly exported by jUCMNav, for higher execution efficiency.

Listing 6.1: Python/SymPy code generated in SimplifiedadaptiveCar.py

```
from MathTo import *
from sympy import *
import sys
import os

# Creating a folder using Model name
ModelName= 'SimplifiedadaptivecaR'
if not os.path.exists(ModelName):
    os.makedirs(ModelName)
os.chdir(ModelName)

# GRL GOAL MODEL
#Initalize all the variables
Reduce_the_cosT = Symbol('Reduce_the_cosT')
Comfortable_drivinG = Symbol('Comfortable_drivinG')
DrivE = Symbol('DrivE')
ControL = Symbol('ControL')
AccelerationN = Symbol('AccelerationN')
Fuel_enginE = Symbol('Fuel_enginE')
Electric_enginE = Symbol('Electric_enginE')
```

```
Manage_distancE = Symbol('Manage_distancE')
Manage_speeD = Symbol('Manage_speeD')
VibratioN = Symbol('VibratioN')
DistancE = Symbol('DistancE')
Power_source_managemenT = Symbol('Power_source_managemenT')
BatterY = Symbol('BatterY')
FueL = Symbol('FueL')
Power_transportatioN = Symbol('Power_transportatioN')
PipeS = Symbol('PipeS')
WireS = Symbol('WireS')
Reduce_the_cosT = Max(0.0 , Min(100.0 , (-50*Fuel_enginE+50*
    Electric_enginE) / 100.0))
Comfortable_drivinG = Max(0.0 , Min(100.0 , (50*Piecewise((100 ,
    VibratioN<=0.0) , (abs( (VibratioN - 10.0) / 10.0) * 50 + 50 , (0.0<
    VibratioN)&(VibratioN<=10.0)) , (-abs( (VibratioN - 10.0) / -10.0) *
    50 + 50 , (10.0<VibratioN)&(VibratioN<20.0)) , (0 , True))+-50*
    Electric_enginE+50*Fuel_enginE) / 100.0))
DrivE = Min(Min(Max(0.0 , Min(100.0 , (-50*Manage_speeD+Max(Fuel_enginE ,
     Electric_enginE)*100.0) / 100.0)),Max(Max(0.0 , Min(100.0 , (100*
    Piecewise((100 , DistancE>=25.0) , (abs( (DistancE - 10.0) / 15.0) *
    50 + 50 , (10.0<=DistancE)&(DistancE<25.0)) , (-abs( (DistancE - 10.0)
     / -5.0) * 50 + 50 , (5.0<DistancE)&(DistancE<10.0)) , (0 , True))) /
    100.0)) , Manage_speeD)),Min(Max(PipeS , WireS),Max(FueL , BatterY)))
ControL = Max(Max(0.0 , Min(100.0 , (100*Piecewise((100 , DistancE>=25.0)
     , (abs( (DistancE - 10.0) / 15.0) * 50 + 50 , (10.0<=DistancE)&(
    DistancE<25.0)) , (-abs( (DistancE - 10.0) / -5.0) * 50 + 50 , (5.0<
    DistancE)&(DistancE<10.0)) , (0 , True))) / 100.0)) , Manage_speeD)
AcceleratioN = Max(0.0 , Min(100.0 , (-50*Manage_speeD+Max(Fuel_enginE ,
    Electric_enginE)*100.0) / 100.0))
Manage_distancE = Max(0.0 , Min(100.0 , (100*Piecewise((100 , DistancE
    >=25.0) , (abs( (DistancE - 10.0) / 15.0) * 50 + 50 , (10.0<=DistancE)
    &(DistancE<25.0)) , (-abs( (DistancE - 10.0) / -5.0) * 50 + 50 , (5.0<
    DistancE)&(DistancE<10.0)) , (0 , True))) / 100.0))
Power_source_managemenT = Max(FueL , BatterY)
Power_transportatioN = Max(PipeS , WireS)

#Actor function
SysteM = (Min(Min(Max(0.0 , Min(100.0 , (-50*Manage_speeD+Max(Fuel_enginE
     , Electric_enginE)*100.0) / 100.0)),Max(Max(0.0 , Min(100.0 , (100*
    Piecewise((100 , DistancE>=25.0) , (abs( (DistancE - 10.0) / 15.0) *
    50 + 50 , (10.0<=DistancE)&(DistancE<25.0)) , (-abs( (DistancE - 10.0)
     / -5.0) * 50 + 50 , (5.0<DistancE)&(DistancE<10.0)) , (0 , True))) /
    100.0)) , Manage_speeD)),Min(Max(PipeS , WireS),Max(FueL , BatterY)))
    *100.0) / 100
#Actor function
UseR = (Max(0.0 , Min(100.0 , (-50*Fuel_enginE+50*Electric_enginE) /
    100.0))*50.0+Max(0.0 , Min(100.0 , (50*Piecewise((100 , VibratioN
    <=0.0) , (abs( (VibratioN - 10.0) / 10.0) * 50 + 50 , (0.0<VibratioN)
    &(VibratioN<=10.0)) , (-abs( (VibratioN - 10.0) / -10.0) * 50 + 50 ,
    (10.0<VibratioN)&(VibratioN<20.0)) , (0 , True))+-50*Electric_enginE
    +50*Fuel_enginE) / 100.0))*50.0) / 100

#The function of Model
SimplifiedadaptivecaR = ((Min(Min(Max(0.0 , Min(100.0 , (-50*Manage_speeD
```

```
    +Max(Fuel_enginE , Electric_enginE)*100.0) / 100.0)),Max(Max(0.0 , Min
    (100.0 , (100*Piecewise((100 , DistancE>=25.0) , (abs( (DistancE -
    10.0) / 15.0) * 50 + 50 , (10.0<=DistancE)&(DistancE<25.0)) , (-abs( (
    DistancE - 10.0) / -5.0) * 50 + 50 , (5.0<DistancE)&(DistancE<10.0)) ,
     (0 , True))) / 100.0)) , Manage_speeD)),Min(Max(PipeS , WireS),Max(
    FueL , BatterY)))*100.0) / 100*60+(Max(0.0 , Min(100.0 , (-50*
    Fuel_enginE+50*Electric_enginE) / 100.0))*50.0+Max(0.0 , Min(100.0 ,
    (50*Piecewise((100 , VibratioN<=0.0) , (abs( (VibratioN - 10.0) /
    10.0) * 50 + 50 , (0.0<VibratioN)&(VibratioN<=10.0)) , (-abs( (
    VibratioN - 10.0) / -10.0) * 50 + 50 , (10.0<VibratioN)&(VibratioN
    <20.0)) , (0 , True))+-50*Electric_enginE+50*Fuel_enginE) / 100.0))
    *50.0) / 100*40) / 100
GRLDiagramName = 'GrladaptivecaR'

#Variable list
List = ['Manage_speeD','VibratioN','FueL','WireS','Electric_enginE','
    DistancE','BatterY','PipeS','Fuel_enginE']
LANG = ''
langList = ['python','c','c++','java","javascript","matlab','r','cp']
def allPrint():
  for j in langList:
    LANG = str(j)
    Translate('((Min(Min(Max(0.0 , Min(100.0 , (-50*Manage_speeD+Max(
        Fuel_enginE , Electric_enginE)*100.0) / 100.0)),Max(Max(0.0 , Min
        (100.0 , (100*Piecewise((100 , DistancE>=25.0) , (abs( (DistancE -
         10.0) / 15.0) * 50 + 50 , (10.0<=DistancE)&(DistancE<25.0)) , (-
        abs( (DistancE - 10.0) / -5.0) * 50 + 50 , (5.0<DistancE)&(
        DistancE<10.0)) , (0 , True))) / 100.0)) , Manage_speeD)),Min(Max(
        PipeS , WireS),Max(FueL , BatterY)))*100.0) / 100*60+(Max(0.0 ,
        Min(100.0 , (-50*Fuel_enginE+50*Electric_enginE) / 100.0))*50.0+
        Max(0.0 , Min(100.0 , (50*Piecewise((100 , VibratioN<=0.0) , (abs(
         (VibratioN - 10.0) / 10.0) * 50 + 50 , (0.0<VibratioN)&(VibratioN
        <=10.0)) , (-abs( (VibratioN - 10.0) / -10.0) * 50 + 50 , (10.0<
        VibratioN)&(VibratioN<20.0)) , (0 , True))+-50*Electric_enginE+50*
        Fuel_enginE) / 100.0))*50.0) / 100*40) / 100' , GRLDiagramName ,
        List , LANG)
if(len(sys.argv)==1):
  allPrint()
else:
  for i in sys.argv:
    if(sys.argv.index(i)==0):continue
    if  (i.lower() not in langList):
      for j in langList:
        LANG = str(j)
        allPrint()
    else:
      print 'in'
      LANG = str(i.lower())
      Translate('((Min(Min(Max(0.0 , Min(100.0 , (-50*Manage_speeD+Max(
          Fuel_enginE , Electric_enginE)*100.0) / 100.0)),Max(Max(0.0 ,
          Min(100.0 , (100*Piecewise((100 , DistancE>=25.0) , (abs( (
          DistancE - 10.0) / 15.0) * 50 + 50 , (10.0<=DistancE)&(DistancE
          <25.0)) , (-abs( (DistancE - 10.0) / -5.0) * 50 + 50 , (5.0<
          DistancE)&(DistancE<10.0)) , (0 , True))) / 100.0)) ,
```

```
            Manage_speeD)),Min(Max(PipeS , WireS),Max(FueL , BatterY)))
            *100.0) / 100*60+(Max(0.0 , Min(100.0 , (-50*Fuel_enginE+50*
            Electric_enginE) / 100.0))*50.0+Max(0.0 , Min(100.0 , (50*
            Piecewise((100 , VibratioN<=0.0) , (abs( (VibratioN - 10.0) /
            10.0) * 50 + 50 , (0.0<VibratioN)&(VibratioN<=10.0)) , (-abs( (
            VibratioN - 10.0) / -10.0) * 50 + 50 , (10.0<VibratioN)&(
            VibratioN<20.0)) , (0 , True))+-50*Electric_enginE+50*
            Fuel_enginE) / 100.0))*50.0) / 100*40) / 100' , GRLDiagramName ,
             List , LANG)

# FEATURE MODEL
#Initalize all the variables
Fuel_enginE= Symbol('Fuel_enginE')
Electric_enginE= Symbol('Electric_enginE')
Manage_distancE= Symbol('Manage_distancE')
Manage_speeD= Symbol('Manage_speeD')
Hybrid_caR= Symbol('Hybrid_caR')
ManeuveR= Symbol('ManeuveR')
Electric_supplieR= Symbol('Electric_supplieR')
BatterY= Symbol('BatterY')
FueL= Symbol('FueL')
PoweR= Symbol('PoweR')
Gas_supplieR= Symbol('Gas_supplieR')
PipeS= Symbol('PipeS')
WireS= Symbol('WireS')
Hybrid_caR= Min((Manage_speeD + Manage_distancE) / Max(1 ,Manage_speeD +
    Manage_distancE)*100.0,(Min(Electric_enginE,Min(WireS,BatterY)) + Min(
    FueL,Min(PipeS,Fuel_enginE))) / Max(1 ,Max(Electric_enginE,Max(WireS,
    BatterY)) + Max(FueL,Max(PipeS,Fuel_enginE)))*100.0)
ManeuveR= Min(100,(Manage_speeD + Manage_distancE) / Max(1 ,Manage_speeD
    + Manage_distancE)*100.0)
Electric_supplieR= Min(Electric_enginE,Min(WireS,BatterY))
PoweR= Min(100,(Min(Electric_enginE,Min(WireS,BatterY)) + Min(FueL,Min(
    PipeS,Fuel_enginE))) / Max(1 ,Max(Electric_enginE,Max(WireS,BatterY))
    + Max(FueL,Max(PipeS,Fuel_enginE)))*100.0)
Gas_supplieR= Min(FueL,Min(PipeS,Fuel_enginE))
#The function of Model
AdaptivecarF= Min(Min(100,(Manage_speeD + Manage_distancE) / Max(1 ,
    Manage_speeD + Manage_distancE)*100.0),Min(100,(Min(Electric_enginE,
    Min(WireS,BatterY)) + Min(FueL,Min(PipeS,Fuel_enginE))) / Max(1 ,Max(
    Electric_enginE,Max(WireS,BatterY)) + Max(FueL,Max(PipeS,Fuel_enginE))
    )*100.0))
FMDiagramName =  'AdaptivecarF'

#Variable list
List= ['Manage_speeD','FueL','Manage_distancE','WireS','Electric_enginE',
    'BatterY','PipeS','Fuel_enginE']
LANG = ''
langList = ['python','c','c++','java","javascript","matlab','r','cp']
def allPrint():
  for j in langList:
    LANG = str(j)
    Translate('Min(Min(100,(Manage_speeD + Manage_distancE) / Max(1 ,
        Manage_speeD + Manage_distancE)*100.0),Min(100,(Min(
```

```
        Electric_enginE,Min(WireS,BatterY)) + Min(FueL,Min(PipeS,
        Fuel_enginE))) / Max(1 ,Max(Electric_enginE,Max(WireS,BatterY)) +
        Max(FueL,Max(PipeS,Fuel_enginE)))*100.0))',FMDiagramName,List,LANG
        )
if(len(sys.argv)==1):
  allPrint()
else:
  for i in sys.argv:
    if(sys.argv.index(i)==0):continue
    if  (i.lower() not in langList):
      for j in langList:
        LANG = str(j)
        allPrint()
    else:
      print 'in'
      LANG = str(i.lower())
      Translate('Min(Min(100,(Manage_speeD + Manage_distancE) / Max(1 ,
          Manage_speeD + Manage_distancE)*100.0),Min(100,(Min(
          Electric_enginE,Min(WireS,BatterY)) + Min(FueL,Min(PipeS,
          Fuel_enginE))) / Max(1 ,Max(Electric_enginE,Max(WireS,BatterY))
          + Max(FueL,Max(PipeS,Fuel_enginE)))*100.0))',FMDiagramName,List,
          LANG)
```

Using the model name, the embedded code creates a folder that includes two other folders named by the involved diagrams' names (GRL and feature diagrams) to hold the generated files of the targeted programming languages for each diagram. Figure 6.11 shows the structure of the generated files, where *SimplifiedadaptivecaR* is the model name while *GrladaptivecaR* and *AdaptivecarF* are the names of GRL and feature diagrams, respectively.

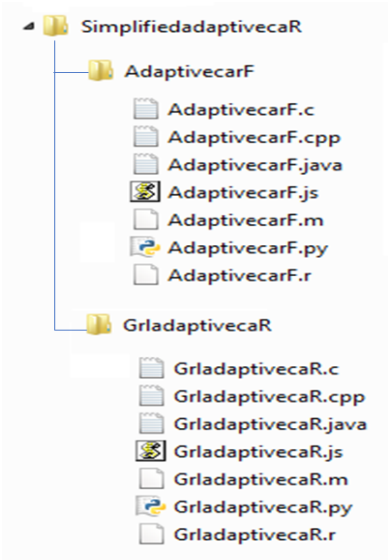

Figure 6.11: Structure of the generated files.

The rest of this section explains how the rules presented in Chapter 4 were used by the mechanism to generate the mathematical functions of GRL and feature models of the example (Listing 6.1).

Generation of the Mathematical Function of Example Goal Model: To transform the GRL model of the simplified hybrid car example (Figure 6.2) into a mathematical function, the proposed mechanism uses the transformation rules for GRL models defined in Section 4.3.1 as follows.

Based on Eq. 4.11, the global mathematical function of the model becomes:

$$v(SAdaptiveCar) = \frac{(v(User) \times 40 + v(System) \times 60)}{100} \tag{6.1}$$

where (from Equations 4.3 and 4.10 applied to the User actor):

$$\begin{aligned} v(User) = & \Big(Max\big(0.0, Min(100.0, \\ & (-50 \times Fuel\_engine + 25 \times Electric\_engine)/100.0)\big) \times 50.0 \\ & + Max\big(0.0, Min(100.0, (50 \times v(Vibration) + \\ & (-25 \times Electric\_engine) + (50 \times Fuel\_engine))/100.0)\big) \\ & \times 50.0\Big)/100 \end{aligned} \tag{6.2}$$

and (from Eq. 4.7 because the target value of the Vibration indicator is less than its worst value):

$$v(Vibration) = \begin{cases} 100 & \text{if } C \leq 0 \\ 0 & \text{if } C \geq 20 \\ Abs(\frac{C-10}{10-0}) \times 50 + 50 & \text{if } 0 < C \leq 10 \\ -Abs(\frac{C-10}{10-20}) \times 50 + 50 & \text{if } 10 < C < 20 \end{cases} \tag{6.3}$$

and where (from Equations 4.1, 4.2, and 4.3 applied to the System actor):

$$\begin{aligned} v(System) = & v(Drive) \\ = & Min\Big(Max\big(Max(0.0, Min(100.0, v(Distance))), Manage\_speed\big), \\ & Max(Pipes, Wires), Max(Fuel, Battery), \\ & Max\big(0.0, Min(100.0, (-50 \times Manage\_speed + \\ & Max(Fuel\_engine, Electric\_engine) \times 100.0)/100.0)\big)\Big) \end{aligned} \tag{6.4}$$

and finally (from Eq. 4.5 because the target value of the Distance indicator is higher than its worst value):

$$v(Distance) = \begin{cases} 100 & \text{if } C \geq 25 \\ 0 & \text{if } C \leq 5 \\ Abs(\frac{C-10}{25-10}) \times 50 + 50 & \text{if } 10 \leq C < 25 \\ -Abs(\frac{C-10}{5-10}) \times 50 + 50 & \text{if } 5 < C < 10 \end{cases} \tag{6.5}$$

Generation of the Mathematical Function of Example Feature Model To transform the feature model of the simplified hybrid car example (Figure 6.2) into a mathematical function, the proposed mechanism uses the transformation rules for feature models defined in Section 4.3.2 as follows.

Based on Eq. 4.21 (multiple links), the overall arithmetic function of the feature part of the model is:

$$v(Hybrid\_car) = Min(v(Power), v(Maneuver)) \tag{6.6}$$

By applying Eq. 4.18 (OR-decomposition):

$$v(Maneuver) = Min\Big(100, Max\big(0, \frac{v(Manage\_speed) + v(Manage\_distance)}{s(Manage\_speed) + s(Manage\_distance)}\big) \times 100\Big) \tag{6.7}$$

By applying Eq. 4.13, because $Manage\_speed$ and $Manage\_distance$ are leaf features:

$$v(Manage\_speed) = s(Manage\_speed) = Manage\_speed \tag{6.8}$$

$$v(Manage\_distance) = s(Manage\_distance) = Manage\_distance \tag{6.9}$$

Then:

$$v(Maneuver) = Min\Big(100, Max\big(0, \frac{Manage\_speed + Manage\_distance}{Manage\_speed + Manage\_distance}\big) \times 100\Big) \tag{6.10}$$

By applying Eq. 4.18 (OR-decomposition):

$$v(power) = Min\Big(100, Max\big(0, \frac{v(Gas\_supplier) + v(Electric\_supplier)}{s(Gas\_supplier) + s(Electric\_supplier)}\big) \times 100\Big) \tag{6.11}$$

By applying Eq. 4.17 (AND-decomposition):

$$v(Gas\_supplier) = Min\big(v(Pipes), v(Fuel\_engine), v(Fuel)\big) \tag{6.12}$$

and

$$v(Electric\_supplier) = Min\big(v(Battery), v(Electric\_engine), v(Wires)\big) \tag{6.13}$$

$s(f)$ functions are calculated using Eq. 4.13 as follows:

$$s(Gas\_supplier) = Max(Pipes, Fuel\_engine, Fuel) \tag{6.14}$$

$$s(Electric\_supplier) = Max(Battery, Electric\_engine, Wires) \tag{6.15}$$

By propagating the $v(Fuel\_supplier)$ and $v(Electric\_supplier)$ to the next level, $v(Power)$ becomes:

$$\begin{aligned} v(Power) = \\ &Min\Big(100, \big(Min(Pipes, Fuel\_engine, Fuel) + \\ &\quad Min(Battery, Electric\_engine, Wires)\big) \\ &\quad /(Max(1, Max(Pipes, Fuel\_engine, Fuel) \\ &\quad + Max(Battery, Electric\_engine, Wires))\big) \\ &\times 100\Big) \end{aligned} \tag{6.16}$$

By propagating $v(Power)$ and $v(Maneuver)$ to the next level, the overall function becomes:

$$
\begin{aligned}
v(Hybrid\_car) = \\
&Min\Big(Min\big(100, Min(Pipes, Fuel\_engine, Fuel) + \\
&\quad Min(Battery, Electric\_engine, Wires) \\
&\quad /(Max(1, Max(Pipes, Fuel\_engine, Fuel) \\
&\qquad + Max(Battery, Electric\_engine, Wires))) \times 100\big), \\
&\quad Min\big(100, (Manage\_speed + Manage\_distance)/ \\
&\quad Max(1, (Manage\_speed + Manage\_distance)) \times 100\big)\Big)
\end{aligned}
\tag{6.17}
$$

To illustrate how the function of the feature model is used to validate the generated strategies, we assume the selected features of the generated strategy to be [Fuel, Fuel engine, Battery, Electric engine, Wires, Manage speed, Manage distance].

The value of a selected feature is 100 while it is 0 for an unselected one. In this strategy, only the Pipes feature is unselected, so its value is 0 while the value of the other features is 100. By assigning those values to the related variables of the mathematical function of the illustrative example (Equation 6.17), the result becomes:

$$
\begin{aligned}
v(Hybrid\_car) = \\
&Min\Big(Min\big(100, Min(0, 100, 100) + \\
&\qquad Min(100, 100, 100)/(Max(1, 100 + 100)) \times 100\big), \\
&\quad Min\big(100, 200/200 \times 100\big)\Big) \\
&= Min\big((100/200) \times 100, 100\big) \\
&= Min(50, 100) \\
&= 50
\end{aligned}
\tag{6.18}
$$

The strategy is hence **invalid** because the overall result is *less than 100*. Although the Gas supplier feature is involved with the Electric supplier in an OR-decomposition, and the value of the Electric supplier is 100, the solution is still considered an invalid one because the Gas supplier cannot be used without Pipes. For instance, if the system is looking for a solution to a situation caused by defective pipes, executing this strategy would burn the car if allowed to be carried out.

6.2.4 Multi-Language Transformation of Goal/Feature Functions

By executing the SimplifiedadaptiveCar.py file (Listing 6.1) using Python, the functions corresponding to the GRL and feature views of the model can be translated to Java, JavaScript, Python, C, C++, R, or Matlab. As an example, Listing 6.2 shows the code generated for Java (GRLadaptivecaR.java) whereas Listing 6.3 shows the Java function of the feature model (AdaptivecarF.java). Note that only the leaf intentional elements from

the GRL/feature model are left as method parameters (i.e., the parameters of the GRL function are Vibration, Distance, Manage Speed, Fuel Engine, Fuel, Pipes, Electric Engine, Battery, and Wires); this is the information that must be provided upon invocation. Note also that SymPy symbolically simplified constant expressions. For example, the 40/100 and 60/100 kept as is in the SymPy code were converted in Java to 0.4 and 0.6, respectively.

Appendix A includes other examples of generated code, where there are some small rounding differences in the real numbers due to the symbolic processing done by SymPy.

Listing 6.2: Java function for the GRL model in GRLadaptivecaR.java

```
public class Model {
    public double GrladaptivecaR(double Manage_speeD, double VibratioN,
        double FueL, double WireS, double Electric_enginE, double DistancE
        , double BatterY, double PipeS, double Fuel_enginE) {
        double expr = 0.4 * Math.max(0, Math.min(100.0, 0.5 *
            Electric_enginE - 0.5 * Fuel_enginE)) + 0.4 * Math.max(0, Math
            .min(100.0, -0.5 * Electric_enginE + 0.5 * Fuel_enginE + 0.5 *
             ((VibratioN <= 0.0) ? (
                100.0
            ) :
            ((VibratioN <= 10.0 && VibratioN > 0.0) ? (
                    50.0 * Math.abs(0.1 * VibratioN - 1.0) + 50.0
                ) :
                ((VibratioN > 10.0 && VibratioN < 20.0) ? (-50.0 * Math.
                    abs(0.1 * VibratioN - 1.0) + 50.0) :
                    (
                        0
                    ))))) + 0.6 * Math.min(Math.max(BatterY, FueL), Math
                        .max(PipeS, WireS), Math.max(0, Math.min(100.0,
                        -0.5 * Manage_speeD + 1.0 * Math.max(
                        Electric_enginE, Fuel_enginE), Math.max(
                        Manage_speeD, 1.0 * ((DistancE >= 25.0) ? (
                100.0
            ) :
            ((DistancE >= 10.0) ? (
                    50.0 * Math.abs(0.0666666666666667 * DistancE -
                        0.666666666666667) + 50.0
                ) :
                ((DistancE > 5.0) ? (-50.0 * Math.abs(0.2 * DistancE -
                    2.0) + 50.0) :
                    (
                        0
                    ))))))));
        return expr;
    }
}
```

Listing 6.3: Java function for the feature model in AdaptivecarF.java

```
public class Model {
```

```
    public double AdaptivecarF(double Manage_speeD, double FueL, double
        Manage_distancE, double WireS, double Electric_enginE, double
        BatterY, double PipeS, double Fuel_enginE) {
        double expr = Math.min(Math.min(100, 100.0 * (Manage_distancE +
            Manage_speeD) / Max(1.0, Manage_distancE + Manage_speeD)),
            Math.min(100, 100.0 * (Math.min(BatterY, Math.min(
            Electric_enginE, WireS)) + Math.min(FueL, Math.min(Fuel_enginE
            , PipeS))) / Math.max(1.000, Math.max(BatterY, Math.max(
            Electric_enginE, WireS)) + Math.max(FueL, Math.max(Fuel_enginE
            , PipeS)))));
        return expr;
    }
}
```

6.2.5 Optimization Model

The mathematical functions generated from GRL and feature models can be composed and used in multiple contexts, including *simulations* (where different values are provided to the function parameters) that are performed standalone or within the context of another model (e.g., in SysML), and *optimizations* where a solver or optimizer tool is used to find values for the function parameters that will maximize or minimize some aspects of these functions. Both simulations and optimizations have a role in the development of adaptive SCPSs.

In the hybrid car engine example, the parameters of the Java method generated for the feature model (Manage Distance, Manage Speed, Fuel Engine, Fuel, Pipes, Electric Engine, Battery, and Wires) are the leaf features of the feature model and the only information that needs to be passed through method invocation. For the method generated from the GRL model, the current environmental conditions (monitored by the Distance and Vibration indicators) are additional parameters through which the function is able to monitor the environment. This enables the detection of violations at design time (e.g., via simulation) and at runtime (e.g., in a system implementation).

Optimization can be useful to find the best valid adaptation strategies in common environmental conditions, i.e., in foreseen contexts (defined by the indicators) that are frequent and likely to be observed. The generated functions can be supplied with all combinations of the possible values (0 and 100) of their parameters, and with specific indicator values (e.g., the four environmental situations listed in Table 6.3 in our example). The arithmetic functions for system actors were embedded in the test code in case suitable strategies for individual actors are needed. The function of the feature model alone can be satisfied by the 9 feasible configurations listed in Table 6.3 (i.e., no other configurations is feasible according to the feature model), whereas the goal model function generated 81 possible strategies, including the 9 from the feature models. Hence, by combining the feature model with the goal model, 72 goal-based strategies (89% here) can be ignored. Note that having the 9 configurations from the feature model all covered by the initial 81 goal model strategies is a coincidence here; the two sets of solutions could be partially overlapping or be disjoint in other models.

We sorted the strategies by their satisfaction values of the GRL function, and we combined them with their satisfaction values of the mathematical function of each system actor (System and User). Table 6.3 also displays the output of the test where only feasible strategies (MAPE plans) were considered as candidate solutions.

Figure 6.12 shows the performance of the feasible strategies according to the environmental conditions that are listed in Table 6.3. Strategy 1 is better than strategies 2 and 3 because its satisfaction level for each of the above violation contexts (colored bars) is higher or equal than those of strategies 2 and 3, Similarly, strategies 4 and 5 are equivalent and each is better than strategies 6, 7, 8, and 9. Hence, strategies 1 and 4 (or 1 and 5) are suitable and valid solutions that cover all the tested environmental contexts, with different trade-offs.

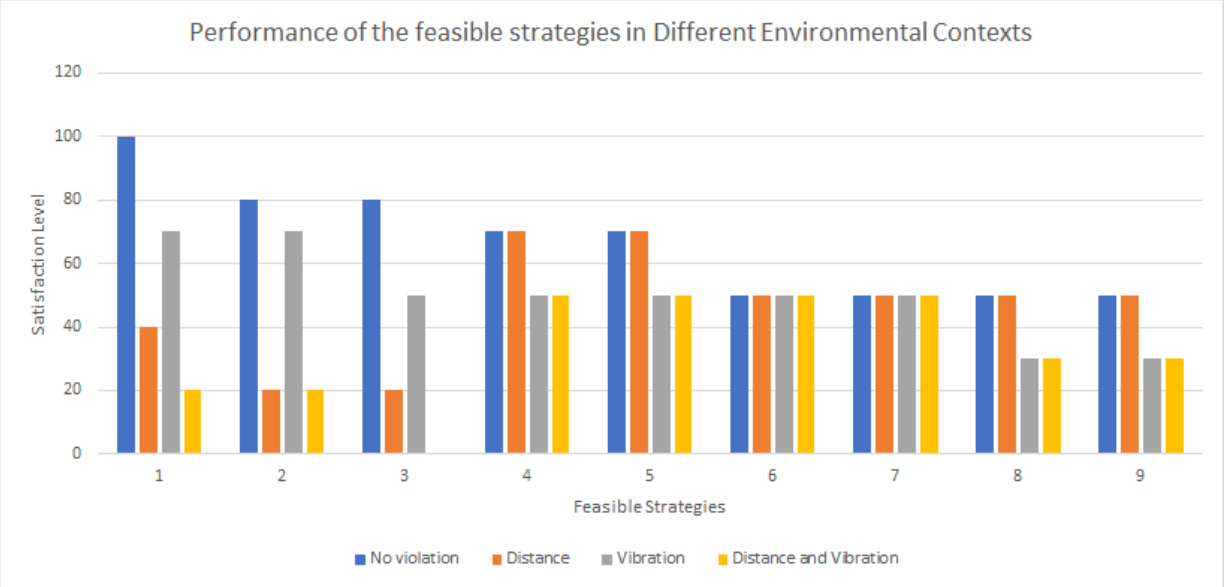

Figure 6.12: Performance of a simple hybrid car's strategies.

However, evaluating all feasible solutions for a wide range of contexts is impractical for realistic models due to the large number of possible configurations, leading to a time-consuming and costly simulation process. Optimization models can help here select optimal strategies for each context [49, 53, 71, 120]. For this reason, an optimization model is built based on the goal/feature functions and then solved.

Functions generated from goal/feature models can be fed to an optimization engine. To transform the functions into the OPL language [?] used by CPLEX, the C++ version of the functions are used, with a renaming of the `max` and `min` functions to OPL's `maxl` and `minl`. From the function code (with an extract in Listing 6.1), the number of decision variables (n) in the model is 7 while the number of ancillary variables (m) is 2. In the goal model's function, the DistancE parameter is used instead of the Manage distancE variable, whose satisfaction value is not 0/100 but is calculated using the monitored distance between the car and the surrounding objects. In the feature model's function, the variable Manage distancE is passed as a parameter to validate the selected configurations. However,

Table 6.3: Feasible solutions for specific environmental conditions
D=Distance, V=Vibration, Ms=Manage speed, Md=Manage distance, Wrs=Wires, Ee=Electric engine, Btry=Battery, Pps=Pipes, Fe=Fuel engine, Obj=Objective function, Hgt=Highest value, Fsb=Feasible solution, Sys=System, 100=Selected feature, 0=Unselected feature, N=Number, Y=Yes.

D	**V**	**Strategy**									**Obj**	**Hgt**	**Fsb**	**User**	**Sys**
		N	Ms	Fuel	Md	Wrs	Ee	Btry	Pps	Fe					
25	0	1	0	100	100	0	0	0	100	100	100	Y	Y	100	100
		2	0	0	100	100	100	100	0	0	80		Y	50	100
		3	0	100	100	100	100	100	100	100	80		Y	50	100
		4	100	100	0	0	0	0	100	100	70		Y	100	50
		5	100	100	100	0	0	0	100	100	70		Y	100	50
		6	100	0	0	100	100	100	0	0	50		Y	50	50
		7	100	0	100	100	100	100	0	0	50		Y	50	50
		8	100	100	0	100	100	100	100	100	50		Y	50	50
		9	100	100	100	100	100	100	100	100	50		Y	50	50
1	0	5	100	100	100	0	0	0	100	100	70	Y	Y	100	50
		4	100	100	0	0	0	0	100	100	70	Y	Y	100	50
		6	100	0	0	100	100	100	0	0	50		Y	50	50
		7	100	0	100	100	100	100	0	0	50		Y	50	50
		8	100	100	0	100	100	100	100	100	50		Y	50	50
		9	100	100	100	100	100	100	100	100	50		Y	50	50
		1	0	100	100	0	0	0	100	100	40		Y	100	0
		3	0	100	100	100	100	100	100	100	20		Y	50	0
		2	0	0	100	100	100	100	0	0	20		Y	50	0
20	30	2	0	0	100	100	100	100	0	0	70	Y	Y	50	83.3
		1	0	100	100	0	0	0	100	100	70	Y	Y	50	83.3
		3	0	100	100	100	100	100	100	100	50		Y	0	83.3
		6	100	0	0	100	100	100	0	0	50		Y	50	50
		7	100	0	100	100	100	100	0	0	50		Y	50	50
		4	100	100	0	0	0	0	100	100	50		Y	50	50
		5	100	100	100	0	0	0	100	100	50		Y	50	50
		8	100	100	0	100	100	100	100	100	30		Y	0	50
		9	100	100	100	100	100	100	100	100	30		Y	0	50
3	25	6	100	0	0	100	100	100	0	0	50	Y	Y	50	50
		7	100	0	100	100	100	100	0	0	50	Y	Y	50	50
		5	100	100	100	0	0	0	100	100	50	Y	Y	50	50
		4	100	100	0	0	0	0	100	100	50	Y	Y	50	50
		8	100	100	0	100	100	100	100	100	30		Y	0	50
		9	100	100	100	100	100	100	100	100	30		Y	0	50
		2	0	0	100	100	100	100	0	0	20		Y	50	0
		1	0	100	100	0	0	0	100	100	20		Y	50	0
		3	0	100	100	100	100	100	100	100	0		Y	0	0

the value of the alternative Manage speeD is used in calculating the satisfaction value if it was selected and if its value (100) is higher than or equal to the current value of the Manage distancE choice. Therefore, by applying the objective equation of Section 4.22, the objective function of the illustrative example becomes:

Maximize
$\mathsf{GS4VS}(x_1, ., x_7, y_1, y_2) = Goal\_function(x_1, ., x_7, y_1, y_2)$
Subject To
$Feature\_function(x_1, ., x_7, y_1) == 100$
$(Manage\_speeD == 0)$ **or** $(Manage\_speeD == 100)$
$(FueL == 0)$ **or** $(FueL == 100)$
$(WireS == 0)$ **or** $(WireS == 100)$
$(Electric\_enginE == 0)$ **or** $(Electric\_enginE == 100)$
$(BatterY == 0)$ **or** $(BatterY == 100)$
$(PipeS == 0)$ **or** $(PipeS == 100)$
$(Fuel\_enginE == 0)$ **or** $(Fuel\_enginE == 100)$

where x is the set of decision variables and y the set of the ancillary variables:

$$x = \{Manage\_speeD, FueL, WireS, Electric\_enginE, BatterY, PipeS, Fuel\_enginE\}$$

$$y = \{DistancE, VibratioN\}$$

Note that Manage_distancE is a function of DistancE and its values are calculated by the environmental context (Distance indicator function). If this value is greater than zero, the related feature/task is still considered as a selected feature while the satisfaction values (the value of the goal function excluding this feature) of its alternatives are less than its own satisfaction value. Also, VibratioN is not passed to the feature function because that indicator is not linked to the feature (see Fig 6.1).

Listing 6.4 shows the optimization model of the hybrid car example using CPLEX's Constraint Programming (CP) optimizer. This optimizer covers discrete integer decision variables and a wide set of mathematical operators, including divisions and multiplications (used in our arithmetic semantics). We can solve the optimization model for different ranges of values for the Distance and Vibration indicators that monitor current environmental conditions. In this case, since the car model monitors only two situations (vibration and distance), we have solved the model first using well-known conditions (to show we can get the same results as those obtained through brute force testing) and then using all possible environmental conditions (to enable dealing with unforeseen situations that may emerge at runtime). The provided environmental conditions (indicators' values) and the corresponding optimal solutions issued by the model using specific conditions are listed in Table 6.4 while Table 6.5 displays all possible contexts using different ranges of values for the Distance and Vibration indicators.

Strategies 1 and 4 together covered all the tested contexts while strategies 2, 5, 6,

Table 6.4: Optimal strategies for specific environmental conditions using the CP Optimizer

Distance	**Vibration**	**Objective function**	**Optimal strategies**	**Violation**
25.0	0.0	100	Manage distance, Fuel,Fuel engine, Pipes	No
1.0	0.0	70	Manage speed, Fuel, Fuel engine, Pipes	Distance
20.0	30.0	70	Manage distance, Battery, Electric engine, Wires	Vibration
3.0	25.0	50	Manage speed, Battery, Electric engine, Wires	Distance and Vibration

Table 6.5: Optimal strategies computed for specific environmental conditions

Distance	**Vibration**	**Objective function**	**Optimal strategies**	**Violation**	
				Distance	**Vibration**
$[25.0, \infty]$	$[0.0, -\infty]$	100	1	No	No
$[25.0, \infty]$	(0.0,10.0]	$[90.0, 100)$	1	No	Partially
$[25.0, \infty]$	(10.0,20.0)	(80,90)	1	No	Partially
$[25.0, \infty]$	$[20.0, \infty]$	80	1, 2	No	Yes
$[0.0, 5.0]$	$[0.0, -\infty]$	70	4	Yes	No
$[0.0, 5.0]$	(0.0,10.0]	$[60.0, 70.0)$	4	Yes	Partially
$[0.0, 5.0]$	(10.0,20.0)	(50.0,60.0)	4	Yes	Partially
$[0.0, 5.0]$	$[20.0, \infty]$	50	4, 6	Yes	Yes
(5.0,10.0)	$[0.0, -\infty]$	70	4, 5	Partially	No
(5.0,10.0)	(0.0,10.0]	$[60.0, 70.0)$	4, 5	Partially	Partially
$(5.0, 10.0)$	$(10.0, 20.0)$	$(50.0, 60.0)$	4, 5	Partially	Partially
$(5.0, 10.0)$	$[20.0, \infty]$	50	4, 5, 6, 7	Partially	Yes
$[10.0, 25.0)$	$[0.0, -\infty]$	$[70.0, 100)$	1	Partially	No
$[10.0, 25.0)$	$(0.0, 10.0]$	$[60.0, 100)$	1	Partially	Partially
$[10.0, 25.0)$	$(10.0, 20.0)$	$(50.0, 90.0)$	1	Partially	Partially
$[10.0, 25.0)$	$[20.0, \infty]$	$(80.0, 50.0]$	1, 2	Partially	Yes

and 7 are alternatives that could be used to reduce the cost of the adaptation process at runtime by assessing their cost and the cost of the transformation process from the current configuration into the most suitable one. For instance, if the current strategy is 2 and the vibration constraint is violated, we do not need to adapt to strategy 1 because both have the same model satisfaction value.

The evaluation of the objective function (column # 3 in Table 6.4) changes according to the current values of the distance and vibration indicators and their related worst, threshold, and target parameters, as defined in the URN standard [**?**]. For example, when the current value of the distance indicator is less than the threshold value (10 m) and the current value of the vibration indicator is greater than its threshold value (10 dm/s^2), the objective function evaluation will range between 50 and 60. The function records the worst performance (50) when the current distance is less than or equal to the worst value and the current vibration is greater than or equal to the worst value. The indicator functions will return different satisfaction values if different parameter values are used (e.g., if one varies the threshold value), but this would result in different models and objective functions. Although the optimization model could possibly handle indicator thresholds as decision parameters (e.g., to tune thresholds), this is not the type of decision targeted in this paper.

In addition to generating the optimal strategies for the predicted contexts, using this method, the weaknesses of the selected configurations can be discovered by examining different environmental conditions while strengthening the design of the developed system by adding/removing specific tasks. For example, the overall performance of the car is currently not fully satisfied ($Satisfaction = 50$) using the selected features when the distance and vibration constraints are violated. Providing such information at design time (within complex and large systems especially) can guide the discovery of new requirements, leading designers in providing suitable features and selecting suitable designs that better satisfy the goals of stakeholders and of the developed system.

Moreover, using real data at runtime with the proposed method, the system can *adapt* to the suitable solutions when facing foreseen and unforeseen contexts. For example, when the distance between the car and the surrounding objects is too close, the system can adapt by choosing another task (Manage speed) to deal with this situation (strategy 4 in Table 6.4). As such model reflects the effects of the features on each other and reduces their side effects [53], it also monitors the impact of environmental changes on the system performance and decides whether to reconfigure or not. Such method can further help designers discover weaknesses and limitations of the selected design under different runtime conditions while narrowing the design space to an acceptable number of suitable solutions (e.g., from 81 strategies in the GRL model to only 2 in our example).

Note that these environment conditions cover all possible contexts according to the indicators' parameters (target, threshold, and worst), which is often not possible when dealing with real-world models. In practice, only the most common contexts should be used to compute predetermined optimal solutions, whereas unforeseen contexts will be a subject to runtime optimization (bounded by a time budget, e.g., 1 second).

Listing 6.4: CPLEX CP Optimizer model of the hybrid car in SimplifiedadaptiveCar.cplex

```
using CP;
// Ancillary variables
float VibratioN=...;
float DistancE=...;
// Decision variables
dvar int Manage_speeD ;
dvar int FueL;
dvar int WireS;
dvar int Electric_enginE;
dvar int BatterY;
dvar int PipeS;
dvar int Fuel_enginE;
// Expressions and related variables
int D1;
int D2 ;
int D3 ;
int Vb2 ;
int Vb1 ;
int Vb0;
execute{
D1=(DistancE>=25.0);
D2=(10.0<=DistancE)&&(DistancE<25.0);
D3=(5.0<DistancE)&&(DistancE<10.0);
Vb2=VibratioN > 10.0 && VibratioN < 20.0;
Vb1=VibratioN <= 10.0 && VibratioN > 0.0;
Vb0= VibratioN <= 0.0;
}
// The environmental condition of the distance sensor
dexpr float Manage_distancE =maxl(0.0 , minl(100.0 , (100*((D1==1) ? 100 : (D2==1) ? (
    abs( (DistancE - 10.0) / 15.0) * 50 + 50) : (D3==1) ? (-abs( (DistancE - 10.0) /
    -5.0) * 50 + 50 ) : 0)) / 100.0)) > 0? 100:0.0;
// Feature model function used as a contraint
dexpr float FeatureModel=minl(minl(100,100*maxl(0.0, (Manage_distancE + Manage_speeD) /
     minl(1.0,(Manage_distancE + Manage_speeD)))), minl(100,100.0*maxl(0, (minl(BatterY
    , minl(Electric_enginE , WireS)) + minl(FueL, minl(Fuel_enginE , PipeS)))/maxl(1.0,
    maxl(BatterY, maxl(Electric_enginE , WireS)) + maxl(FueL, maxl(Fuel_enginE , PipeS)))
    )));
// Overall GRL model function
dexpr float SimplifiedadaptivecaR = ((minl(minl(maxl(0.0 , minl(100.0 , (-50*
    Manage_speeD+maxl(Fuel_enginE , Electric_enginE)*100.0) / 100.0)),maxl(maxl(0.0 ,
    minl(100.0 , (100*((D1==1) ? 100 : (D2==1) ? (abs( (DistancE - 10.0) / 15.0) * 50 +
     50) : (D3==1) ? (-abs( (DistancE - 10.0) / -5.0) * 50 + 50 ) : 0)) / 100.0)) ,
    Manage_speeD)),minl(maxl(PipeS , WireS),maxl(FueL , BatterY)))*100.0) / 100*60+(
    maxl(0.0 , minl(100.0 , (-50*Fuel_enginE+50*Electric_enginE) / 100.0))*100.0+maxl
    (0.0 , minl(100.0 , (50*((Vb0==1) ?100 : (Vb1==1) ? (abs( (VibratioN - 10.0) /
    10.0) * 50 + 50) :  (Vb2==1) ? (-abs( (VibratioN - 10.0) / -10.0) * 50 + 50) : 0)+
    -50*Electric_enginE+50*Fuel_enginE) / 100.0))*100.0) / 100*40) / 100 ;
// The objective function
maximize SimplifiedadaptivecaR;
// Constraints
subject to {
 FeatureModel==100;
 decisionVar:
 Manage_speeD ==0 || Manage_speeD ==100;
 FueL ==0 || FueL ==100 ;
 WireS ==0 || WireS ==100 ;
 Electric_enginE ==0 || Electric_enginE ==100 ;
 BatterY ==0 || BatterY ==100 ;
 PipeS ==0 || PipeS ==100 ;
 Fuel_enginE ==0 || Fuel_enginE ==100 ;
}
```

6.2.6 Simulation Model

To explore the results of different feature and goal models, we extended the hybrid car models with two additional optional features, Light and Air conditioner, as shown in Fig. 6.13. Figure 6.14 presents an additional Provide services goal in the GRL model. This goal is achieved through the two new features.

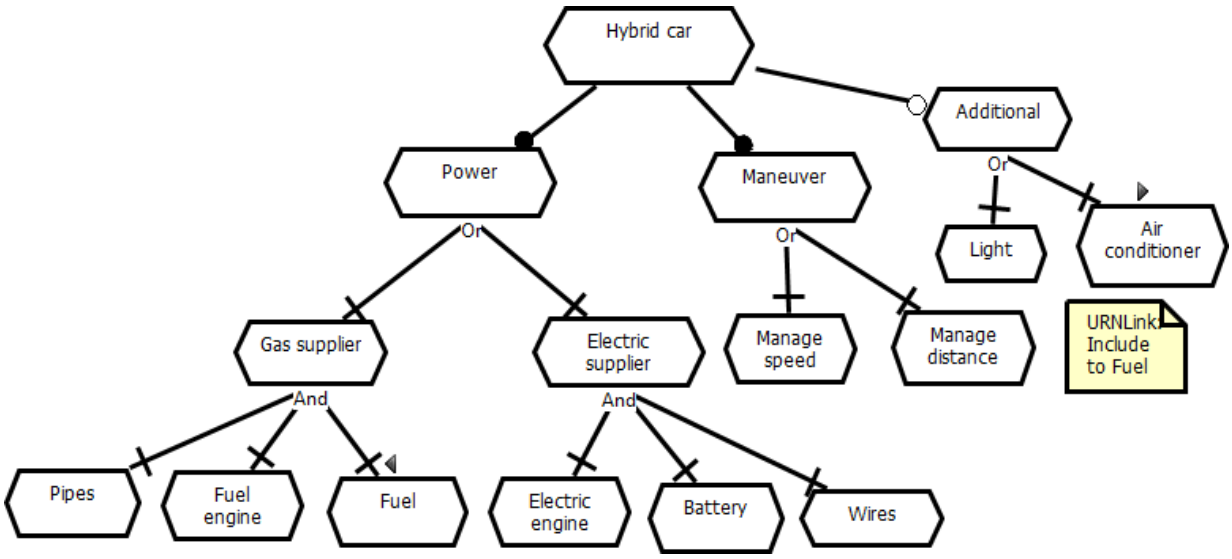

Figure 6.13: Extended feature model of a simplified hybrid car system example.

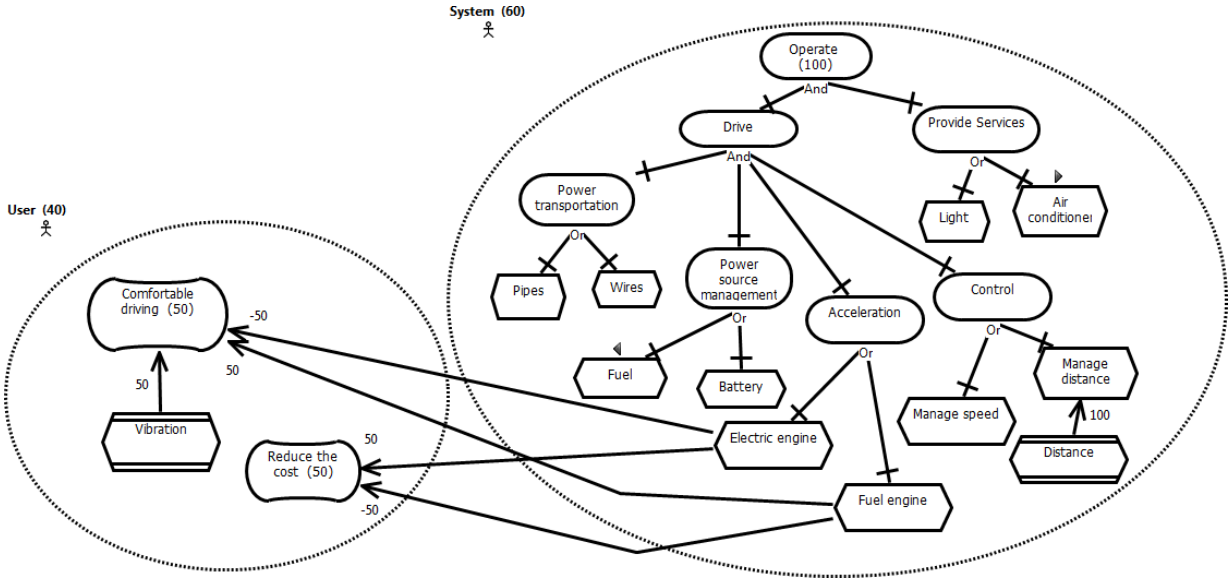

Figure 6.14: Extended GRL model of a simple hybrid car system example.

The arithmetic functions can be used to support CPS design and runtime adaptation activities (MAPE) by embedding executable goal/feature models in SysML models that can be simulated, to mitigate bad surprises when adaptation functions are integrated to the

larger system description. The objective of this section is to demonstrate the feasibility and the usefulness of integrating the generated functions (i.e., executable requirements) with system design in the context of self-adaptation.

Simulation Design: In order to detect design weaknesses as well as to Monitor and Analyze the current context (in MAPE terms), the goal function is used to assess the performance of the current configuration against quality requirements and stakeholders' goals. In MAPE's Plan activity, the embedded strategies generated from the optimization model are used for well-known conditions while the combined function from the goal/feature model is used to generate, on the fly, the best feasible solutions for unknown conditions. We embedded the functions in a SysML model with a parametric diagram integrated to the system's structure (block definition diagram – BDD) and behaviour (state machine).

We used *MagicDraw Cameo Systems Modeler* [126] here as it supports SysML simulations. We first added a BDD that contains the main block (Adaptive car) and related constraint blocks, as shown in Fig. 6.15. The latter are used to embed our goal/feature arithmetic functions and create related parametric diagrams; the one for the goal function is shown in Fig. 6.16 while Fig. 6.17 shows the parametric diagram for the feature function.

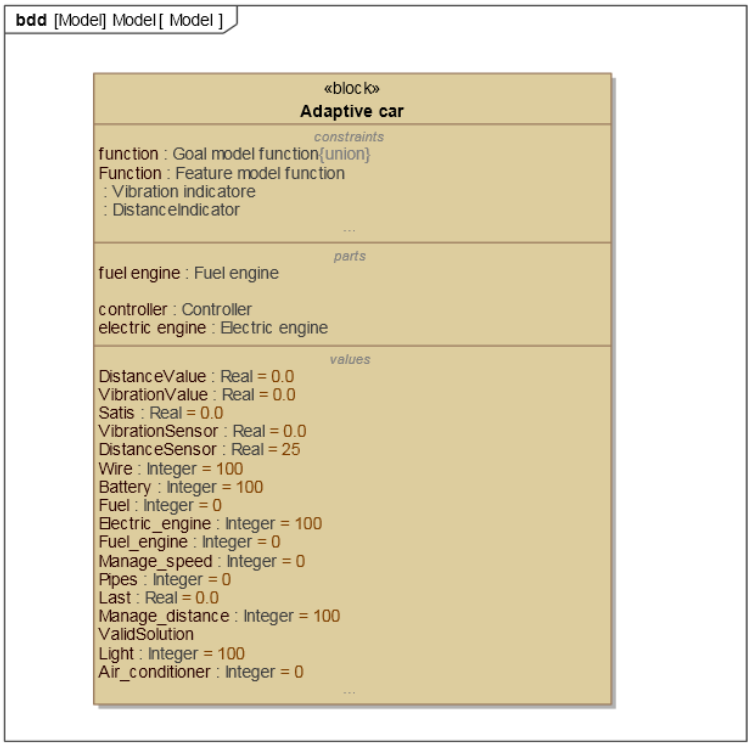

Figure 6.15: SysML block definition diagram of the hybrid car example

Environmental conditions (DistanceSensor and VibrationSensor), together with the alternatives that are represented by the properties of the parent block, supply the constraint blocks with real data through binding connections. The binding connectors also transform

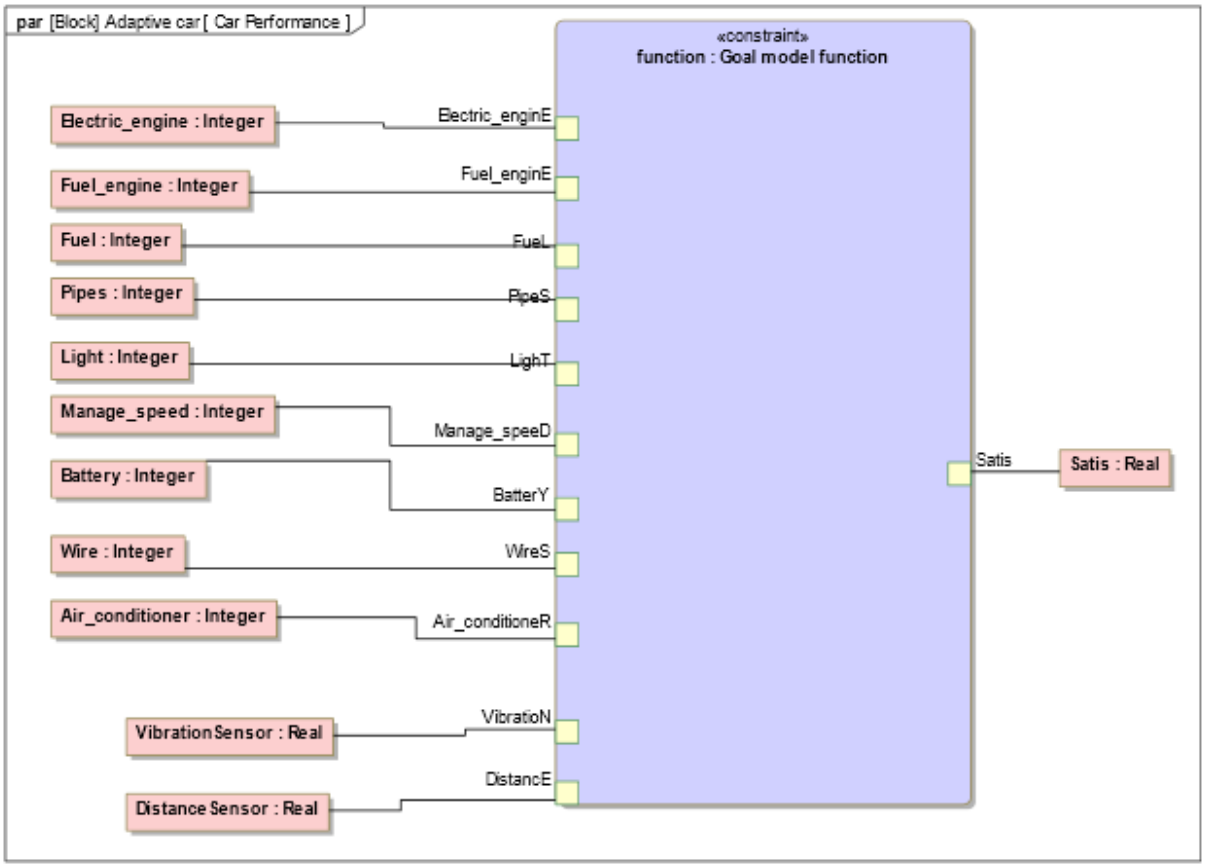

Figure 6.16: SysML parametric diagram of the goal model function for the hybrid car example

the results of evaluating the goal and feature functions to the parent block's properties (Satis and ValidSolution). These two properties are used to control the flow of the state machine diagram (the behavioural classifier of the Adaptive car block). This way, a change in state is a result of different values of the equations that are recalculated automatically according to the different environmental conditions and system variables. This can be done interactively through the tool's GUI (Figure 6.18 and Figure 6.19).

Using the properties of the two functions, we assigned guarding constraints to enable the transitions from one state to the next state, as shown on the transitions between the states (right part of Fig. 6.18). For example, to transit from the Monitoring and Analyze state to the Planning state, the following constraint must be true: $[Satis < 100 \,||\, ValidSolution < 100]$. This means the satisfaction level must be less than 100 (changes in the environmental conditions degraded the system performance) or that the user has selected an invalid configuration (i.e., using the Fuel engine without Gas). The Using Embedded Strategies state has three transitions paths: 1) to the Solver state if $Satis < 100$ and there is time to search other candidate solutions, 2) to the Working state if $Satis == 100$, and 3) to the Stop state if there is no time left to use the solver for a solution and $[Satis < 50]$ or the optimal strategy produced by the solver has $[Satis < 50]$ (transition from the Solver state).

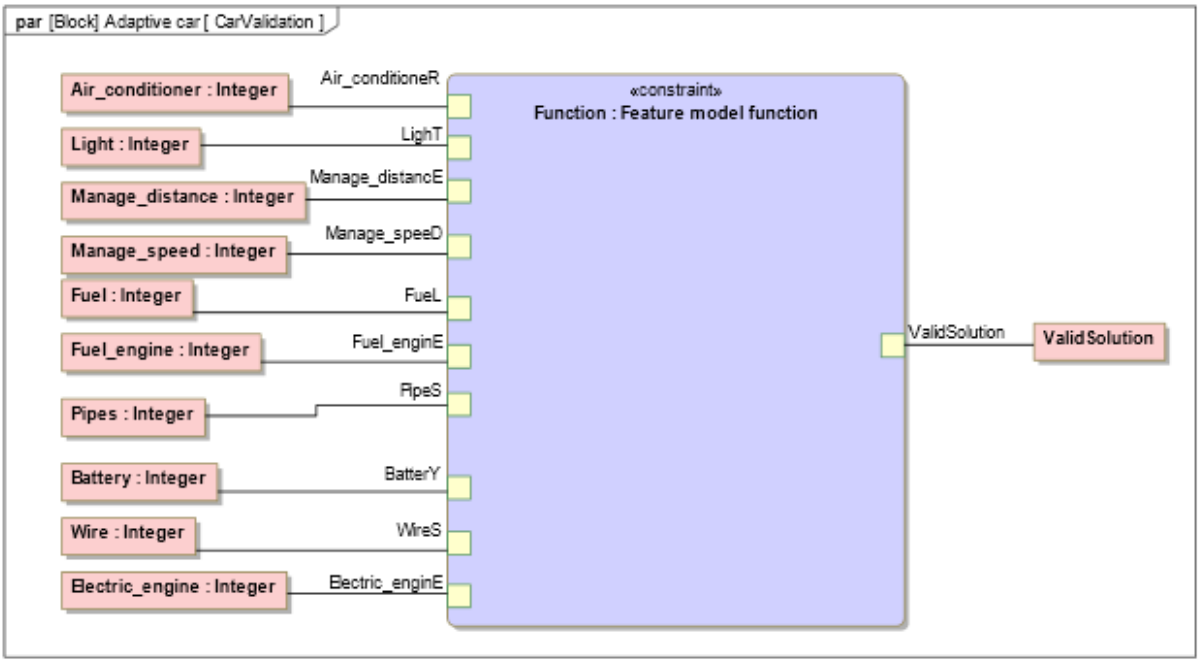

Figure 6.17: SysML parametric diagram of the feature model function for the hybrid car example

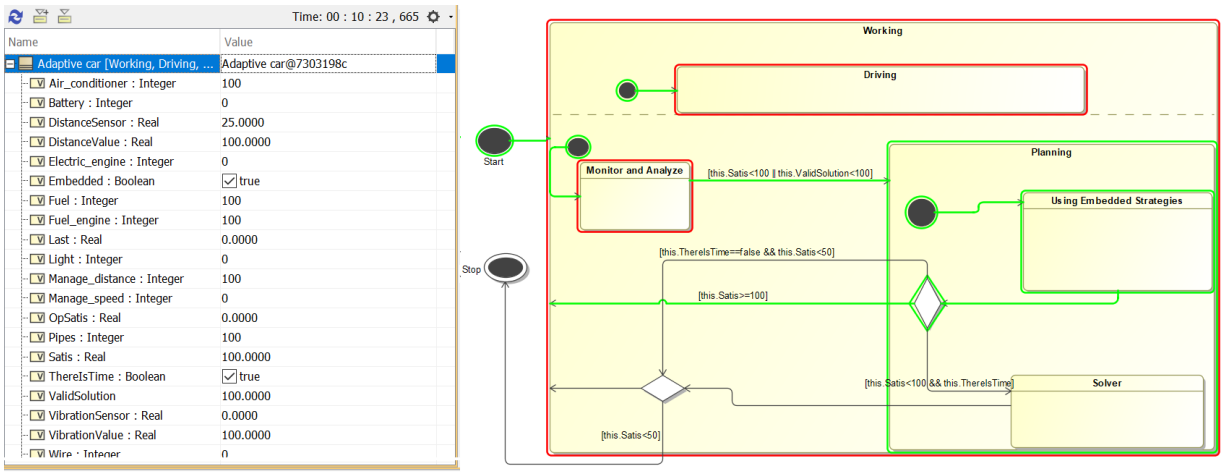

Figure 6.18: Interaction between the parametric diagram (left) and the executable SysML state machine diagram (right).

Simulation Steps and Results: We exposed the simulated system to different environmental conditions. Each time the values of the vibration or distance sensors were changed, the system started searching for solutions if the satisfaction level of the goal model was less than 100. Table 6.6 shows some sample values of input parameters, the current strategy number, and the related system output and reactions. Figure 6.19 shows the time series chart of the overall satisfaction of the system generated while executing the model. As shown in Table 6.6, when the system started with strategy 1, the satisfaction level was 80. The Monitor and Analyze state detected the violation and decided

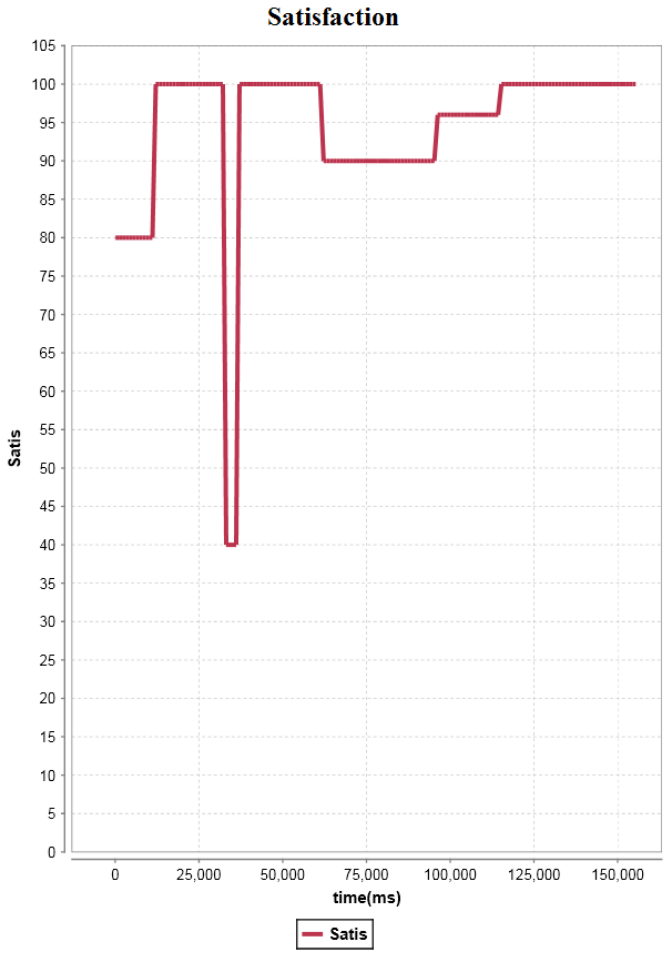

Figure 6.19: System satisfaction according to environmental conditions, as a time series.

to adapt by transitioning to the Planning super-state (see Fig. 6.18, right). The Planning activity looks for suitable solutions for the current situation, provided by the Distance and Vibration values, either Using Embedded Strategies optimized for well-known contexts, or using the combination of the two functions (with a Solver) to generate the best solutions for unforeseen contexts. The result here is that an embedded strategy (strategy 2) was selected and the system satisfaction was improved to 100 (second row of the table). The system continued Driving and Monitoring the environment until the context got changed ([1,0] input), which reduced the system performance (goal function satisfaction reduced to 40). Consequently, Planing was activated and it searched for the best solution using the Solver since the condition was not covered by the embedded strategies.

As shown in the adaptation column of Table 6.6, the reactions of the system were different depending on the selected features or system alternatives that form the possible strategies or configurations. For example, when the input got to [1,10] and then to [25,4], the best feasible strategies could not increase system performance to become 100 and hence the system shut down the electric engine to better satisfy user preferences (Comfortable driving softgoal in Fig. 6.14). In this case, the combined function detected a

Table 6.6: Sample simulation parameters and related system reactions.

Input		Output		Adapt?	Weakness?
Distance	**Vibration**	**Strategy**	**Satisfaction**		
25	0	1	80	Embedded	No
25	0	2	100	No need	No
1	0	2	40	Solver	No
1	0	3	100	No need	No
1	10	3	90	Best	Yes
25	4	3	96	Best	Yes
25	0	3	100	No need	No

weakness in the current design (the available alternatives or the selected feature cannot satisfy the objectives of the developed system and its stakeholders) in dealing with these conditions. Improvements to the current design (adding/removing/replacing features to reduce vibration) are needed to increase the system ability to deal with such situations at runtime. As shown in Table 6.6, we have attributed several values to the vibration and distance properties to examine the system reaction using the generated functions. Each time the system performance was degraded by new conditions, the functions helped the system adapt to the observed condition, as shown in the time series of Fig. 6.19.

This simple scenario demonstrates that the arithmetic functions generated from goal and feature models can be integrated with system design to support decision making during design selection and runtime adaptation, via optimization and simulation models.

Lessons learned: By modeling and exploring this simple adaptive system example, we have learned the following:

1. The optimization process, which generates optimal strategies covering predefined environmental conditions (e.g., using the functions in a CPLEX model), and the simulation process, which checks the coverage of these strategies against system design, can be used to help keep systems stable through adaptation. They can also help detect weaknesses requiring less restrictive goals or better designs.

2. Feedback is needed to update the embedded strategies when the solver generates new solutions that enhance the current conditions.

3. A simple embedded solver or optimizer is needed to increase the ability of systems to deal with unforeseen conditions at runtime (for system reliability), perhaps under a time budget (i.e., best effort).

4. The state machine diagram in Figure 6.18 is actually very important in the CGS4Adaptation approach as it captures the monitoring, analysis, and planning behaviour of the MAPE cycle. This state machine structure is generic and reusable in other SCPSs.

6.3 Chapter Summary

This chapter provided a complete illustrative example suggesting the feasibility of CGS4Adaptation and its methods. In order to evaluate the correctness and effectiveness of the approach for a realistic and larger system, the next chapter uses a Smart Home case study (representing a real-world adaptive SCPS), and experiment, where test cases, and optimizations will be assessed in design-time and runtime adaptation contexts.

Chapter 7

Smart Home Modeling Case Study

In this chapter, we introduce an *IoT-based Smart Home* case study whose first informal description was provided by collaborators from the SM@RT Team (IRIT, France) [87]. We describe the roles of the control system in the management of smart devices for smart homes towards the achievement of inhabitants' objectives in a real context. First, we design complementary feature and goal models of a Smart Home (Sections 7.3 and 7.4) using an informal description and requirements (Sections 7.1 and 7.2), with an explanation of the rationale behind the selected design options. Next, we use CGS4Adaptation to create the corresponding mathematical function for each model and convert these functions to programming language code (Section 7.5). We also explain how the functions are integrated and employed to build an optimization model, with an implementation based on a commercial solver, namely IBM CPLEX Optimizer (Section 7.6). This optimization model is validated using specific environmental contexts, random contexts, as well as all possible contexts in order to detect and fix model vulnerabilities while strengthening the self-adaptability of the smart home management system during the development and runtime stages (Section 7.7). SysML requirement diagrams and block definition diagrams that show the system's structure are also described and modeled in this chapter (Section 7.8).

7.1 Introduction

A *Smart Home* (SH) is considering the in-home support services that i) support elderly and disabled people with the care needed to live independently in the home environment, while ii) providing them with a comfortable and secure life despite age and physical limitations. The aim here is to design a self-adaptive *Smart Home Management System* (SHMS) that can effectively control most smart devices such as smart thermostats, smart windows, and home electronics, with the ability to control electricity consumption and production, as well as the comfort, privacy, and security of the inhabitants. This case study is based on the informal description presented in Appendix B.1.

7.2 Smart Home Requirements

We use Smart Home requirements that express adaptability and require a decision to be made by the management system. Table 7.1 shows the requirements for the self-adaptive SHMS.

Table 7.1: Requirements of the Smart Home case study, from [44].

Categ.	Id	Description
Security	SH01-010	The Smart Home shall support prevention and detection of unauthorized physical intrusions.
	SH-01-020	The Smart Home shall support prevention and detection of unauthorized computer intrusions.
	SH-01-050	The Smart Home shall be able to detect signs of fire.
	SH-01-070	The Smart Home shall allow physical access to emergency services (firemen, hospital service, etc.).
Accommodation	SH-02-010	The Smart Home shall accommodate physical, medical, and mental conditions of the inhabitants.
	SH-02-020	The Smart Home shall accommodate specific preferences entered by the inhabitants.
	SH-02-030	The Smart Home shall learn from the behaviour of inhabitants.
	SH-02-040	The Smart Home shall assist inhabitants with certain everyday tasks.
Economy	SH-03-010	The Smart Home shall be energy efficient.
	SH-03-020	The Smart Home's annual energy consumption shall be less than the maximum imposed by local regulations.
	SH-03-030	The Smart Home shall support the production and efficient use of energy via solar panels, wind turbines, etc.

7.3 Smart Home Feature Model

In this section, we create the feature model by introducing the required features of the SHMS and their constraints.

The **features** represent the characteristics of the software and hardware required by stakeholders, including sensors, smart devices, and software tasks capturing functionalities. To choose suitable features for our SHMS, we first explored various smart devices present in today's markets (e.g., on security [96], heating and conditioning [39], and doors and windows [130]), as well as the literature on the design of smart homes [29, 75, 80], with a focus on the Canadian context.

To decrease energy consumption, we focus on the management of air conditioner, heater, water heater, and dishwasher appliances because they represent the main energy consumers

in Canadian households [57]. Also, according to [57], older windows allow heat energy to radiate through them and increase energy consumption by increasing the operating time of heaters and air conditioners, thus, smart windows are added to the set of managed devices.

Each smart device has its own *controller* that can switch the device on and off. We ignore features unhelpful in a centralized management context, such as activating the alarm of the washing machine when the laundry is finished, because such features are already well managed by the devices themselves. To increase the comfort of inhabitants and the system's reliability, Remote access and Schedule features are used to enable people to turn on/off their devices with/without an Internet connection. Also, a Learning technique is an optional feature used to control SH devices based on the inhabitants' behaviours and usage patterns. Figure 7.1 shows the mandatory and optional features for our Smart Home feature model.

There are also several **exclusion and inclusion constraints** among some of these features, realized in our model using URN Links of types *Exclude* and *Include*:

1. Include links from the On air conditioner and On heater features to the Close windows and Off air ventilator features.
2. Exclude link from the On air conditioner feature to the On heater feature.
3. Include links from all remote access features to the Internet feature.

7.4 Smart Home Goal Model

Rather than adjusting each device for each condition and making decisions according to the local context associated with each device, the *Smart Home Management System* controls daily household activities using the available features and devices to maximize goal satisfaction of inhabitants according to a global context. Thus, SHMS must monitor the environment to assess the overall performance of the house, and then decide whether to reconfigure/adapt or not. We create the GRL model of SHMS according to the SH case study description in Appendix B, the requirements in Table 7.1, and the features described in the previous section. The model is split over several diagrams, shown in Figures 7.2 to 7.4.

The GRL model has two actors: SmartHomeSystem and Inhabitant. The system's main goal (Manage home) is decomposed into many sub-goals: Monitor the environment, Manage energy, Normalize inside temperature, Provide services, and Manage appliances. Each goal is refined into the related tasks that participate in calculating its satisfaction value, and these goals correspond to leaf features in the feature model of Figure 7.1. The inhabitant's softgoals are Maximize energy production, Maximize privacy, Minimize consumption, Maximize comfort, and Maximize security. Each is provided a relative importance weight, between parentheses. Some of the tasks/features in the system have positive or negative contributions towards the satisfaction of the inhabitant's softgoals. The contribution levels are approximate values based on the thesis author's understanding of the features. More

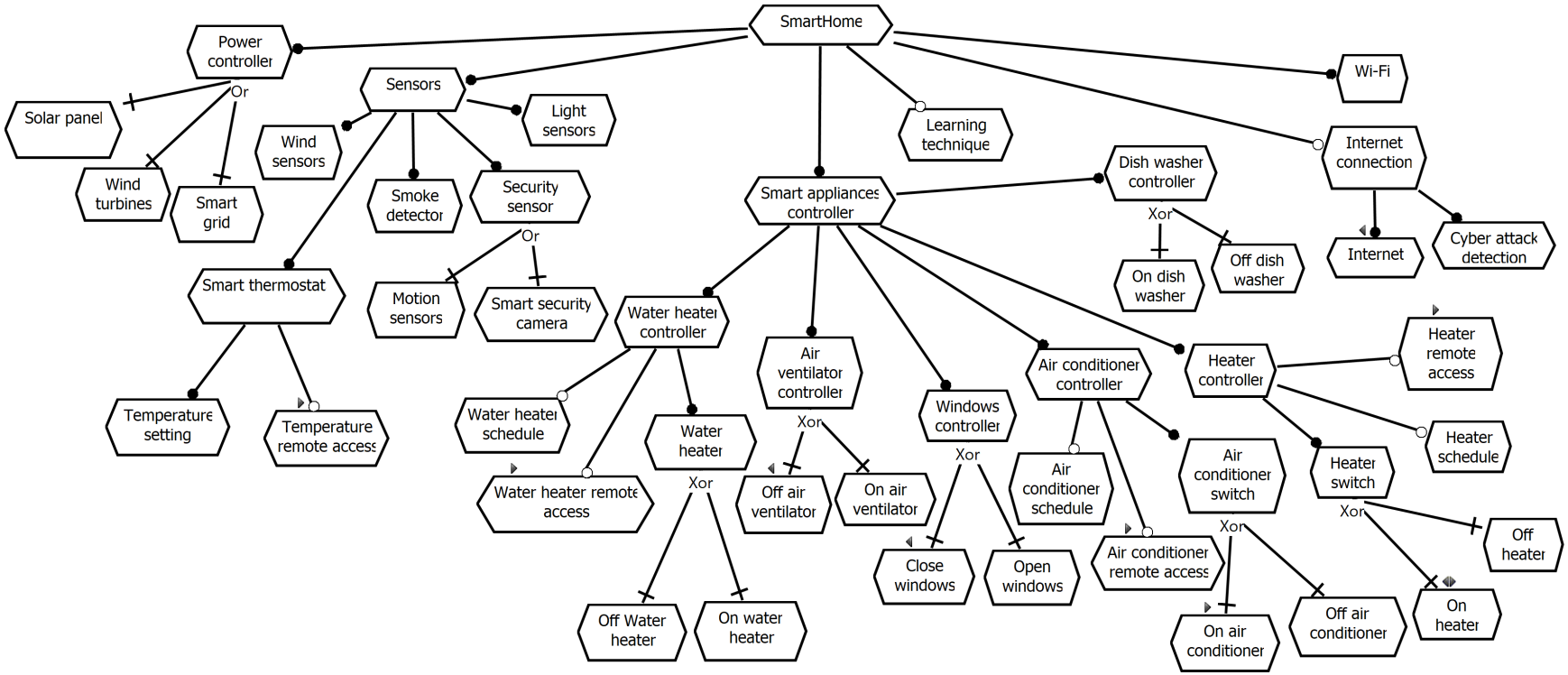

Figure 7.1: Feature model of the Smart Home case study.

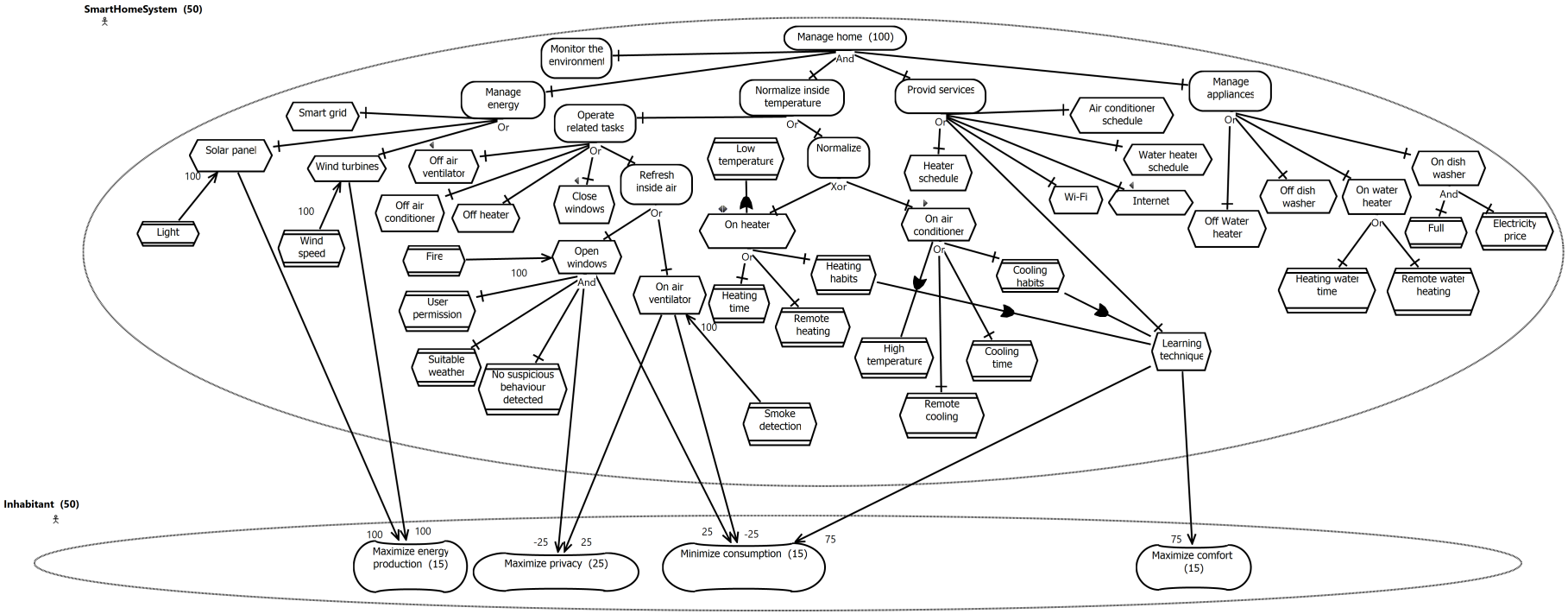

Figure 7.2: GRL model of the top-level Manage home goal.

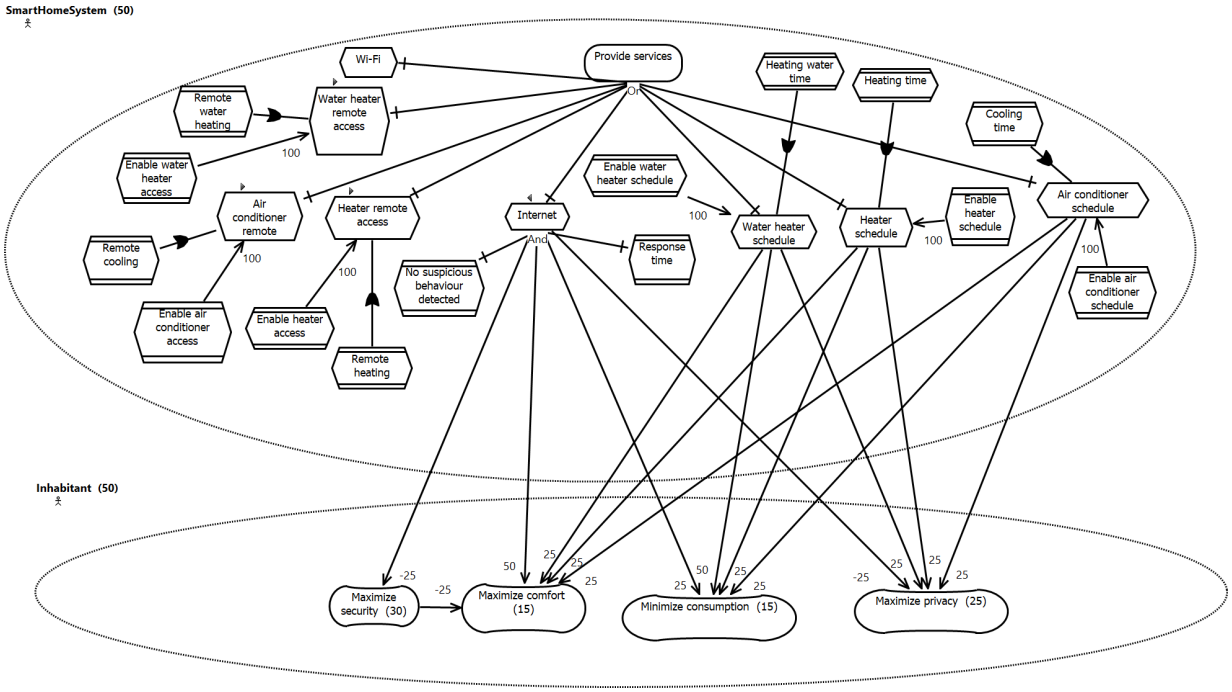

Figure 7.3: GRL model of the Provide services sub-goal.

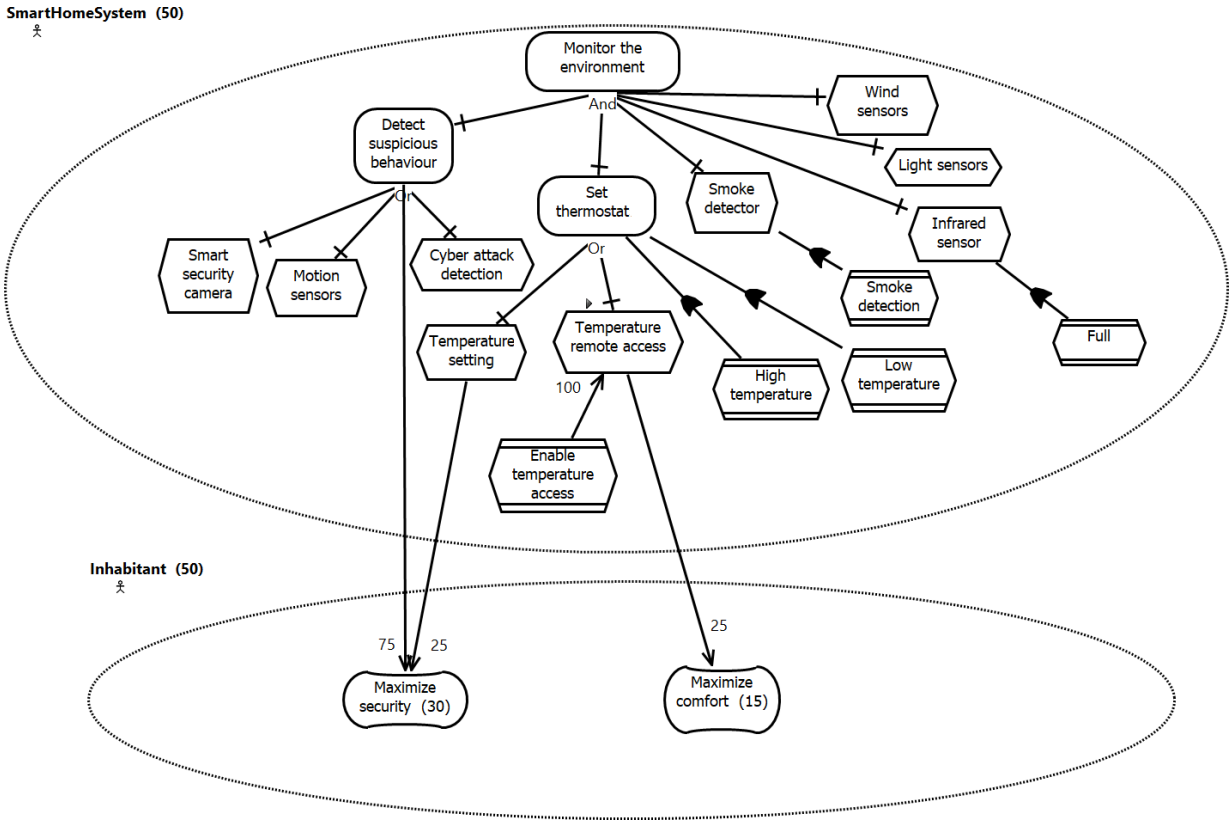

Figure 7.4: GRL model of the Monitor the environment sub-goal.

advanced techniques based on consensus-building among experts [10, 73] could be used to find more precise importance and contribution values, but such precision is not necessary for the purpose of this case study.

To provide the rationale behind our design and the relationship between the generated model and SH requirements from Table 7.2, we describe some features and behaviours of the created GRL model of the SHMS together with the motivated requirements.

- **SH-01-010 & SH-01-020**: To achieve security and privacy goals, we follow the security tips to secure smart homes provided by the Norton Community [128]. The SHMS model provides the detect suspicious behaviour goal as well as the related tasks and indicators, shown in Figure 7.4, to monitor the Internet connection and the surrounding area. If the value of the No suspicious behaviour detected indicator is 0/false, all the remote access features will be disabled by disabling the Internet connection (Figure 7.3). As shown in Figure 7.2, the Close windows selection will be forced by disabling the Open windows selection (related by an XOR relationship in the feature model).

- **SH-01-050**: To ensure the safety and security of the inhabitants, SHMS uses Fire and Smoke detection indicators to detect any sign of fire.

- **SH-01-070**: Also for safety and security reasons, SHMS does not Open windows unless there is a User permission or emergency (fire detection) to allow physical access to emergency services.

- **SH-02-010 & SH-02-040**: The Normalize inside temperature and Manage appliances goals and related tasks enable elderly people and people with disabilities to control and manage daily activities such as washing dishes and providing hot water in time.

- **SH-02-020**: To increase the comfort of inhabitants through preferences, the Schedule and Remote access features allow inhabitants to enable, set and control remotely their devices as they prefer while the Provide services goal (Figure 7.3) and its related indicators operate the related tasks (i.e., On heater, On air conditioner, Open windows, etc.) at desired times.

- **SH-02-030**: Also, the Learning technique feature is provided and its result is used by SHMS to activate the heater and air conditioner based on past behaviour using the Cooling habits and Heating habits indicators.

- **SH-03-010 & SH-03-20**: From an economic perspective, the Learning technique, Remote access and Schedule features as well as some system behaviour such as closing windows and the ventilator while the heater or air conditioner are operating, contribute positively to energy efficiency.

- **SH-03-030**: In addition, the Manage energy goal manages the production of energy from Solar panels and Wind turbines according to the current weather through Wind speed and the available light through Light indicators.

Table 7.2: Boolean indicators (last letter capitalized by our function generation tool).

Indicator	**Condition**
Electricity_pricE	≤ 6.4
Wind_speeD	≥ 12 and < 90 kilometer per hour [81]
High_tempreturE	≥ 27 degrees Celsius
Low_tempreturE	≤ 15 degrees Celsius
Cooling_habitS	==True
Cooling_timE	==True
FirE	==True
FulL	==True
Heating_habitS	==True
Heating_timE	==True
Heating_water_timE	==True
LighT	==True
No_suspicious_behaviour_detecteD	==True
Remote_coolinG	==True
Remote_water_heatinG	==True
Response_timE	≤ 2 milliseconds [12]
Smoke_detectioN	==True
Suitable_weatheR	==True
User_permissioN	==True
Remote_heatinG	==True
Enable_water_heater_accesS	==True
Enable_water_heater_schedulE	==True
Enable_temperature_accesS	==True
Enable_air_conditioner_accesS	==True
Enable_heater_schedulE	==True
Enable_heater_accesS	==True
Enable_air_conditioner_schedulE	==True

7.5 Automatic Generation of Arithmetic Functions for GRL and Feature Models

To convert the GRL and feature models of the SHMS into mathematical functions, we export the model as a split functions whose formulas of leaf features and indicators were separated and integrated with the main GRL and feature functions, as discussed in Section 4.3. A file named SmartHome.py is automatically generated, and can be found in Appendix B.2.

7.6 SHMS Model Validation and Enhancement

To enhance the SHMS quality, correctness, and reliability, its feature, goal, and optimization models (described in Section 7.3, Section 7.4, and Listing B.2 respectively) and their solutions have been tested and improved through several iterations of the modeling and optimization processes.

To validate the modeled solutions, we build an optimization model of SHMS that returns the optimal solution for each context. When no solution (where there is no feasible solution) or low-quality solutions (where the model's satisfaction value is very low) are issued by the optimizer, causes are investigated by checking the current context (i.e., the values of indicators and related leaf-features), then the GRL and/or feature models are modified, and finally new mathematical functions are generated. The optimization model is updated and tested using the same contexts and then different ones. Unacceptable solutions must be improved until they become satisfactory.

7.6.1 Generation of Contexts

In order to cover as many contexts as possible at the time of development and increase system quality when dealing with these contexts, the created models have been tested through three steps, with increasing coverage:

1. **Specific context scenarios** to meet specific user requirements. These scenarios are extracted from system requirements where specific system behaviours and actions are required according to specific contexts (i.e., in case of fire, cyber and physical violations and inhabitants with disability). The system is checked with the required scenarios (i.e. in the case of fire, fire indicator = 100) while randomly changing the values of the unrelated indicators. These scenarios and the improved solutions are listed in Tables 7.3 and 7.4 respectively.

2. **Random contexts** to deal with unexpected contexts, when covering all possible contexts is impossible because of the large number of indicator value combinations describing environmental situations. This method is used mainly to discover models' flaws and situations without any solution, often caused by contradictions, and to

fix them individually. Since the outcome of the optimization process is saved (e.g., in an Excel sheet) and includes the value of the objective function, the values of random data for contexts (indicators and related features) that lead to unacceptable solutions can be tracked and disclosed. Random testing is repeated several times. When an unacceptable solution is discovered, the context is tracked, the relevant model is updated, and the optimization process is performed by the context that causes this solution using the violating context as a specific scenario (previous step). We repeat this step and modify the models until unacceptable solutions no longer appear in the results. The main result is that the issued solutions are satisfactory when performing this test, and the optimization model is updated by adding feature formulas to resolve conflicts, as described in Subsection 7.6.2.

3. **All possible contexts** to deal with unexpected contexts at runtime. This is an *exhaustive* way to check all possible contexts (including unexpected ones) by examining system behaviour according to these contexts and providing a suitable solution for each of them. Note however that this step is only applicable when the number of combinations of variable/indicator values describing the contexts is small enough. SHMS has 27 indicators and each has two possible values: 100 or 0. An exhaustive coverage of system behaviour would require $2^{27} = 134,217,728$ contexts to be optimized, which cannot be done easily. However, dependency, include, exclude and decomposition relationships between these indicators and related features can be used to eliminate unfeasible or redundant combinations/contexts as follows:

   (a) AND-decompositions: We test the system using the same values for all the related indicators, i.e., 0 AND 0, and 100 AND 100. The remaining two combinations will have the same result as the 0-0 combination so they are eliminated.

   (b) OR-decompositions and Exclude relationships: We test the system using 100 OR 0, 0 OR 100, and 0 OR 0 as inputs. We eliminated the 100 OR 100 combination because it has the same result as 100 OR 0 using OR-decomposition. Similarly, 100 Excludes 100 is eliminated because this is the same result as using 0 Excludes 0.

   (c) Dependencies and Include relationships: In both cases, we remove the 100-0 combination here as it is already covered by the 0-0 combination.

   Notice that the above rules are not applicable for indicators that are part of multiple relationships.

   For example, in the SHMS model, the Air conditioner remote access feature can be enabled or disabled by users based on the Enable air conditioner access indicator, and the Remote cooling indicator depends on it. Since this feature has an include relationship to the Internet connection and is turned off by the feature model formula (even if it is enabled by the user with Enable air conditioner access = 100) when no Internet connection is available, we examine the system using only 0 as entry for these indicators (as Internet = 0). In addition, since examining them using different values is not rational, both of them are represented by one iteration (i.e., Remote cooling =

Enable remote access). This is also applicable to other indicators that have similar relationships, such as Remote heating and Remote water heating.

As the Internet is AND-decomposed by the No suspicious behaviour detected and Response time indicators, we do not need to examine all remote indicators (Remote cooling, Remote heating, and Remote water heating) and the related enable remote access indicators (Enable air conditioner access, Enable water heater access, and Enable heater access) with the value 100 when No suspicious behaviour detected and Response time indicators are not both equal to 100. As a result, each time the Internet is disabled, all combinations of remote accesses and their related indicators that use 100 as a satisfaction value are eliminated.

By applying these rules through all model elements, the number of contexts to examine is reduced by 99.99% to only 14,400 contexts, and their optimization requires about three hours to be solved. Listing 7.1 shows the program used to generate the data needed for testing the model, with dependencies embedded in the *for* loops. This program creates the `SmartEquationL` class, whose method `Optimizefunction` calls the CPLEX CP optimizer instance to solve the model using the passed parameters (indicator values describing each context) and saves the result in an Excel document.

Listing 7.1: Java code to generate all possible contexts for the Smart Home case study

```
for (int Electricity_pricES = 6; Electricity_pricES <= 16;
    Electricity_pricES += 10)
 for (int TemperaturES = 10; TemperaturES <= 30; TemperaturES += 10)
  for (int Response_timES = 1; Response_timES <= 3; Response_timES +=
      2)
   for (int No_suspicious_behaviour_detecteDS = 0;
       No_suspicious_behaviour_detecteDS <= 100;
       No_suspicious_behaviour_detecteDS += 100)
    for (int Cooling_habitSS = 0; Cooling_habitSS <= 100;
        Cooling_habitSS += Math.min(No_suspicious_behaviour_detecteDS
        , (Response_timES <= 2) ? 100 : 0) == 100 ? 200 : 100)
     for (int Heating_habitSS = 0; Heating_habitSS <= 100;
         Heating_habitSS += Math.min(
         No_suspicious_behaviour_detecteDS, (Response_timES <= 2) ?
         100 : 0) == 100 ? 200 : 100)
      for (int LighT = 0; LighT <= 100; LighT += 100)
       for (int FirE = 0; FirE <= 100; FirE += 100)
        for (int Smoke_detectioNS = 0; Smoke_detectioNS <= 100;
            Smoke_detectioNS += 100)
         for (int Wind_speeDS = 10; Wind_speeDS < 90; Wind_speeDS +=
             70)
          for (int Remote_coolinGS = 0; Remote_coolinGS <= 100;
              Remote_coolinGS += Math.min(
              No_suspicious_behaviour_detecteDS, (Response_timES <=
              2) ? 100 : 0) == 100 ? 100 : 200)
           for (int Cooling_timES = 0; Cooling_timES <= 100;
               Cooling_timES += Remote_coolinGS == 0 ? 100 : 200)
```

```
for (int Remote_water_heatinGS = 0; Remote_water_heatinGS
    <= 100; Remote_water_heatinGS += Math.min(
   No_suspicious_behaviour_detecteDS, (Response_timES <=
    2) ? 100 : 0) == 100 ? 100 : 200)
 for (int Heating_water_timES = 0; Heating_water_timES <=
     100; Heating_water_timES += Remote_water_heatinGS
    == 0 ? 100 : 200)
  for (int Remote_heatinGS = 0; Remote_heatinGS <= 100;
      Remote_heatinGS += Math.min(
      No_suspicious_behaviour_detecteDS, (Response_timES
      <= 2) ? 100 : 0) == 100 ? 100 : 200)
   for (int Heating_timES = 0; Heating_timES <= 100;
       Heating_timES += Remote_heatinGS == 0 ? 100 : 200)
    for (int Enable_temperature_accesS = 0;
        Enable_temperature_accesS <= 100;
        Enable_temperature_accesS += Math.min(
        No_suspicious_behaviour_detecteDS, (
        Response_timES <= 2) ? 100 : 0) == 100 ? 100 :
        200) {
     Indx++;
     SmartEquationL optimizeModel = new SmartEquationL(
         Electricity_pricES, Cooling_habitSS, FirE,
         Heating_habitSS, LighT, TemperaturES,
         No_suspicious_behaviour_detecteDS,
         Response_timES, Smoke_detectioNS, Wind_speeDS,
         Remote_coolinGS, Remote_water_heatinGS,
         Remote_heatinGS, Cooling_timES, Heating_timES,
         Heating_water_timES, Enable_temperature_accesS,
         Indx);
     optimizeModel.Optimizefunction(arg);
     optimizeModel = null;
     System.gc();
    }
```

The result of each test, including the selected and unselected features, and the resulting value of the objective function together with the related context variables, are saved in an Excel document and imported in a database file for analysis. Section 7.7 describes the result of this step.

7.6.2 Optimization Model Construction

To build the model, we need to define the objective function, the ancillary variables to supply the model with data, the free features (decision variables), the constraints on these variables, and the integration type to share the values of features related to indicators.

Objective function: maximizes the Smart Home GRL function generated by jUCMNav and translated to several programming languages, see Listing B.2.

```
Maximize:
    SmartHome()
```

Ancillary variables: represent the indicators' current values, using the same names as their related indicators. Accordingly, Table 7.2 lists the names of the indicators, which are the names of the ancillary variables.

Decision variables: according to the GRL and feature models, we have 17 decision variables representing *free features*, i.e., features that are not related to indicators, and hence that represent choices for the configuration:

```
Close_windowS
Cyber_attack_detectioN
Infrared_sensoR
Learning_techniquE
Light_sensorS
Motion_sensorS
Off_air_conditioneR
Off_air_ventilatoR
Off_dish_washeR
Off_heateR
Off_water_heateR
Smart_griD
Smart_security_camerA
Smoke_detectoR
Temperature_settinG
WifI
Wind_sensorS
```

Constraints: specify that the value of the SHMS feature function must be 100, with the following restrictions on the values of the decision variables, which should equal 0 (unselected) or 100 (selected):

```
Subject To:
    FeatureModel()==100
    Learning_techniquE==0 OR Learning_techniquE==100
    Off_air_ventilatoR==0 OR Off_air_ventilatoR==100
    Temperature_settinG==0 OR Temperature_settinG==100
    Wind_sensorS==0 OR Wind_sensorS==100
    Off_water_heateR==0 OR Off_water_heateR==100
    Cyber_attack_detectioN==0 OR Cyber_attack_detectioN==100
    Light_sensorS==0 OR Light_sensorS==100
    Off_dish_washeR==0 OR Off_dish_washeR==100
    Motion_sensorS==0 OR Motion_sensorS==100
    Smart_griD==0 OR Smart_griD==100
    WifI==0 OR WifI==100
    Close_windowS==0 OR Close_windowS==100
    Smart_security_camerA==0 OR Smart_security_camerA==100
    Infrared_sensoR==0 OR Infrared_sensoR==100
    Smoke_detectoR==0 OR Smoke_detectoR==100
    Off_heateR==0 OR Off_heateR==100
    Off_air_conditioneR==100 OR Off_air_conditioneR==0
```

Integration type to share the values of features related to indicators: since we have already exported the GRL function as a group of formulas, including the formulas of these features and the main GRL function, we evaluate the formulas of these features before evaluating the GRL and feature functions to obtain the overall satisfaction values for the model. To synchronize their values between GRL and feature functions, as discussed in Section 5.3, we have to deal with two types of non-free features. For binary features (0 or 100), synchronization can be easily achieved by calculating their GRL formulas first, before evaluating the global GRL and feature functions. Their values will be automatically passed to the two functions, where they are represented by the same variables. This enables computing the feature functions less often, without repetition, and hence with faster results. The GRL formulas for this type of features in SHMS are the following, where variables that have an indicator name with a S suffix represent the current value of each indicator:

```
Response_timE = Response_timES<=2 ? 100: 0
No_suspicious_behaviour_detecteD = No_suspicious_behaviour_detecteDS
InterneT = minl(No_suspicious_behaviour_detecteD, Response_timE)
Water_heater_schedulE = Enable_water_heater_schedulE
Heater_schedulE = Enable_heater_schedulE
Air_conditioner_schedulE = Enable_air_conditioner_schedulE
Cooling_habitS = minl(Learning_techniquE, Cooling_habitSS)
Cooling_timE = minl(Air_conditioner_schedulE, Cooling_timES)
Heating_habitS = minl(Learning_techniquE, Heating_habitSS)
Heating_timE = minl(Heater_schedulE, Heating_timES)
Heating_water_timE = minl(Water_heater_schedulE, Heating_water_timES)
High_temperaturE = minl(maxl(Temperature_settinG,
    Enable_temperature_accesS), TemperaturES>=27?100:0)
Low_temperaturE = minl(maxl(Temperature_settinG,
    Enable_temperature_accesS), TemperaturES<=15?100:0)
Remote_coolinG = minl(Air_conditioner_remote_accesS, Remote_coolinGS)
Remote_heatinG = minl(Heater_remote_accesS, Remote_heatinGS)
Remote_water_heatinG = minl(Water_heater_remote_accesS,
    Remote_water_heatinGS)
Smoke_detectioN = minl(Smoke_detectoR, Smoke_detectioNS)
Wind_speeD = (Wind_speeDS>=12) && (Wind_speeDS<90)? 100:0
Wind_turbineS = Wind_speeD
Solar_paneL = LighT
On_air_ventilatoR = Smoke_detectioN
Open_windowS = maxl(0.0, minl(100.0, (100*FirE+minl(Suitable_weatheR,
    minl(No_suspicious_behaviour_detecteD,User_permissioN))*100.0)/ 100.0)
    )
On_water_heateR = maxl(Remote_water_heatinG, Heating_water_timE)
On_dish_washeR = minl(Electricity_pricE, FulL)
```

In the end, we have already integrated the two functions through their non-free features and built the optimization model. The SHMS optimization model, found in Listing B.2 is implemented using IBM CPLEX's Constraint Programming (CP) optimizer. Next, we solve the optimization model using different current values for the indicators, describing different environmental conditions (contexts).

7.7 Optimization Results and Discussion

This section introduces the results of the testing process, first using *random contexts*, *specific scenarios*, and then by covering *all possible contexts*, and provides insights into the captured data in the discussion subsection.

7.7.1 Using Random Contexts

We went through several random contexts/scenarios in order to discover system weaknesses and how far they were from satisfying the smart home's inhabitants' goals. The main result is that several modifications were made to the GRL and feature models, and consequently to the proposed method itself by using features formulas in solving features' conflicts.

While testing the models, we examined specific contexts individually. Any context that did not generate any solution was resolved individually by adding negative and positive contribution links to disable or enable the relevant features. When we obtained the desired solutions, we examined more complex contexts by selecting and deselecting the non-free features via indicators.

It is worth noting that if invalid features are enabled by users or if a conflict between different solutions emerges for unforeseen conditions, the optimization model returns no available solution. Since the values of the non-free features are enforced by the context, validating them is necessary before optimizing the model to avoid having no solution or adding more contribution links to solve the emergent conflicts. Because the validity of the selected features (plan or configuration) is detected by the feature function, using features' formulas before the optimization process disables the invalid features automatically (i.e., when the formula returns values < 100 for invalid features) and solves the emergent conflicts. This consequently leads to modify the GRL model by deleting all the added contribution links, which reduces the size of the generated function. The benefits of reducing the size of the functions are further illustrated in Chapter 9.

To increase the reliability of the SHMS using the feature formulas of non-free features, we switched off the invalid features using their internal dependencies instead of returning no solution, as follows:

1. An *invalid feature* has an unsatisfied value from its feature formula (i.e., less than 100). This can be caused, for example, by an *include* constraint on a feature disabled by the context, or by an *exclude* constraint on a feature enabled by the context. When we developed the SHMS optimization model to test system behaviour using random contexts, we did not get any solution when the Internet service was not selected (response time > 2). We examined the problematic contexts via the ancillary variables and the related non-free features and found that some remote access features (Air conditioner remote access, Heater remote access, or Water heater remote access) were enabled by the inhabitant via the Enable Air conditioner access, Enable heater access, or Enable water heater access indicators (Figure 7.3). In the feature model, all the remote access features have *include* constraints with the Internet service (as explained

at the end of Section 7.3) while the current context enables them, and disables the Internet. Consequently, the optimizer cannot find any solution to this situation. This situation was not taken into account initially when modeling the system. It emerged during a test of the model through random input contexts. To avoid this situation and disable remote access features when the Internet service is not available, feature formulas (generated by jUCMNav as a separated formula for each feature and combined to generate the whole function of the feature model) are used as follows:

```
//From GRL function
Temperature_remote_accesSGRL = Enable_temperature_accesS
//From feature function
Temperature_remote_accesS = (((Temperature_remote_accesSGRL
                                + InterneT) / 200.0) * 100.0)
                            < 100 ? 0 : 100

//From GRL function
Air_conditioner_remote_accesSGRL = Enable_air_conditioner_accesS
//From feature function
Air_conditioner_remote_accesS = (((Air_conditioner_remote_accesSGRL
                                + InterneT) / 200.0) * 100.0)
                            < 100 ? 0 : 100

//From GRL function
Water_heater_remote_accesSGRL = Enable_water_heater_accesS
//From feature function
Water_heater_remote_accesS = (((Water_heater_remote_accesSGRL
                                + InterneT) / 200.0) * 100.0)
                            < 100 ? 0 : 100

//From GRL function
Heater_remote_accesSGRL = Enable_heater_accesS
//From feature function
Heater_remote_accesS = (((Heater_remote_accesSGRL + InterneT)
                                / 200.0) * 100.0)
                            < 100 ? 0 : 100
```

2. To resolve conflicts between different solutions due to *unexpected contexts*, feature formulas from the feature model are used to turn off the invalid features, as described in Section 5.3. In one case, when modeling the SHMS case study, we decided to automatically open the smart house windows in case of fire, to allow access to emergency services, as shown in Figure 7.2. However, for some random contexts, we got no solution as a result. The feature function told us that the solution we have initially designed was not valid. When we checked the input data (current values of indicators) and the related features, we found that the windows had opened due to the fire, but the temperature was 42C, which also triggered the air conditioner. These two features have an *exclude* constraint in the feature model, while fire and high temperature indicators had turned them on at the same time, which forces the optimizer to look for valid configurations including these two features. Therefore, the feature function always returns values less than 100 (invalid solution) and no solution

provided by the GRL function could satisfy this constraint. To solve this conflict, we added negative and positive contribution links from the indicators causing the conflict to the conflicted features. This in turn required us to remodel the constraints of the feature model in the goal model through the involved indicators (e.g., the on heater feature alone is enabled by three indicators, as seen in Figures 7.2). Instead, evaluating both formulas (GRL and feature models formulas of the Open windows and On air conditioner) turns off the air conditioner, resolves the conflict before optimizing the model and releases the optimal solutions. This leads to delete all the added contribution links and takes advantage of the relationships designed in feature models without duplicating the work. The used formulas as follows:

```
On_heateRGRL = minl(Low_temperaturE, maxl(0.0, minl(100.0,
                    (maxl(Remote_heatinG, maxl(Heating_habitS,
                        Heating_timE)) * 100.0) / 100.0)))

//From GRL function
On_heateRGRL = minl(Low_temperaturE, maxl(0.0, minl(100.0,
                    (maxl(Remote_heatinG, maxl(Heating_habitS,
                        Heating_timE)) * 100.0) / 100.0)))

//From feature function
On_heateR = (((On_heateRGRL + minl(Close_windowS,
                                   Off_air_ventilatoR)) / 200.0) *
                                      100.0)
                < 100 ? 0 : On_heateRGRL

//From GRL function
On_air_conditionerGRL = minl(High_temperaturE, maxl(0.0, minl(100.0,
                    (maxl(Remote_coolinG, maxl(Cooling_habitS,
                        Cooling_timE)) * 100.0) / 100.0)))

//From feature function
On_air_conditioneR = minl(maxl(0, On_air_conditionerGRL - On_heateR),
                          (((On_air_conditionerGRL + minl(Close_windowS
                              , Off_air_ventilatoR)) / 200.0) * 100.0))
                          < 100 ? 0 : On_air_conditioneRGRL
```

The final SHMS functions and the optimization model are shown in Listing B.2. The knowledge gained through the use of randomly generated contexts also influenced the completeness and complexity of the specific scenarios derived from the informal requirements, described in the following subsection.

7.7.2 Using Specific Scenarios

To examine the system reactions, and the potential interactions between the system features, we solved the optimization model for different values of the indicators that describe environmental condition scenarios/contexts covering important aspects of the informal requirements. These ten expected contexts are listed in Table 7.3 while the resulting optimal

solutions computed from the model are listed in Table 7.4. We explain the expected contexts and the related strategies as follows:

Scenario 1 describes the context where there is insufficient light or wind speed to operate solar panels or wind turbines. The Internet service is significantly delayed, the temperature is too high, and the air conditioner schedule feature was enabled by residents. According to the air conditioner schedule, the cooling time is now (Cooling time indicator=100) while the inhabitants opened the windows (user permission=100). This context enables two conflicting features: open windows and on air conditioner. This stimulates the SHMS to re-configure the home by issuing Strategy 1, as shown in Table 7.4. In this strategy, the SHMS activates the smart grid to provide electricity to the home, and solves the conflict by opening the windows and turning off the air conditioner. Although the dishwasher is full, it is not turned on because the price of electricity is high. Also, the water heater is not turned on even though this is water heating time (water heating time=100) because residents have disabled the feature (enable water heater schedule=0). Similarly, the Cyber attack detection feature was unselected because of the Internet connection delay. The satisfaction value with this strategy is 85.2.

In **Scenario 2**, the inhabitants have enabled all remote access and scheduling features of the devices. However, the heating time is not specified. The light is good enough to operate the solar panels while the temperature is 2C. Also, the Internet is deactivated (response time=3) even if the user is trying to control the water heater remotely. The response of the SHMS model (Strategy 2 with satisfaction level=96.0) is that it used the learning technique to power the heater, closed the windows, activated solar panels, deactivated the smart grid, and operated the dishwasher because the price of electricity equals the target price. Due to the poor Internet connection, the water heater cannot be operated.

In **Scenario 3**, we assume that the wind speed is not high enough to generate electricity while there is enough light to produce it. The temperature is very low (-20C), and there is fire and smoke in the house. Fire and smoke detection sensors cause the opening of windows and ventilator, which are incompatible with the heater activated by low temperatures and heating habits. As a reaction, the system closes the heater based on the on heater feature's formula, opens the windows to provide access to an emergency team without user permission, and turns the air ventilator on. The dishwasher is activated because it is full and the price of electricity is lower than the target value. The smart grid and wind turbines are turned off and solar panels are activated (Strategy 3, satisfaction level=96.0).

In **Scenario 4**, the level of satisfaction raises to 100 when there is no fire and the Internet is deactivated (because of a long response time=3). In addition, the light and wind speed are suitable to generate the needed electricity while the smoke detector activates the air ventilator, which consequently turns the heater off based on the formula of the on heater feature. The dishwasher gets activated because it is full and the price of electricity is lower than the target value. The smart grid is turned off since the wind turbines and solar panels are activated.

As the Internet service works well (response time=1) in **Scenario 5**. where the remote access and schedule features were also enabled by inhabitant, the system is able to use the remote access features of the devices, turns the heater and water heater on in response to

the user's remote commands and the low temperature (temperature=-10), and activates the Cyber attack detection task. As in the previous strategy, the dishwasher is activated because it is full and the price of electricity is lower than the target value. The smart grid gets turned off since wind turbines and solar panels are turned on. The overall satisfaction level of the model here is 89.1.

In **Scenario 6**, we assume that the previous solution is the current context, where the heater was turned on by remote residents who also opened the windows remotely. In addition to activating the dishwasher, solar panels, and wind turbines, because the weather is suitable, the system opens the windows and turns off the heater (Strategy 6, with satisfaction level=85.2).

In contrast, **Scenario 7** supposes that the weather is unsuitable, with a wind speed of 90 km/h and a temperature at -10C, and that the heating time is now (heating time=100). SHMS (Strategy 7) does not open the windows but opens the heater when the residents are out of the house and tried to open the windows remotely (user permission=100). Moreover, the wind turbine is turned off by the system to avoid damages. This solution has a stisfaction level=85.9.

In **Scenario 8**, the wind speed is sufficient to generate electricity without damaging the turbines, the residents have enabled all the schedule and remote access features of the devices, and the temperature is too low at -30C. However, the system detects suspicious behaviours. The reaction (Strategy 8, with a satisfaction level=96.0) is that the SHMS closes the windows and disables all remote access features by disabling Internet services, in order to prevent any violation [128]. To increase the quality of the provided services, the system activates the learning technique task that was used to open the heater while activating the water heater using a user-defined schedule (enable water heater schedule and water heating time indicators are at 100).

In **Scenario 9**, the temperature is 22C (neither too hot nor too cold). Strategy 9 is released for SHMS, with a satisfaction level=89.1. The air conditioner is not turned on even though it is remotely opened (SHMS uses the user setting of the air conditioner when the high temperature is $\geq$ 27C).

The model can be used to find solutions when some features stop working at runtime. In **Scenario 10**, we assume that motion sensors are invalid. For the optimal solution that includes an alternative to these malfunctioning sensors, we run the model using *motion sensors == 0* as a constraint. This forces the optimizer to look for optimal solutions that exclude these sensors. Strategy 10, with a satisfaction level=89.1, is the released solution and includes the use of a security camera instead of motion sensors.

7.7.3 Using All Possible Contexts

For the SHMS, this option is using 14,400 individual optimizations whose results were imported in a database. Several queries were conducted to get a deeper insight about how far the designed model satisfies the goals of its actors (system and inhabitant).

Table 7.3: Tested scenarios and the related indicators' values

Indicator	**Scenarios**									
	1	2	3	4	5	6	7	8	9	10
Electricity_pricE	13.4	6.4	6.4	6.4	6.4	6.4	15	13.4	13.4	13.4
Wind speeD	11	10	10	20	20	19	90	23	19	20
High_tempreturE	100	0	0	0	0	0	0	0	0	100
Low_tempreturE	0	100	100	100	100	100	100	100	0	0
Cooling habitS	100	0	0	0	0	0	0	0	0	0
Cooling timE	100	0	0	0	0	0	0	0	100	100
FirE	0	0	100	0	0	0	0	0	0	0
FulL	100	100	100	100	100	100	0	0	0	0
Heating habitS	0	100	100	100	0	0	0	100	0	0
Heating timE	0	0	0	0	0	0	100	0	0	0
Heating_Water_timE	100	0	0	0	0	0	100	100	0	0
LighT	0	100	100	100	100	100	0	100	100	0
No_suspicious_behaviour_detecteD	100	100	100	100	100	100	100	0	100	100
Remote_coolinG	100	0	0	0	0	0	0	0	100	100
Remote_water_heatinG	0	100	100	100	100	0	0	0	0	0
Response_timE	3	3	3	3	1	1	1	1.5	1.5	1.8
Smoke_detectioN	0	0	100	100	0	0	0	0	0	0
Suitable_weatheR	100	0	0	0	0	100	0	100	100	100
User_permissioN	100	0	0	0	0	100	100	100	0	0
Remote_heatinG	0	0	0	0	100	100	0	0	0	0
Enable_water_heater_access	0	100	100	100	100	100	100	100	100	100
Enable_water_heater_schedulE	0	100	100	100	100	100	100	100	100	100
Enable_temperature_accesS	0	100	100	100	100	100	100	100	100	100
Enable_air_conditioner_accesS	100	100	100	100	100	100	100	100	100	100
Enable_heater_schedule	0	100	100	100	100	100	100	100	100	100
Enable_heater_accesS	0	100	100	100	100	100	100	100	100	100
Enable_air_conditioner_schedulE	100	100	100	100	100	100	100	100	100	100

Table 7.4: Optimal strategies for specific environmental conditions using the CP Optimizer (Y means the feature is selected, while N means it is unselected)

Feature	**Optimal strategy for each scenario**									
	1	2	3	4	5	6	7	8	9	10
Cyber attack detectioN	N	N	N	N	Y	Y	Y	N	Y	Y
Motion sensorS	Y	Y	N	N	Y	Y	Y	Y	Y	N
Smart security camerA	N	N	Y	Y	N	N	N	N	N	Y
Temperature settinG	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Smoke detectoR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Learning techniquE	Y	Y	Y	Y	N	N	N	Y	N	N
Infrared sensoR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Light sensorS	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Wind sensorS	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Smart griD	Y	N	N	N	N	N	N	N	N	N
Off dish washeR	Y	N	N	N	N	N	Y	Y	Y	Y
Off water heateR	Y	Y	Y	Y	N	N	N	N	Y	Y
Close windowS	N	N	Y	Y	Y	N	Y	Y	Y	Y
Off air conditioneR	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
Off air ventilatoR	Y	Y	N	N	Y	Y	Y	Y	Y	Y
Off heateR	Y	N	Y	Y	N	Y	N	N	Y	Y
WifI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
InterneT	N	N	N	N	Y	Y	Y	N	Y	Y
On_air_conditioneR	N	N	N	N	N	N	N	N	N	Y
On_heateR	N	Y	N	N	Y	N	Y	Y	N	N
Wind_turbines	N	N	N	Y	Y	Y	N	Y	Y	Y
Solar_paneL	N	Y	Y	Y	Y	Y	Y	Y	Y	N
On_air_ventilatoR	N	N	Y	Y	N	N	N	N	N	N
Open_windowS	Y	N	Y	N	N	Y	N	N	N	N
On_water_heateR	N	N	N	N	Y	Y	Y	Y	N	N
On_dish_washeR	N	Y	Y	Y	Y	Y	N	N	N	N
Water_heater_schedulE	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
Heater_schedulE	Y	N	N	Y	Y	Y	Y	Y	Y	Y
Air_conditioner_schedulE	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Temperature_remote_accesS	N	N	N	N	Y	Y	Y	N	Y	Y
Air_conditioner_remote_accesS	N	N	N	N	Y	Y	Y	N	Y	Y
Water_heater_remote_accesS	N	N	N	N	Y	Y	Y	N	Y	Y
Heater_remote_accesS	N	N	N	N	Y	Y	Y	N	Y	Y
Objective function	85.2	96.0	96.0	100	89.1	85.2	85.9	96.0	89.1	89.1

The most noteworthy result is that no solution with a null satisfaction was issued, which suggests that the use of random contexts was effective in detecting errors and that the features formula helped solving the conflicts (step 2 in Section 7.6). There are also 144 contexts (1%) with a fully satisfied model. All of these solutions include neither the Internet service nor the smart grid, both the air conditioner and the heater are off, the solar panel and/or the wind turbines are on, and the dishwasher and water heater are also on. These 144 contexts satisfy the inhabitant's softgoals, which are to maximize energy production, privacy, and comfort, while minimizing energy consumption.

Another query shows that 144 other contexts have the same lowest satisfaction level, which is 74.2. All these contexts exclude the Internet service and all schedule features are disabled by the user. Moreover, these contexts are characterized by light and wind speeds insufficient for generating electricity.

To get an indication of how often a feature formula is used to solve contextual conflicts (e.g., the on_ heater feature formula), we ran a query that selected solutions where *on heater = 0* and at least one of the schedule, remote access, and/or learning technique features and its related indicator (heating time, remote heating, and/or heating habits) are enabled, with the temperature less than or equal to 15C. This context describes that the temperature is low, the inhabitants have turned the heater on (using remote access, setting the schedule, or using learning technique), and the SHMS turned it off. All of the resulting 3,072 solutions include opening windows and/or opening a ventilator. When we group these solutions by relevant context variables, the result shows only three causes of conflict: fire, smoke, and/or user permission and suitable weather, in that order.

7.7.4 Discussion

From Tables 7.3 and 7.4, we note that to improve the satisfaction of the model's goals, each time the Internet service is disabled, the system activates the learning technique task to enable or disable home devices according to inhabitants' preferences (Strategies 1, 2, 3, 4, and 8). Moreover, the Cyber attacks detection task is always selected when the Internet service is available, in order to increase the privacy and security of the population. Furthermore, the system enables the mandatory features, such as infrared sensor and smoke detector, through all the issued solutions while choosing between the optional features (e.g., Internet or learning technique) or alternatives (e.g., motion sensors or smart security camera) to increase the overall quality of the service.

Although the context enables some conflicting features to be present together (such as contexts 1, 3, 4, and 6), feature formulas solve these conflicts before searching for solutions, hence avoiding solutions with a satisfaction level of 0. As the result of exploring all contexts shows the effectiveness of using feature formulas in solving the conflicts caused by contexts or modeling errors, it also shows how these formulas can be used to solve causes of conflicts related to each feature and help developers to avoid solving each conflict individually (since many of these conflicts have already been modeled in the feature model). Moreover, these formulas reduce the time needed to optimize all these contexts since each time we add one

of these formulas, the number of contexts without a solution is greatly reduced (e.g., the *on_heater* formula alone solves conflicts caused by 3 factors in 3,072 contexts).

In summary, optimizing the model for each context evaluates the system satisfaction regarding this context and helps developers validate the developed system and improve its quality. Moreover, employing GRL and feature functions in this model specify clearly the impact of each feature on the entire system and helps developers discover weaknesses in their designs, diagnose the causes of problems, and improve the solutions ahead of time, before building the system. The overall result is that the SHMS functions are able to manage 42 alternatives while monitoring 27 environmental conditions (producing 34,217,728 contexts). They reconfigure the smart house according to changes in the environment, taking into account the preferences of the inhabitants, by enabling all mandatory features and choosing between alternatives that maximize these preferences.

7.8 SysML Model

This section describes the SysML model of the SHMS, including its block definition and requirement diagrams. These two diagrams are designed with the SysML tool *Magic-Draw Cameo Systems Modeler*, according to the system features and goals, to meet system requirements.

7.8.1 SysML Block Definition Diagram

The structure of the SHMS is shown in a block definition diagram (Figures 7.5, 7.6, and 7.7). These figures show the relationships between system blocks, including hierarchy relationships, associations, dependencies, and decompositions, in addition to the system requirements satisfied by these blocks. Figure 7.5 illustrates the first level of decomposition of the Smart Home block, while the second level of decomposition is described in Fig. 7.7, where the Monitor block and its related blocks and constraint blocks are introduced. Also, one level of generalization between the Smart device and is specific devices is shown in Figure 7.6, where the blocks related to the Appliance controller block are described.

7.8.2 SysML Requirement Diagram

Figure 7.8 shows the requirement diagram of the SHMS according to the requirements listed in Table 7.1. The diagram shows a compound requirement (SmartHomeSystem) and the related sub-requirements (Economy, SecurityAndSafety, and Accommodation), in addition to their own derived and sub-requirements.

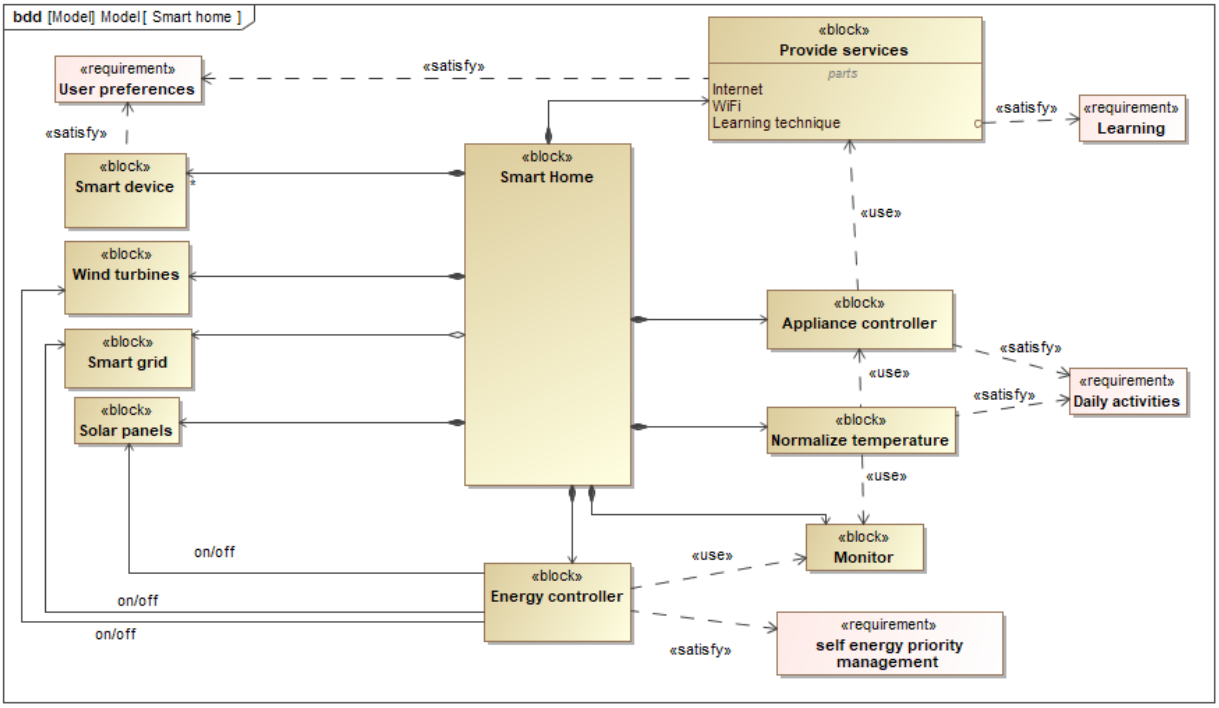

Figure 7.5: Block definition diagram of the SHMS.

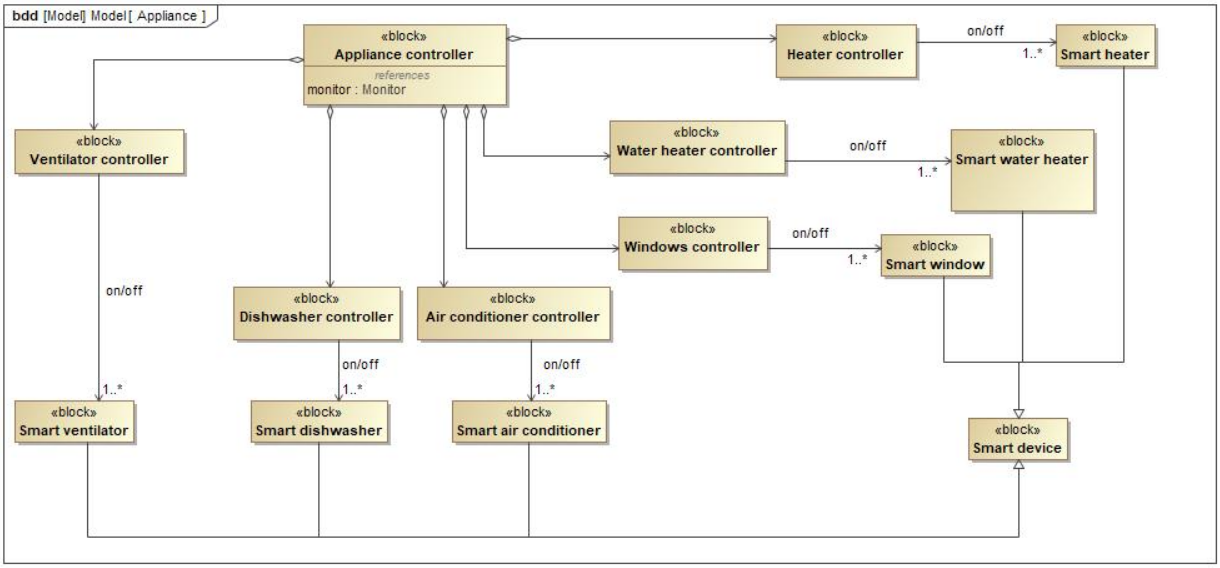

Figure 7.6: Block definition diagram of the Appliance controller block from Fig. 7.5.

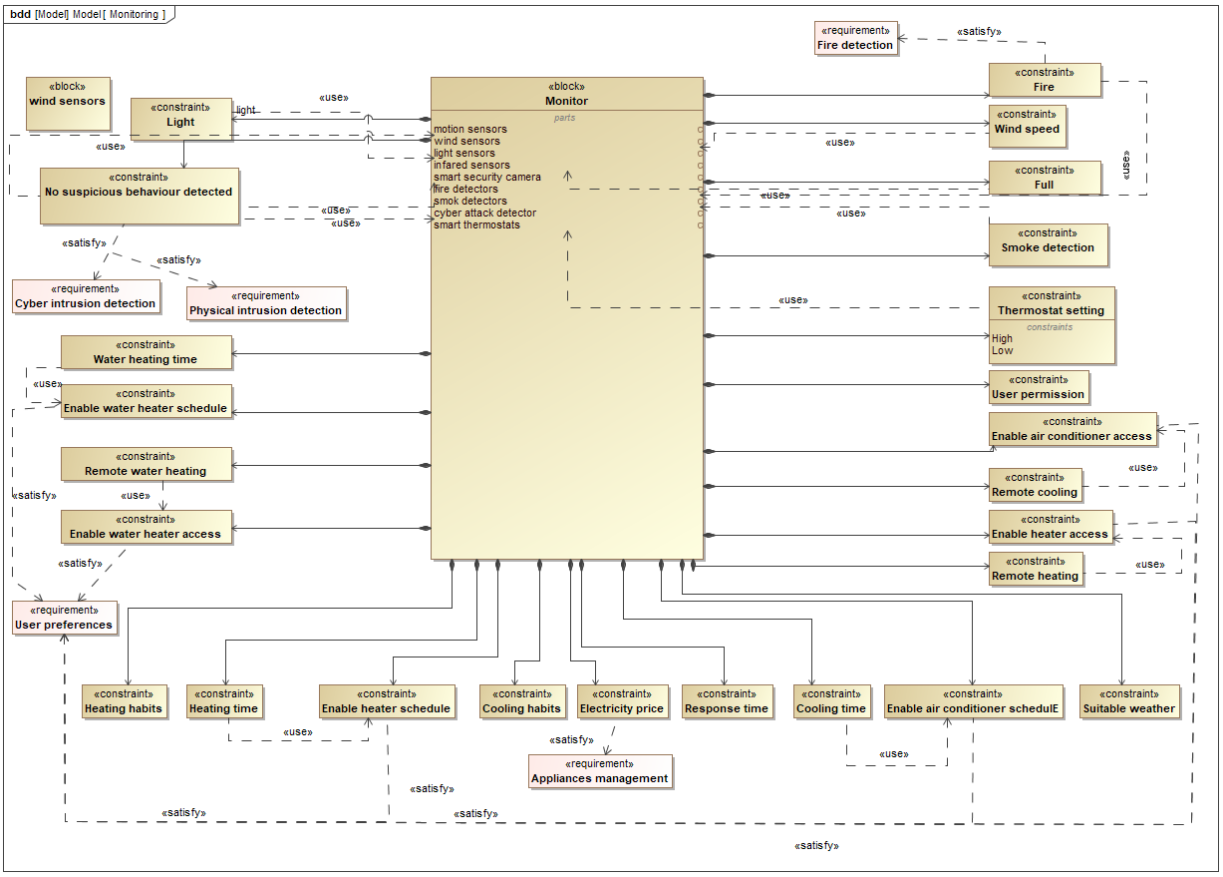

Figure 7.7: Block definition diagram of the Monitor block from Fig. 7.5.

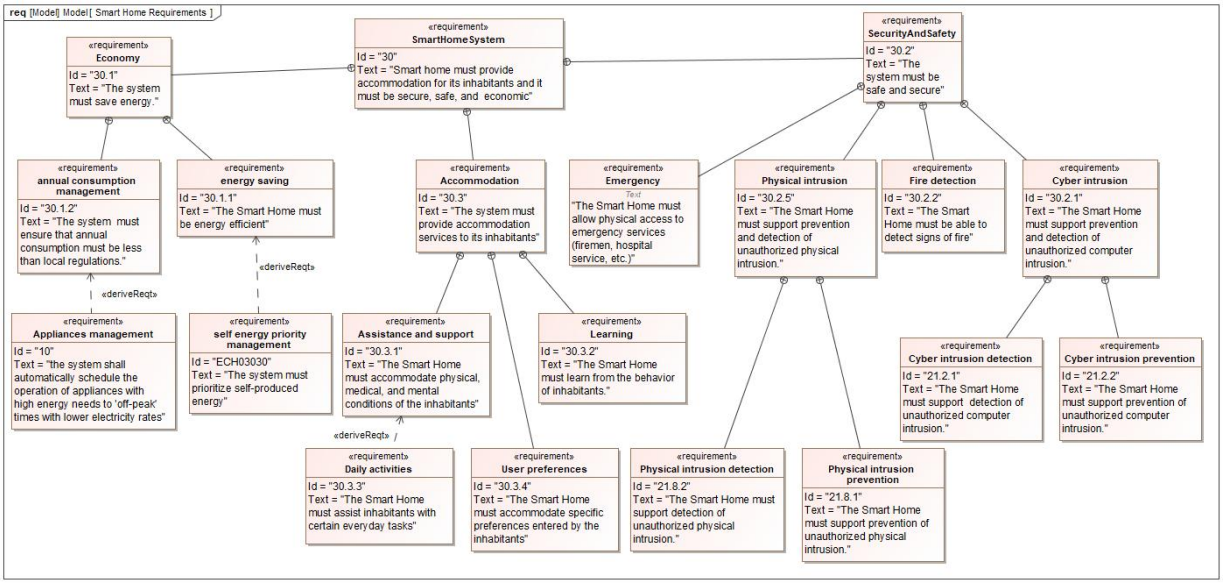

Figure 7.8: Requirement diagram of the SHMS.

7.9 Chapter Summary

Using a sophisticated Smart Home Management System case study, we demonstrated how CGS4Adaptation can be used to manage the selection among system alternatives according to monitored environmental conditions, while ensuring system validity. In particular, we built an optimization model that adapts the system from one configuration to the valid and best one when changes in the surrounding context require it. The goal and feature models of the SHMS were validated and enhanced through an iterative process using three type of tests: 1) using random contexts, 2) using specific scenarios, and 3) using all possible contexts while exploiting the relationships between the monitored elements. During the validation process, contexts without valid solutions due to conflicts between system features that were activated by these contexts are discovered, and the conflict causes are tracked. We also illustrated how these conflicts can be resolved before optimizing the model to increase model reliability.

In the next chapter, we exploit traceability links between the SHMS models using IBM Rational DOORS in order to help developers assess completeness and consistency across system requirements, goal models, and SysML design models.

Chapter 8

Traceability Management in the Smart Home Models

In this chapter, we export GRL models from jUCMNav and SysML models from Cameo Systems Modeler [126] to DXL scripts, with application to the Smart Home Management System. IBM Rational DOORS is used to import and store the elements of these models and their relationships, and manage traceability links between the models. A traceability information model and automated DXL rules are used to run consistency and completeness checks using the direct and indirect relationships between system goals, requirements, and blocks, in order to find alignment issues early and on a continuous basis. This chapter also illustrates the benefits of using DOORS views and filters to visualize the results of these consistency and completeness checks and to solve highlighted issues through appropriate actions.

8.1 Importing the SHMS Models into DOORS

To import the SysML model of the SHMS in the DOORS database, the same automated mechanism used for the illustrative example in Chapter 6 was reused here, without any observed issue. The report template created to produce DXL scripts from SysML models from Cameo Systems Modeler (Section 5.1.2) was used to export all the identified elements of the block definition and requirement diagrams. The resulting DXL script (Listing C.2) then invokes the SysML library produced from the MI-DSL specification in Section 5.1.1.

Executing this DXL script with DOORS produces 6 formal modules (Blocks, Requirements, Block definition diagram, Requirements diagram, Block Relationships, and Requirements Relationships) and 8 link modules (Aggregation, Association, Composition, ContainedBy, Dependency, Derive, Generalization, and Satisfy). The objects of each module have the same name, identifier, and links as the design elements created by Cameo Systems Modeler.

To import the GRL model of SHMS in the DOORS database, a DXL script was generated from jUCMNav and executed by IBM DOORS, with the usual formal and link

modules provided by Rahman's import mechanism [139].

Figure 8.1 shows the created *SysMLSmartHomeModel* folder for the SysML model, the *URNSmartHome* folder for the GRL model, and their related modules in DOORS.

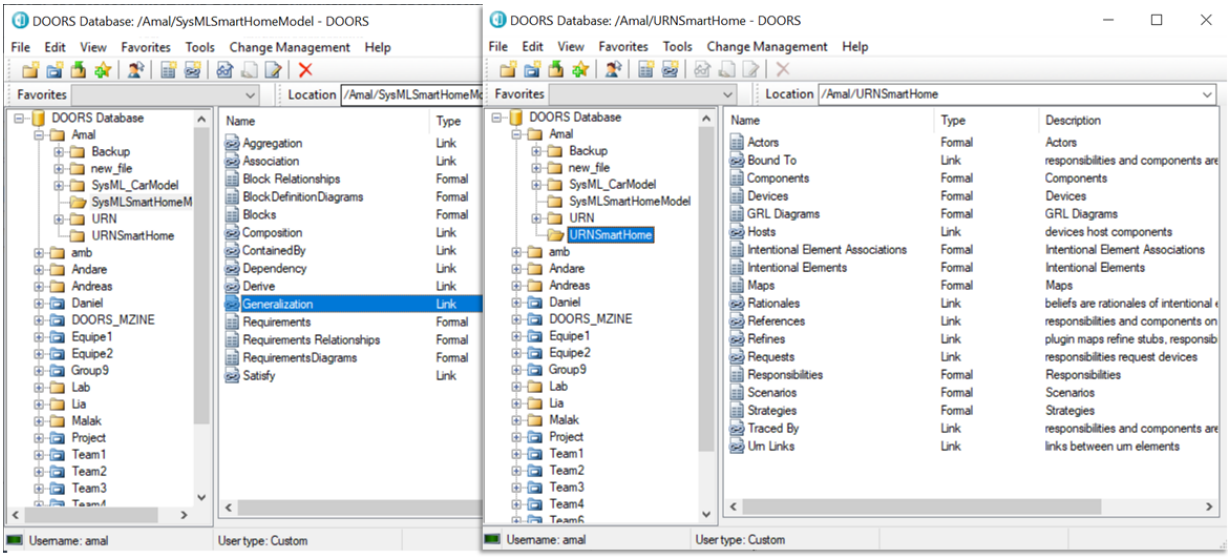

Figure 8.1: SysML and GRL models in the DOORS database.

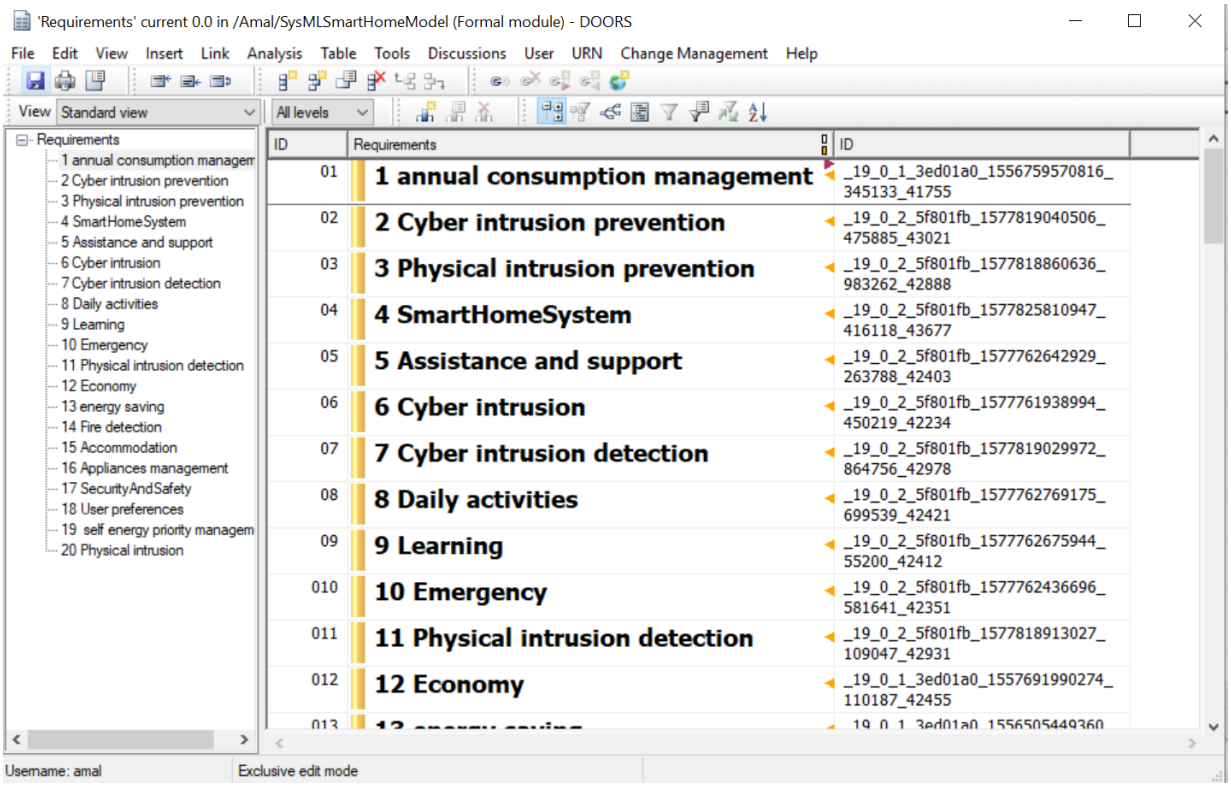

Figure 8.2: Requirements formal module.

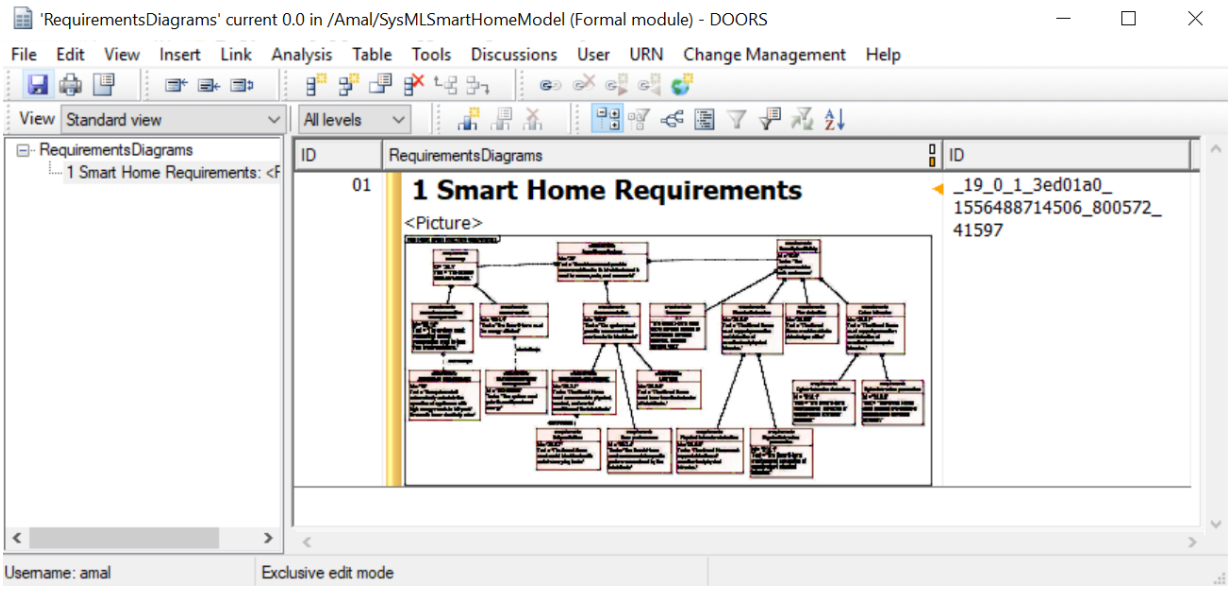

Figure 8.3: Requirements diagram formal module.

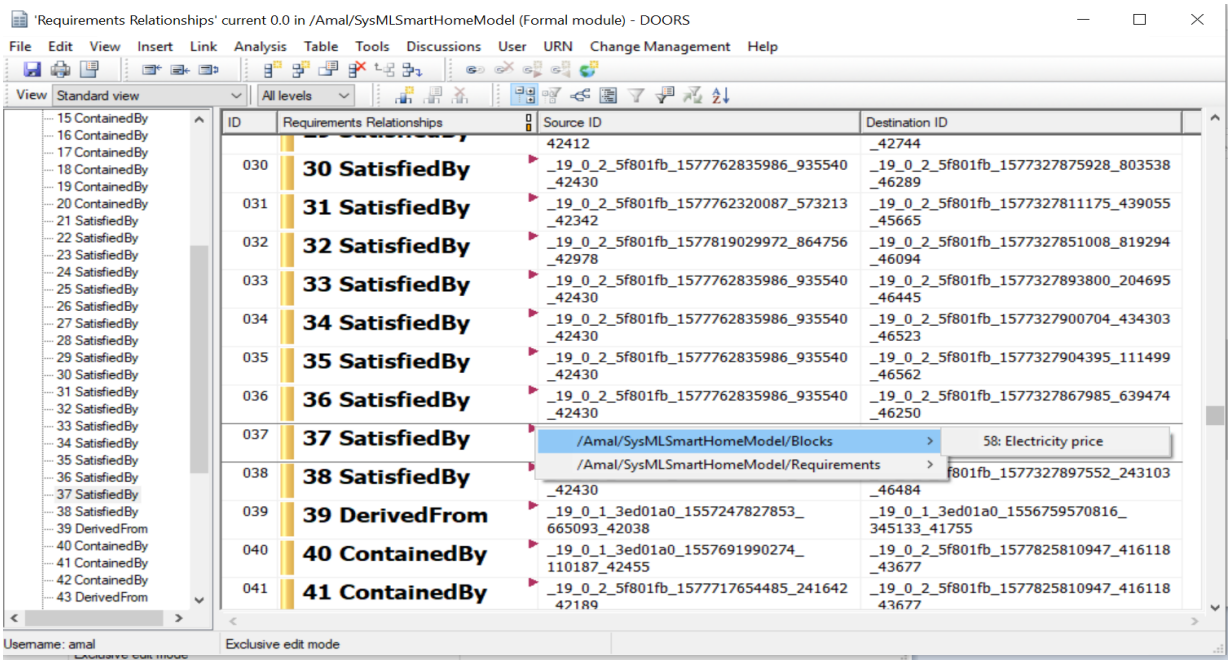

Figure 8.4: Requirements relationships formal module.

The requirements module (Fig. 8.2) has 20 requirements and 58 links involving requirements and other SysML elements. The relationships between requirements and blocks are shown in Fig. 8.4 and classified using three link modules (types): Satisfy, Derive, and ContainedBy. Figure 8.3 shows the imported requirements diagram including its name, ID, image, and the related relationships.

The block definition diagram module (Fig. 8.5) has 3 diagrams linked to 60 blocks (Fig. 8.6) through 60 ContainedBy links, whereas the relationships between blocks them-

selves are described via 79 links. All of these relationships are saved in the block relationships formal module described in Fig. 8.7.

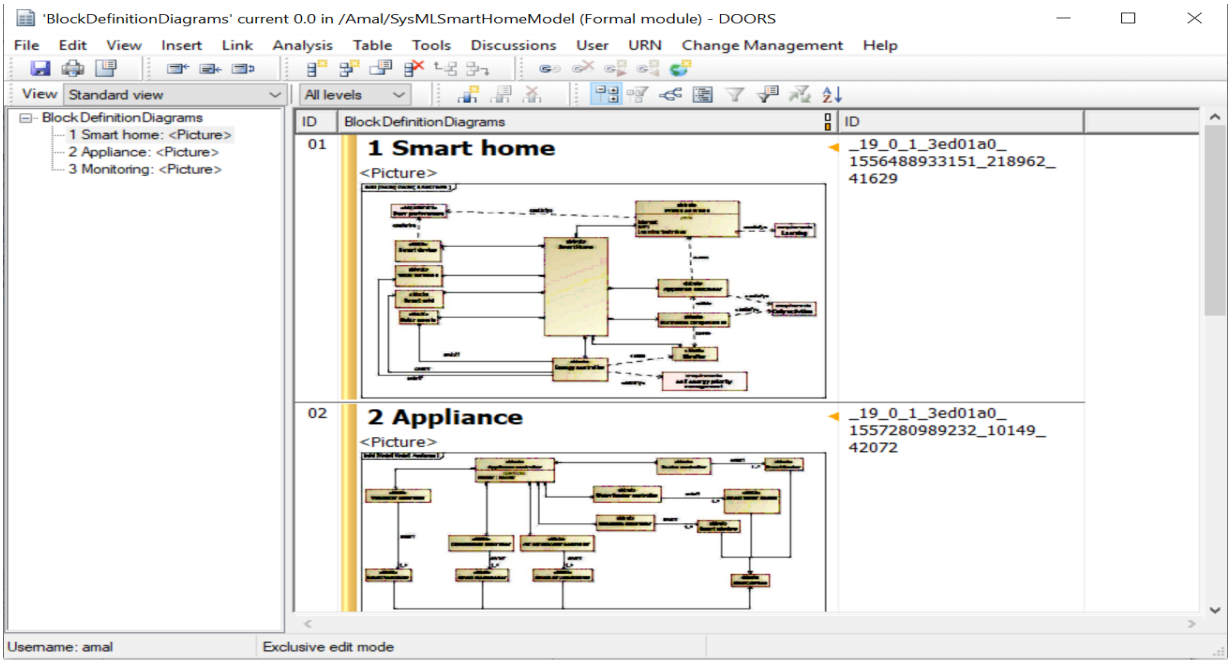

Figure 8.5: Block definition diagrams formal module.

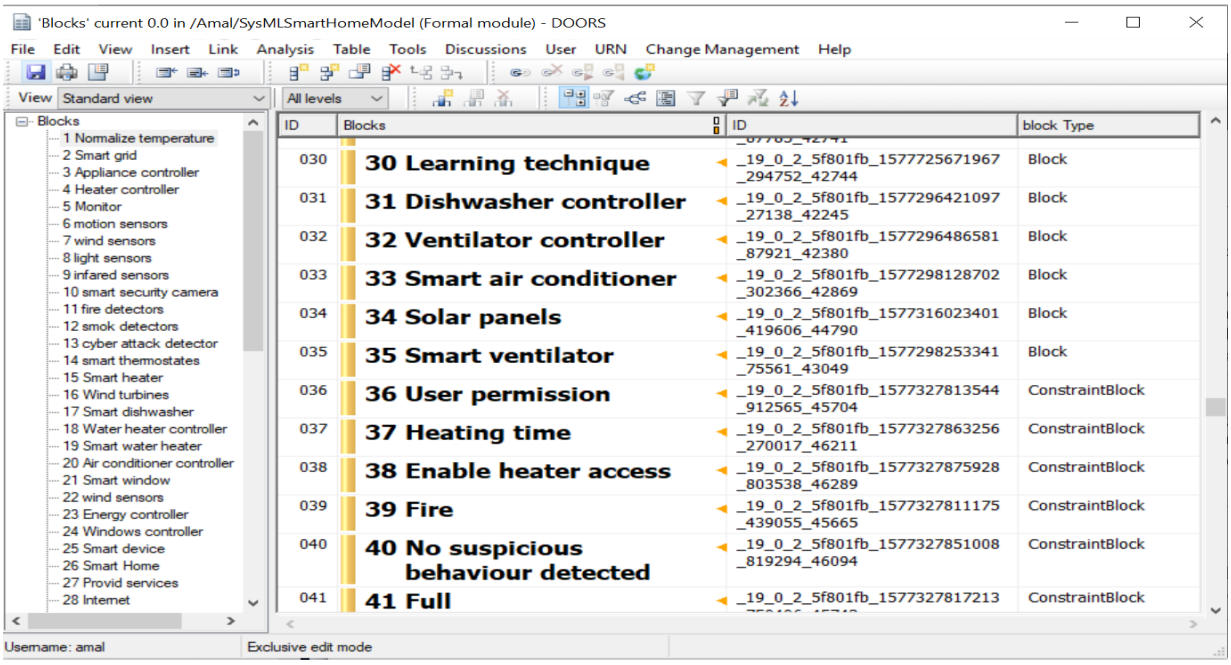

Figure 8.6: Blocks formal module.

Moreover, all relationships are achieved using DOORS links represented by small red or orange triangles next to each object. In this way, we automatically capture the parts

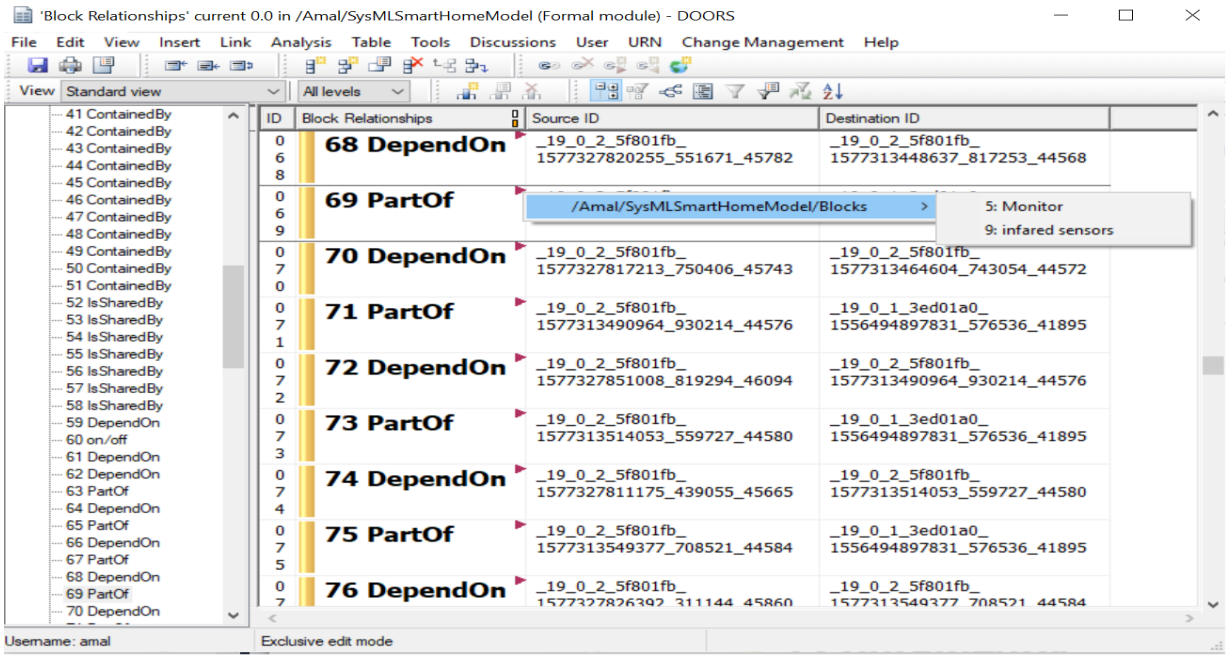

Figure 8.7: Block relationships formal module.

considered relevant of a SysML model in the DOORS database, including diagrams, blocks, requirements, and their relationships.

Importing the GRL model results in 76 intentional elements and two actors imported in their respective formal modules (Fig. 8.8). These elements are linked to their diagrams via 76 links and linked to each other using 155 links imported into the intentional elements associations module, as shown in Fig. 8.9.

The main result of this step is that GRL and SysML models of SHMS are imported into the DOORS database. At this point, traceability between the imported models can be created manually and then managed to check system completeness and consistency through these different views.

8.2 Checking Consistency and Completeness

In addition to the imported relationships, developers can take advantage of DOORS to create, manage, and exploit traceability links between the imported modules and bridge the gaps between different system views, including goals, requirements, design elements, and possibly others. DOORS comes out of the box with many impact and traceability analysis features that can be used here. However, DOORS and other RMSs, by default, do not understand the semantics of the objects being traced, and hence cannot out of the box detect inconsistency and incompleteness issues. Such detection mechanism must be created to exploit intra-model links (imported automatically, as described in the previous section) and inter-model links that must be added manually (or possibly semi-automatically using

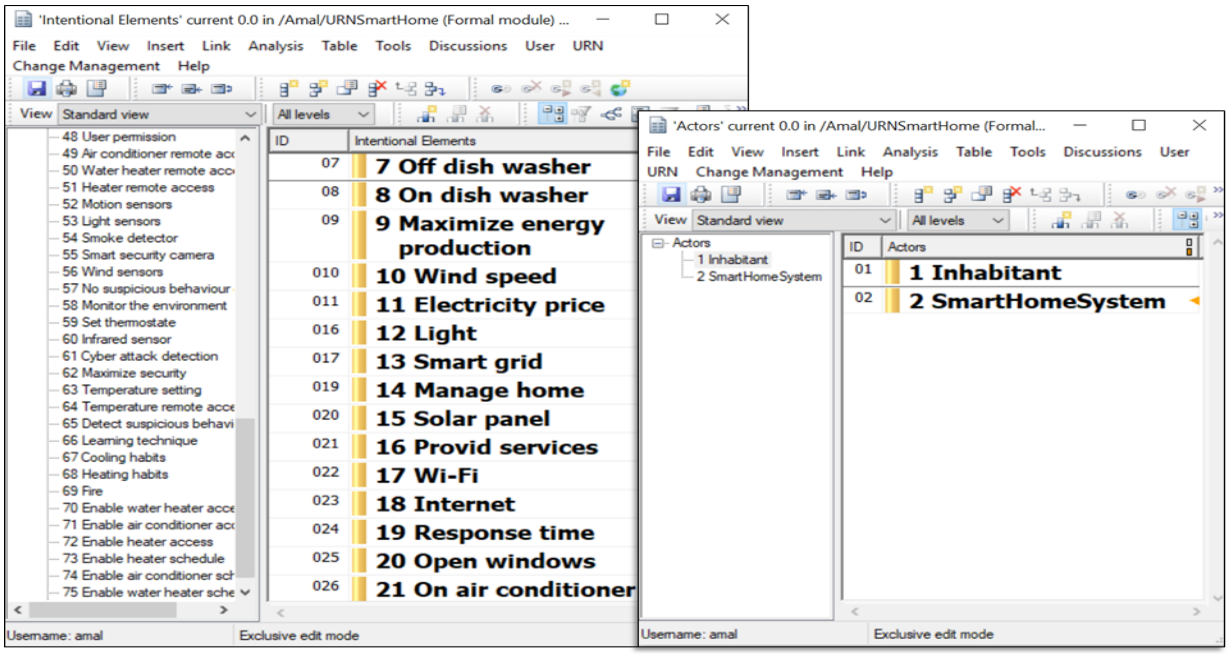

Figure 8.8: Intentional elements and Actors formal modules.

specialized information retrieval techniques such as those described in a recent journal special issue [110]).

In order to automate analysis that exploit inter-model and intra-model links, we first need to identify the types and directions of these links using a Traceability Information Model (TIM) and consistency rules. This was previously done in our case in Section 4.2.3. Second, we manually connect the inter-model elements using the identified typed links. Then, we convert the identified rules into executable programs (in DXL for DOORS) that navigate these links and report on inconsistency and incompleteness issues.

8.2.1 Managing Inter-Model Traceability Links

In order to check the completeness and consistency of the SHMS, we exploit Rational IBM DOORS tool to create traceability links between the GRL, requirements, and block definition diagrams based on the specified TIM (blue links in Fig. 4.4). To satisfy the directions and types of the links specified in the TIM, we have exploited the 9 link modules imported from GRL and SysML models to create specific DOORS linksets, as follows:

1. A linkset in the Satisfy link module from the Intentional elements module to the Blocks module, as shown in Fig. 8.10. In this linkset, we have created the following *Satisfy* links between intentional elements and blocks modules.

    - From the Maximize security softgoal to the Smart security camera block;
    - From the Low-temperature and High-temperature indicators to the Smart thermostat block;

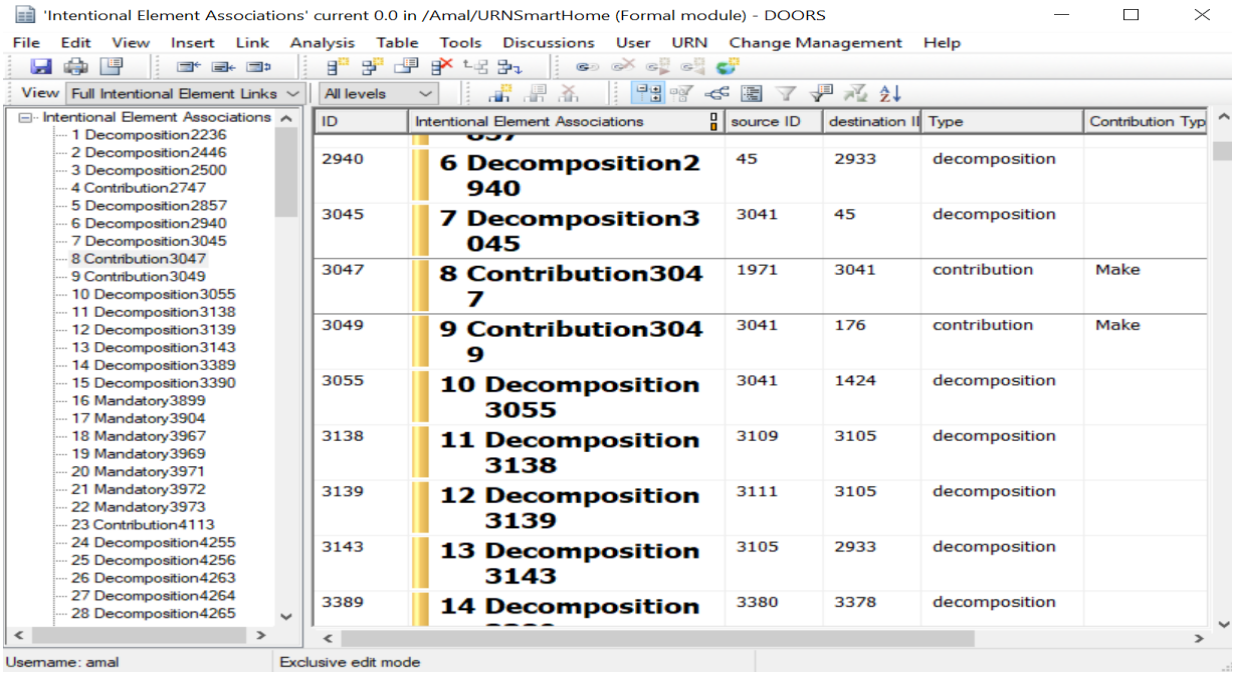

Figure 8.9: Intentional elements association formal module.

- From the On ventilator and Off ventilator tasks to the Smart ventilator block;
- From the Learning technique task to the Learning technique block;
- From the Maximize comfort softgoal to the Smart device block;
- From the Monitor the environment goal to the Monitor block (but we did not link all their children to the related block/intentional element);
- From the Normalize goal to the Normalize temperature block (but we did not link its related goals and tasks to any block);
- From the Internet task to the Internet block; and
- From the Wi-Fi task to the Wi-Fi block. However, we did not link the owner block, Provide services, to any intentional element.

2. A linkset in the Satisfy link module from the Blocks module to the Actors module (Fig. 8.11). We have linked the System actor with the Smart home block and we assigned "True" to the ignore attribute of the Inhabitants actor in the Actors formal module (so it would not create completeness or consistency issues).

3. A linkset in the Satisfy link module from the Blocks module to the Requirements module (Fig. 8.12). These *Satisfy* links are created as follows: the Monitor block is linked with the Security and safety and Self-energy priority management requirements while the Assistance and support requirement is linked to the Provide services block. However, there are no links between their children (requirements or blocks) except for the links already imported from the SysML model between these children.

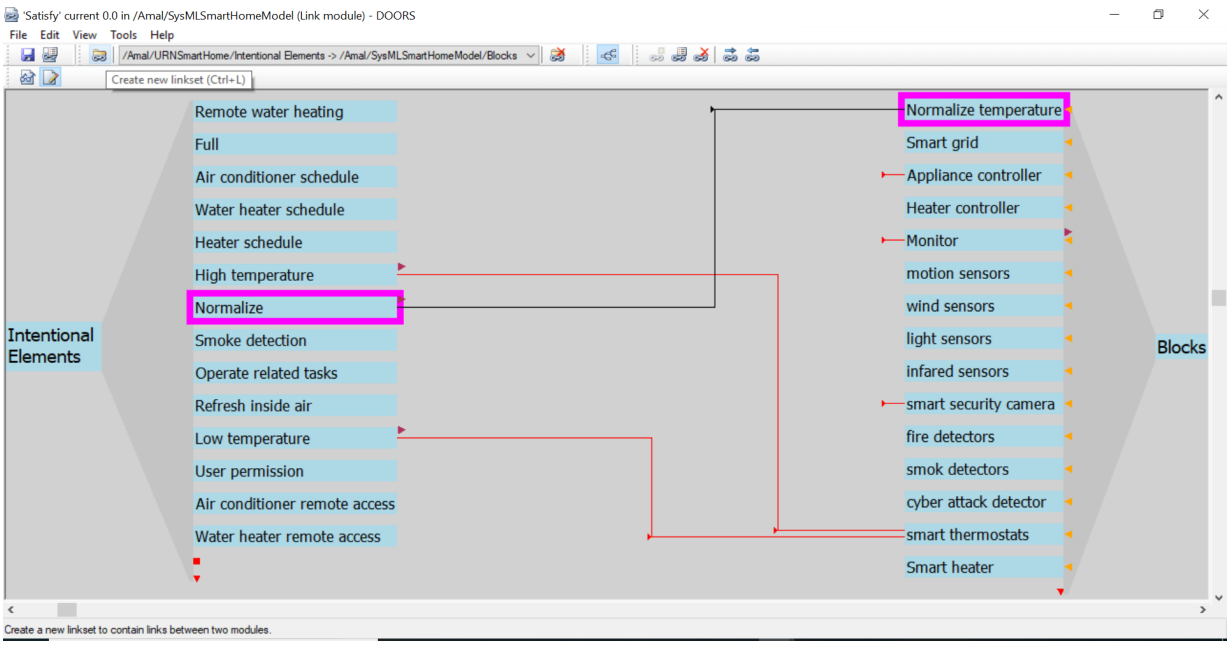

Figure 8.10: *Satisfy* linkset from the Intentional elements module to the Blocks module.

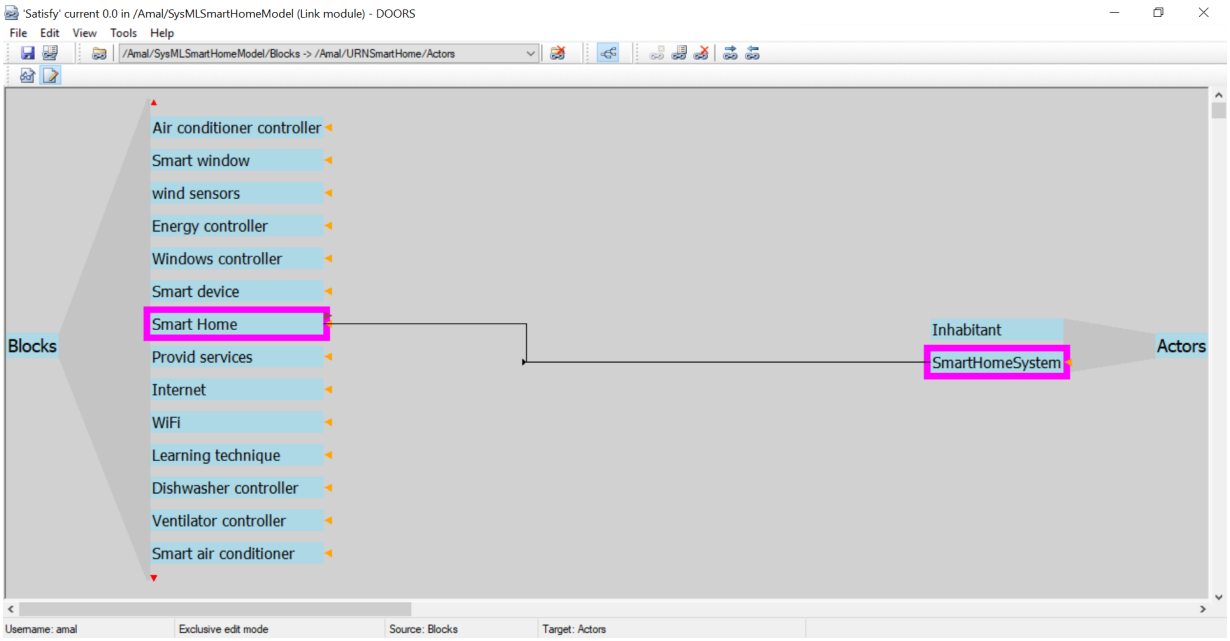

Figure 8.11: *Satisfy* linkset from the Blocks module to the Actors module.

4. A linkset in the Traced by link module from the Requirements module to the Intentional elements module. In this linkset (Fig. 8.13), the Manage appliances goal is linked to the Appliances management requirement while the Manage energy goal and the Maximize energy production softgoal are linked to the Self-energy priority management requirement. Also, we connected the Minimize consumption softgoal with the Annual consumption management requirement.

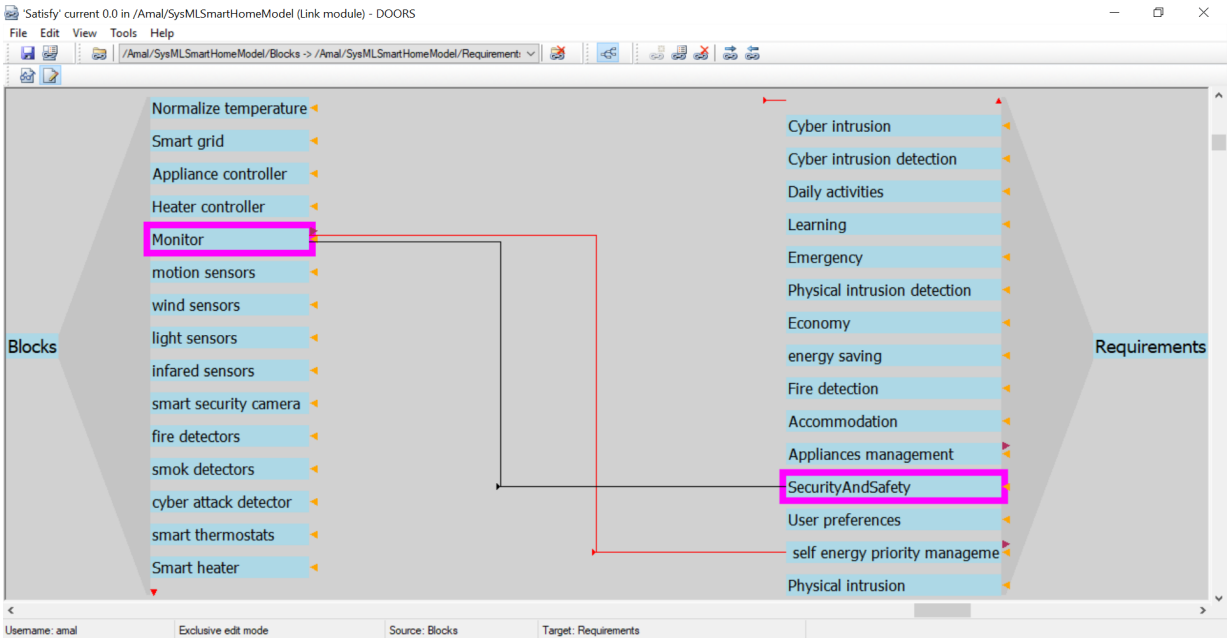

Figure 8.12: *Satisfy* linkset from the Blocks module to the Requirements module.

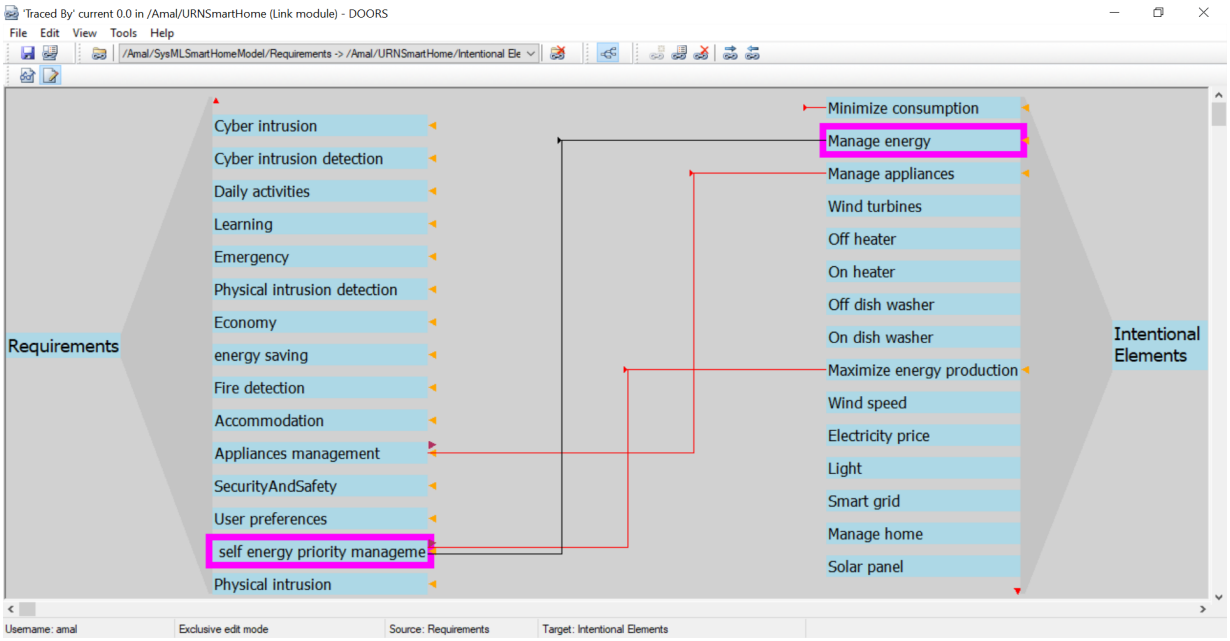

Figure 8.13: *Trace* linkset from the Requirements module to the Intentional element module.

Note that we have not connected all the elements together to explain the proposed rules and the benefits of the imported relationships. Moreover, establishing new traceability links can be done semi-automatically, for example, using natural language processing, which is outside the scope of this thesis.

8.2.2 Conducting Consistency and Completeness Checks

In order to access the objects of each formal module and check their links, DOORS' DXL language is used to analyze this information based on the specified TIM (Fig. 4.4) as well as the seven consistency and completeness rules informally defined for the CGS4Adaptation approach in Section 4.2.3. Listing C.3 on page 267 provides the DXL code used to read, analyze, and report issues about the consistency and completeness of GRL/SysML models. This program goes through each object in each formal module (Block, Intentional elements, Actors, and Requirements) following the existing links, according to the proposed rules. If the links of an object and its parents' links do not satisfy a rule, a problem is reported in a new attribute called Consistency and completeness of the violating objects. Recall from the last rule in Section 4.2.3 that parent relationships are inferred from ContainedBy, PartOf, and ChildOf links in SysML, and from Decomposition links in GRL.

After checking all explicit links for each object, the *implicit links* are explored in order to uncover potentially missing links between objects. Here, an implicit link is a relationship that could not be discovered easily by developers. For example, assume a block is linked to a requirement but not to an intentional element. We check whether the requirement is related to an intentional element. If yes, then this block has an implicit link to the same intentional element. The result of this analysis is reported in the Implicit link attribute for each object. Developers can then filter/flag these new attributes to add potentially missing links.

In order to save developers time and efforts, these automated checks can be conducted before and after creating new links in DOORS; developers can efficiently build their traceability links based on the analysis results, as explained in the next subsection.

8.2.3 Results

We take advantage of the DOORS RMS to create a new view called *consistency and completeness* for each formal module, in order to report the results of these checks in a usable way. In addition to the object names, this view shows the Ignore flag, the ConsistencyAndCompleteness and Implicit Link attributes populated in the previous section, as well as the identifier and name of the objects linked to the current object. For the linked objects, the Requirements module provides Block Info and Intentional Elements Info attributes, the Blocks module provides Requirements Info and Intentional Elements Info attributes, and the Intentional elements module provides Block info and Requirements info attributes.

Figure 8.14 shows part of the consistency and completeness view of the Intentional elements module. The Provide services goal is not linked to any block or requirement, as seen in the ConsistencyAndCompleteness attribute, while its children (the WiFi and Internet tasks) are connected explicitly to the WiFi and Internet blocks and implicitly to the User preference requirement. This requirement is satisfied by the Provide services block, which is the parent of the WiFi and Internet blocks in the Blocks module (Rule #7 in Section 4.2.3). This relationship is an implicit one that is imported from the SysML model (see Fig. 7.5)

and passed to all parts of the Provide services block during the consistency and completeness checks because these parts do not have any direct link to the requirements model.

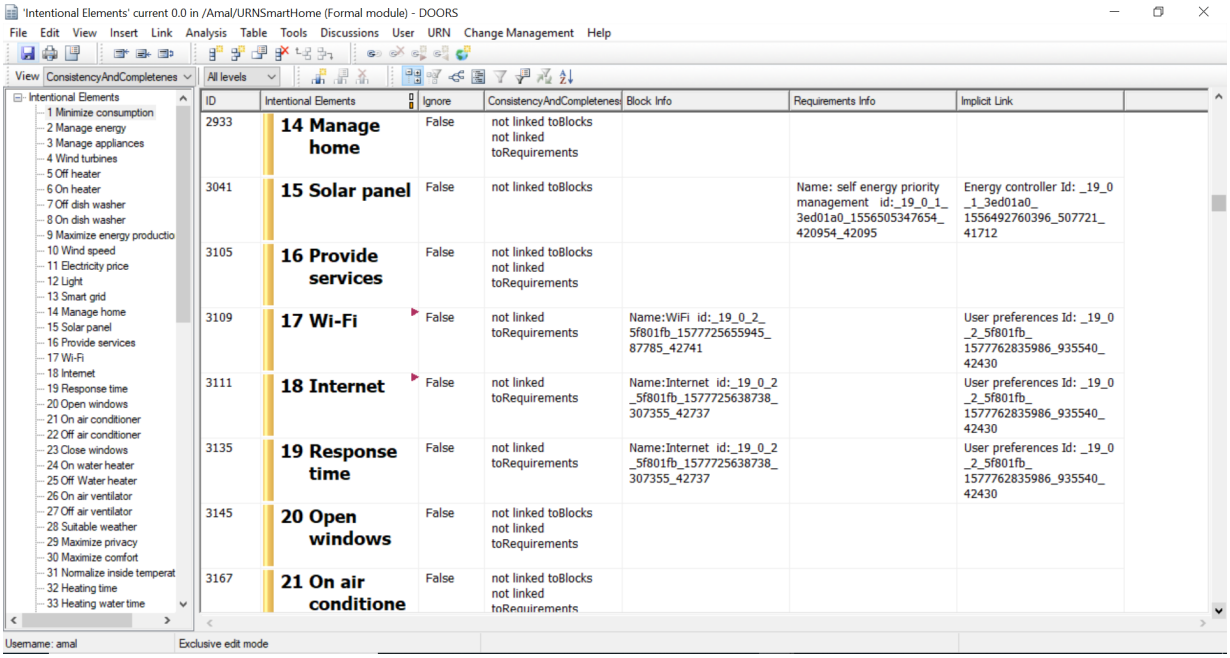

Figure 8.14: Results of the consistency and completeness checks of the Intentional elements module.

Moreover, the Solar panel task does not have any link (there is no triangles next to the object name, as shown in Fig. 8.14) but it is considered as linked to the Requirements and Blocks modules because its parent, the Manage energy goal, is linked to the Self-energy priority management requirement via a *Trace* link (Fig. 8.13). Using Rule #7, consistency and completeness checks propagated this relationship to all elements belonging to the Manage energy goal, including Solar panel, Smart grid, and Wind turbines. Moreover, the implicit link check discovered a *Satisfy* link from the Self-energy priority management requirement to the Energy controller Block (see Fig. 7.5), so it passed this link to all the intentional elements related to this requirement. As a result, the Solar panel, Smart grid, and Wind turbines tasks are linked to the Requirements and Intentional elements modules using indirect links that developers could not easily document.

Similarity, the Security and safety requirement is linked to the Monitor block (Fig. 8.12), so this relationship is passed on to all of its constituent requirements that have no relationships with the Blocks module. Figure 8.15 shows the test results where the Emergency and Physical intrusion detection requirements inherit the relationship with the Monitor block from their parent (the Security and safety requirement) while Fire detection does not since it is already connected to the Fire constraint block in the SysML model. In the same manner, implicit links were discovered through the Monitor the environment goal that was linked to the Monitor block, so these links are passed on to the Emergency and Physical intrusion detection requirements. However, the Fire detection requirement obtained the same implicit

link with the Monitor the environment goal but from the Fire block, which is part of the Monitor block.

Furthermore, please note that the Appliances management requirement is a *complete* requirement as it does not have any consistency or completeness issue.

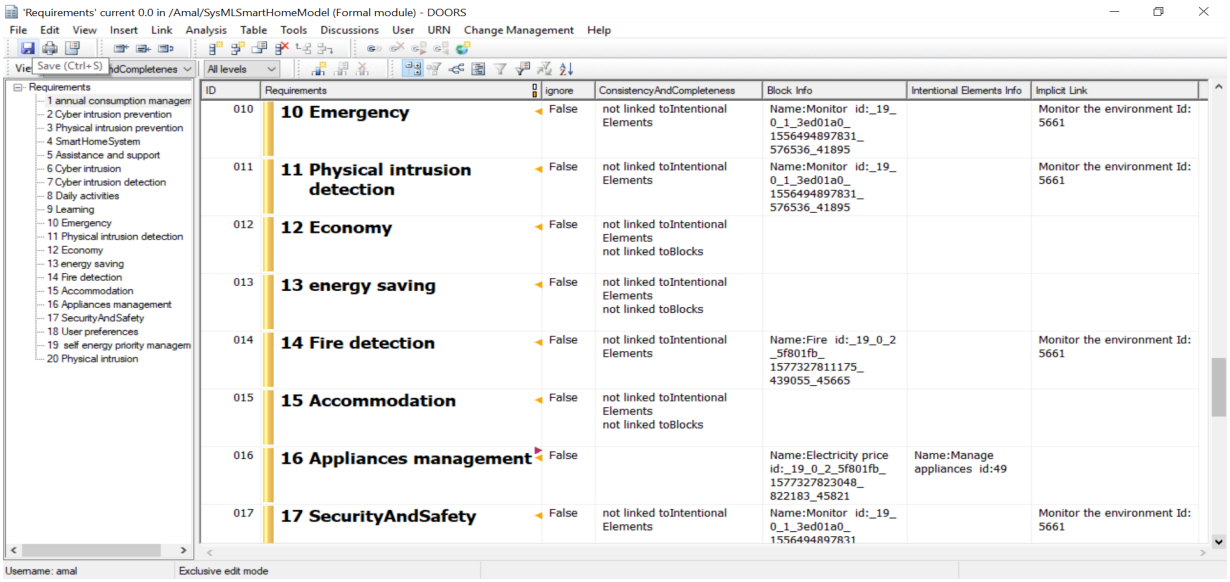

Figure 8.15: Results of the consistency and completeness checks of the Requirements module.

From the consistency and completeness view of the Blocks module (Fig. 8.16), the Monitor block is consistent with the Intentional elements and Requirements modules, and its condition is transferred to its parts/children, including Motion sensors and Wind sensors, during the check.

Figure 8.17 illustrates the consistency and completeness view of the Actors module, where the Inhabitant actor was ignored through the analysis by assigning "True" to its Ignore attribute. On its side, the Smart home system actor was linked to the Smart home block. The ConsistencyAndCompleteness attribute shows no problem, and the actors are considered consistent with the SHMS models according to the TIM and defined rules.

Automating the verification of consistency and completeness rules and of implicit links enables better support for conventional impact analysis and change management analysis by helping ensure that no link is missing and by showing relevant information about related elements in one usable view.

8.3 Further Actions Based on DOORS Filters

When checking the consistency and completeness of the GRL and SysML models, developers can further explore the results using *DOORS filters* to 1) display the results for specific conditions, and 2) apply suitable resolutions in the same view. Some filters useful for the CGS4Adaptation approach are presented below.

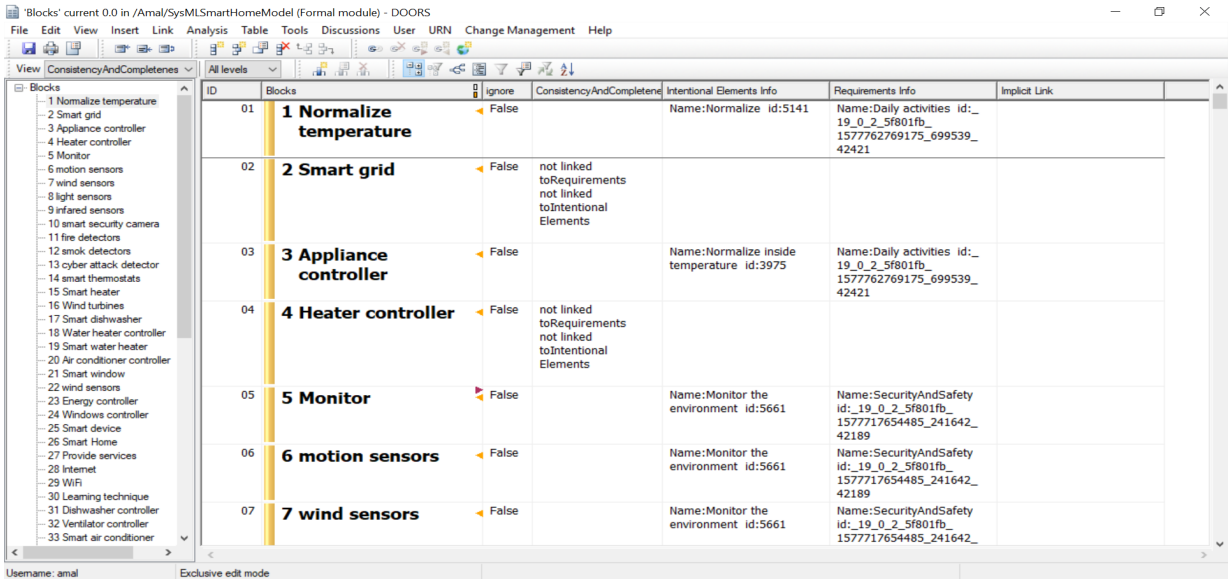

Figure 8.16: Results of the consistency and completeness checks of the Blocks module.

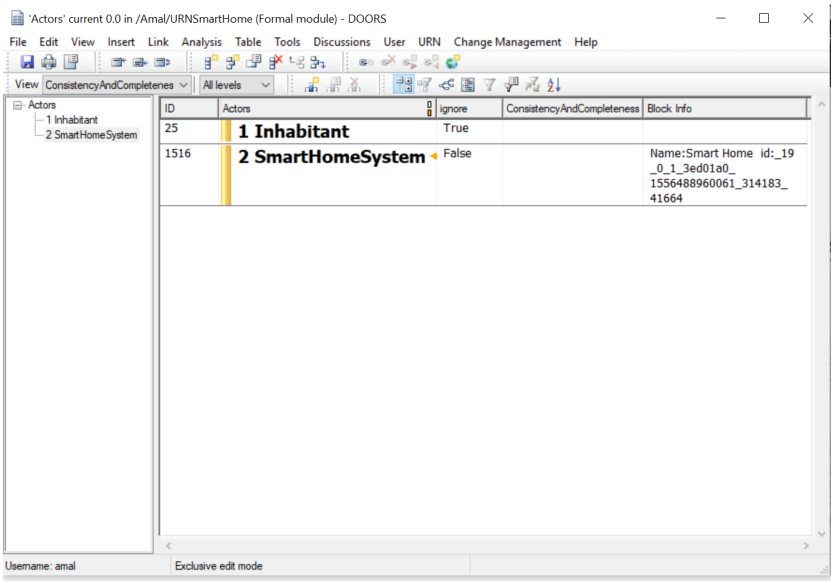

Figure 8.17: Results of the consistency and completeness checks of the Actors module.

To display design elements that are not linked to goals and requirements, a filter can be applied to the Blocks module to select the objects with empty Intentional elements info and empty Requirements info attributes, as illustrated in Fig. 8.18. The result lists 16 blocks and constraint blocks without a rationale (i.e., without a link to a requirement or goal). For each problematic element, the developer may then choose one of the following two resolutions:

1. Add a new Satisfy link;

2. Ignore this element during the checks by assigning "True" to its Ignore attribute; or

3. Delete the element from the system design (and such deletion will be communicated to the other developers through DOORS).

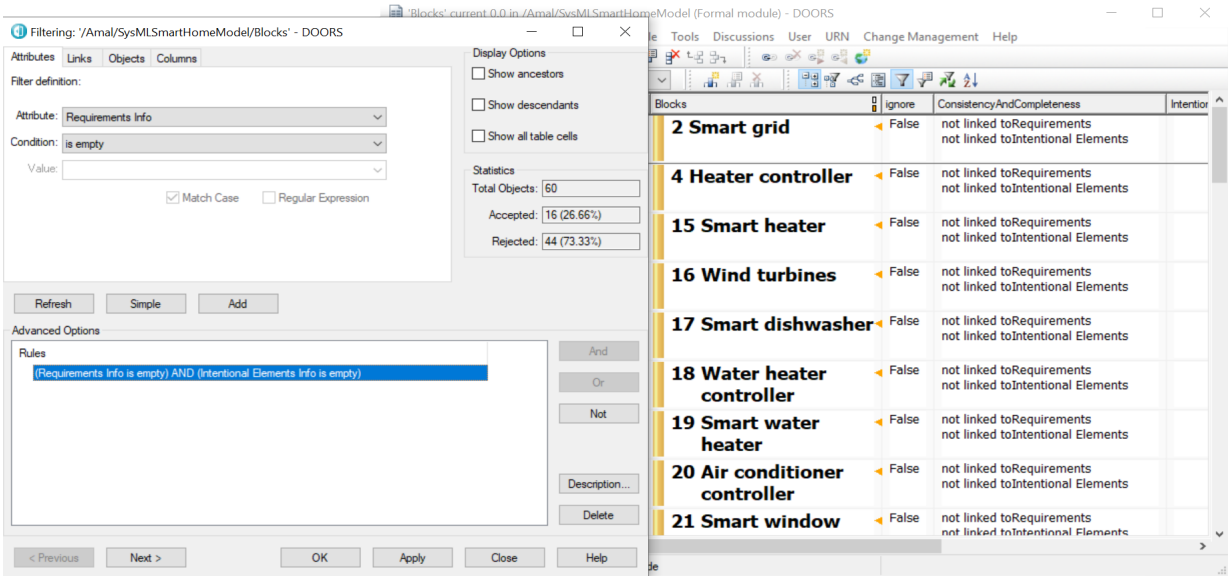

Figure 8.18: Filter: Blocks without a rationale.

To calculate and review only the satisfied (linked to design) and traced objects (linked to requirements) in the Intentional elements module, a filter that extracts objects with a non-empty block info and a non-empty requirements info, or a non-empty implicit link, can be applied. The result indicates 37 elements (softgoals, goals, indicators, or tasks) are complete and consistent through the GRL and SysML models, as displayed in Fig. 8.19 (right), while there are 39 incomplete or inconsistent elements (Fig. 8.19 (left)).

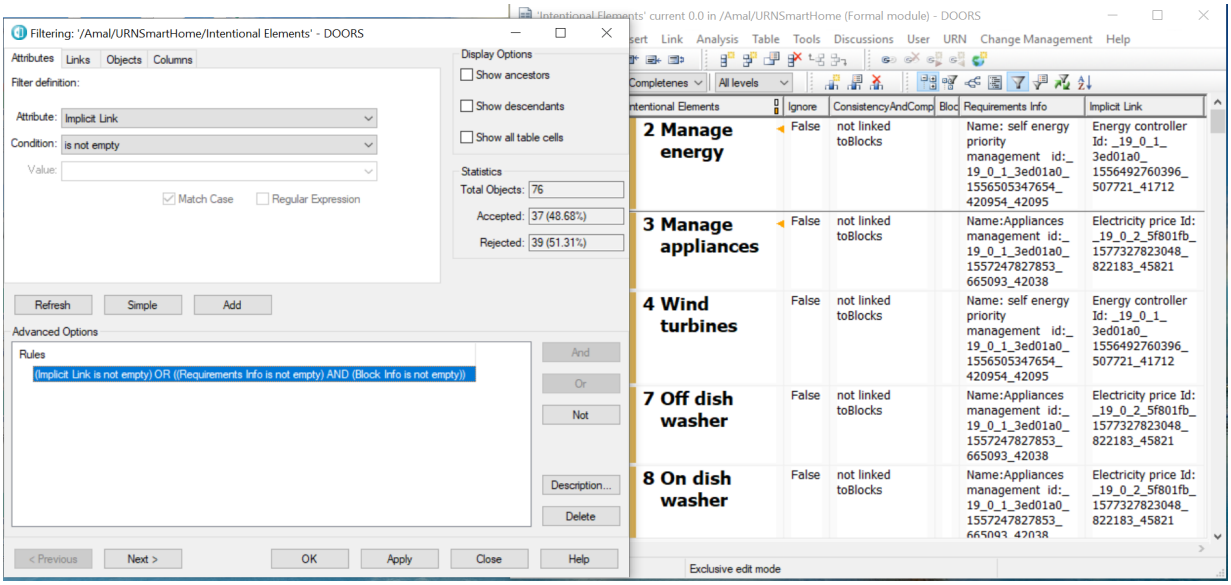

Figure 8.19: Filter: Consistent intentional elements.

The same filter is applied on the Requirements module to extract all the complete and consistent requirements. Figure 8.20 shows the result of this filter, including 14 complete requirements (satisfied by blocks and traced to goals) and 6 inconsistent or incomplete requirements.

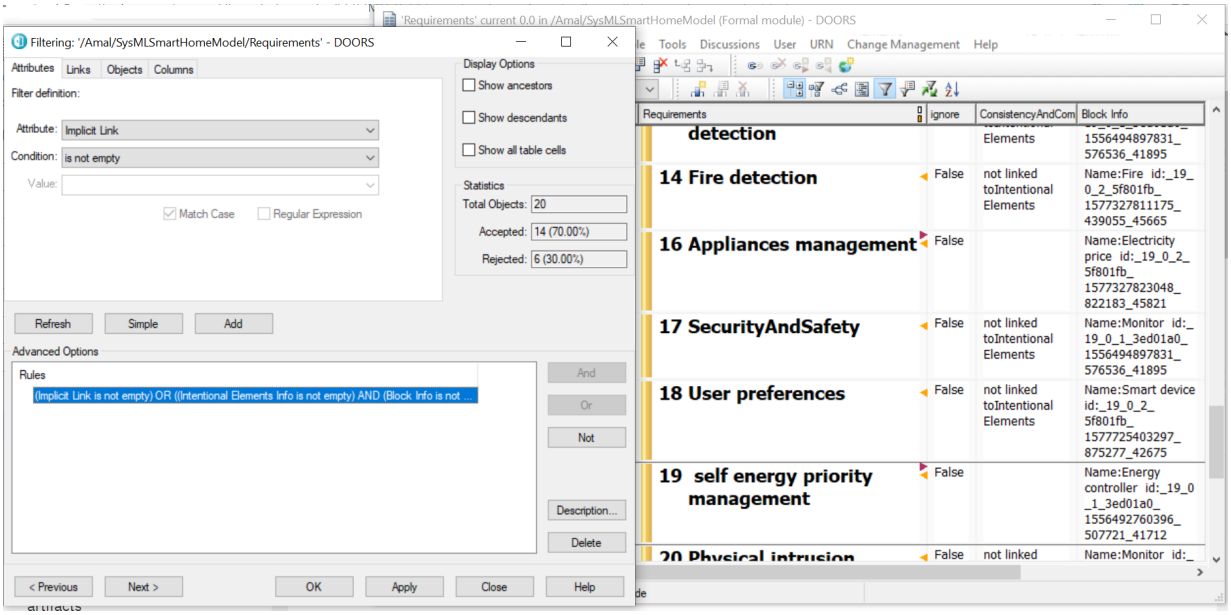

Figure 8.20: Filter: Complete requirements.

To list the requirement that have a rationale (linked to the goal model) but that are incomplete (not linked implicitly or explicitly to blocks), a filter that extracts the requirements linked to intentional elements but not blocks is applied, as shown in Fig. 8.21. The filter condition, which also exploits the Implicit Link, Block Info, and Intentional Elements info attributes, is shown in the Advanced Options of the filtering window. The result of applying this filter appears in the same figure, in the requirements window, where annual consumption management is the only incomplete requirement with a rationale. In this case, developers can 1) bridge the gap and link this requirement to a related block, 2) ignore this requirement during the checks by assigning "True" to its Ignore attribute (which is a good option for non-operationalized requirements such as "The color of the system shall be blue"), 3) delete the requirement, or 4) tolerate this issue and consider the project as not yet completed.

To list the incomplete requirements without a rationale, a complementary filter is applied, as shown in Fig. 8.22. The result shows 4 requirements with such issues.

8.4 Chapter Summary

In this chapter, we explained how GRL and SysML models are exported from modeling tools and imported into DOORS via DXL scripts. We then took advantage of DOORS

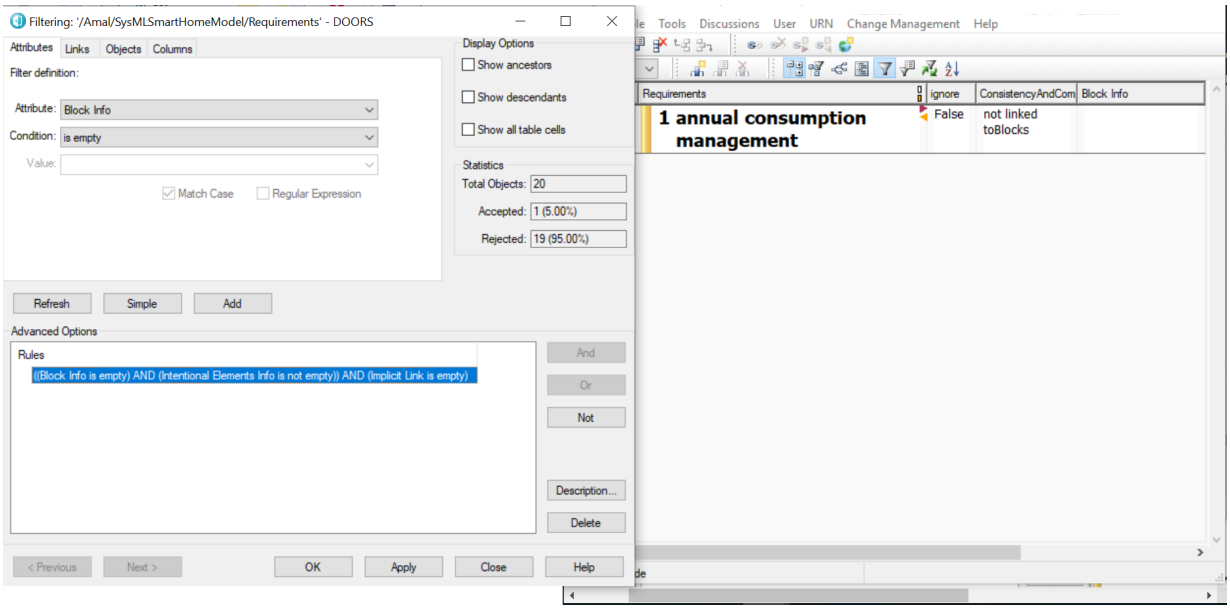

Figure 8.21: Filter: Incomplete requirements with a rationale.

to manage traceability between the imported models while demonstrating how imported and manually-created links can be exploited to automatically check the consistency and completeness of the GRL and SysML models of the SHMS, possibly leading to additional changes in the models so they are kept properly aligned.

The overall results show that importing GRL and SysML models, including block definition and requirement diagrams, into DOORS is feasible and beneficial. Moreover, the import of the elements and relationships specified in the TIM enables checking the consistency and completeness of GRL and SysML models through explicit links but also *implicit* links, often resulting from transitive relationships, which cannot easily be spotted by designers in large and complex models. The use of DOORS views and filters support usable management of the traceability information, as well as resolutions (linking, deleting, or ignoring an element) that must be made by designers. Clean, consistent, and complete links further facilitate common change management and impact analysis processes, likely saving developers' time and effort.

In the next chapter, we examine the scalability and performance of our approach to realize dynamic adaptations at runtime.

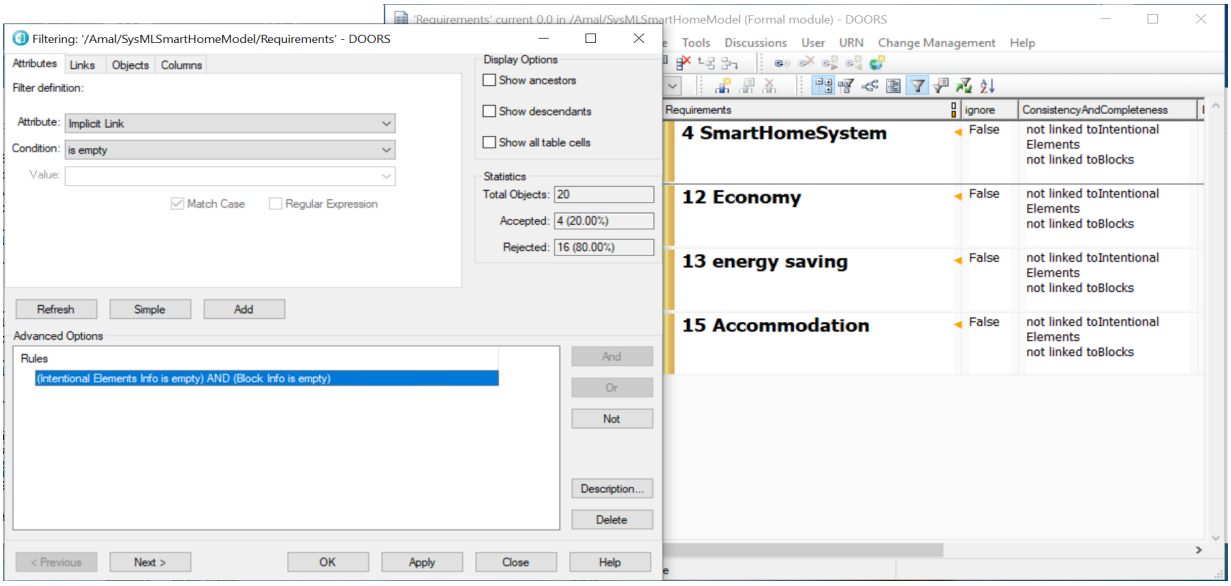

Figure 8.22: Filter: Incomplete requirements without a rationale.

Chapter 9

Runtime Performance and Scalability

This chapter presents an experiment conducted to assess the scalability and performance of the most computationally intensive part of the CGS4Adaptation approach, namely its optimization method, for runtime adaptation. In order to build large coherent models for this experiment, the GRL and feature functions of the SHMS models are duplicated and relationships between them are created using the GRL actor mathematical function presented in Equation 4.10. This enables to create models that are up to 10,000 times larger than the original SHMS model presented in Chapter 7, and hence to get empirical performance and scalability results for a wide range of model sizes.

9.1 Performance and Scalability Experiment

In order to evaluate the scalability and performance of our method for runtime adaptation, we optimize a management system that controls a set of smart home functions initially created in Chapter 7. For this evaluation of runtime adaptation, code in a programming language running in an embedded optimizer is used instead of the interactive IBM CPLEX environment used in the previous chapters. In particular, we selected the *CP Optimizer* [84] library of IBM CPLEX ILOG Optimization Studio, which provides a constraint programming solver with a Java interface.

9.1.1 Experiment's Goal and Metrics

Because the optimization process is the most computationally intensive part of CGS4Adaptation, the goal is to asses how it affects system performance using models of various sizes at runtime. For each issued solution, we assess these four metrics:

1. System performance, measured by:
   (a) **Solving Time**, which is the time spent by the optimizer to extract and solve the model.

(b) **Total Program Time**, which is the time spent in the Java program that invokes the optimizer library. The Total Program Time hence includes the Solving Time plus the time used to start and execute the Java program.

2. Memory required, measured by:

   (a) **Optimizer Memory Usage**, which is the memory used by the optimizer instance and related dynamic objects.

   (b) **Program Memory Usage**, which is the memory used by the program that contains the model and invokes the optimizer library.

9.1.2 Experiment Design

To measure the performance of the optimization method, we create large models by joining several copies of the SHMS model. To create one management system model that controls many smart homes, the generated large models combine many copies of the GRL and feature models, with renamed elements, and belonging to different actors integrated using Equations 4.10 and 4.11. Using this method, each new large model is a management system that controls a set of houses. To duplicate the same designed solutions (created in Chapter 7) for each new smart house (SH), we recreate the three parts of the optimization model (the objective function, decision variables, and constraints) by:

1. Creating new variables to represent the decision variables/free features.
2. Duplicating the same GRL and feature functions, but using the new decision variables and the original ancillary variables (which capture the context through indicators).
3. Rebuilding the constraints using the new variables.
4. Passing the ancillary variables to each new SH to express the same context for all homes every time we optimize the model.

Since the indicators' variables represent the context and are passed as constants to the optimizer via the ancillary variables, they are *not* duplicated in the large model and hence the same context is shared between all created SH models. This way, the overall satisfaction can be computed and the generated SH models can be validated. Note that ancillary variables do not impact the optimization process as they are evaluated using the current context, before optimizing the model.

Using this duplication approach, the number of duplicates of each managed device equals the size of this model. For example, a model with 100 smart homes contains 100 windows, 100 heaters, 100 water heaters, etc. Each of these devices has two tasks and is managed by the generated equation.

Note that the optimization model of a single smart home has 17 decision variables and 18 constraints, corresponding to a GRL model with 76 intentional elements and 110

relationships (59 decompositions, 37 contributions, and 14 dependencies) combined to its feature model with 48 features (including 32 leaf features shared with the GRL model) and 56 relationships (38 mandatory, 9 optional, 8 include and 1 exclude). The total number of goal/feature elements and relationships in a single SH model is 258, after removing the shared 32 tasks/features.

9.1.3 Experiment Procedure

The experiment procedure is described in Algorithm 1.

Algorithm 1 Experiment procedure

```
1: Contexts = {SameContext, RandomContext}                    ▷ Context sources
2: ModelSizes = {1, 3, 10, 32, 100, 316, 1000, 3162, 10000}   ▷ Predetermined model sizes
3: procedure EXPERIMENT
4:     Create new Excel document
5:     for all context ∈ Contexts do
6:         for all size ∈ ModelSizes do
7:             for iteration = 1 to 10 do
8:                 Create decision variables (×size)
9:                 Create GRL function using decision and ancillary variables
10:                Create feature function (×size) using decision variables
11:                Rebuild constraints using the new variables
12:                Feed context to the generated model
13:                Invoke CP Optimizer on the model
14:                Add the context and the related features to the Excel sheet
15:                if the model is solved then
16:                    Add runtime measurements to Excel document
17:                end if
18:                Clear the model
19:            end for
20:        end for
21:    end for
22: end procedure
```

We selected nine sizes representing the numbers of SH models in one integrated model (1, 3, 10, 32, 100, 316, 1000, 3162, 10000). These sizes correspond to different powers of 10, representable on a logarithmic scales (10^0, $10^{0.5}$, 10^1, $10^{1.5}$, ..., 10^4). The largest size represents a goal/feature model with $10,000 \times 258 = 2.58$ million elements, which is several orders of magnitudes larger than most GRL/feature models for realistic models.

We implemented this procedure using Java and the Constraint Programming (CP) Optimizer library, under Eclipse Modeling 2019. The computing environment used to conduct this experiment and collect the data was a laptop with MS Windows10-x64, an Intel Core i5-8250U CPU at 1.60GHz, and 8 GB of memory.

In order to check that the generated models are correctly represented by GRL reasoning functions for different sizes of smart home sets and that there is no missing operator or element, we tested the behaviour of these models and functions according to several environmental contexts. The models of our nine pre-selected sizes were all validated using the same *specific scenarios* used to validate the SHMS optimization model (Section 7.7.2). The solutions issued by each model were inspected and compared to the solutions of the original SH model until we obtained the same results, including the same satisfaction levels and selected strategies (the selected strategies could be different in the chosen alternatives whose contributions have the same impact on the overall satisfaction level). We also validated all created models using *random contexts* to check their reliability, and no unsolved context was detected.

After validating the generated models, many data points were collected and saved in the Excel document while executing the experiment: the identified metrics, the model size, the context variables including the indicators' variables and the related features, the model satisfaction level, and the selected features. At runtime, the Solving Time and Optimizer Usage Memory metrics were obtained from the CP Optimizer instance, while the Total Program Time and Program Usage Memory metrics were tracked externally to the library.

9.2 Experiment Results

A summary of the collected data (the identified measurements) is presented in Table 9.1, with visualizations in Figures 9.1 to 9.4. The **Msr** column in Table 9.1 refers to the four metrics defined in Section 9.1.1, the first two being measured in seconds and the last two in megabytes.

Table 9.1: Averages of collected data, for 10 iterations
Measurement=Msr, S=Sample Type, R=Random, T=The same context, A=All samples

Msr	**S**	**1**	**3**	**10**	**32**	**100**	**326**	**1000**	**3162**	**10000**
1.a (s)	T	0.012	0.015	0.021	0.052	0.139	0.478	3.51	67.12	801.56
	R	0.012	0.02	0.023	0.066	0.144	0.510	4.13	64.23	728.17
	A	0.012	0.017	0.022	0.059	0.141	0.494	3.82	65.68	764.87
1.b (s)	T	0.052	0.048	0.058	0.105	0.272	0.764	4.46	71.06	835.78
	R	0.067	0.06	0.057	0.176	0.255	0.794	5.08	68.16	761.88
	A	0.059	0.055	0.057	0.140	0.263	0.779	4.77	69.61	798.83
2.a (MB)	T	3.88	4.72	7.71	17.99	47.40	145.76	364.10	1102.08	2147.48
	R	3.77	4.42	6.74	14.60	40.85	118.93	337.31	1039.12	2147.48
	A	3.78	4.44	6.87	15.07	42.18	117.55	340.19	1060.23	2147.48
2.b (MB)	T	6.22	6.34	6.5	6.64	6.93	15.51	21.33	26.64	63.56
	R	6.2	6.35	6.51	8.23	8.46	17.52	21.35	25.09	57.05
	A	6.21	6.35	6.5	7.43	7.69	16.51	21.34	25.86	60.31

Figure 9.1 shows the log of the average Solve Time (in seconds, and based on 10 iterations) for each model size using the same predetermined scenario/context, while Fig. 9.2

shows the log of the average Solve Time for 10 different random contexts. When using the same input (Fig. 9.1), there are small variations within regions where the model size is smaller than 100 while the areas with a model size higher than 100 show almost no differences. However, there are clear variations in all regions when using random contexts, as expected (Fig. 9.2).

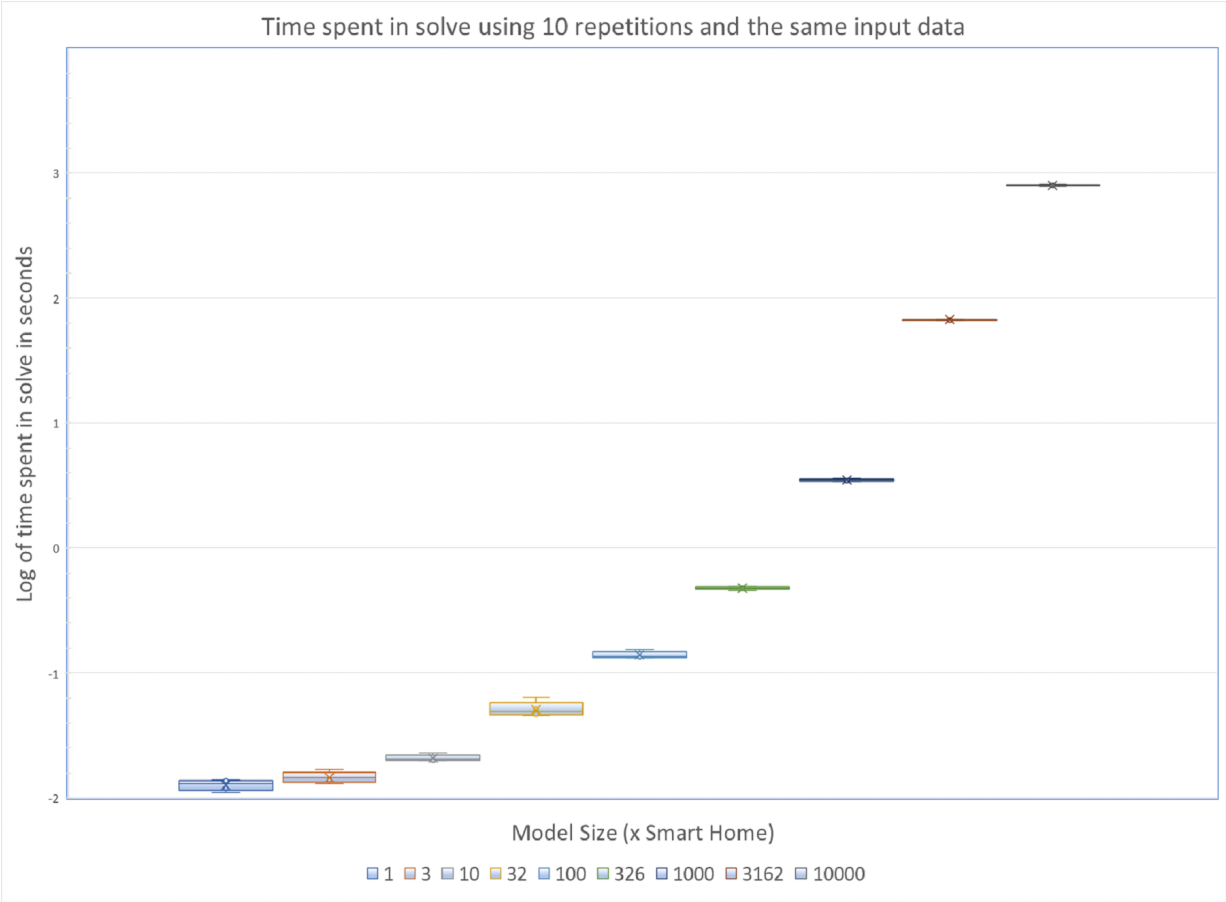

Figure 9.1: *Solving Time* for the nine model sizes, averaged over 10 iterations of the same context

Similarly, Fig. 9.3 presents the logs of the average Solving Time and average Total Program Time, for each model size. Each data point in the figure represents an average of 10 iterations, using the same context (where variation was minimal). The figure shows that the Solving Time and Total Program Time have the same pattern, with a fairly stable program overhead that is more noticeable for small sizes (in proportion to the Solving Time). Both Solve Time and Total Program Time increase exponentially with the model size.

The Program Memory Usage the Optimizer Memory Usage have however different patterns, as shown in Fig. 9.4. The figure shows the average of the usage memory of 10 optimization processes using the same context for each model size.

There are important observations to be made from the figures and Table 9.1:

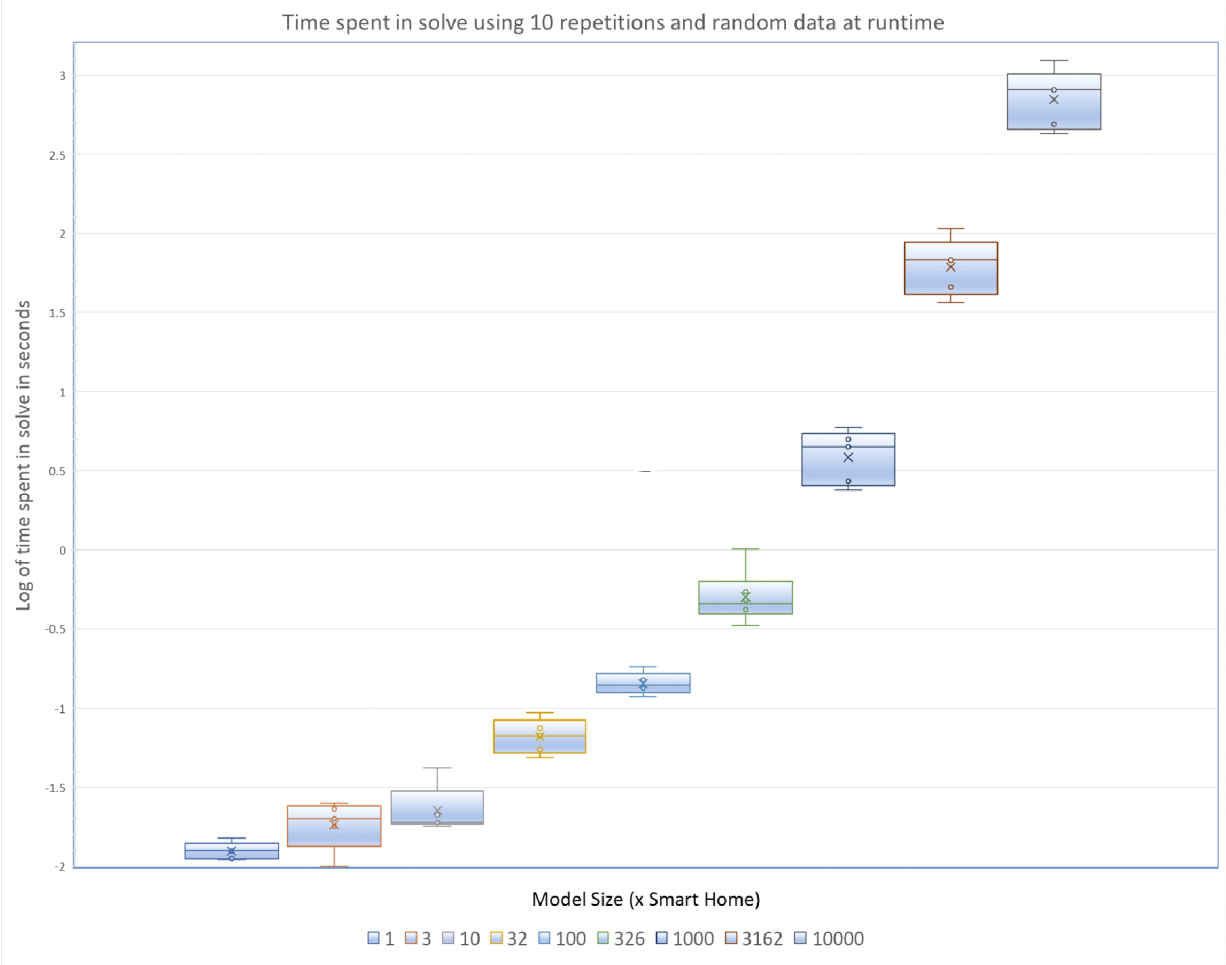

Figure 9.2: *Solving Time* for the nine model sizes, averaged over 10 random contexts

- The optimization method for runtime adaptation provides a fast average Solving Time for one smart house model (12 ms). Even a model the size of 100 SHs (i.e., containing 25,800 modeling elements) can be solved in about one tenth of a second.
- The performance is good even in the presence of a specific time budget. For example, given a half-second time budget, the results suggest that models with about 84,000 modeling elements (326 SHs) can be handled if only Solving Time matters (i.e., when the CP Optimizer is already running), and that about 50,000 modeling elements (about 200 SHs) can be handled in terms of Total Program Time (i.e., when the Java program also needs to be started and the model created in memory).
- For model sizes greater than 10 SHs, the Optimizer Memory Usage, related to the searching of the solution space, quickly outgrows the Program Memory Usage, which remains much more stable.
- When the model size reaches 10,000 SHs (i.e., 2,580,000 modeling elements leading to 170,000 decision variables and 170,001 constraints), the sum of the Program Memory Usage and Optimizer Memory Usage (as we need both) increases from a handful of

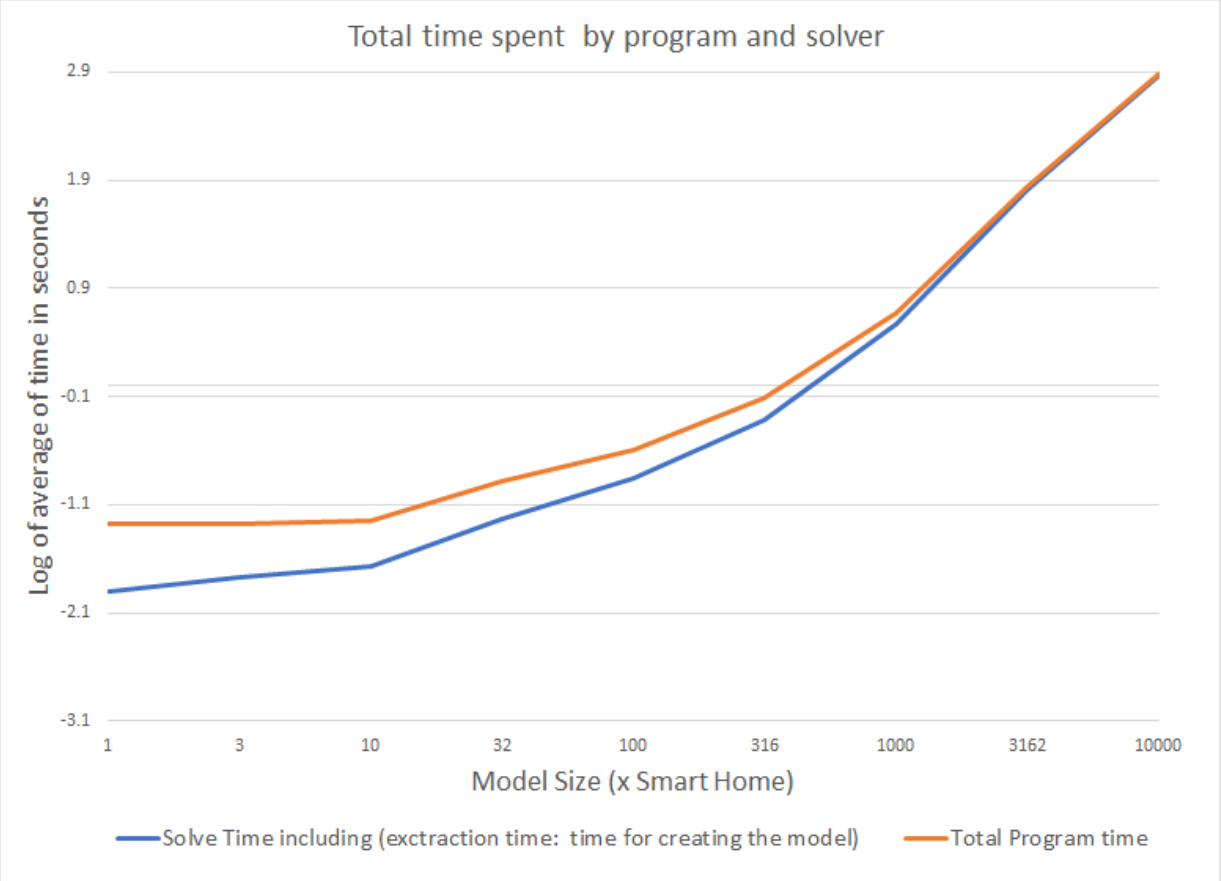

Figure 9.3: *Solving Time* and *Total Program Time* for the nine model sizes, averaged over 10 iterations of the same context

megabytes to about 2.2 gigabytes, with a Solving Time of about 12 minutes. This obviously goes beyond what is usually affordable for runtime adaptation in most systems, so there exists model sizes where this approach shows its limits.

The overall result shows the scalability of using CGS4Adaptation's optimization method at runtime for realistic models, and beyond. In addition, because function performance decreases as the size of the function increases, it is important to keep simplifying the generated functions as much as possible, and further opportunities to do so will be mentioned in the future work (Section 11.3).

9.3 Chapter Summary

In this chapter, we assessed the performance and scalability of CGS4Adaptation's optimization method for runtime adaptation. The GRL and feature model functions generated from the SHMS case study were duplicated to build large models and solved using Java and the *CP Optimizer* [84] library of IBM CPLEX ILOG Optimization Studio. The results demon-

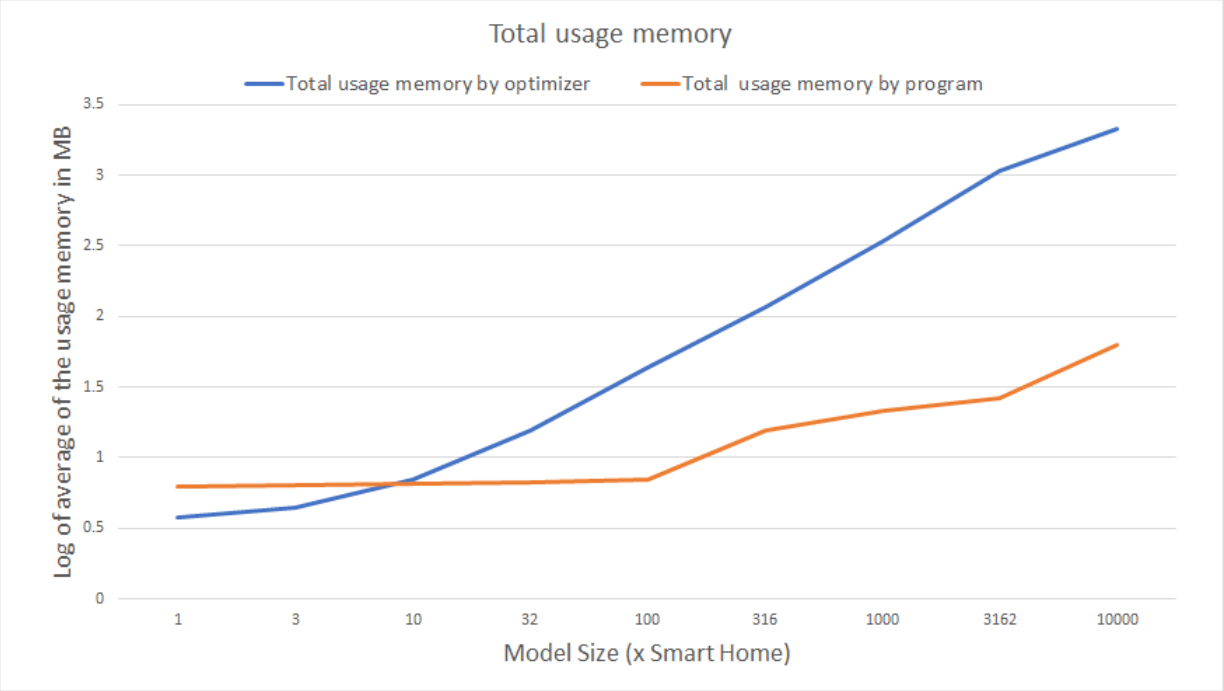

Figure 9.4: *Program Memory Usage* and *Optimizer Memory Usage*, averaged over 10 iterations of the same context

strate the good performance of the method for large realistic models and the scalability to models larger than what is typically found in a goal/feature modeling context.

The next chapter will further compare the CGS4Adaptation approach as a whole with closely-related work, and discuss the main threats to the validity of this thesis.

Chapter 10

Discussion

In this chapter, we compare the CGS4Adaption approach with related work along the main criteria identified in the literature review (Chapter 3). We also discuss important threats to the validity of the experimentation.

10.1 Comparison with Related Work

This section adds the CGS4Adaption approach to the comparison of related work presented in Chapter 3. It also explains how the approach brought forward in this thesis fills part of the identified research gap.

The papers selected in the literature review were assessed via four types of criteria: adaptation terms, modeling dimensions, model integration, and reasoning methods. Tables 10.1 to 10.4 highlight the results of the CGS4Adaption evaluation against these criteria, along with the original content of Tables 3.16 to 3.19 on the evaluation of related work, for convenience.

10.1.1 Adaptation Terms

In terms of adaptation (Table 10.1), the research gaps observed for related work include the following:

1. all these model-based approaches (Note that approaches of Qian et al. [138] and Soyler and Sala-Diakanda [149] are not model-based approaches) embedded their adaptation strategies at design time (Close adaptation type);
2. all the model-based approaches but one (Baresi et al. [31]) used rules or conditions to trigger the adaptation process (Decision (Analyze activities));
3. all the approaches but one (Qian et al. [138]) used rules or conditions to select among the strategies at runtime (static selection decision), which required other techniques to solve the conflicts between these conditions/rules; and

4. none of the adaptation approaches enhanced the decision-making mechanism itself (rules, conditions, or equations) at runtime (making adaptation approach).

Table 10.1: Adaptation terms related to each selected method, including CGS4Adaption. (C=Closed, O=Open, ?=Not provided, Y=Yes, N=No, P=Partially, Dy=Dynamic, Sta=Static, M=Making, A=Achieving, Md=Middleware, Ap=Application, Sr=Service, St=Structure, W=Weak, Rt=Reactive, Pt=Proactive, Co=Continuous, Ad=Adaptive)

Adaptation terms	Ingram et al. [88]	Ahmad et al. [3]	Morandini et al. [116]	Qian et al. [138]	Ramnath et al. [141]	Meacham [113]	Ribeiro et al. [142]	Baresi et al. [31]	Lopes et al. [106]	Soyler and Sala-Diakanda [149]	Baresi and Pasquale [32]	CGS4Adaptation
Adaptation Type	C	N	C	O	C	C	C	C	?	O	C	O
Model-based	Y	Y	Y	P	Y	Y	Y	Y	Y	Y	Y	Y
Decision (Analyze / Selection activities)	?	Sta	Sta	Dy	Sta	Sta	?	Sta	?	?	Dy/ Sta	Dy
Adaptation approach	M	M	M	M	M	M	M	M	M	M	M	Semi A
Layer	Md	Ap	Ap	Ap	Ap	Ap	?	?	Md	Ap	Ap	Ap
Artifact	Sr/St	Sr	Sr	Sr	Sr	Sr	?	St	Sr/St	Sr	Sr	Sr
Cost-impact	W	W	W	W	W	W	?	W	W	W	W	W
Temporal adaptation	Rt	Rt	Pt/Rt	Rt	Rt	Rt	?	Rt	Rt	Rt	Rt	Pt/Rt
Temporal monitoring	Ad	Co	Co	Co	?	Co	?	Co	Co	Co	Co	Co
Human intervention	N	N	P	N	N	N	?	P	?	?	P	N

CGS4Adaption provides an *open* adaptation approach that can deal with unforseen situations at runtime by producing, through the runtime optimization method, new suitable strategies that have not been foreseen or embedded into the system design at development time. The decision-making mechanism is *dynamic* since the equations corresponding to goal and feature models are used instead of conditions/rules to activate the adaptation process and select the best among many alternatives, at runtime.

In term of adaptation approach, although the functions themselves are embedded at design time and their structure is not changed at runtime ("Making" approach), the available features are managed through these functions. Developers can take advantage of this characteristic to develop decisions made at runtime by using tasks that change their impact depending on the runtime data fed to the formulas and representing specific tasks/features. This then leads to change the contexts that activate these features/tasks at runtime (*Achieving* approach).

For instance, although CGS4Adaption uses the arithmetic functions of goal and feature models as a decision mechanism, the decisions made by this mechanism can be influenced at runtime when the system benefits from inputs such as the Learning technique from the Smart home case study (Section 7.7) and activates some tasks using their results. In this case, the indicators were supplied with data from several sources: 1) sensors that monitor the environmental contexts, such as low temperature, 2) user input such as the status of the remote access and scheduling features (enabled or disabled) and the values of the heating or cooling times, and 3) the results of the LEARNING TECHNIQUES task that activate certain tasks (heater, water heater, and air conditioner devices) using population behaviours. Thus, the SHMS uses the results of Learning techniques when the temperature is low and all the remote access and the schedule features are disabled or not set (unforeseen context). This strengthens the adaptation approach by increasing its flexibility when facing unexpected contexts at runtime (a form of uncertainty).

Consequently, compared to the related work, CGS4Adaption is the only model-based approach that provides open and dynamic adaptation at runtime, without human intervention, while potentially enhancing itself via managed features that provide a "Semi-achieving" adaptation approach.

10.1.2 Modeling Dimensions

From the perspective of modeling dimensions (Table 10.2), by enabling the use and analysis of all goal model elements without involving rules or conditions on the reasoning process, CGS4Adaption is the only approach that manages multiple coherent goals and softgoals (FR and NFR) together, without restrictions (using free alternatives), and with guaranteed timeliness. This is because the produced strategies are validated by the feature function using external data (e.g., sensors), as well as the impact of the involved elements on each other and on the overall performance of the system. Additionally, CGS4Adaption increases system reliability by:

1. validating and enhancing system models using optimization and simulation methods to ensure that the system behaves as required by stakeholders;
2. providing formulas (a dynamic method) to resolve feature conflicts due to unforeseen contexts, instead of using conditions/rules that would reduce flexibility in avoiding these conflicts [93]; and
3. optimizing the generated functions at runtime using runtime data.

10.1.3 Model Integration

Using the model integration criteria (AC1-AC10, specified in Section 3.2.4 on page 54 and used in Table 10.3), unlike the related work, the CGS4Adaption approach integrates all

Table 10.2: Modeling dimension of the selected methods and CGS4Adaption. (Sta=Static, Dy=Dynamic, SDy=Semi-Dynamic, Rgd=Rigid, Cns=Constrained, S=Single, Ncn=Unconstrained, Tmp=Temporary, Prs=Persistent, Mlti= Multiple, E=External, I=Internal, FR=Functional requirement, NFR=Non-functional requirement, Gt=Guaranteed, NGt=Not guaranteed, ?=Unknown)

Adaptation terms	Ingram et al. [88]	Ahmad et al. [3]	Morandini et al. [116]	Qian et al. [138]	Ramnath et al. [141]	Meacham [113]	Ribeiro et al. [142]	Baresi et al. [31]	Lopes et al. [106]	Soyler and Sala-Diakanda [149]	Baresi and Pasquale [32]	CGS4Adaptation
Goal												
Evolution	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Dy	SDy
Flexibility	Rgd	Cns	Cns	Ncn	Rgd	Rgd	Rgd	Rgd	Rgd	Rgd	Cns	Ncn
Duration	Prs	Prs	Prs	Tmp	Prs	Prs	Prs	Prs	Prs	Prs	Prs	Tmp
Multiplicity	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	S	Mlti	Mlti	Mlti	Mlti
Dependency	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	S	Mlti	Mlti	Mlti	Mlti
Change												
Source	E	E	E&I	I	E	E	?	I	E&I	E	E&I	E&I
Type	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	-	NFR	FR& NFR
Mechanisms												
Timeliness	Gt	None	Gt	NGt	Gt	Gt	?	Gt	Gt	?	NGt	Gt

goal model elements with the SysML model elements in a RMS (DOORS), hence supporting traceability management, change management, and impact analysis. Automated consistency and completeness checks, together with the detection of implicit links, further improve the quality of the traceability links and the alignment of designs with goals.

CGS4Adaption also integrates the goal/feature functions with the SysML model (design) and system code (implementation) to support self-adaptation by conducting goal reasoning and trade-off analysis outside of goal modeling tools and inside SysML tools and system implementations. This is done automatically with tools support and code generation, which saves developers time and effort. However, using a third-party RMS tool to store the models and manage traceability reduces the usability of CGS4Adaption when compared to having only one or two tools involved in supporting similar functionalities.

Finally, concurrent modeling is partially supported (through the use of DOORS in between) while increases its scalability to handle large models in one environment.

Table 10.3: Comparison of adaptation methods, including CGS4Adaption, against integration criteria. (Y=Yes, N=No, P=partially, –=No integration, W=Weak)

Study	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	AC10
Goals and SysML										
Cui and Paige [51]	P	P	P	N	P	N	N	N	N	N
Tsadimas et al. [152]	N	P	P	N	P	N	N	N	N	N
Apvrille and Roudier [26]	N	P	P	N	P	N	N	N	N	N
Gnaho et al. [69]	P	P	P	N	P	N	N	N	N	N
Spyropoulos and Baras [150]	P	P	P	N	P	N	N	N	N	W
Ingram et al. [88]	N	P	P	N	P	N	N	N	N	W
Ahmad et al. [3]	P	P	P	N	P	N	N	N	N	W
Fan et al. [60]	P	P	P	N	P	N	N	N	N	N
Goals only										
Morandini et al. [115, 116]	N	Y	Y	N	Y	N	N	–	N	W
Ramnath et al. [141]	N	P	P	N	P	N	N	–	N	W
Baresi and Pasquale [32]	N	N	N	N	N	N	N	–	P	W
Qian et al. [138]	N	N	N	N	N	Y	N	–	P	W
SysML only										
Soyler and Sala-Diakanda [149]	N	P	P	N	P	N	N	–	N	W
Lopes et al. [106]	N	N	P	N	P	N	N	–	N	W
Ribeiro et al. [142]	N	P	P	N	P	N	N	–	N	W
Baresi et al. [31]	N	N	N	N	N	N	N	–	N	W
Meacham [113]	N	N	N	N	N	N	N	–	N	W
CGS4Adaptation	Y	Y	Y	Y	Y	Y	P	P	Y	W

10.1.4 Reasoning Methods

To support self-adaptation at runtime, *qualitative* reasoning is not enough to assess system performance (Section 3.5) while high-level goals help guide the evolution of requirements from elicitation to runtime adaptation [5, 38]. Moreover, goal-based reasoning could produce unfeasible solutions, which would consequently lead to additional effort and time needed to validate them during design (for design-time adaptation). However, combining the reasoning with complementary technique that prunes out invalid configurations and strategies (i.e., feature models) helps taking advantage of the goal model at runtime to produce valid adaptation strategies when faced with unforeseen changes.

Unlike the related work, CGS4Adaption is the only approach that includes all elements of goal and feature models and their relationships in the *quantitative* reasoning process through their arithmetic functions (Table 10.4). Moreover, these functions are automatically created and translated into several programming languages to be executed outside of goal modeling tools, in support of design time and runtime adaptation. In this way, the adaptation decision does not depend on local violations that may have a small impact on the overall performance of the system and lead to an expensive adaptation process. This for both design-time and run-time activities (C7), are the primary characteristics that distinguish the CGS4Adaption approach from other related work.

Table 10.4: Assessment of the collected quantitative reasoning methods with the CGS4Adaption approach (using criteria C1-C7 defined in Section 3.5).

Method	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**
Chitra et al. [50, 49]	Yes	Yes	Partially	No	No	Yes	Design
Noorian et al. [127]	No	Yes	Partially	Yes	No	Yes	Design
Nguyen et al. [124]	No	Partially	Partially	No	No	Yes	Design
Chatzikonstantinou and Kontogiannis [47]	Partially	Partially	Partially	No	No	No	Runtime
Ramirez and Cheng [140]	Yes	Partially	Partially	No	No	No	Runtime
Asadi et al. [28]	Partially	No	No	Yes	No	Yes	Design
CGS4Adaptation	Yes	Yes	Yes	Yes	Yes	Yes	Design& Runtime

10.1.5 Conclusions

Compared to the approaches discussed in Chapter 3, the CGS4Adaption is the only approach that helps discovering suitable adaptations for predictable contexts at design time, and for predictable and unpredictable contexts at runtime. It also takes advantage of all GRL operators in its mathematical functions (including contributions, actors, and KPIs) while taking into account validity restrictions imposed by a companion feature model. Furthermore, the same functions (available in 7 languages) can also be embedded in the SCPS implementation to support runtime adaptation. Finally, the method supports many RE activities considered mandatory for adaptive SCPS development, including comprehensive traceability management [100].

10.2 Threats to Validity

The validity of any study focuses on the degree of its conclusion's correctness, considering bias and generalization [61]. This section highlights the main construct, internal, and external threats to the validity of the experimentation. The threats to the validity of the systematic literature review are covered in Sections 3.3 and are not repeated here.

10.2.1 Construct Validity

Construct validity assesses to what degree the evaluation method measures what the proposed approach claims to do. One threat here is that case studies may not capture the full complexity of real-world problems and models. This is partially mitigated by the case study based on existing descriptions and requirements provided by a research team in France (SM@RT Team), with expertise in smart homes and in modeling. Additionally, we have used the descriptions and some of the values of existing commercial smart home features in order to create more realistic models.

Another threat is the unavailability of existing and aligned goal, feature, and SysML models that would have enabled further experiments. In addition, the performance and scalability evaluation relies on artificially created models because none that was sufficiently large was available. The performance results could indirectly be affected, for example, by some symmetry resulting from the duplication of one base model (a smart home GRL/feature model) to form larger models, although care was taken to minimize overlap by having different decision variables for their interpretation in the optimization model.

10.2.2 Internal Validity

Internal validity examines bias and other confounding factors in order to assess the degree to which conclusions can be reached based on the experimental settings and measures collected. One obvious threat here is that the thesis author developed the approach and supporting tools, designed the models, conducted the evaluation, and analyzed and reported the results. This may increase the perceived bias related to these methodological steps. However, this threat was partially mitigated by having a second person (the thesis supervisor) review these steps and results, and by having published several peer-reviewed papers on the approach.

The thesis author is also the only person who actually used the full CGS4Adaptation approach. However, this is partially mitigated by having two other researchers who used part of the approach (the automatic generation of arithmetic functions from GRL models [66, 13], and the use of these functions in CPLEX-based optimizations [13]) successfully in their work.

10.2.3 External Validity

This type of validity is related to the generalization of the results out of the thesis area. One threat is that we used only one case study to evaluate the proposed approach, and we hence anticipate limitations to the generalization of our conclusions to other examples and other domains.

Similarly, some specific tools are used for SysML, goal, and feature modeling, as well as for traceability management. Although many parts of the thesis attempt to provide tool-independent contributions, our conclusions may again not generalize well to other

modeling or RMS tools. Similarly, although the CGS4Adaption approach works well for GRL, it may not generalize easily to other goal-oriented modeling languages.

One particular traceability information model (TIM) involving GRL and SysML model elements was used in the current incarnation of CGS4Adaptation. Although other TIM could very likely be used for better answering the particular needs of given SCPS projects (by incorporating model elements and relationships different from those explored here), no other TIM was actually explored in this thesis, so generalization along that dimension has yet to be demonstrated.

Chapter 11

Conclusion and Future Work

This chapter recalls the main contributions of the thesis, answers the research questions, and discusses future work opportunities.

11.1 Contributions

Socio-cyber-physical systems consider human goals as a part of their concerns. Hence, these concerns should also be considered while configuring these systems at design time and during adaptation at runtime. Although mapping or remodeling goals at design time to realize a goal/SysML integration is the most common approach found in the literature, it is also time-consuming, limited, and prone to errors. Despite the fact that combining goal and SysML modeling into one tool is a desirable solution for solving traceability issues, achieving such an integration with current design tools remains a challenge.

Because adjusting requirements engineering activities to better design adaptive SCPSs is an urgent need, this thesis improves RE activities in that context by:

1. Proposing a new model-based development approach (*CGS4Adaptation*) that integrates social and goal-oriented concepts (in GRL) with independent methods for:
    - System design in SysML, to support the documentation and consideration of goal-based rationales behind requirements and design elements, traceability and change management, impact analysis, and consistency and completeness checks between the goal and SysML views (Sections 4.2, 5.1, and 6.2.2, as well as Chapter 8).
    - Optimization and simulation, to select and validate system designs, as well as adaptation strategies predefined at design time for common/foreseen contexts and generated at runtime for unforeseen contexts (Sections 4.3, 5.2, 6.2.6, 6.2.5 and 7.6.2).
    - Implementation, to support systems in choosing between adaptation alternatives at runtime while monitoring their quality and compliance (Chapter 9).

CGS4Adaptation also takes advantage of existing tools and techniques to minimize coding errors and the effort needed for its realization.

2. Introducing new arithmetic semantics for GRL, and optionally feature models, to produce mathematical functions (Sections 4.3.2 and 4.3.1). These functions can be used to conduct goal-based analysis and decision making outside goal modeling tools.

3. Transforming GRL models, optionally combined with feature models, automatically to mathematical functions in several programming languages without work repetition or information loss (Sections 4.3.3, 5.2.2, 5.2.1, 6.2.3, and 6.2.4). These functions can be embedded in SysML models and implementations, which supports consideration for human concerns in the decision-making process while improving the development and execution phases.

This thesis also contributes new tools[1] for automating parts of the CGS4Adaptation approach, including:

- A jUCMNav plug-in that exports GRL and feature models to arithmetic functions in SymPy.
- Modifications and extensions to SymPy to handle $Min()$ and $Max()$ functions properly and to export functions to Java code.
- A tool to import SysML requirement and block definition diagrams from Cameo Systems Modeler into IBM Rational DOORS, composed of 1) a MI-DSL traceability information model (Listing 5.1) from which a DXL library is produced, and 2) a report template in VTL (Listing C.1) that exports SysML elements, attributes, intra-model links, and figures, to a DXL script that invokes the library previously produced.
- Automated consistency and completeness checks for GRL and SysML models, implemented for DOORS as a DXL script (Listing C.3), accompanied by views and filters to improve the usability of the analysis.

Note that these contributions do not all require the simultaneous existence of GRL, SysML, and feature models. Feature models are actually optional in all three contributions, and the last two contributions do not require SysML models. In addition, within SysML, different subset than the one used here could be used. For example, requirement diagrams could be skipped if textual requirements exist elsewhere (as they would be redundant). This flexibility provides more opportunities for the above contributions to be used outside the SCPS context targeted by CGS4Adaptation.

11.2 Answers to the Research Questions

The research questions were answered through the CGS4Adaptation approach as follows.

[1] Available online at https://github.com/andaamal/ThesisArtifacts

***Q1**: How should the integration between goal, feature, and SysML models be done in early stages to support effective SCPS requirements definition, modeling, and analysis?*

This question was answered by integrating goal and feature models with SysML model diagrams that represent system structure, requirements, parametric views, and behaviour for different purposes (manage traceability, select suitable strategies for well-known contexts, and simulate system behaviour).

Three types of integrations are realized to cover these objectives:

1. Integration between the elements, attributes, and relationships of goal and SysML models via a third-party requirements management systems (DOORS);

2. Integration between the mathematical function representing criteria for goal evaluation and the mathematical function representing the feature model, in optimization models and in implementation code (via specific programming languages); and

3. Integration of the mathematical functions of goal and feature models with SysML models in a SysML tool that evaluates these functions and uses their values to simulate system behaviour and select appropriate design alternatives.

To support comprehensive consistency and completeness checks, change management, and impact analysis through managing traceability links across system goals, requirements, and design, the integration #1 is made between i) goal model elements (goals, softgoals, tasks, indicators, and actors) and their relationships, and ii) SysML model elements (blocks, constraint blocks, and requirements) and their relationships, in addition to the block definition and requirement diagrams themselves. This is achieved through importing them automatically into an RMS database as objects and links (Chapters 6 and 8). To support concurrent modeling, the information from these models (Section 5.1) that is manipulated in the RMS database is read from the original models, exported, and imported in DOORS automatically. Re-importing new versions of these models leads to updated information in the database, not just mere replacements, while keeping the changes made by the developers on this data (e.g., links to other objects). Consistency and completeness checks, in addition to implicit link checking (Chapter 8), further demonstrate the benefits of this integration for developers.

To identify several possible correct solutions, with GRL strategies constrained by feature model configurations, integration #2 combines the functions generated from a GRL model and a feature model into an *optimization model*. Feature model functions provide the feasible solutions through the relationships between the selectable features while goal model functions specify the impact of these features on each other and on the overall system quality in specific environmental conditions (Section 6.2.5 and 7.6.2). This method is used to validate the created models while narrowing the number of possible solutions to a few correct solutions (that can also be tested trough simulation in a larger modeling or execution context). An experiment also demonstrated the good performance and scalability of this approach.

To simulate system behaviour and detect system flaws prior to implementation, the mathematical functions of goal and feature models are combined with parametric and state machine diagrams (integration #3) in a SysML tool that evaluates the functions and performs system transition from one state to another using the values of the functions (Section 6.2.6). This integration also supports most of the MAPE adaptation activities.

The introduced integrations also support the following mandatory requirements engineering activities for socio-cyber-physical systems, identified by Lace and Kirikova [100]:

1. "Focus on business requirements and goals", with a goal model;
2. "Identify several possible correct solutions", with GRL strategies constrained by feature model configurations;
3. "Define how the system can monitor itself", with KPIs in goal models, that can be embedded in SysML models.
4. "Define how different requirements can impact each other", with links in the goal model and in the SysML model;
5. "Focus on defining constraints", with the feature model;
6. "Define qualitative requirements in measurable ways", with KPIs;
7. "Simulate system behaviour" with GRL/feature functions in SysML simulations and in CPLEX optimizations, to detect system defects before implementation.

***Q2**: What information should be carried from goal and feature models to SysML models and implementations in order to support SCPSs at runtime effectively, especially in a self-adaptation context?*

This question was answered by incorporating *all* of the concepts of GRL goal models with the SysML model and system implementation to perform precise goal-based reasoning outside of goal modeling tools and support many MAPE activities. This includes GRL *indicators* (for monitoring the environment and assessing symptoms of violations using real data), *intentional elements* (tasks, goals, and softgoals), their containing *actors*, their *importance* levels, and their *relationships* (AND/OR decompositions, dependencies, and weighted contributions). These concepts are used to propagate the values observed by the indicators to the high-level softgoals/goals, their actors, and the entire model. Section 4.3.1 presents the arithmetic semantics used to transform these concepts (elements and propagation) into one mathematical function. The resulting functions can be used to monitor goal violations, determine whether to adapt or not, and choose the best strategy according to the current context and system resources (Section 6.2.6 and Chapter 9).

To further restrict the candidate solutions allowed by goal functions to only valid ones, arithmetic functions generated from feature models carry system features, their relationships (mandatory, optional, and AND/OR/XOR decompositions), and cross-tree constraints (exclude and include) are combined with goal functions in several contexts, including optimization, simulation, and implementation (Section 4.3.2).

In conclusion, all of the information from goal and feature models should be carried, in an integrated and executable way.

11.3 Future Work

To better support the proposed approach, the following research areas are opened for further improvement and support.

- Further evaluation of the CGS4Adaptation approach could lead to additional improvements, especially in terms of generality (e.g., regarding other traceability information models and application areas). This could include industrial case studies and usability studies. An application of CGS4Adaptation that would cover the entire development, implementation, and deployment cycles of a real-world self-adaptive system would particularly be relevant.

- Providing better support for the adaptation activities by assessing the actual *cost* of an adaptation strategy given the current context. Such cost could be computed in terms of "distance" between the current context and what is considered for a given strategy, with additional knowledge of moving from local alternative to another one for all decision points. Some strategies would have a higher global satisfaction level, but at a higher cost. Trade-offs at that level could be integrated to the optimization method.

- Further simplifying the generated functions and enhancing their scalability. As noticed from the experiment in Chapter 9, the performance of the optimization decreases as the function size increases. There are several opportunities to follow up in this area. For example, contribution relationships cause the same formula for the source element to be repeated in the target element formula. Separating the formula from the main function and assigning it to a specific variable calculated before the main function and using the variable instead would reduce the overall number of operations in the main function, thus improving its performance and scalability. Similarity, in the optimization model, any formula not related to a decision variable should be separated from the main function and calculated before optimizing the model.

- Enhancing the functions used at runtime based on acquired knowledge. There is an opportunity to explore a variant of the methods that would better support *learning* as part of the adaptation process. At the moment, learning is seen as an activity performed outside of the CGS4Adaptation approach, which informs models by providing values to GRL indicators. However, feedback, knowledge acquisition, and learning activities could more systematically be considered, in a *MAPE-K*-like fashion.

- Exploring opportunities to bring GRL/feature modeling concepts and techniques directly into future versions of the SysML standard [131], in order to minimize the number of tools needed to support adaptive SCPS approaches such as GS4Adaptation.

References

[1] Ahmad, M. First step towards a domain specific language for self-adaptive systems. In *New Technologies of Distributed Systems (NOTERE), 2010 10th Annual International Conference on* (2010), IEEE, pp. 285–290.

[2] Ahmad, M., Araújo, J., Belloir, N., Bruel, J.-M., Gnaho, C., Laleau, R., and Semmak, F. Self-adaptive systems requirements modelling: Four related approaches comparison. In *Comparing Requirements Modeling Approaches Workshop (CMA@ RE), 2013 International* (2013), IEEE, pp. 37–42.

[3] Ahmad, M., Belloir, N., and Bruel, J.-M. Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems. *Journal of Systems and Software 107* (2015), 50–70.

[4] Ahmad, M., and Bruel, J.-M. bCMS requirements modelling using RELAX/SysML/KAOS. In *3rd CMA Workshop at RE'2013* (2013).

[5] Ahmad, M., and Bruel, J.-M. A comparative study of RELAX and SysML/KAOS. Tech. rep., Institut de Recherche en Informatique de Toulouse, France, 2014.

[6] Ahmad, M., Bruel, J.-M., Laleau, R., and Gnaho, C. Using RELAX, SysML and KAOS for ambient systems requirements modeling. In *Procedia Computer Science* (2012), vol. 10, pp. 474–481.

[7] Ahmad, M., Dragomir, I., Bruel, J.-M., Ober, I., and Belloir, N. Early analysis of ambient systems SysML properties using OMEGA2-IFx. In *Proceedings of the 3rd International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH-2013)* (2013).

[8] Akbas, A. S., and Karwowski, W. A systems engineering approach to modeling and simulating software training management efforts. In *25th European Modeling and Simulation Symposium, EMSS 2013* (2013), pp. 264–269.

[9] Akbas, A. S., Mykoniatis, K., Angelopoulou, A., and Karwowski, W. A model-based approach to modeling a hybrid simulation platform (Work in Progress). In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative* (2014), vol. 46, pp. 223–228.

[10] Akhigbe, O., Alhaj, M., Amyot, D., Badreddin, O., Braun, E., Cartwright, N., Richards, G., and Mussbacher, G. Creating quantitative goal models: Governmental experience. In *Conceptual Modeling* (Cham, 2014), E. Yu, G. Dobbie, M. Jarke, and S. Purao, Eds., Springer International Publishing, pp. 466–473.

[11] Akhigbe, O., Amyot, D., Anda, A. A., Lessard, L., and Xiao, D. Consistency analysis for user requirements notation models. In *Proceedings of the Ninth International i* Workshop (iStar 2016)* (2016), CEUR-WS Vol. 1674, pp. 43–48.

[12] Allen, J. How 5G could make your home smarter, 2019. https://www.techradar.com/news/how-5g-could-make-your-home-smarter.

[13] Alwidian, S. A. R. *Union Models: Support of Variability Modeling and Efficient Reasoning About Model Families Over Space and Time*. PhD thesis, University of Ottawa, Canada, 2020.

[14] Amyot, D., Anda, A. A., Baslyman, M., Lessard, L., and Bruel, J.-M. Towards Improved Requirements Engineering with SysML and the User Requirements Notation. In *2016 IEEE 24th International Requirements Engineering Conference (RE)* (2016), pp. 329–334.

[15] Amyot, D., Becha, H., Bræk, R., and Rossebø, J. E. Next generation service engineering. In *2008 First ITU-T Kaleidoscope Academic Conference - Innovations in NGN: Future Network and Services* (2008), pp. 195–202.

[16] Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E. Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems 25*, 8 (2010), 841–877.

[17] Amyot, D., and Mussbacher, G. User Requirements Notation: the first ten years, the next ten years. *JSW 6*, 5 (2011), 747–768.

[18] Amyot, D., Shamsaei, A., Kealey, J., Tremblay, E., Miga, A., Mussbacher, G., Alhaj, M., Tawhid, R., Braun, E., and Cartwright, N. Towards Advanced Goal Model Analysis with jUCMNav. In *Advances in Conceptual Modeling* (2012), Springer Berlin Heidelberg, pp. 201–210.

[19] Anda, A. A. Modeling adaptive socio-cyber-physical systems with goals and SysML. In *Doctoral Symposium, 2018 IEEE 26th International Requirements Engineering Conference (RE)* (2018), IEEE CS, pp. 442–447.

[20] Anda, A. A., and Amyot, D. Arithmetic Semantics of Feature and Goal Models for Adaptive Cyber-Physical Systems. In *27th IEEE International Requirements Engineering Conference (RE'19)* (2019), IEEE CS, pp. 245–256.

[21] Anda, A. A., and Amyot, D. An optimization modeling method for adaptive systems based on goal and feature models. In *Proceedings of the Tenth International Model-Driven Requirements Engineering Workshop (MoDRE)* (2020), IEEE CS, pp. 10–20.

[22] Anda, A. A., and Amyot, D. Traceability management of GRL and SysML models. In *SAM'20: 12th System Analysis and Modeling Conference Proceedings* (2020), ACM.

[23] Andersson, J., De Lemos, R., Malek, S., and Weyns, D. Modeling dimensions of self-adaptive software systems. *Software engineering for self-adaptive systems* (2009), 27–47.

[24] Apache Velocity Project. Velocity Template Language, 2020. http://velocity.apache.org/engine/devel/user-guide.html.

[25] Apel, S., Batory, D., Kästner, C., and Saake, G. *Feature-oriented software product lines*. Springer, 2016.

[26] Apvrille, L., and Roudier, Y. SysML-Sec: A SysML environment for the design and development of secure embedded systems. *APCOSEC, Asia-Pacific Council on Systems Engineering* (2013), 8–11.

[27] Apvrille, L., and Roudier, Y. Designing safe and secure embedded and cyber-physical systems with SysML-Sec. In *International Conference on Model-Driven Engineering and Software Development* (2015), Springer, pp. 293–308.

[28] Asadi, M., Bagheri, E., Gasevic, D., and Hatala, M. Goal-oriented requirements and feature modeling for software product line engineering. In *Ninth International Conference on Generative Programming and Component Engineering (GPCE'10)* (2010).

[29] Aybar, D., Azor, O., Nguyen, L., and Samedi, L. Smart home management system. Tech. rep., University of Central Florida, 2014.

[30] Badreddin, O., Abdelzad, V., Lethbridge, T. C., and Elaasar, M. FSysML: Foundational executable SysML for cyber-physical system modeling. In *CEUR Workshop Proceedings* (2016), vol. 1731, pp. 38–51.

[31] Bareiss, P., Schütz, D., Priego, R., Marcos, M., and Vogel-Heuser, B. A model-based failure recovery approach for automated production systems combining SysML and industrial standards. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)* (2016), pp. 1–7.

[32] Baresi, L., and Pasquale, L. Adaptive goals for self-adaptive service compositions. In *2010 IEEE International Conference on Web Services* (2010), IEEE, pp. 353–360.

[33] Baresi, L., and Pasquale, L. Live goals for adaptive service compositions. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (New York, NY, USA, 2010), SEAMS '10, ACM, pp. 114–123.

[34] Baresi, L., Pasquale, L., and Spoletini, P. Fuzzy goals for requirements-driven adaptation. In *2010 18th IEEE International Requirements Engineering Conference (RE)* (2010), IEEE, pp. 125–134.

[35] Belloir, N., Chiprianov, V., Ahmad, M., Munier, M., Gallon, L., and Bruel, J.-M. Using RELAX operators into an MDE security requirement elicitation process for systems of systems. In *Proceedings of the 2014 European Conference on Software Architecture Workshops* (2014), ACM, p. 32.

[36] Berenbach, B., Paulish, D., Kazmeier, J., and Rudorfer, A. *Software & systems requirements engineering: in practice*. McGraw-Hill, Inc., 2009.

[37] Blair, G., Bencomo, N., and France, R. B. Models@Run.Time. *Computer 42*, 10 (2009), 22–27.

[38] Bocanegra, J., Pavlich-Mariscal, J., and Carrillo-Ramos, A. On the role of model-driven engineering in adaptive systems. In *Computing Conference (CCC), 2016 IEEE 11th Colombian* (2016), IEEE, pp. 1–8.

[39] Boonstra. Boonstra Heating and Air conditioning, 2019. https://boonstraright-time.ca/specials/.

[40] Bouchard, K., Bouchard, B., and Bouzouane, A. Guidelines to efficient smart home design for rapid AI prototyping: a case study. In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments* (2012), pp. 1–8.

[41] Bousse, E. Requirements management led by formal verification. Master's thesis, University of Rennes, France, 2012.

[42] Braun, P., Broy, M., Houdek, F., Kirchmayr, M., Müller, M., Penzenstadler, B., Pohl, K., and Weyer, T. Guiding requirements engineering for software-intensive embedded systems in the automotive industry. *Computer Science-Research and Development 29*, 1 (2014), 21–43.

[43] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems 8*, 3 (2004), 203–236.

[44] Bruel, J.-M. Smart Home Challenge Problem, 2018. https://docs.google.com/document/d/1D9ii7bGAdCCspapVxCN3lLIvGO4o-EyWZxOIzTml5fI/edit.

[45] Cantu, E. Method and system for solution of models, 05 2013. S.A. Patent WO 2013/061 166 A2.

[46] Cavalcante, E., Batista, T., Bencomo, N., and Sawyer, P. Revisiting goal-oriented models for self-aware systems-of-systems. In *2015 IEEE International Conference on Autonomic Computing (ICAC)* (July 2015), pp. 231–234.

[47] Chatzikonstantinou, G., and Kontogiannis, K. Run-time requirements verification for reconfigurable systems. *Information and Software Technology 75* (2016), 105–121.

[48] Cheng, B. H. C., De Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., and Others. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software engineering for self-adaptive systems* (2009), vol. LNCS 5525, Springer, pp. 1–26.

[49] Chitra, Subramanian, M., Krishna, A., and Kaur, A. Optimal goal programming of softgoals in goal-oriented requirements engineering. In *PACIS* (2016), p. 202.

[50] Chitra, S., Krishna, A., and Kaur, A. Optimal reasoning of goals in the i* framework. In *Asia-Pacific Software Engineering Conference, APSEC* (2015), pp. 346–353.

[51] Cui, X., and Paige, R. F. An integrated framework for system/software requirements development aligning with business motivations. In *Proceedings - 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, ICIS 2012* (2012), pp. 547–552.

[52] De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., Vogel, T., et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, pp. 1–32.

[53] DeVries, B., and Cheng, B. H. Run-time monitoring of self-adaptive systems to detect n-way feature interactions and their causes. In *Proceedings of the 13th International Conference on Software Engineerings for Adaptive and Self-Managing Systems* (2018), ACM, pp. 94–100.

[54] Duran, M. B., and Mussbacher, G. Investigation of feature run-time conflicts on goal model-based reuse. *Information Systems Frontiers 18*, 5 (2016), 855–875.

[55] Eclipse Foundation. Xtext: Language Engineering for Everyone!, 2018. http://www.eclipse.org/Xtext/.

[56] Emmerich, W., Butchart, B., Chen, L., Wassermann, B., and Price, S. L. Grid service orchestration using the business process execution language (bpel). *Journal of Grid Computing 3*, 3-4 (2005), 283–304.

[57] energyrates.ca. Residential Electricity and Natural Gas Plans, 2019. https://energyrates.ca/residential-electricity-natural-gas/.

[58] Fan, Y. GRLToMath plugin for jUCMNav, 2018. https://github.com/AAmberFan/GRLToMath.

[59] Fan, Y., Anda, A. A., and Amyot, D. An arithmetic semantics for grl goal models with function generation. In *System Analysis and Modeling. Languages, Methods, and Tools for Systems Engineering (SAM 2018)* (2018), vol. LNCS 11150, Springer, pp. 144–162.

[60] Fan, Z., Yue, T., and Zhang, L. SAMM: an architecture modeling methodology for ship command and control systems. *Software and Systems Modeling 15*, 1 (2016), 71–118.

[61] Feldt, R., and Magazinius, A. Validity Threats in Empirical Software Engineering Research-An Initial Survey. In *SEKE* (2010), pp. 374–379.

[62] Fisher, Z. C., Cooksey, K. D., and Mavris, D. A model-based systems engineering approach to design automation of SUAS. In *2017 IEEE Aerospace Conference* (mar 2017), pp. 1–15.

[63] Frazzon, E. M., Hartmann, J., Makuschewitz, T., and Scholz-Reiter, B. Towards socio-cyber-physical systems in production networks. *Procedia CIRP 7* (2013), 49–54. Forty Sixth CIRP Conference on Manufacturing Systems 2013.

[64] Friedenthal, S., Moore, A., and Steiner, R. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[65] Ghanavati, S., Amyot, D., and Peyton, L. A requirements management framework for privacy compliance. In *Proceedings of the 10th Workshop on Requirements Engineering (WER'07)* (2007), pp. 149–159.

[66] Ghasemi, M., and Amyot, D. Data preprocessing for goal-oriented process discovery. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)* (2019), pp. 200–206.

[67] Giorgini, P., Mylopoulos, J., Nicchiarelli, E., and Sebastiani, R. Reasoning with goal models. In *International Conference on Conceptual Modeling* (2002), Springer, pp. 167–181.

[68] Gnaho, C., Laleau, R., Semmak, F., and Bruel, J.-M. bCMS requirements modelling using SysML/KAOS, 2013. https://goo.gl/QU9Tgn.

[69] Gnaho, C., Semmak, F., and Laleau:, R. An overview of a SysML extension for goal-oriented NFR modelling: Poster paper. In *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)* (may 2013), pp. 1–2.

[70] Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J., and Mäder, P. *Traceability Fundamentals*. Springer London, London, 2012, pp. 3–22.

[71] Guo, J., Yang, D., Siegmund, N., Apel, S., Sarkar, A., Valov, P., Czarnecki, K., Wasowski, A., and Yu, H. Data-efficient performance learning for configurable systems. *Empirical Software Engineering 23*, 3 (2018), 1826–1867.

[72] Haghighatkhah, A., Oivo, M., Banijamali, A., and Kuvaja, P. Improving the state of automotive software engineering. *IEEE Software 34*, 5 (2017), 82–86.

[73] Hassine, J., and Amyot, D. A questionnaire-based survey methodology for systematically validating goal-oriented models. *Requirements Engineering 21*, 2 (2016), 285–308.

[74] Hevner, A. R., March, S. T., Park, J., and Ram, S. Design Science in Information Systems Research. *MIS Q. 28*, 1 (March 2004), 75–105.

[75] Hoelle, D. Frameworks for a general-purpose smart home operating system, 2014.

[76] Holt, J., and Perry, S. *SysML for systems engineering*, vol. 7. IET, 2008.

[77] Horkoff, J., Salay, R., Chechik, M., and Di Sandro, A. Supporting early decision-making in the presence of uncertainty. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)* (2014), IEEE, pp. 33–42.

[78] Horkoff, J., and Yu, E. Comparison and evaluation of goal-oriented satisfaction analysis techniques. *Requirements Engineering 18*, 3 (Sep 2013), 199–222.

[79] Horváth, I. What the Design Theory of Social-Cyber-Physical Systems Must Describe, Explain and Predict? In *An Anthology of Theories and Models of Design*. Springer, 2014, pp. 99–120.

[80] Hsu, Y.-L., Chou, P.-H., Chang, H.-C., Lin, S.-L., Yang, S.-C., Su, H.-Y., Chang, C.-C., Cheng, Y.-S., and Kuo, Y.-C. Design and implementation of a smart home system using multisensor data fusion technology. *Sensors 17*, 7 (2017), 1631.

[81] Hydro-Québec. Wind Power, 2019. http://www.hydroquebec.com/learning/eolienne/.

[82] IBM. IBM ILOG CPLEX Optimization Studio, 2018. https://www.ibm.com/ca-en/marketplace/ibm-ilog-cplex.

[83] IBM. Rational DOORS v9.6.1, 2018. http://goo.gl/yGWpze.

[84] IBM. IBM ILOG CPLEX Optimization Studio - CP Optimizer User's Manual, V12.10.0, 2019. https://ibm.co/2YRj8oy.

[85] IBM Knowledge Center. IBM Rational DOORS DXL Reference Manual Release 9.5, 2018.

[86] IBM Knowledge Center. Rational DOORS database structure schema, 2018. https://goo.gl/CCcLGX.

[87] Informatics Research Institute of Toulouse. SM@RT Team, 2018. https://www.irit.fr/en/departement/dep-reliability-systems-and-software/smrt-team/.

[88] Ingram, C., Andrews, Z., Payne, R., and Plat, N. SysML fault modelling in a traffic management system of systems. In *System of Systems Engineering (SOSE), 2014 9th International Conference on* (2014), IEEE, pp. 124–129.

[89] Ito, Y., Tomura, S., and Moriya, K. Vibration-reducing motor control for hybrid vehicles. *R&D Review of Toyota CRDL 40*, 2 (2005), 37–43.

[90] ITU-T. Recommendation Z.151 (10/18): User Requirements Notation (URN) - Language Definition, 2018. http://www.itu.int/rec/T-REC-Z.151/en.

[91] Jureta, I. J., Borgida, A., Ernst, N. A., and Mylopoulos, J. The requirements problem for adaptive systems. *ACM Transactions on Management Information Systems (TMIS) 5*, 3 (2015), 17.

[92] Kephart, J. O., and Chess, D. M. The vision of autonomic computing. *Computer 36*, 1 (2003), 41–50.

[93] Kephart, J. O., and Walsh, W. E. An artificial intelligence perspective on autonomic computing policies. In *Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)* (2004), IEEE, pp. 3–12.

[94] Kifetew, F. M., Muñante, D., Gorroñogoitia, J., Siena, A., Susi, A., and Perini, A. Grammar based genetic programming for software configuration problem. In *International Symposium on Search Based Software Engineering* (2017), Springer, pp. 130–136.

[95] Kitchenham, B., and Charters, S. Guidelines for performing systematic literature reviews in software engineering. Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[96] Klubnikin, A. Smart Home Feature News, 2018. https://www.iotevolutionworld.com/smart-home/articles/438395-building-smarter-connected-homes-with-machine-learning.htm.

[97] Kolesnikov, S., Siegmund, N., Kästner, C., Grebhahn, A., and Apel, S. Tradeoffs in modeling performance of highly configurable software systems. *Software & Systems Modeling* (2018), 1–19.

[98] Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., and Becker, C. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing 17* (2015), 184–206.

[99] Kuchcinski, K., and Szymanek, R. JaCoP: Java Constraint Programming Library, 2018. http://jacop.cs.lth.se/.

[100] Lace, K., and Kirikova, M. Required changes in requirements engineering approaches for socio-cyber-physical systems. In *CRE+FIRE, REFSQ-JP 2018* (2018), CEUR-WS, Vol-2075.

[101] Laleau, R., Semmak, F., Matoussi, A., Petit, D., Hammad, A., and Tatibouët, B. A first attempt to combine SysML requirements diagrams and B. *Innovations in Systems and Software Engineering 6*, 1 (2010), 47–54.

[102] Lane, J. A., and Bohn, T. Using sysml modeling to understand and evolve systems of systems. *Systems Engineering 16*, 1 (2013), 87–98.

[103] Lee, K., Kang, K. C., and Lee, J. Concepts and guidelines of feature modeling for product line software engineering. In *ICSR-7: Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools* (2002), Springer, pp. 62–77.

[104] Liaskos, S., Jalman, R., and Aranda, J. On eliciting contribution measures in goal models. In *2012 20th IEEE International Requirements Engineering Conference (RE)* (Sep. 2012), pp. 221–230.

[105] Liu, L., and Yu, E. Designing information systems in social context: a goal and scenario modelling approach. *Information Systems 29*, 2 (2004), 187–203.

[106] Lopes, A. J., Lezama, R., and Pineda, R. Model Based Systems Engineering for Smart Grids as systems of systems. In *Procedia Computer Science* (2011), vol. 6, pp. 441–450.

[107] Lung, C.-H., Zhang, X., and Rajeswaran, P. Improving software performance and reliability in a distributed and concurrent environment with an architecture-based self-adaptive framework. *Journal of Systems and Software 121* (2016), 311–328.

[108] Luo, H., and Amyot, D. Towards a declarative, constraint-oriented semantics with a generic evaluation algorithm for GRL. In *Proceedings of the Fifth International i* Workshop (iStar 2011)* (2011), CEUR-WS Vol-766, pp. 26–31. http://ceur-ws.org/Vol-766/paper05.pdf.

[109] Macías-Escrivá, F. D., Haber, R., del Toro, R., and Hernandez, V. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications 40*, 18 (2013), 7267–7279.

[110] Mäder, P., Olivetto, R., and Marcus, A. Empirical studies in software and systems traceability. *Empirical Software Engineering 22* (2017), 963–966.

[111] Mammar, A., and Laleau, R. On the use of domain and system knowledge modeling in goal-based Event-B specifications. In *International Symposium on Leveraging Applications of Formal Methods (ISoLA 2016)* (2016), vol. LNCS 9952, pp. 325–339.

[112] Matoussi, A., Gervais, F., and Laleau, R. A goal-based approach to guide the design of an abstract Event-B specification. In *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on* (2011), IEEE, pp. 139–148.

[113] Meacham, S. Towards self-adaptive IoT applications: Requirements and adaptivity patterns for a fall-detection ambient assisting living application. In *Components and Services for IoT Platforms*. Springer, 2017, pp. 89–102.

[114] Mistrik, I., Ali, N., Kazman, R., Grundy, J., and Schmerl, B. *Managing Trade-offs in Adaptable Software Architectures*. Morgan Kaufmann, 2016.

[115] Morandini, M., Penserini, L., and Perini, A. Automated mapping from goal models to self-adaptive systems. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering* (2008), IEEE Computer Society, pp. 485–486.

[116] Morandini, M., Penserini, L., Perini, A., and Marchetto, A. Engineering requirements for adaptive systems. *Requirements Engineering 22*, 1 (2017), 77–103.

[117] Muñoz-Fernández, J. C., Mazo, R., Salinesi, C., and Tamura, G. 10 challenges for the specification of self-adaptive software. In *2018 12th International Conference on Research Challenges in Information Science (RCIS)* (May 2018), pp. 1–12.

[118] Murty, K. G. *Optimization models for decision making: Volume 1*. University of Michigan, Ann Arbor, 2003.

[119] Mussbacher, G., Amyot, D., Breu, R., Bruel, J.-M., Cheng, B. H. C., Collet, P., Combemale, B., France, R. B., Heldal, R., Hill, J., Kienzle, J., Schöttle, M., Steimann, F., Stikkolorum, D., and Whittle, J. The relevance of model-driven engineering thirty years from now. In *Model-Driven Engineering Languages and Systems* (Cham, 2014), J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfran, Eds., Springer International Publishing, pp. 183–200.

[120] Nair, V., Yu, Z., Menzies, T., Siegmund, N., and Apel, S. Finding faster configurations using flash. *arXiv preprint arXiv:1801.02175* (2018).

[121] Ncube, C., and Lim, S. L. On systems of systems engineering: A requirements engineering perspective and research agenda. In *2018 IEEE 26th International Requirements Engineering Conference (RE)* (Aug 2018), IEEE CS, pp. 112–123.

[122] Nešić, D., Krüger, J., Stănciulescu, u., and Berger, T. Principles of feature modeling. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (New York, NY, USA, 2019), ESEC/FSE 2019, Association for Computing Machinery, p. 62–73.

[123] Nguyen, C. M. *Efficient Reasoning with Constrained Goal Models*. PhD thesis, University of Trento, Italy, 2017.

[124] Nguyen, C. M., Sebastiani, R., Giorgini, P., and Mylopoulos, J. Multi-objective reasoning with constrained goal models. *Requirements Engineering* (2016), 1–37.

[125] Nguyen, C. M., Sebastiani, R., Giorgini, P., and Mylopoulos, J. Multi-objective reasoning with constrained goal models. *Requirements Engineering 23*, 2 (2018), 189–225.

[126] No Magic Inc. Cameo Systems Modeler, 2019. https://www.nomagic.com/products/cameo-systems-modeler#resources.

[127] Noorian, M., Bagheri, E., and Du, W. Toward automated quality-centric product line configuration using intentional variability. *Journal of Software: Evolution and Process 29*, 9 (2017), e1870.

[128] Norton Community. 12 tips to help secure your smart home and IoT devices, 2019. https://us.norton.com/internetsecurity-iot-smart-home-security-core.html.

[129] Olaechea, R., Rayside, D., Guo, J., and Czarnecki, K. Comparison of exact and approximate multi-objective optimization for software product lines. In *Proceedings of the 18th International Software Product Line Conference-Volume 1* (2014), ACM, pp. 92–101.

[130] Olid Chinen Tech. Leading Auto Doors Manufacture in China, 2019. https://www.olideautodoor.com/.

[131] OMG. Systems Modeling Language (SysML) v2 Request For Proposal (RFP), 2017. http://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-2.

[132] OMG. Systems Modeling Language (SysML) v1.6, 2019. https://www.omg.org/spec/SysML/1.6/.

[133] OSLC. Open services lifecycle collaboration, 2018. https://open-services.net/specifications/.

[134] Ozkaya, I. Representing requirement relationships. In *First International Workshop on Visualization in Requirements Engineering, REV* (2007).

[135] Palmieri, A., Collet, P., and Amyot, D. Handling regulatory goal model families as software product lines. In *International Conference on Advanced Information Systems Engineering* (2015), Springer, pp. 181–196.

[136] Pnueli, A. The temporal logic of programs. In *Foundations of Computer Science, 18th Annual Symposium on* (1977), IEEE, pp. 46–57.

[137] Pourshahid, A., Johari, I., Richards, G., Amyot, D., and Akhigbe, O. S. A goal-oriented, business intelligence-supported decision-making methodology. *Decision Analytics 1*, 9 (2014).

[138] Qian, W., Peng, X., Chen, B., Mylopoulos, J., Wang, H., and Zhao, W. Rationalism with a dose of empiricism: combining goal reasoning and case-based reasoning for self-adaptive software systems. *Requirements Engineering 20*, 3 (2015), 233–252.

[139] Rahman, A., and Amyot, D. A DSL for importing models in a requirements management system. In *4th International Model-Driven Requirements Engineering Workshop (MoDRE)* (2014), IEEE, pp. 37–46.

[140] Ramirez, A. J., and Cheng, B. H. C. Automatic Derivation of Utility Functions for Monitoring Software Requirements. In *Model Driven Engineering Languages and Systems (MODELS)* (2011), Springer Berlin Heidelberg, pp. 501–516.

[141] Ramnath, R., Gupta, V., and Ramanathan, J. RED-Transaction and Goal-Model Based Analysis of Layered Security of Physical Spaces. In *2008 32nd Annual IEEE International Computer Software and Applications Conference* (2008), IEEE, pp. 679–685.

[142] Ribeiro, F. G. C., Misra, S., and Soares, M. S. Application of an Extended SysML Requirements Diagram to Model Real-Time Control Systems. In *International Conference on Computational Science and Its Applications* (2013), Springer, pp. 70–81.

[143] Roudier, Y., and Apvrille, L. SysML-Sec: A model driven approach for designing safe and secure systems. In *Model-Driven Engineering and Software Development (MODELSWARD), Third International Conference on* (2015), IEEE, pp. 655–664.

[144] Salay, R., Famelis, M., and Chechik, M. Language independent refinement using partial modeling. In *Fundamental Approaches to Software Engineering* (2012), J. de Lara and A. Zisman, Eds., Springer Berlin Heidelberg, pp. 224–239.

[145] Salehie, M., and Tahvildari, L. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS) 4*, 2 (2009), 14.

[146] Salehie, M., and Tahvildari, L. Towards a goal-driven approach to action selection in self-adaptive software. *Software: Practice and Experience 42*, 2 (2012), 211–233.

[147] Sanchez, L. E., Diaz-Pace, J. A., Zunino, A., Moisan, S., and Rigault, J.-P. An approach based on feature models and quality criteria for adapting component-based systems. *Journal of Software Engineering Research and Development 3*, 1 (2015), 10.

[148] Smirnov, A., Kashevnik, A., and Ponomarev, A. Multi-level self-organization in cyber-physical-social systems: Smart home cleaning scenario. *Procedia CIRP 30* (2015), 329–334. 7th Industrial Product-Service Systems Conference - PSS, industry transformation for sustainability and business.

[149] Soyler, A., and Sala-Diakanda, S. A model-based systems engineering approach to capturing disaster management systems. In *2010 IEEE International Systems Conference* (2010), pp. 283–287.

[150] Spyropoulos, D., and Baras, J. S. Extending Design Capabilities of SysML with Trade-off Analysis: Electrical Microgrid Case Study. *Procedia Computer Science 16* (2013), 108–117.

[151] SymPy Development Team. SymPy, 2018. http://www.sympy.org/.

[152] Tsadimas, A., Nikolaidou, M., and Anagnostopoulos, D. Extending SysML to explore non-functional requirements: the case of information system design. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (2012), ACM, pp. 1057–1062.

[153] Tueno Fotso, S. J., Frappier, M., Laleau, R., Mammar, A., and Leuschel, M. Formalisation of sysml/kaos goal assignments with b system component decompositions. In *Integrated Formal Methods* (Cham, 2018), C. A. Furia and K. Winter, Eds., Springer International Publishing, pp. 377–397.

[154] van Lamsweerde, A. *Requirements engineering: From system goals to UML models to software*, vol. 10. Chichester, UK: John Wiley & Sons, 2009.

[155] Vanderperren, Y., and Dehaene, W. SysML and systems engineering applied to UML-based SoC design. In *Proceedings of the 2nd UML-SoC Workshop at 42nd DAC, USA* (2005).

[156] Warmer, J. B., and Kleppe, A. G. *The object constraint language: Precise modeling with UML*. Addison-Wesley Professional, 1998.

[157] Welsh, K., Bencomo, N., Sawyer, P., and Whittle, J. Self-explanation in adaptive systems based on runtime goal-based models. In *Transactions on Computational Collective Intelligence XVI*. Springer, 2014, pp. 122–145.

[158] Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H. C., and Bruel, J.-M. RELAX: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering 15*, 2 (2010), 177–196.

[159] Woldeamlak, S., Diabat, A., and Svetinovic, D. Goal-oriented requirements engineering for research-intensive complex systems: A case study. *Systems Engineering 19*, 4 (2016), 322–333.

[160] Yu, E. S. K. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997, Proceedings of the Third IEEE International Symposium on* (1997), pp. 226–235.

[161] Zambonelli, F. Towards a general software engineering methodology for the internet of things. *CoRR abs/1601.05569* (2016).

[162] Zitzler, E., and Künzli, S. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature* (2004), Springer, pp. 832–842.

Appendix A

Hybrid Car's Engine Example

This appendix presents the DXL scripts (Section 6.2.2) and the arithmetic functions generated from the GRL/feature model of the hybrid car's engine system example (Chapter 6), in several programming languages (Section 6.2.4).

A.1 DXL Scripts of the Hybrid Car's Engine Example

Listing A.1 shows the DXL scripts of the hybrid car's engine system example, generated by running the report template in Listing C.1.

Listing A.1: DXL script of the adaptive care in SimplifiedadaptiveCar.DXL

```
# include "addins/DSL/lib/Utilities.dxl"
pragma runLim, 0
beginImport("AdaptiveCarOriginal")

//Requirement Diagram
requirementsDiagram("_19_0_2_5f801fb_1585023798206_250545_42563","
    AdaptiveCarRequirements", "", "file:/C:/CaseStudy/
    SmartReportNote_files/_19_0_2_5f801fb_1585023798206_250545_42563.png",
    "diagram.diagramType of Smart Home Case study" )
requirementToDiagram("_19_0_2_5f801fb_1585024370822_154602_43024_19_0_2_
     5f801fb_1585023798206_250545_42563","ContainedBy","","
    _19_0_2_5f801fb_ 1585024370822_154602_43024","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024351846_858122_42809_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024351846_858122_42809","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585026061603_466277_43358_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585026061603_466277_43358","
    _19_0_2_5f801fb_1585023798206_250545_42563")
```

```
requirementToDiagram("_19_0_2_5f801fb_1585024392253_566642_43282_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024392253_566642_43282","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024380953_268910_43153_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024380953_268910_43153","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024358490_669314_42895_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024358490_669314_42895","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024366507_363993_42981_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
      1585024366507_363993_42981","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024355190_929810_42852_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024355190_929810_42852","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024347829_832841_42766_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024347829_832841_42766","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024361756_6376_42938_19_0_2_ 5
    f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
    1585024361756_6376_42938","_19_0_2_5f801fb_1585023798206_250545_42563"
    )
requirementToDiagram("_19_0_2_5f801fb_1585024374357_354966_43067_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024374357_354966_43067","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024344154_915734_42723_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024344154_915734_42723","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024388846_380075_43239_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024388846_380075_43239","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024341233_278133_42680_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024341233_278133_42680","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024377890_862239_43110_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024377890_862239_43110","
    _19_0_2_5f801fb_1585023798206_250545_42563")
requirementToDiagram("_19_0_2_5f801fb_1585024333418_777761_42637_19_0_2_
    5f801fb_1585023798206_250545_42563","ContainedBy","","_19_0_2_5f801fb_
     1585024333418_777761_42637","
    _19_0_2_5f801fb_1585023798206_250545_42563")
//SysML Block Definition Diagram
blockDefinitionDiagram("_19_0_2_5f801fb_1585023855608_147157_42596", "
```

```
    AdaptiveCarBlocks", "", "file:/C:/CaseStudy/SmartReportNote_files/
    _19_0_2_5f801fb_1585023855608_147157_42596.png", "SysML Block
    Definition Diagram of Smart Home Case study" )
blockToDiagram("
    _19_0_2_5f801fb_1585026225973_235045_43713_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026225973 _235045_43713","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026248141_903744_44028_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026248141_ 903744_44028","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026219261_659151_43623_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026219261 _659151_43623","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026208125_983537_43443_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026208125 _983537_43443","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026238757_873968_43893_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026238757 _873968_43893","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026254293_890928_44118_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026254293 _890928_44118","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026205653_780803_43398_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026205653 _780803_43398","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026211301_555359_43488_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026211301 _555359_43488","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026216677_300905_43578_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026216677_ 300905_43578","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026245309_260416_43983_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026245309 _260416_43983","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("_19_0_2_5f801fb_1585026242060_759745_43938_19_0_2_5f801fb
```

```
      _1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026242060 _759745_43938","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026235310_748842_43848_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026235310 _748842_43848","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026258692_649637_44163_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026258692 _649637_44163","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026213621_616879_43533_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026213621 _616879_43533","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026250890_682235_44073_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026250890 _682235_44073","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026229229_557851_43758_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026229229_ 557851_43758","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026222693_146070_43668_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026222693_ 146070_43668","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585072222132_884521_42167_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585072222132_ 884521_42167","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("
    _19_0_2_5f801fb_1585026265149_947387_44211_19_0_2_5f801fb_
    1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026265149 _947387_44211","
    _19_0_2_5f801fb_1585023855608_147157_42596")
blockToDiagram("_19_0_2_5f801fb_1585026273526_752148_44250_19_0_2_5f801fb
     _1585023855608_147157_42596","ContainedBy","","
    _19_0_2_5f801fb_1585026273526 _752148_44250","
    _19_0_2_5f801fb_1585023855608_147157_42596")
//Blocks
block("_19_0_2_5f801fb_1585026219261_659151_43623","Battery","","Block")

block("_19_0_2_5f801fb_1585026205653_780803_43398","Electric Engine","","
    Block")

BlockDependency("_19_0_2_5f801fb_1585073578361_357627_42359","DependOn","
```

```
    "," _19_0_2_5f801fb_1585026245309_260416_43983","
    _19_0_2_5f801fb_1585026205653_780803_43398")

block("_19_0_2_5f801fb_1585026208125_983537_43443","Steering","","Block")

BlockDependency("_19_0_2_5f801fb_1585073312806_155898_42341","DependOn","
    ","_19_0_2_5f801fb_1585026238757_873968_43893","
    _19_0_2_5f801fb_1585026208125_983537_43443")

block("_19_0_2_5f801fb_1585026213621_616879_43533","Distance Sensor","","
    Block")

BlockDependency("_19_0_2_5f801fb_1585072914854_101179_42319","DependOn","
    ","_19_0_2_5f801fb_1585026273526_752148_44250","
    _19_0_2_5f801fb_1585026213621_616879_43533")

block("_19_0_2_5f801fb_1585026250890_682235_44073","Brake","","Block")

BlockDependency("_19_0_2_5f801fb_1585073318834_548200_42347","DependOn","
    ","_19_0_2_5f801fb_1585026238757_873968_43893","
    _19_0_2_5f801fb_1585026250890_682235_44073")

block("_19_0_2_5f801fb_1585026242060_759745_43938","AdaptiveCar","","
    Block")


  blockparts("_19_0_2_5f801fb_1585072463108_702702_42211","PartOf","","
      _19_0_2_5f801fb_1585026238757_873968_43893","
      _19_0_2_5f801fb_1585026242060_759745_43938")
  blockparts("_19_0_2_5f801fb_1585072476569_537542_42243","PartOf","","
      _19_0_2_5f801fb_1585026229229_557851_43758","
      _19_0_2_5f801fb_1585026242060_759745_43938")
  blockparts("_19_0_2_5f801fb_1585072481547_865184_42259","PartOf","","
      _19_0_2_5f801fb_1585026258692_649637_44163","
      _19_0_2_5f801fb_1585026242060_759745_43938")
block("_19_0_2_5f801fb_1585026216677_300905_43578","Economic power
    consumption","","Block")
requirementToBlock("_19_0_2_5f801fb_1585076876625_810721_42827","
    SatisfiedBy","","_19_0_2_5f801fb_1585024355190_929810_42852","
    _19_0_2_5f801fb_1585026216677_300905_43578")

BlockDependency("_19_0_2_5f801fb_1585073323638_244848_42353","DependOn","
    ","_19_0_2_5f801fb_1585026238757_873968_43893","
    _19_0_2_5f801fb_1585026216677_300905_43578")

block("_19_0_2_5f801fb_1585026254293_890928_44118","Power Supply","","
    Block")

Blockaggregation("_19_0_2_5f801fb_1585073604618_498732_42375","IsSharedBy
    ","","_19_0_2_5f801fb_1585026219261_659151_43623","
    _19_0_2_5f801fb_1585026254293_890928_44118")
Blockaggregation("_19_0_2_5f801fb_1585073611531_51964_42391","IsSharedBy"
    ,"","_19_0_2_5f801fb_1585026222693_146070_43668","
    _19_0_2_5f801fb_1585026254293_890928_44118")
```

```
block("_19_0_2_5f801fb_1585026229229_557851_43758","Power Subsystem","","
    Block")


  blockparts("_19_0_2_5f801fb_1585073629835_442021_42407","PartOf","","
      _19_0_2_5f801fb_1585026245309_260416_43983","
      _19_0_2_5f801fb_1585026229229_557851_43758")
  blockparts("_19_0_2_5f801fb_1585073640338_133373_42423","PartOf","","
      _19_0_2_5f801fb_1585026254293_890928_44118","
      _19_0_2_5f801fb_1585026229229_557851_43758")
  blockparts("_19_0_2_5f801fb_1585073648450_288244_42439","PartOf","","
      _19_0_2_5f801fb_1585026211301_555359_43488","
      _19_0_2_5f801fb_1585026229229_557851_43758")
  blockparts("_19_0_2_5f801fb_1585073655514_553515_42455","PartOf","","
      _19_0_2_5f801fb_1585026248141_903744_44028","
      _19_0_2_5f801fb_1585026229229_557851_43758")
block("_19_0_2_5f801fb_1585026222693_146070_43668","Fuel","","Block")


block("_19_0_2_5f801fb_1585026235310_748842_43848","Fuel engine","","
    Block")

BlockDependency("_19_0_2_5f801fb_1585073583897_529454_42365","DependOn","
    ","_19_0_2_5f801fb_1585026245309_260416_43983","
    _19_0_2_5f801fb_1585026235310_748842_43848")

block("_19_0_2_5f801fb_1585026245309_260416_43983","Engine","","Block")


block("_19_0_2_5f801fb_1585026225973_235045_43713","Vibration Sensor","",
    "Block")

BlockDependency("_19_0_2_5f801fb_1585072652483_423004_42303","DependOn","
    ","_19_0_2_5f801fb_1585026265149_947387_44211","
    _19_0_2_5f801fb_1585026225973_235045_43713")

block("_19_0_2_5f801fb_1585026238757_873968_43893","Controller","","Block
    ")
requirementToBlock("_19_0_2_5f801fb_1585076404737_557035_42693","
    SatisfiedBy","","_19_0_2_5f801fb_1585024361756_6376_42938","
    _19_0_2_5f801fb_1585026238757_873968_43893")


block("_19_0_2_5f801fb_1585026211301_555359_43488","Wire","","Block")


block("_19_0_2_5f801fb_1585026258692_649637_44163","Environmental
    conditions","","Block")

Blockaggregation("_19_0_2_5f801fb_1585072635060_342708_42275","IsSharedBy
    ","","_19_0_2_5f801fb_1585026213621_616879_43533","
    _19_0_2_5f801fb_1585026258692_649637_44163")
Blockaggregation("_19_0_2_5f801fb_1585072953480_450784_42329","IsSharedBy
```

```
    ","","_19_0_2_5f801fb_1585026225973_235045_43713","
    _19_0_2_5f801fb_1585026258692_649637_44163")

block("_19_0_2_5f801fb_1585026248141_903744_44028","Pipe","","Block")


// ConstraintBlock
block("_19_0_2_5f801fb_1585026265149_947387_44211","Vibration indicator",
    "","ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1585076748296_472212_42775","
    SatisfiedBy","","_19_0_2_5f801fb_1585024333418_777761_42637","
    _19_0_2_5f801fb_1585026265149_947387_44211")

Blockparent("_19_0_2_5f801fb_1585072291331_463982_42205","ChildOf","","
    _19_0_2_5f801fb_1585026265149_947387_44211","
    _19_0_2_5f801fb_1585072222132_884521_42167")
block("_19_0_2_5f801fb_1585072222132_884521_42167","Indicator","","
    ConstraintBlock")

block("_19_0_2_5f801fb_1585026273526_752148_44250","Distance indicator","
    ","ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1585076625528_162871_42734","
    SatisfiedBy","","_19_0_2_5f801fb_1585024377890_862239_43110","
    _19_0_2_5f801fb_1585026273526_752148_44250")

Blockparent("_19_0_2_5f801fb_1585072266321_513403_42202","ChildOf","","
    _19_0_2_5f801fb_1585026273526_752148_44250","
    _19_0_2_5f801fb_1585072222132_884521_42167")
//Requirements
requirement("_19_0_2_5f801fb_1585024341233_278133_42680","Driving","The
    system SHALL support driving the car using acceleration, control, and
    power","R-08")
subrequirement("
    _19_0_2_5f801fb_1585024361756_6376_42938_19_0_2_5f801fb_1585024341233_
     278133_42680","ContainedBy","","
    _19_0_2_5f801fb_1585024361756_6376_42938","_19_0_2_ 5
    f801fb_1585024341233_278133_42680")
subrequirement("
    _19_0_2_5f801fb_1585026061603_466277_43358_19_0_2_5f801fb_1585024341233_
     278133_42680","ContainedBy","","
    _19_0_2_5f801fb_1585026061603_466277_43358","_19_0_2_ 5
    f801fb_1585024341233_278133_42680")
subrequirement("
    _19_0_2_5f801fb_1585024388846_380075_43239_19_0_2_5f801fb_1585024341233_
     278133_42680","ContainedBy","","
    _19_0_2_5f801fb_1585024388846_380075_43239","_19_0_2_ 5
    f801fb_1585024341233_278133_42680")
subrequirement("
    _19_0_2_5f801fb_1585024392253_566642_43282_19_0_2_5f801fb_1585024341233
     _278133_42680","ContainedBy","","
    _19_0_2_5f801fb_1585024392253_566642_43282","_19_0_2_ 5
    f801fb_1585024341233_278133_42680")

requirement("_19_0_2_5f801fb_1585026061603_466277_43358","Acceleration
```

```
    Req", "The system SHALL accelerate using two engines, fuel engine and
    electric engine","R-05")
requirementlink("_19_0_2_5f801fb_1585075444561_602495_42499","DerivedFrom
    ","", "_19_0_2_5f801fb_1585024355190_929810_42852","
    _19_0_2_5f801fb_1585026061603_466277_43358")

requirement("_19_0_2_5f801fb_1585024380953_268910_43153","Adaptation for
    distance", "The system MAY use braking OR steering to keep the
    distance between the car and the other cars over 5 meters.","R-15")

requirement("_19_0_2_5f801fb_1585024351846_858122_42809","Power network",
    "The system SHALL contain two transportation nets, pipes and wires","R
    -10")
requirementlink("_19_0_2_5f801fb_1585076208361_390721_42640","DerivedFrom
    ","","_19_0_2_5f801fb _1585024344154_915734_42723","
    _19_0_2_5f801fb_1585024351846_858122_42809")
requirementlink("_19_0_2_5f801fb_1585076204847_600890_42625","DerivedFrom
    ","","_19_0_2_5f801fb _1585024366507_363993_42981","
    _19_0_2_5f801fb_1585024351846_858122_42809")

requirement("_19_0_2_5f801fb_1585024374357_354966_43067","Dynamic control
     for adaptation","The system MAY use brakes OR steering and Distance
    sensors provide the distance between the surrounding objects and the
    car.","R-13")

requirement("_19_0_2_5f801fb_1585024347829_832841_42766","Steering Req","
    The system MAY use steering OR braking to manage the distance between
    the car and the surrounding objects","R-03")

requirement("_19_0_2_5f801fb_1585024388846_380075_43239","Comfortable
    driving","The system SHALL minimize the vibration level","R-15")
requirementlink("_19_0_2_5f801fb_1585076023514_913760_42581","DerivedFrom
    ","","_19_0_2_5f801fb_1585024333418_777761_42637","
    _19_0_2_5f801fb_1585024388846_380075_43239")
requirementlink("_19_0_2_5f801fb_1585076052288_946292_42596","DerivedFrom
    ","","_19_0_2_5f801fb_1585024358490_669314_42895","
    _19_0_2_5f801fb_1585024388846_380075_43239")

requirement("_19_0_2_5f801fb_1585024366507_363993_42981","Fuel network","
    The system SHALL use pipes to transport fuel power","R-11")

requirement("_19_0_2_5f801fb_1585024377890_862239_43110","Accepted
    distance","The system SHALL keep the distance between the car and
    other cars AS CLOSE AS POSSIBLE to 25 meters and >= 10 meters.","R-14"
    )

requirement("_19_0_2_5f801fb_1585024358490_669314_42895","Accepted
    vibration","The system SHALL keep the vibration level AS CLOSE AS
    POSSIBLE to 0 dm/s2 and <= 10 dm/s2","R-07")

requirement("_19_0_2_5f801fb_1585024370822_154602_43024","Brakes req","
    The system SHALL use brakes to manage the speed","R-02")

requirement("_19_0_2_5f801fb_1585024333418_777761_42637","Adaptation for
```

```
    vibration","The system MAY use fuel engine to keep the vibration level
     less than 20 dm/s2","R-04")

requirement("_19_0_2_5f801fb_1585024355190_929810_42852","Economic
    driving","The system MAY use fuel OR battery to reduce the cost","R-06
    ")

requirement("_19_0_2_5f801fb_1585024344154_915734_42723","Electric
    network","The system SHALL use wires to transport electric power","R
    -12")

requirement("_19_0_2_5f801fb_1585024361756_6376_42938","Dynamic control",
    "The system SHALL monitor the environment and react based on the
    current situation","R-01")
requirementlink("_19_0_2_5f801fb_1585075743016_357732_42514","DerivedFrom
    ","","_19_0_2_5f801fb_1585024347829_832841_42766","
    _19_0_2_5f801fb_1585024361756_6376_42938")
requirementlink("_19_0_2_5f801fb_1585075969768_415106_42568","DerivedFrom
    ","","_19_0_2_5f801fb_1585024377890_862239_43110","
    _19_0_2_5f801fb_1585024361756_6376_42938")
requirementlink("_19_0_2_5f801fb_1585075748016_202994_42529","DerivedFrom
    ","","_19_0_2_5f801fb_1585024370822_154602_43024","
    _19_0_2_5f801fb_1585024361756_6376_42938")
requirementlink("_19_0_2_5f801fb_1585075753768_192764_42542","DerivedFrom
    ","","_19_0_2_5f801fb_1585024374357_354966_43067","
    _19_0_2_5f801fb_1585024361756_6376_42938")
requirementlink("_19_0_2_5f801fb_1585075760232_337773_42555","DerivedFrom
    ","","_19_0_2_5f801fb_1585024380953_268910_43153","
    _19_0_2_5f801fb_1585024361756_6376_42938")

requirement("_19_0_2_5f801fb_1585024392253_566642_43282","Power sources",
    "The system MAY use fuel OR battery","R-09")
requirementlink("_19_0_2_5f801fb_1585076166768_290618_42610","DerivedFrom
    ","","_19_0_2_5f801fb_1585024351846_858122_42809","
    _19_0_2_5f801fb_1585024392253_566642_43282")
endImport
```

A.2 Arithmetic Functions of the Hybrid Car's Engine Example

Listings from A.2 to A.14 show the arithmetic functions generated from the GRL/feature model of the hybrid car's engine system example, in several programming languages.

Listing A.2: C++ mathematical function in GRLadaptivecaR.cpp

```
#include <iostream>
using namespace std;
double GrladaptivecaR( double Manage_speeD,double VibratioN,double FueL,
    double WireS,double Electric_enginE,double DistancE,double BatterY,
```

```
    double PipeS,double Fuel_enginE){
  double expr = 0.40000152587890625*std::max(0, std::min(100.0, 0.5*
      Electric_enginE - 0.5*Fuel_enginE)) + 0.40000152587890625*std::max
      (0, std::min(100.0, -0.5*Electric_enginE + 0.5*Fuel_enginE + 0.5*((
      VibratioN <= 0.0) ? (
    100.0
)
: ((VibratioN <= 10.0 && VibratioN > 0) ? (
    50.0*std::fabs(0.10000000000000001*VibratioN - 1.0) + 50.0
)
: ((VibratioN > 10.0 && VibratioN < 20.0) ? (
    -50.0*std::fabs(0.10000000000000001*VibratioN - 1.0) + 50.0
)
: (
    0
))))))) + 0.59999847412109375*std::min(Max(BatterY, FueL), std::min(Max(
    PipeS, WireS), std::max(0, std::min(100.0, std::min(-0.5*Manage_speeD
    + 1.0*Max(Electric_enginE, Fuel_enginE), std::max(Manage_speeD, 1.0*((
    DistancE >= 25.0) ? (
    100.0
)
: ((DistancE >= 10.0) ? (
    50.0*std::fabs(0.066666666666666666*DistancE - 0.66666666666666663) +
        50.0
)
: ((DistancE > 5.0) ? (
    -50.0*std::fabs(0.20000000000000001*DistancE - 2.0) + 50.0
)
: (
    0
))))))))));
  return expr;
}
```

Listing A.3: C mathematical function in GRLadaptivecaR.c

```
#include <stdio.h>

double GrladaptivecaR( double Manage_speeD,double VibratioN,double FueL,
    double WireS,double Electric_enginE,double DistancE,double BatterY,
    double PipeS,double Fuel_enginE){
  double expr = 0.40000152587890625*maxl(0, minl(100.0, 0.5*
      Electric_enginE - 0.5*Fuel_enginE)) + 0.40000152587890625*maxl(0,
      minl(100.0, -0.5*Electric_enginE + 0.5*Fuel_enginE + 0.5*((VibratioN
       <= 0.0) ? (
    100.0
)
: ((VibratioN <= 10.0 && VibratioN > 0) ? (
    50.0*abs(0.10000000000000001*VibratioN - 1.0) + 50.0
)
: ((VibratioN > 10.0 && VibratioN < 20.0) ? (
    -50.0*abs(0.10000000000000001*VibratioN - 1.0) + 50.0
```

```
)
: (
    0
))))))) + 0.59999847412109375*minl(maxl(BatterY, FueL), maxl(PipeS, WireS)
    , maxl(0, minl(100.0, -0.5*Manage_speeD + 1.0*maxl(Electric_enginE,
    Fuel_enginE), maxl(Manage_speeD, 1.0*((DistancE >= 25.0) ? (
    100.0
)
: ((DistancE >= 10.0) ? (
    50.0*abs(0.06666666666666666*DistancE - 0.66666666666666663) + 50.0
)
: ((DistancE > 5.0) ? (
    -50.0*abs(0.20000000000000001*DistancE - 2.0) + 50.0
)
: (
    0
))))))));
  return expr;
}
```

Listing A.4: JavaScript mathematical function in GRLadaptivecaR.js

```
function GrladaptivecaR( Manage_speeD,VibratioN,FueL,WireS,
    Electric_enginE,DistancE,BatterY,PipeS,Fuel_enginE){
  expr = 0.4*Math.max(0, Math.min(100.0, 0.5*Electric_enginE - 0.5*
      Fuel_enginE)) + 0.4*Math.max(0, Math.min(100.0, -0.5*Electric_enginE
       + 0.5*Fuel_enginE + 0.5*((VibratioN <= 0.0) ? (
    100.0
)
: ((VibratioN <= 10.0 && VibratioN > 0) ? (
    50.0*Math.abs(0.1*VibratioN - 1.0) + 50.0
)
: ((VibratioN > 10.0 && VibratioN < 20.0) ? (
    -50.0*Math.abs(0.1*VibratioN - 1.0) + 50.0
)
: (
    0
))))))) + 0.6*Math.min(Math.max(BatterY, FueL), Math.max(PipeS, WireS),
    Math.max(0, Math.min(100.0, -0.5*Manage_speeD + 1.0*Math.max(
    Electric_enginE, Fuel_enginE), Math.max(Manage_speeD, 1.0*((DistancE
    >= 25.0) ? (
    100.0
)
: ((DistancE >= 10.0) ? (
    50.0*Math.abs(0.0666666666666667*DistancE - 0.666666666666667) + 50.0
)
: ((DistancE > 5.0) ? (
    -50.0*Math.abs(0.2*DistancE - 2.0) + 50.0
)
: (
    0
))))))));
```

```
  return expr;
}
```

Listing A.5: Matlab mathematical function in GRLadaptivecaR.m

```
function expr=GrladaptivecaR( Manage_speeD,VibratioN,FueL,WireS,
    Electric_enginE,DistancE,BatteryY,PipeS,Fuel_enginE)
  expr = 0.4*max(0, min(100.0, 0.5*Electric_enginE - 0.5*Fuel_enginE)) +
      0.4*max(0, min(100.0, -0.5*Electric_enginE + 0.5*Fuel_enginE +
      0.5*((VibratioN <= 0.0).*(100.0) + (~(VibratioN <= 0.0)).*( ...
(VibratioN <= 10.0 & VibratioN > 0).*(50.0*abs(0.1*VibratioN - 1.0) +
    50.0) + (~(VibratioN <= 10.0 & VibratioN > 0)).*( ...
(VibratioN > 10.0 & VibratioN < 20.0).*(-50.0*abs(0.1*VibratioN - 1.0) +
    50.0) + (~(VibratioN > 10.0 & VibratioN < 20.0)).*(0)))))) + 0.6*min(
    Max(BatteryY, FueL), min(Max(PipeS, WireS), max(0, min(100.0, min(-0.5*
    Manage_speeD + 1.0*Max(Electric_enginE, Fuel_enginE), max(Manage_speeD
    , 1.0*((DistancE >= 25.0).*(100.0) + (~(DistancE >= 25.0)).*( ...
(DistancE >= 10.0).*(50.0*abs(0.0666666666666667*DistancE -
    0.666666666666667) + 50.0) + (~(DistancE >= 10.0)).*( ...
(DistancE > 5.0).*(-50.0*abs(0.2*DistancE - 2.0) + 50.0) + (~(DistancE >
    5.0)).*(0)))))))))
end
```

Listing A.6: Python mathematical function in GRLadaptivecaR.py

```
def GrladaptivecaR( Manage_speeD,VibratioN,FueL,WireS,Electric_enginE,
    DistancE,BatteryY,PipeS,Fuel_enginE):
  expr = 0.4*max(0, min(100.0, 0.5*Electric_enginE - 0.5*Fuel_enginE)) +
      0.4*max(0, min(100.0, -0.5*Electric_enginE + 0.5*Fuel_enginE +
      0.5*((100.0) if (VibratioN <= 0.0) else (((50.0*abs(0.1*VibratioN -
      1.0) + 50.0) if (VibratioN <= 10.0 and VibratioN > 0) else (((-50.0*
      abs(0.1*VibratioN - 1.0) + 50.0) if (VibratioN > 10.0 and VibratioN
      < 20.0) else (((0) if (True) else None))))))))) + 0.6*min(max(
      BatteryY, FueL), max(PipeS, WireS), max(0, min(100.0, -0.5*
      Manage_speeD + 1.0*max(Electric_enginE, Fuel_enginE), max(
      Manage_speeD, 1.0*((100.0) if (DistancE >= 25.0) else (((50.0*abs
      (0.0666666666666667*DistancE - 0.666666666666667) + 50.0) if (
      DistancE >= 10.0) else (((-50.0*abs(0.2*DistancE - 2.0) + 50.0) if (
      DistancE > 5.0) else (((0) if (True) else None)))))))))))
  return expr
```

Listing A.7: R mathematical function in GRLadaptivecaR.r

```
GrladaptivecaR <- function( Manage_speeD,VibratioN,FueL,WireS,Electric_
    enginE,DistancE,BatteryY,PipeS,Fuel_enginE){
  expr = 0.4*max(0, min(100.0, 0.5*Electric_enginE - 0.5*Fuel_enginE)) +
      0.4*max(0, min(100.0, -0.5*Electric_enginE + 0.5*Fuel_enginE + 0.5*
      ifelse(VibratioN <= 0.0,100.0,ifelse(VibratioN <= 10.0 & VibratioN >
       0,50.0*abs(0.1*VibratioN - 1.0) + 50.0,ifelse(VibratioN > 10.0 &
```

```
VibratioN < 20.0,-50.0*abs(0.1*VibratioN - 1.0) + 50.0,0))))) + 0.6*
min(max(BatterY, FueL), max(PipeS, WireS), max(0, min(100.0, -0.5*
Manage_speeD + 1.0*max(Electric_enginE, Fuel_enginE), max(Manage_
speeD, 1.0*ifelse(DistancE >= 25.0,100.0,ifelse(DistancE >=
10.0,50.0*abs(0.0666666666666667*DistancE - 0.666666666666667) +
50.0,ifelse(DistancE > 5.0,-50.0*abs(0.2*DistancE - 2.0) + 50.0,0)))
))))) }
```

Listing A.8: C++ mathematical function in AdaptivecarF.cpp

```
#include <iostream>
using namespace std;

double AdaptivecarF( double Manage_speeD,double FueL,double
    Manage_distancE,double WireS,double Electric_enginE,double BatterY,
    double PipeS,double Fuel_enginE){
  double expr = std::min(std::min(100.0,100.0*(Manage_distancE +
      Manage_speeD)/Max(1.0, Manage_distancE + Manage_speeD)), std::min
      (100.0,100.0*(std::min(BatterY, std::min(Electric_enginE, WireS)) +
      std::min(FueL, std::min(Fuel_enginE, PipeS)))/std::max(1.000, std::
      max(BatterY, std::max(Electric_enginE, WireS)) + std::max(FueL, std
      ::max(Fuel_enginE, PipeS)))));
  return expr;
}
```

Listing A.9: JavaScript mathematical function in AdaptivecarF.js

```
function AdaptivecarF( Manage_speeD,FueL,Manage_distancE,WireS,
    Electric_enginE,BatterY,PipeS,Fuel_enginE){
  expr = Math.min(Math.min(100.0,100.0*(Manage_distancE + Manage_speeD)/
      Math.max(1.0, Manage_distancE + Manage_speeD)), Math.min
      (100.0,100.0*(Math.min(BatterY, Electric_enginE, WireS) + Math.min(
      FueL, Fuel_enginE, PipeS))/Math.max(1.0, Math.max(BatterY,
      Electric_enginE, WireS) + Math.max(FueL, Fuel_enginE, PipeS))));
  return expr;
}
```

Listing A.10: Matlab mathematical function in AdaptivecarF.m

```
function expr=AdaptivecarF( Manage_speeD,FueL,Manage_distancE,WireS,
    Electric_enginE,BatterY,PipeS,Fuel_enginE)
  expr = min(min(100.0,100.0*(Manage_distancE + Manage_speeD)/Max(1.0,
      Manage_distancE + Manage_speeD)), min(100.0,100.0*(min(BatterY, min(
      Electric_enginE, WireS)) + min(FueL, min(Fuel_enginE, PipeS))./max
      (1.000, max(BatterY, max(Electric_enginE, WireS)) + max(FueL, max(
      Fuel_enginE, PipeS)))))
end
```

Listing A.11: Python mathematical function in AdaptivecarF.py

```
def AdaptivecarF( Manage_speeD,FueL,Manage_distancE,WireS,Electric_enginE
    ,BatterY,PipeS,Fuel_enginE):
  expr = min(min(100.0,100.0*(Manage_distancE + Manage_speeD)/max(1.0,
      Manage_distancE + Manage_speeD)), min(100.0,100.0*(min(BatterY,
      Electric_enginE, WireS) + min(FueL, Fuel_enginE, PipeS))/max(1.0,
      max(BatterY, Electric_enginE, WireS) + max(FueL, Fuel_enginE, PipeS)
      )))
  return expr
```

Listing A.12: R mathematical function in AdaptivecarF.r

```
AdaptivecarF <- function( Manage_speeD,FueL,Manage_distancE,WireS,
    Electric_enginE,BatterY,PipeS,Fuel_enginE){
  expr = min(min(100.0.100.0*(Manage_distancE + Manage_speeD)/max(1.0,
      Manage_distancE + Manage_speeD)), min(100.0,100.0*(min(BatterY,
      Electric_enginE, WireS) + min(FueL, Fuel_enginE, PipeS))/max(1.0,
      max(BatterY, Electric_enginE, WireS) + max(FueL, Fuel_enginE, PipeS)
      ))) }
```

Listing A.13: C mathematical function in AdaptivecarF.c

```
#include <stdio.h>

double AdaptivecarF( double Manage_speeD,double FueL,double
    Manage_distancE,double WireS,double Electric_enginE,double BatterY,
    double PipeS,double Fuel_enginE){
  double expr = fmin(fmin(100.0,100.0*(Manage_distancE + Manage_speeD)/
      fmax(1.0, Manage_distancE + Manage_speeD)), fmin(100.0,100.0*(fmin(
      BatterY, fmin(Electric_enginE, WireS)) + fmin(FueL, fmin(Fuel_enginE
      , PipeS)))/fmax(1.0, fmax(BatterY, fmax(Electric_enginE, WireS)) +
      fmax(FueL, fmax(Fuel_enginE, PipeS)))));
  return expr;
}
```

Listing A.14: Java program used to test the generated functions of the Adaptive car system

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
public class ModelAll{
// Feature Model's function
  public double AdaptivecarF(double Manage_speeD,double FueL,double
      Manage_distancE,double WireS,double Electric_enginE,double BatterY,
      double PipeS,double Fuel_enginE){
```

```
    double expr = Math.min(Math.min(100.0,100.0*Math.max(0, (
        Manage_distancE + Manage_speeD)/Math.max(1.0, Manage_distancE +
        Manage_speeD))), Math.min(100.0,100.0*Math.max(0, (Math.min(
        BatterY, Math.min(Electric_enginE, WireS)) + Math.min(FueL, Math.
        min(Fuel_enginE, PipeS)))/Math.max(1.000, Math.max(BatterY, Math.
        max(Electric_enginE, WireS)) + Math.max(FueL, Math.max(Fuel_enginE
        , PipeS))))));
    return expr;
  }
// GRL function
public double GrladaptivecaR(double Manage_speeD,double VibratioN,double
    FueL,double WireS,double Electric_enginE,double DistancE,double
    BatterY,double PipeS,double Fuel_enginE){
    double expr = ((Math.min(Math.min(Math.max(0.0 , Math.min(100.0 ,
        (-50*Manage_speeD+Math.max(Fuel_enginE , Electric_enginE)*100.0) /
         100.0)),Math.max(Math.max(0.0 , Math.min(100.0 , (100*((DistancE
        >=25.0) ? 100 : (10.0<=DistancE)&&(DistancE<25.0) ? (Math.abs( (
        DistancE - 10.0) / 15.0) * 50 + 50) : (5.0<DistancE)&&(DistancE
        <10.0) ? (-Math.abs( (DistancE - 10.0) / -5.0) * 50 + 50 ) : 0)) /
         100.0)) , Manage_speeD)),Math.min(Math.max(PipeS , WireS),Math.
        max(FueL , BatterY)))*100.0) / 100*60+(Math.max(0.0 , Math.min
        (100.0 , (-50*Fuel_enginE+50*Electric_enginE) / 100.0))*100.0+Math
        .max(0.0 , Math.min(100.0 , (50*((VibratioN<=0.0) ?100 : (0.0<
        VibratioN)&&(VibratioN<=10.0) ? (Math.abs( (VibratioN - 10.0) /
        10.0) * 50 + 50) :  (10.0<VibratioN)&&(VibratioN<20.0) ? (-Math.
        abs( (VibratioN - 10.0) / -10.0) * 50 + 50) : 0)+ -50*
        Electric_enginE+50*Fuel_enginE) / 100.0))*100.0) / 100*40) / 100 ;

    return expr;
  }
// User Actor's function
public double user(double VibratioN, double Fuel_enginE, double
    Electric_enginE){
  double UseR = (Math.max(0.0 , Math.min(100.0 , (-50*Fuel_enginE+50*
      Electric_enginE) / 100.0))*100.0+Math.max(0.0 , Math.min(100.0 ,
      (50* (( VibratioN<=0.0) ? 100 : (0.0<VibratioN)&&(VibratioN<=10.0) ?
       (Math.abs( (VibratioN - 10.0) / 10.0) * 50 + 50) : (10.0<VibratioN)
      &&(VibratioN<20.0) ? (-Math.abs( (VibratioN - 10.0) / -10.0) * 50 +
      50) : 0 )+-50*Electric_enginE+50*Fuel_enginE) / 100.0))*100.0) /
      100;

  return UseR;
}
// System Actor's function
public double System(double Manage_speeD,double FueL,double WireS,double
    Electric_enginE,double DistancE,double BatterY,double PipeS,double
    Fuel_enginE){
  double SysteM = (Math.min(Math.min(Math.max(0.0 , Math.min(100.0 ,
      (-50*Manage_speeD+Math.max(Fuel_enginE , Electric_enginE)*100.0) /
      100.0)),Math.max(Math.max(0.0 , Math.min(100.0 , (100* ((DistancE >=
       25.0) ? (
  100.0
)
: ((DistancE >= 10.0) ? (
```

```
    50.0*Math.abs(0.06666666666666666*DistancE - 0.66666666666666663) +
        50.0
)
: ((DistancE > 5.0) ? (
    -50.0*Math.abs(0.20000000000000001*DistancE - 2.0) + 50.0
)
: (
    0
))))) / 100.0)) , Manage_speeD)),Math.min(Math.max(PipeS , WireS),Math.
    max(FueL , BatterY)))*100.0) / 100;

  return SysteM;
}
// Main function passes different environmental contexts (distance and
    vibration) with all possible strategies while printing only the valid
    solutions

public static void main(String[] args){
  FileWriter w = null;
        try {
            w = new FileWriter("ResultGRF_V25_D3.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
        BufferedWriter bw = new BufferedWriter(w);
        PrintWriter wr = new PrintWriter(bw);

        try {

  ModelAll ValidModel= new ModelAll();

  double VibratioN, DistancE,Userv,SysteM;
  VibratioN=25.0;
  DistancE=3.0;
  Manage_distancE=Math.max(0.0 , Math.min(100.0 , (100*((DistancE>=25.0)
      ? 100 : (10.0<=DistancE)&&(DistancE<25.0) ? (Math.abs( (DistancE -
      10.0) / 15.0) * 50 + 50) : (5.0<DistancE)&&(DistancE<10.0) ? (-Math.
      abs( (DistancE - 10.0) / -5.0) * 50 + 50 ) : 0)) / 100.0))>0?100:0;
  int ValidSolution=0;
  int NumberOfSolutions=0;
  double result;
  for( int Manage_speeD=0; Manage_speeD<=100; Manage_speeD+=100)
    for( int FueL=0; FueL<=100; FueL+=100)
      for( int WireS=0; WireS<=100; WireS+=100)
        for( int Electric_enginE=0; Electric_enginE<=100; Electric_enginE
            +=100)
          for( int BatterY=0; BatterY<=100; BatterY+=100)
            for( int PipeS=0; PipeS<=100; PipeS+=100)
              for( int Fuel_enginE=0; Fuel_enginE<=100; Fuel_enginE+=100)
                  {
                  if (ValidModel.AdaptivecarF( Manage_speeD,  FueL,
                      Manage_distancE,  WireS,  Electric_enginE,  BatterY,
                        PipeS,  Fuel_enginE)==100.0)
                  {
```

```
                    result=ValidModel.GrladaptivecaR( Manage_speeD,
                        VibratioN, FueL, WireS, Electric_enginE, DistancE,
                        BatterY, PipeS,Fuel_enginE);
                    SysteM=ValidModel.System(Manage_speeD, FueL, WireS,
                        Electric_enginE, DistancE, BatterY, PipeS,
                        Fuel_enginE);
                    Userv=ValidModel.user(VibratioN, Fuel_enginE,
                        Electric_enginE);
                    ValidSolution+=1;
                              wr.write("Manage_speeD    FueL
                                  Manage_distancE      WireS
                                  Electric_enginE    BatterY    PipeS
                                  Fuel_enginE      FeatureModel UseR System"+
                                  '\n');
                    wr.write(String.valueOf(Manage_speeD));
                     wr.write("        ");
                     wr.write(String.valueOf(FueL));
                     wr.write("          ");
                     wr.write(String.valueOf(Manage_distancE));
                     wr.write("          ");
                     wr.write(String.valueOf(WireS));
                     wr.write("        ");
                     wr.write(String.valueOf(Electric_enginE));
                     wr.write("          ");
                     wr.write(String.valueOf(BatterY));
                     wr.write("         ");
                     wr.write(String.valueOf(PipeS));
                     wr.write("       ");
                     wr.write(String.valueOf(Fuel_enginE));
                     wr.write("         ");

                     wr.write(String.valueOf(result));
                     wr.write(String.valueOf(Userv));
                     wr.write(String.valueOf(SysteM));
                     wr.append('\n');
                     wr.write("
                         ____________________________________________________________
                         "+'\n');
                    }
                    NumberOfSolutions+=1;
                  }



wr.write("Valid solutions= "+ String.valueOf(ValidSolution) + "  Out of "
   + String.valueOf(NumberOfSolutions));
wr.close();
            bw.close();
        } catch (IOException e) {
            e.printStackTrace();
        } }
 }
```

Appendix B

Smart Home Case Study Description and Functions

This appendix supplements Chapter 7 and provides an informal description of the Smart Home case study (based on material from the SM@RT Team [87]), the related arithmetic functions generated from the corresponding GRL/feature model by jUCMNav (Section 7.5), and the corresponding optimization model for IBM CPLEX' constraint programming optimizer (Section 7.6).

B.1 Description

A Smart Home (SH) aims to simplify the life of its inhabitants, especially if the latter are elderly people or people with disabilities. An SM has to address a number of security, societal, legislative, economical, and ecological challenges and constraints. This part sketches some ways in which the services in an SM could address these challenges.

1. **Safety and security**: As any other home, Smart Homes must provide a certain level of protection against unwanted physical access by burglars. Yet, Smart Homes are particularly vulnerable to cyber-attacks and must offer effective protection against them. Security breaches or poor authentication mechanisms may allow hackers to take control of the system or steal personal information. However, compared to classical homes, smart homes can leverage sensors and devices to detect intrusions and respond by alerting, e.g., inhabitants, care givers, or the police, as appropriate. Note that this functionality could also be used to protect the home while the inhabitants are on vacation.

   Other emergencies could be detected and dealt with in similar fashion. For instance, the mobility of the home inhabitant is often so compromised that they cannot get up after a fall; data collected from sensors, microphones, or cameras could be used to automatically detect the fall and request help accompanied by relevant information such as location of the fall and the person's vital signs. Similar responses could be initiated after certain medical emergencies such as heart attacks, strokes, or seizures.

Another important concern of Smart Homes is that the information collected is stored safely and securely and in a way that prevents the violation of any privacy or confidentiality rules to an acceptable degree. This concern is exacerbated by the fact that the software architecture of the Smart Home may, essentially, be that of a distributed system that allows some remote parties not only to connect, but also to collect data and exert a certain amount of control. Measures must therefore be taken to ensure that, e.g., proper authentication, encryption, monitoring, and data storage techniques are employed.

A related concern is that system reliability should be sufficiently high. In cases where critical functionality is lost (e.g., due to software bugs, broken hardware, power outages), adverse effects should be kept to an acceptable level and not, e.g., jeopardize the health, well-being, and assets of the inhabitants. The system should take, or at least support, mitigating or correcting measures as appropriate.

2. **Accommodations to physical, medical, and mental restrictions of inhabitants**: A Smart Home should be able to adapt to the physical, medical, or mental restrictions of its inhabitants and to provide appropriate accommodations to them to ensure a safe and convenient living environment. Examples include voice controlled appliances (e.g., telephone, TV, coffee machine), height-adjustable cabinets, warning or auto-off functionality for stoves that have been left on, and warning or auto-lock functionality for doors that have been left open. Inhabitants with dementia or Alzheimer's could be supported in the completion of certain tasks through suitable, automatically given cues and reminders provided by audio or video. Anomalous, potentially harmful behaviour such as leaving the home at an usual time or for an unusual duration could be detected. Moreover, violations of important rules specifying, e.g., a healthy diet or the proper timing and dosage of medication could be detected, and used to trigger appropriate action.

   Finally, there is significant potential to leverage data mining or learning to allow the Smart Home to identify its inhabitants' habits, schedules, preferences, and changing physical or cognitive abilities. This information could be used to adapt the home to reduce costs without compromising comfort (by, e.g., keeping unused rooms at a lower temperature in Winter), to increase safety (by, e.g., facilitating the detection of unusual behaviour), or to support care givers (by, e.g., providing information supporting a diagnosis or indicating an improvement or deterioration of a medical condition).

3. **Environment and economy**: Smart Homes should be energy efficient to meet local laws and regulations governing energy consumption, but also to reduce costs in general. The reduction of room temperature when inhabitants are sleeping is one example. Moreover, a smart home might also be able to leverage information about the behaviour of its inhabitants to, e.g., reduce the use of heating or air conditioning in rooms that are expected to be unused for a sufficiently long time. Remote temperature control would allow inhabitants to adjust the temperature while on vacation or during an unexpected trip. Automatic blinds could be used to help heat or cool the home while inhabitants are absent. If appropriate, certain appliances

with high energy needs (such as washing machines, dryers, and dishwashers) might automatically schedule their operation to 'off-peak' times with lower electricity rates.

But, apart from reducing energy consumption, a Smart Home of the future is also likely to contribute to the production of energy. Solar panels and wind turbines might automatically charge batteries in the home or in the car. Warm water could be produced by a solar water heater. Information from local weather forecasts could be used to optimize energy storage capacity and cost savings given certain expected usage demands.

B.2 Python/SymPy Code Generated using jUCMNav

Listing B.1 displays the arithmetic functions of the Smart Home Management System generated from its GRL and feature models by jUCMNav.

Listing B.1: Python/SymPy code generated in SmartHome.py

```
from MathTo import *
from sympy import *
import sys
import os

# Creating a folder using Model name
ModelName= 'UrnspeC'
if not os.path.exists(ModelName):
    os.makedirs(ModelName)
os.chdir(ModelName)

# Initalize all the variables
Minimize_consumptioN = Symbol('Minimize_consumptioN')
Manage_energY = Symbol('Manage_energY')
Manage_applianceS = Symbol('Manage_applianceS')
Wind_turbines = Symbol('Wind_turbines')
Off_heateR = Symbol('Off_heateR')
On_heateR = Symbol('On_heateR')
Off_dish_washeR = Symbol('Off_dish_washeR')
On_dish_washeR = Symbol('On_dish_washeR')
Maximize_energy_productioN = Symbol('Maximize_energy_productioN')
Wind_speeD = Symbol('Wind_speeD')
Electricity_pricE = Symbol('Electricity_pricE')
LighT = Symbol('LighT')
Smart_griD = Symbol('Smart_griD')
Manage_homE = Symbol('Manage_homE')
Solar_paneL = Symbol('Solar_paneL')
Provided_serviceS = Symbol('Provided_serviceS')
WifI = Symbol('WifI')
InterneT = Symbol('InterneT')
Response_timE = Symbol('Response_timE')
Open_windowS = Symbol('Open_windowS')
On_air_conditioneR = Symbol('On_air_conditioneR')
```

```
Off_air_conditioneR = Symbol('Off_air_conditioneR')
Close_windowS = Symbol('Close_windowS')
On_water_heateR = Symbol('On_water_heateR')
Off_water_heateR = Symbol('Off_water_heateR')
On_air_ventilatoR = Symbol('On_air_ventilatoR')
Off_air_ventilatoR = Symbol('Off_air_ventilatoR')
Suitable_weatheR = Symbol('Suitable_weatheR')
Maximize_privacY = Symbol('Maximize_privacY')
Maximize_comforT = Symbol('Maximize_comforT')
Normalize_inside_temperaturE = Symbol('Normalize_inside_temperaturE')
Heating_timE = Symbol('Heating_timE')
Heating_water_timE = Symbol('Heating_water_timE')
Cooling_timE = Symbol('Cooling_timE')
Remote_coolinG = Symbol('Remote_coolinG')
Remote_heatinG = Symbol('Remote_heatinG')
Remote_water_heatinG = Symbol('Remote_water_heatinG')
FulL = Symbol('FulL')
Air_conditioner_schedulE = Symbol('Air_conditioner_schedulE')
Water_heater_schedulE = Symbol('Water_heater_schedulE')
Heater_schedulE = Symbol('Heater_schedulE')
High_temperaturE = Symbol('High_temperaturE')
NormalizE = Symbol('NormalizE')
Smoke_detectioN = Symbol('Smoke_detectioN')
Operate_related_taskS = Symbol('Operate_related_taskS')
Refresh_inside_aiR = Symbol('Refresh_inside_aiR')
Low_temperaturE = Symbol('Low_temperaturE')
User_permissioN = Symbol('User_permissioN')
Air_conditioner_remote_accesS = Symbol('Air_conditioner_remote_accesS')
Water_heater_remote_accesS = Symbol('Water_heater_remote_accesS')
Heater_remote_accesS = Symbol('Heater_remote_accesS')
Motion_sensorS = Symbol('Motion_sensorS')
Light_sensorS = Symbol('Light_sensorS')
Smoke_detectoR = Symbol('Smoke_detectoR')
Smart_security_camerA = Symbol('Smart_security_camerA')
Wind_sensorS = Symbol('Wind_sensorS')
No_suspicious_behaviour_detecteD = Symbol('
    No_suspicious_behaviour_detecteD')
Monitor_the_environmenT = Symbol('Monitor_the_environmenT')
Set_thermostatE = Symbol('Set_thermostatE')
Infrared_sensoR = Symbol('Infrared_sensoR')
Cyber_attack_detectioN = Symbol('Cyber_attack_detectioN')
Maximize_securitY = Symbol('Maximize_securitY')
Temperature_settinG = Symbol('Temperature_settinG')
Temperature_remote_accesS = Symbol('Temperature_remote_accesS')
Detect_suspicious_behaviouR = Symbol('Detect_suspicious_behaviouR')
Learning_techniquE = Symbol('Learning_techniquE')
Cooling_habitS = Symbol('Cooling_habitS')
Heating_habitS = Symbol('Heating_habitS')
FirE = Symbol('FirE')
Enable_water_heater_accesS = Symbol('Enable_water_heater_accesS')
Enable_air_conditioner_accesS = Symbol('Enable_air_conditioner_accesS')
Enable_heater_accesS = Symbol('Enable_heater_accesS')
Enable_heater_schedulE = Symbol('Enable_heater_schedulE')
Enable_air_conditioner_schedulE = Symbol('Enable_air_conditioner_schedulE
```

```
    ')
Enable_water_heater_schedulE = Symbol('Enable_water_heater_schedulE')
Enable_temperature_accesS = Symbol('Enable_temperature_accesS')
Minimize_consumptioN = Max(0.0 , Min(100.0 , (25*Open_windowS+50*InterneT
    +25*Water_heater_schedulE+25*Heater_schedulE+25*
    Air_conditioner_schedulE+75*Learning_techniquE+-25*On_air_ventilatoR)
    / 100.0))
Wind_turbines = Max(0.0 , Min(100.0 , (100*Wind_speeD) / 100.0))
Manage_energY = Max(Wind_turbines,Max(Solar_paneL,Smart_griD))
Manage_applianceS = Max(Max(On_water_heateR,On_dish_washeR),Max(
    Off_water_heateR,Off_dish_washeR))
On_heateR = Min(Low_temperaturE,Max(0.0 , Min(100.0 , (-100*
    Smoke_detectioN+Max(Remote_heatinG,Max(Heating_habitS,Heating_timE))
    *100.0) / 100.0)))
On_dish_washeR = Min(Electricity_pricE , FulL)
Maximize_energy_productioN = Max(0.0 , Min(100.0 , (100*Wind_turbines
    +100*Solar_paneL) / 100.0))
Manage_homE = Min(Min(Min(Min(Max(Smart_security_camerA,Max(
    Motion_sensorS,Cyber_attack_detectioN)),Smoke_detectoR),Min(Min(
    Wind_sensorS,Light_sensorS),Min(Infrared_sensoR,Max(
    Temperature_settinG , Temperature_remote_accesS)))),Max(Max(
    On_air_conditioneR , On_heateR) , Max(Max(Max(Open_windowS ,
    On_air_ventilatoR),Off_heateR),Max(Off_air_ventilatoR,Max(
    Off_air_conditioneR,Close_windowS))))),Min(Max(Wind_turbines,Max(
    Solar_paneL,Smart_griD)),Min(Max(Max(On_water_heateR,On_dish_washeR),
    Max(Off_water_heateR,Off_dish_washeR)),Max(Max(WifI,Max(
    Heater_schedulE,InterneT)),Max(Max(Learning_techniquE,
    Air_conditioner_remote_accesS),Max(Max(Water_heater_schedulE,
    Air_conditioner_schedulE),Max(Heater_remote_accesS,
    Water_heater_remote_accesS)))))))
Solar_paneL = Max(0.0 , Min(100.0 , (100*LighT) / 100.0))
Provided_serviceS = Max(Max(WifI,Max(Heater_schedulE,InterneT)),Max(Max(
    Learning_techniquE,Air_conditioner_remote_accesS),Max(Max(
    Water_heater_schedulE,Air_conditioner_schedulE),Max(
    Heater_remote_accesS,Water_heater_remote_accesS))))
InterneT = Min(No_suspicious_behaviour_detecteD , Response_timE)
Open_windowS = Max(0.0 , Min(100.0 , (100*FirE+Min(Suitable_weatheR,Min(
    No_suspicious_behaviour_detecteD,User_permissioN))*100.0) / 100.0))
On_air_conditioneR = Min(High_temperaturE,Max(0.0 , Min(100.0 , (-100*
    Smoke_detectioN+Max(Remote_coolinG,Max(Cooling_habitS,Cooling_timE))
    *100.0) / 100.0)))
On_water_heateR = Max(Remote_water_heatinG , Heating_water_timE)
On_air_ventilatoR = Max(0.0 , Min(100.0 , (100*Smoke_detectioN) / 100.0))
Maximize_privacY = Max(0.0 , Min(100.0 , (-25*InterneT+25*
    Water_heater_schedulE+25*Heater_schedulE+25*Air_conditioner_schedulE
    +25*On_air_ventilatoR+-25*Open_windowS) / 100.0))
Maximize_comforT = Max(0.0 , Min(100.0 , (50*InterneT+25*
    Air_conditioner_schedulE+25*Heater_schedulE+25*Water_heater_schedulE
    +-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*Temperature_settinG+75*Max
    (Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN))) /
    100.0))+25*Temperature_remote_accesS+75*Learning_techniquE) / 100.0))
Normalize_inside_temperaturE = Max(Max(On_air_conditioneR , On_heateR) ,
    Max(Max(Max(Open_windowS , On_air_ventilatoR),Off_heateR),Max(
    Off_air_ventilatoR,Max(Off_air_conditioneR,Close_windowS))))
```

```
Heating_timE = Min(Heater_schedulE,Piecewise((100 , Heating_timE>=100.0)
    , (abs( (Heating_timE - 99.99) / 0.010000000000005116) * 50 + 50 ,
    (99.99<=Heating_timE)&(Heating_timE<100.0)) , (-abs( (Heating_timE -
    99.99) / -99.99) * 50 + 50 , (0.0<Heating_timE)&(Heating_timE<99.99))
    , (0 , True)))
Heating_water_timE = Min(Water_heater_schedulE,Piecewise((100 ,
    Heating_water_timE>=100.0) , (abs( (Heating_water_timE - 99.99) /
    0.010000000000005116) * 50 + 50 , (99.99<=Heating_water_timE)&(
    Heating_water_timE<100.0)) , (-abs( (Heating_water_timE - 99.99) /
    -99.99) * 50 + 50 , (0.0<Heating_water_timE)&(Heating_water_timE
    <99.99)) , (0 , True)))
Cooling_timE = Min(Air_conditioner_schedulE,Piecewise((100 , Cooling_timE
    >=100.0) , (abs( (Cooling_timE - 99.99) / 0.010000000000005116) * 50 +
     50 , (99.99<=Cooling_timE)&(Cooling_timE<100.0)) , (-abs( (
    Cooling_timE - 99.99) / -99.99) * 50 + 50 , (0.0<Cooling_timE)&(
    Cooling_timE<99.99)) , (0 , True)))
Remote_coolinG = Min(Air_conditioner_remote_accesS,Piecewise((100 ,
    Remote_coolinG>=100.0) , (abs( (Remote_coolinG - 99.99) /
    0.010000000000005116) * 50 + 50 , (99.99<=Remote_coolinG)&(
    Remote_coolinG<100.0)) , (-abs( (Remote_coolinG - 99.99) / -99.99) *
    50 + 50 , (0.0<Remote_coolinG)&(Remote_coolinG<99.99)) , (0 , True)))
Remote_heatinG = Min(Heater_remote_accesS,Piecewise((100 , Remote_heatinG
    >=100.0) , (abs( (Remote_heatinG - 99.9) / 0.09999999999999432) * 50 +
     50 , (99.9<=Remote_heatinG)&(Remote_heatinG<100.0)) , (-abs( (
    Remote_heatinG - 99.9) / -99.9) * 50 + 50 , (0.0<Remote_heatinG)&(
    Remote_heatinG<99.9)) , (0 , True)))
Remote_water_heatinG = Min(Water_heater_remote_accesS,Piecewise((100 ,
    Remote_water_heatinG>=100.0) , (abs( (Remote_water_heatinG - 99.99) /
    0.010000000000005116) * 50 + 50 , (99.99<=Remote_water_heatinG)&(
    Remote_water_heatinG<100.0)) , (-abs( (Remote_water_heatinG - 99.99) /
     -99.99) * 50 + 50 , (0.0<Remote_water_heatinG)&(Remote_water_heatinG
    <99.99)) , (0 , True)))
FulL = Min(Infrared_sensoR,Piecewise((100 , FulL>=100.0) , (abs( (FulL -
    99.99) / 0.010000000000005116) * 50 + 50 , (99.99<=FulL)&(FulL<100.0))
     , (-abs( (FulL - 99.99) / -99.99) * 50 + 50 , (0.0<FulL)&(FulL<99.99)
    ) , (0 , True)))
Air_conditioner_schedulE = Max(0.0 , Min(100.0 , (100*
    Enable_air_conditioner_schedulE) / 100.0))
Water_heater_schedulE = Max(0.0 , Min(100.0 , (100*
    Enable_water_heater_schedulE) / 100.0))
Heater_schedulE = Max(0.0 , Min(100.0 , (100*Enable_heater_schedulE) /
    100.0))
High_temperaturE = Min(Max(Temperature_settinG ,
    Temperature_remote_accesS),Piecewise((100 , High_temperaturE<=10.0) ,
    (abs( (High_temperaturE - 10.99) / 0.9900000000000002) * 50 + 50 ,
    (10.0<High_temperaturE)&(High_temperaturE<=10.99)) , (-abs( (
    High_temperaturE - 10.99) / -0.009999999999999787) * 50 + 50 , (10.99<
    High_temperaturE)&(High_temperaturE<11.0)) , (0 , True)))
NormalizE = Max(On_air_conditioneR , On_heateR)
Smoke_detectioN = Min(Smoke_detectoR,Piecewise((100 , Smoke_detectioN
    >=100.0) , (abs( (Smoke_detectioN - 99.99) / 0.010000000000005116) *
    50 + 50 , (99.99<=Smoke_detectioN)&(Smoke_detectioN<100.0)) , (-abs( (
    Smoke_detectioN - 99.99) / -99.99) * 50 + 50 , (0.0<Smoke_detectioN)&(
    Smoke_detectioN<99.99)) , (0 , True)))
```

```
Operate_related_taskS = Max(Max(Max(Open_windowS , On_air_ventilatoR),
    Off_heateR),Max(Off_air_ventilatoR,Max(Off_air_conditioneR,
    Close_windowS)))
Refresh_inside_aiR = Max(Open_windowS , On_air_ventilatoR)
Low_temperaturE = Min(Max(Temperature_settinG , Temperature_remote_accesS
    ),Piecewise((100 , Low_temperaturE>=27.0) , (abs( (Low_temperaturE -
    26.99) / 0.010000000000001563) * 50 + 50 , (26.99<=Low_temperaturE)&(
    Low_temperaturE<27.0)) , (-abs( (Low_temperaturE - 26.99) /
    -0.00999999999999801) * 50 + 50 , (26.98<Low_temperaturE)&(
    Low_temperaturE<26.99)) , (0 , True)))
Air_conditioner_remote_accesS = Max(0.0 , Min(100.0 , (100*
    Enable_air_conditioner_accesS) / 100.0))
Water_heater_remote_accesS = Max(0.0 , Min(100.0 , (100*
    Enable_water_heater_accesS) / 100.0))
Heater_remote_accesS = Max(0.0 , Min(100.0 , (100*Enable_heater_accesS) /
     100.0))
No_suspicious_behaviour_detecteD = Piecewise((100 ,
    No_suspicious_behaviour_detecteD) , (0 , True))
Monitor_the_environmenT = Min(Min(Max(Smart_security_camerA,Max(
    Motion_sensorS,Cyber_attack_detectioN)),Smoke_detectoR),Min(Min(
    Wind_sensorS,Light_sensorS),Min(Infrared_sensoR,Max(
    Temperature_settinG , Temperature_remote_accesS))))
Set_thermostatE = Max(Temperature_settinG , Temperature_remote_accesS)
Maximize_securitY = Max(0.0 , Min(100.0 , (-25*InterneT+25*
    Temperature_settinG+75*Max(Smart_security_camerA,Max(Motion_sensorS,
    Cyber_attack_detectioN))) / 100.0))
Temperature_remote_accesS = Max(0.0 , Min(100.0 , (100*
    Enable_temperature_accesS) / 100.0))
Detect_suspicious_behaviouR = Max(Smart_security_camerA,Max(
    Motion_sensorS,Cyber_attack_detectioN))
Cooling_habitS = Min(Learning_techniquE,Piecewise((100 , Cooling_habitS
    <=0.0) , (abs( (Cooling_habitS - 10.0) / 10.0) * 50 + 50 , (0.0<
    Cooling_habitS)&(Cooling_habitS<=10.0)) , (-abs( (Cooling_habitS -
    10.0) / -10.0) * 50 + 50 , (10.0<Cooling_habitS)&(Cooling_habitS<20.0)
    ) , (0 , True)))
Heating_habitS = Min(Learning_techniquE,Piecewise((100 , Heating_habitS
    <=0.0) , (abs( (Heating_habitS - 10.0) / 10.0) * 50 + 50 , (0.0<
    Heating_habitS)&(Heating_habitS<=10.0)) , (-abs( (Heating_habitS -
    10.0) / -10.0) * 50 + 50 , (10.0<Heating_habitS)&(Heating_habitS<20.0)
    ) , (0 , True)))
# Actor function
InhabitantS = (Max(0.0 , Min(100.0 , (100*Wind_turbines+100*Solar_paneL)
    / 100.0))*15+Max(0.0 , Min(100.0 , (25*Open_windowS+50*InterneT+25*
    Water_heater_schedulE+25*Heater_schedulE+25*Air_conditioner_schedulE
    +75*Learning_techniquE+-25*On_air_ventilatoR) / 100.0))*15+Max(0.0 ,
    Min(100.0 , (-25*InterneT+25*Water_heater_schedulE+25*Heater_schedulE
    +25*Air_conditioner_schedulE+25*On_air_ventilatoR+-25*Open_windowS) /
    100.0))*25+Max(0.0 , Min(100.0 , (50*InterneT+25*
    Air_conditioner_schedulE+25*Heater_schedulE+25*Water_heater_schedulE
    +-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*Temperature_settinG+75*Max
    (Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN))) /
    100.0))+25*Temperature_remote_accesS+75*Learning_techniquE) / 100.0))
    *15+Max(0.0 , Min(100.0 , (-25*InterneT+25*Water_heater_schedulE+25*
    Heater_schedulE+25*Air_conditioner_schedulE+25*On_air_ventilatoR+-25*
```

```
    Open_windowS) / 100.0))*25+Max(0.0 , Min(100.0 , (50*InterneT+25*
    Air_conditioner_schedulE+25*Heater_schedulE+25*Water_heater_schedulE
    +-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*Temperature_settinG+75*Max
    (Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN))) /
    100.0))+25*Temperature_remote_accesS+75*Learning_techniquE) / 100.0))
    *15+Max(0.0 , Min(100.0 , (25*Open_windowS+50*InterneT+25*
    Water_heater_schedulE+25*Heater_schedulE+25*Air_conditioner_schedulE
    +75*Learning_techniquE+-25*On_air_ventilatoR) / 100.0))*15+Max(0.0 ,
    Min(100.0 , (-25*InterneT+25*Temperature_settinG+75*Max(
    Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN))) /
    100.0))*30+Max(0.0 , Min(100.0 , (-25*InterneT+25*
    Water_heater_schedulE+25*Heater_schedulE+25*Air_conditioner_schedulE
    +25*On_air_ventilatoR+-25*Open_windowS) / 100.0))*25+Max(0.0 , Min
    (100.0 , (50*InterneT+25*Air_conditioner_schedulE+25*Heater_schedulE
    +25*Water_heater_schedulE+-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*
    Temperature_settinG+75*Max(Smart_security_camerA,Max(Motion_sensorS,
    Cyber_attack_detectioN))) / 100.0))+25*Temperature_remote_accesS+75*
    Learning_techniquE) / 100.0))*15+Max(0.0 , Min(100.0 , (25*
    Open_windowS+50*InterneT+25*Water_heater_schedulE+25*Heater_schedulE
    +25*Air_conditioner_schedulE+75*Learning_techniquE+-25*
    On_air_ventilatoR) / 100.0))*15+Max(0.0 , Min(100.0 , (-25*InterneT
    +25*Temperature_settinG+75*Max(Smart_security_camerA,Max(
    Motion_sensorS,Cyber_attack_detectioN))) / 100.0))*30) / 240
# Actor function
SmarthomesysteM = (Min(Min(Min(Min(Max(Smart_security_camerA,Max(
    Motion_sensorS,Cyber_attack_detectioN)),Smoke_detectoR),Min(Min(
    Wind_sensorS,Light_sensorS),Min(Infrared_sensoR,Max(
    Temperature_settinG , Temperature_remote_accesS)))),Max(Max(
    On_air_conditioneR , On_heateR) , Max(Max(Max(Open_windowS ,
    On_air_ventilatoR),Off_heateR),Max(Off_air_ventilatoR,Max(
    Off_air_conditioneR,Close_windowS))))),Min(Max(Wind_turbines,Max(
    Solar_paneL,Smart_griD)),Min(Max(Max(On_water_heateR,On_dish_washeR),
    Max(Off_water_heateR,Off_dish_washeR)),Max(Max(WifI,Max(
    Heater_schedulE,InterneT)),Max(Max(Learning_techniquE,
    Air_conditioner_remote_accesS),Max(Max(Water_heater_schedulE,
    Air_conditioner_schedulE),Max(Heater_remote_accesS,
    Water_heater_remote_accesS)))))))*100) / 100
# The function of Model
UrnspeC = ((Max(0.0 , Min(100.0 , (100*Wind_turbines+100*Solar_paneL) /
    100.0))*15+Max(0.0 , Min(100.0 , (25*Open_windowS+50*InterneT+25*
    Water_heater_schedulE+25* Heater_schedulE+25*Air_conditioner_schedulE
    +75*Learning_techniquE+-25*On_air_ventilatoR) / 100.0))*15+ Max(0.0 ,
    Min(100.0 , (-25*InterneT+25*Water_heater_schedulE+ 25*Heater_schedulE
    + 25*Air_conditioner_schedulE+25*On_air_ventilatoR+-25*Open_windowS) /
     100.0))*25+
Max(0.0 , Min(100.0 , (50*InterneT+25*Air_conditioner_schedulE+25*
    Heater_schedulE+25*Water_heater_schedulE+-25*Max(0.0 , Min(100.0 ,
    (-25*InterneT+25*Temperature_settinG+ 75*Max(Smart_security_camerA,Max
    (Motion_sensorS,Cyber_attack_detectioN))) / 100.0))+25*
    Temperature_remote_accesS+ 75*Learning_techniquE) / 100.0))*15+Max(0.0
     , Min(100.0 , (-25*InterneT+25*Water_heater_schedulE+ 25*
    Heater_schedulE+25*Air_conditioner_schedulE+25*On_air_ventilatoR+-25*
    Open_windowS) / 100.0))*25+Max(0.0 , Min(100.0 , (50*InterneT+25*
    Air_conditioner_schedulE+25*Heater_schedulE+25*Water_heater_schedulE
```

```
    +-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*Temperature_settinG+ 75*
    Max(Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN)))
     / 100.0))+25*Temperature_remote_accesS+ 75*Learning_techniquE) /
    100.0))*15+Max(0.0 , Min(100.0 , (25*Open_windowS+50*InterneT+25*
    Water_heater_schedulE+25*Heater_schedulE+25*Air_conditioner_schedulE
    +75*Learning_techniquE+-25*On_air_ventilatoR) / 100.0))*15+Max(0.0 ,
    Min(100.0 , (-25*InterneT+25*Temperature_settinG+75* Max(
    Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN))) /
    100.0))*30+Max(0.0 , Min(100.0 , (-25*InterneT+25*
    Water_heater_schedulE+25*Heater_schedulE+25*Air_conditioner_schedulE
    +25*On_air_ventilatoR+-25*Open_windowS) / 100.0))*25+Max(0.0 , Min
    (100.0 , (50*InterneT+25*Air_conditioner_schedulE+25*Heater_schedulE
    +25*Water_heater_schedulE+
-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*Temperature_settinG+75*Max(
    Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN))) /
    100.0))+25*Temperature_remote_accesS+75*Learning_techniquE) / 100.0))
    *15+Max(0.0 , Min(100.0 , (25*Open_windowS+50*InterneT+25*
    Water_heater_schedulE+25*Heater_schedulE+25*Air_conditioner_schedulE
    +75*Learning_techniquE+-25*On_air_ventilatoR) / 100.0))*15+Max(0.0 ,
    Min(100.0 , (-25*InterneT+25*Temperature_settinG+75* Max(
    Smart_security_camerA,Max(Motion_sensorS, Cyber_attack_detectioN))) /
    100.0))*30) / 240*50+(Min(Min(Min(Min(Max(Smart_security_camerA, Max(
    Motion_sensorS,Cyber_attack_detectioN)),Smoke_detectoR), Min(Min(
    Wind_sensorS,Light_sensorS),Min(Infrared_sensoR,Max(
    Temperature_settinG , Temperature_remote_accesS)))),Max(Max(
    On_air_conditioneR , On_heateR) , Max(Max(Max(Open_windowS ,
    On_air_ventilatoR) ,Off_heateR), Max(Off_air_ventilatoR, Max(
    Off_air_conditioneR, Close_windowS))))),
Min(Max(Wind_turbines,Max(Solar_paneL,Smart_griD)), Min(Max(Max(
    On_water_heateR,On_dish_washeR),Max(Off_water_heateR, Off_dish_washeR)
    ), Max(Max(WifI,Max(Heater_schedulE,InterneT)), Max(Max(
    Learning_techniquE, Air_conditioner_remote_accesS), Max(Max(
    Water_heater_schedulE, Air_conditioner_schedulE), Max(
    Heater_remote_accesS, Water_heater_remote_accesS )))))) *100) /
    100*50) / 100

GRLDiagramName = 'Manage_homE'

# Variable list
List = ['LighT','Motion_sensorS','Off_dish_washeR','Heating_water_timE','
    Learning_techniquE','Cooling_timE','Enable_heater_schedulE','
    Off_air_ventilatoR','Temperature_settinG','Low_temperaturE','
    Cooling_habitS','Response_timE','Enable_water_heater_schedulE','
    User_permissioN','Suitable_weatheR','Enable_air_conditioner_schedulE',
    'On_heateR','Heating_habitS','Smart_griD','Wind_speeD','Wind_sensorS',
    'Remote_water_heatinG','Electricity_pricE','On_air_conditioneR','WifI'
    ,'Remote_coolinG','Off_air_conditioneR','High_temperaturE','
    Enable_air_conditioner_accesS','Heating_timE','Off_water_heateR','
    Enable_heater_accesS','No_suspicious_behaviour_detecteD','
    Enable_temperature_accesS','FulL','Cyber_attack_detectioN','FirE','
    Close_windowS','Smoke_detectoR','Light_sensorS','Smart_security_camerA
    ','Smoke_detectioN','Infrared_sensoR','Off_heateR','Remote_heatinG','
    Enable_water_heater_accesS']
LANG = []
```

```
langList = ['python','c','c++','java',"javascript",'matlab','r']
def allPrint():
    List = ['Wind_turbines','Off_dish_washeR','
        Air_conditioner_remote_accesS','Smart_security_camerA','
        Motion_sensorS','Cyber_attack_detectioN','Temperature_settinG','
        Solar_paneL','Open_windowS','Water_heater_schedulE','
        Heater_schedulE','Air_conditioner_schedulE','Learning_techniquE','
        Temperature_remote_accesS','On_air_ventilatoR','On_heateR','
        InterneT','Water_heater_schedulE','Heater_schedulE','
        Air_conditioner_schedulE','Heater_remote_accesS','
        Water_heater_remote_accesS','Smart_griD','On_water_heateR','
        On_dish_washeR','Off_water_heateR','Off_dish_washeR','
        Smoke_detectoR','Wind_sensorS','Light_sensorS','Infrared_sensoR','
        Open_windowS' , 'On_air_ventilatoR','Off_heateR','
        Off_air_ventilatoR','Off_air_conditioneR','Close_windowS','
        On_air_conditioneR','WifI']
    Translate('((Max(0.0 , Min(100.0 , (100*Wind_turbines+100*Solar_paneL
        ) / 100.0))*15+Max(0.0 , Min(100.0 , (25*Open_windowS+50*InterneT
        +25*Water_heater_schedulE+25*Heater_schedulE+25*
        Air_conditioner_schedulE+75*Learning_techniquE+-25*
        On_air_ventilatoR) / 100.0))*15+Max(0.0 , Min(100.0 , (-25*
        InterneT+25*Water_heater_schedulE+25*Heater_schedulE+25*
        Air_conditioner_schedulE+25*On_air_ventilatoR+-25*Open_windowS) /
        100.0))*25+Max(0.0 , Min(100.0 , (50*InterneT+25*
        Air_conditioner_schedulE+25*Heater_schedulE+25*
        Water_heater_schedulE+-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*
        Temperature_settinG+75*Max(Smart_security_camerA,Max(
        Motion_sensorS,Cyber_attack_detectioN))) / 100.0))+25*
        Temperature_remote_accesS+75*Learning_techniquE) / 100.0))*15+Max
        (0.0 , Min(100.0 , (-25*InterneT+25*Water_heater_schedulE+25*
        Heater_schedulE+25*Air_conditioner_schedulE+25*On_air_ventilatoR
        +-25*Open_windowS) / 100.0))*25+Max(0.0 , Min(100.0 , (50*InterneT
        +25*Air_conditioner_schedulE+25*Heater_schedulE+25*
        Water_heater_schedulE+-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*
        Temperature_settinG+75*Max(Smart_security_camerA,Max(
        Motion_sensorS,Cyber_attack_detectioN))) / 100.0))+25*
        Temperature_remote_accesS+75*Learning_techniquE) / 100.0))*15+Max
        (0.0 , Min(100.0 , (25*Open_windowS+50*InterneT+25*
        Water_heater_schedulE+25*Heater_schedulE+25*
        Air_conditioner_schedulE+75*Learning_techniquE+-25*
        On_air_ventilatoR) / 100.0))*15+
    Max(0.0 , Min(100.0 , (-25*InterneT+25*Temperature_settinG+75*Max(
        Smart_security_camerA,Max(Motion_sensorS,Cyber_attack_detectioN)))
         / 100.0))*30+Max(0.0 , Min(100.0 , (-25*InterneT+25*
        Water_heater_schedulE+25*Heater_schedulE+25*
        Air_conditioner_schedulE+25*On_air_ventilatoR+-25*Open_windowS) /
        100.0))*25+Max(0.0 , Min(100.0 , (50*InterneT+25*
        Air_conditioner_schedulE+25*Heater_schedulE+25*
        Water_heater_schedulE+-25*Max(0.0 , Min(100.0 , (-25*InterneT+25*
        Temperature_settinG+75*Max(Smart_security_camerA,Max(
        Motion_sensorS,Cyber_attack_detectioN))) / 100.0))+25*
        Temperature_remote_accesS+75*Learning_techniquE) / 100.0))*15+Max
        (0.0 , Min(100.0 , (25*Open_windowS+50*InterneT+25*
        Water_heater_schedulE+25*Heater_schedulE+25*
```

```
    Air_conditioner_schedulE+75*Learning_techniquE+-25*
    On_air_ventilatoR) / 100.0))*15+Max(0.0 , Min(100.0 , (-25*
    InterneT+25*Temperature_settinG+75*Max(Smart_security_camerA,Max(
    Motion_sensorS,Cyber_attack_detectioN))) / 100.0))*30) / 240*50+(
    Min(Min(Min(Min(Max(Smart_security_camerA,Max(Motion_sensorS,
    Cyber_attack_detectioN)),Smoke_detectoR),Min(Min(Wind_sensorS,
    Light_sensorS),Min(Infrared_sensoR,Max(Temperature_settinG ,
    Temperature_remote_accesS)))),Max(Max(On_air_conditioneR ,
    On_heateR) , Max(Max(Max(Open_windowS , On_air_ventilatoR),
    Off_heateR),Max(Off_air_ventilatoR,Max(Off_air_conditioneR,
    Close_windowS))))),Min(Max(Wind_turbines,Max(Solar_paneL,
    Smart_griD)),Min(Max(Max(On_water_heateR,On_dish_washeR),Max(
    Off_water_heateR,Off_dish_washeR)),Max(Max(WifI,Max(
    Heater_schedulE,InterneT)),Max(Max(Learning_techniquE,
    Air_conditioner_remote_accesS),Max(Max(Water_heater_schedulE,
    Air_conditioner_schedulE),Max(Heater_remote_accesS,
    Water_heater_remote_accesS)))))))*100) / 100*50) / 100' ,
    GRLDiagramName , List , LANG,2)
# Indicators
List=['Wind_speeD']
print 'Wind_speeD'
Translate('Piecewise((100 , (Wind_speeD>=10) & (Wind_speeD<=24)),(0 ,
     True))', List[0], List, LANG,2)
List=['Electricity_pricE']
print 'Electricity_pricE'
Translate('Piecewise((100 , Electricity_pricE<=6.4) , (0 , True))',
    List[0], List, LANG,2)
List=['LighT']
print 'LighT'
Translate('Piecewise((100 , LighT),(0 , True))', List[0], List, LANG
    ,2)
List=['Response_timE']
print 'Response_timE'
Translate('Piecewise((100 , Response_timE<=2),(0 , True))', List[0],
    List, LANG,2)
List=['Suitable_weatheR']
print 'Suitable_weatheR'
Translate('Piecewise((100 ,Suitable_weatheR),(0 , True))', List[0],
    List, LANG,2)
List=['Heating_timE','Enable_heater_schedulE']
print 'Heating_timE'
Translate('Min(Max(0.0 , Min(100.0 , (100*Enable_heater_schedulE) /
    100.0)),Piecewise((100 , Heating_timE),(0 , True)))', List[0],
    List, LANG,2)
List=['Heating_water_timE','Enable_water_heater_schedulE']
print 'Heating_water_timE'
Translate('Min(Max(0.0 , Min(100.0 , (100*
    Enable_water_heater_schedulE) / 100.0)),Piecewise((100 ,
    Heating_water_timE),(0 , True)))', List[0], List, LANG,2)
List=['Cooling_timE','Enable_air_conditioner_schedulE']
print 'Cooling_timE'
Translate('Min(Max(0.0 , Min(100.0 , (100*
    Enable_air_conditioner_schedulE) / 100.0)),Piecewise((100 ,
    Cooling_timE),(0 , True)))', List[0], List, LANG,2)
```

```
List=['Remote_coolinG','Enable_air_conditioner_accesS']
print 'Remote_coolinG'
Translate('Min(Max(0.0 , Min(100.0 , (100*
    Enable_air_conditioner_accesS) / 100.0)),Piecewise((100 ,
    Remote_coolinG),(0 , True)))', List[0], List, LANG,2)
List=['Remote_heatinG','Enable_heater_accesS']
print 'Remote_heatinG'
Translate('Min(Max(0.0 , Min(100.0 , (100*Enable_heater_accesS) /
    100.0)),Piecewise((100 , Remote_heatinG),(0 , True)))', List[0],
    List, LANG,2)
List=['Remote_water_heatinG','Enable_water_heater_accesS']
print 'Remote_water_heatinG'
Translate('Min(Max(0.0 , Min(100.0 , (100*Enable_water_heater_accesS)
     / 100.0)),Piecewise((100 , Remote_water_heatinG),(0 , True)))',
    List[0], List, LANG,2)
List=['FulL','Infrared_sensoR']
print 'FulL'
Translate('Min(Infrared_sensoR,Piecewise((100 , FulL),(0 , True)))',
    List[0], List, LANG,2)
List=['High_temperaturE','Temperature_settinG','Response_timE','
    No_suspicious_behaviour_detecteD','Enable_temperature_accesS']
print 'High_temperaturE'
#>=27
Translate('Min(Max(Temperature_settinG , Max(0.0 , Min(100.0 , (100*
    Enable_temperature_accesS) / 100.0))),100*Min(Max( floor(
    High_temperaturE/27),0),1))', List[0], List, LANG,2)
List=['Smoke_detectioN','Smoke_detectoR']
print 'Smoke_detectioN'
Translate('Min(Smoke_detectoR,Piecewise((100 , Smoke_detectioN),(0 ,
    True)))', List[0], List, LANG,2)
List=['Low_temperaturE','Temperature_settinG','Response_timE','
    No_suspicious_behaviour_detecteD','Enable_temperature_accesS']
print 'Low_temperaturE'
Translate('Min(Max(Temperature_settinG , Max(0.0 , Min(100.0 , (100*
    Enable_temperature_accesS) / 100.0))),100*Min(Max( floor(15/
    Low_temperaturE),0),1))', List[0], List, LANG,2)
List=['User_permissioN']
print 'User_permissioN'
Translate('Piecewise((100 , User_permissioN),(0 , True))', List[0],
    List, LANG,2)
List=['No_suspicious_behaviour_detecteD']
print 'No_suspicious_behaviour_detecteD'
Translate('Piecewise((100 , No_suspicious_behaviour_detecteD),(0 ,
    True))', List[0], List, LANG,2)
List=['Cooling_habitS','Learning_techniquE']
print 'Cooling_habitS'
Translate('Min(Learning_techniquE,Piecewise((100 , Cooling_habitS),(0
     , True)))', List[0], List, LANG,2)
  List=['Heating_habitS','Learning_techniquE']
print 'Heating_habitS'
Translate('Min(Learning_techniquE,Piecewise((100 ,Heating_habitS),(0
    , True)))', List[0], List, LANG,2)
List=['FirE']
print 'FirE'
```

```
    Translate('Piecewise((100 , FirE),(0 , True))', List[0], List, LANG
        ,2)
    List=['Enable_water_heater_accesS']
    print 'Enable_water_heater_accesS'
    Translate('Piecewise((100 , Enable_water_heater_accesS),(0 , True))',
         List[0], List, LANG,2)
    List=['Enable_air_conditioner_accesS']
    print 'Enable_air_conditioner_accesS'
    Translate('Piecewise((100 , Enable_air_conditioner_accesS),(0 , True)
        )', List[0], List, LANG,2)
    List=['Enable_heater_accesS']
    print 'Enable_heater_accesS'
    Translate('Piecewise((100 , Enable_heater_accesS),(0 , True))', List
        [0], List, LANG,2)
    List=['Enable_heater_schedulE']
    print 'Enable_heater_schedulE'
    Translate('Piecewise((100 , Enable_heater_schedulE),(0 , True))',
        List[0], List, LANG,2)
    List=['Enable_air_conditioner_schedulE']
    print 'Enable_air_conditioner_schedulE'
    Translate('Piecewise((100 , Enable_air_conditioner_schedulE),(0 ,
        True))', List[0], List, LANG,2)
    List=['Enable_water_heater_schedulE']
    print 'Enable_water_heater_schedulE'
    Translate('Piecewise((100 , Enable_water_heater_schedulE),(0 , True))
        ', List[0], List, LANG,2)
    List=['Enable_temperature_accesS']
    print 'Enable_temperature_accesS'
    Translate('Piecewise((100 , Enable_temperature_accesS),(0 , True))',
        List[0], List, LANG,2)
if(len(sys.argv)==1):
  LANG = langList
  allPrint()
else:
  for i in sys.argv:
    if(sys.argv.index(i)==0):continue
    if  (i.lower() not in langList):
      LANG = langList
      break
    else:
      LANG.append(str(i.lower()))
  allPrint()

#inital all the variable
Wind_turbineS= Symbol('Wind_turbineS')
Off_heateR= Symbol('Off_heateR')
On_heateR= Symbol('On_heateR')
Off_dish_washeR= Symbol('Off_dish_washeR')
On_dish_washeR= Symbol('On_dish_washeR')
Dish_washer_controlleR= Symbol('Dish_washer_controlleR')
Air_conditioner_controlleR= Symbol('Air_conditioner_controlleR')
Power_controlleR= Symbol('Power_controlleR')
SmarthomE= Symbol('SmarthomE')
Smart_griD= Symbol('Smart_griD')
```

```
Heater_controlleR= Symbol('Heater_controlleR')
Solar_paneL= Symbol('Solar_paneL')
WifI= Symbol('WifI')
InterneT= Symbol('InterneT')
Open_windowS= Symbol('Open_windowS')
On_air_conditioneR= Symbol('On_air_conditioneR')
Off_air_conditioneR= Symbol('Off_air_conditioneR')
Close_windowS= Symbol('Close_windowS')
Water_heater_controlleR= Symbol('Water_heater_controlleR')
On_water_heateR= Symbol('On_water_heateR')
Off_water_heateR= Symbol('Off_water_heateR')
Air_ventilator_controlleR= Symbol('Air_ventilator_controlleR')
On_air_ventilatoR= Symbol('On_air_ventilatoR')
Off_air_ventilatoR= Symbol('Off_air_ventilatoR')
Smart_appliances_controlleR= Symbol('Smart_appliances_controlleR')
Windows_controlleR= Symbol('Windows_controlleR')
Air_conditioner_schedulE= Symbol('Air_conditioner_schedulE')
Water_heater_schedulE= Symbol('Water_heater_schedulE')
Heater_schedulE= Symbol('Heater_schedulE')
Heater_switcH= Symbol('Heater_switcH')
Water_heater_switcH= Symbol('Water_heater_switcH')
Air_conditioner_switcH= Symbol('Air_conditioner_switcH')
Air_conditioner_remote_accesS= Symbol('Air_conditioner_remote_accesS')
Water_heater_remote_accesS= Symbol('Water_heater_remote_accesS')
Heater_remote_accesS= Symbol('Heater_remote_accesS')
SensorS= Symbol('SensorS')
Motion_sensorS= Symbol('Motion_sensorS')
Light_sensorS= Symbol('Light_sensorS')
Smoke_detectoR= Symbol('Smoke_detectoR')
Smart_security_camerA= Symbol('Smart_security_camerA')
Wind_sensorS= Symbol('Wind_sensorS')
Smart_thermostaT= Symbol('Smart_thermostaT')
Infrared_sensoR= Symbol('Infrared_sensoR')
Cyber_attack_detectioN= Symbol('Cyber_attack_detectioN')
Temperature_settinG= Symbol('Temperature_settinG')
Temperature_remote_accesS= Symbol('Temperature_remote_accesS')
Security_sensoR= Symbol('Security_sensoR')
Internet_connectioN= Symbol('Internet_connectioN')
Learning_techniquE= Symbol('Learning_techniquE')
On_heateR= Min(Min(0,On_heateR - On_air_conditioneR),(((On_heateR + Min(
    Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 ))
On_air_conditioneR= Min(Min(0,On_air_conditioneR - On_heateR),(((
    On_air_conditioneR + Min(Close_windowS,Off_air_ventilatoR)) / 200.0 )*
     100.0 ))
Air_conditioner_remote_accesS= (((Air_conditioner_remote_accesS +
    InterneT) / 200.0 )* 100.0 )
Water_heater_remote_accesS= (((Water_heater_remote_accesS + InterneT) /
    200.0 )* 100.0 )
Heater_remote_accesS= (((Heater_remote_accesS + InterneT) / 200.0 )*
    100.0 )
Temperature_remote_accesS= (((Temperature_remote_accesS + InterneT) /
    200.0 )* 100.0 )
Dish_washer_controlleR= (Max(Off_dish_washeR,On_dish_washeR)) / Max(1,
    Off_dish_washeR + On_dish_washeR)*100.0
```

```
Air_conditioner_controlleR= Min((Max(Off_air_conditioneR,Min(Min(0,
    On_air_conditioneR - On_heateR),(((On_air_conditioneR + Min(
    Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 )))) / Max(1,
    Off_air_conditioneR + On_air_conditioneR)*100.0,Piecewise((100,Max(
    Air_conditioner_remote_accesS,Air_conditioner_schedulE)<=0),(((((
    Air_conditioner_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Air_conditioner_schedulE) / Max(1 ,Air_conditioner_remote_accesS +
    Air_conditioner_schedulE)*100.0,True)))
Power_controlleR= Min(100,(Smart_griD + Wind_turbineS + Solar_paneL) /
    Max(1 ,Smart_griD + Wind_turbineS + Solar_paneL)*100.0)
SmarthomE= Min(Min(Min(WifI,Min(100,(Smart_griD + Wind_turbineS +
    Solar_paneL) / Max(1 ,Smart_griD + Wind_turbineS + Solar_paneL)*100.0)
    ),Min(Min(Min(Wind_sensorS,Smoke_detectoR),Min(Light_sensorS,Min(Min(
    Temperature_settinG,Piecewise((100,Temperature_remote_accesS<=0),((((
    Temperature_remote_accesS + InterneT) / 200.0 )* 100.0 ) / Max(1 ,
    Temperature_remote_accesS)*100.0,True))),Min(100,(
    Smart_security_camerA + Motion_sensorS) / Max(1 ,Smart_security_camerA
     + Motion_sensorS)*100.0)))),Min(Min((Max(Close_windowS,Open_windowS))
     / Max(1,Close_windowS + Open_windowS)*100.0,(Max(Off_dish_washeR,
    On_dish_washeR)) / Max(1,Off_dish_washeR + On_dish_washeR)*100.0),Min(
    Min((Max(On_air_ventilatoR,Off_air_ventilatoR)) / Max(1,
    On_air_ventilatoR + Off_air_ventilatoR)*100.0,Min((Max(On_water_heateR
    ,Off_water_heateR)) / Max(1,On_water_heateR + Off_water_heateR)*100.0,
    Piecewise((100,Max(Water_heater_remote_accesS,Water_heater_schedulE)
    <=0),(((((Water_heater_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Water_heater_schedulE) / Max(1 ,Water_heater_remote_accesS +
    Water_heater_schedulE)*100.0,True)))),Min(Min((Max(Min(Min(0,On_heateR
     - On_air_conditioneR),(((On_heateR + Min(Close_windowS,
    Off_air_ventilatoR)) / 200.0 )* 100.0 )),Off_heateR)) / Max(1,
    On_heateR + Off_heateR)*100.0,Piecewise((100,Max(Heater_remote_accesS,
    Heater_schedulE)<=0),(((((Heater_remote_accesS + InterneT) / 200.0 )*
    100.0 ) + Heater_schedulE) / Max(1 ,Heater_remote_accesS +
    Heater_schedulE)*100.0,True))),Min((Max(Off_air_conditioneR,Min(Min(0,
    On_air_conditioneR - On_heateR),(((On_air_conditioneR + Min(
    Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 )))) / Max(1,
    Off_air_conditioneR + On_air_conditioneR)*100.0,Piecewise((100,Max(
    Air_conditioner_remote_accesS,Air_conditioner_schedulE)<=0),(((((
    Air_conditioner_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Air_conditioner_schedulE) / Max(1 ,Air_conditioner_remote_accesS +
    Air_conditioner_schedulE)*100.0,True)))))))),Piecewise((100,Max(
    Cyber_attack_detectioN,Max(Learning_techniquE,InterneT))<=0),((Min(
    Cyber_attack_detectioN,InterneT) + Learning_techniquE) / Max(1 ,Max(
    Cyber_attack_detectioN,InterneT) + Learning_techniquE)*100.0,True)))
Heater_controlleR= Min((Max(Min(Min(0,On_heateR - On_air_conditioneR),(((
    On_heateR + Min(Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 ))
    ,Off_heateR)) / Max(1,On_heateR + Off_heateR)*100.0,Piecewise((100,Max
    (Heater_remote_accesS,Heater_schedulE)<=0),(((((Heater_remote_accesS +
     InterneT) / 200.0 )* 100.0 ) + Heater_schedulE) / Max(1 ,
    Heater_remote_accesS + Heater_schedulE)*100.0,True)))
Water_heater_controlleR= Min((Max(On_water_heateR,Off_water_heateR)) /
    Max(1,On_water_heateR + Off_water_heateR)*100.0,Piecewise((100,Max(
    Water_heater_remote_accesS,Water_heater_schedulE)<=0),(((((
    Water_heater_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Water_heater_schedulE) / Max(1 ,Water_heater_remote_accesS +
```

```
    Water_heater_schedulE)*100.0,True)))
Air_ventilator_controlleR= (Max(On_air_ventilatoR,Off_air_ventilatoR)) /
    Max(1,On_air_ventilatoR + Off_air_ventilatoR)*100.0
Smart_appliances_controlleR= Min(Min((Max(Close_windowS,Open_windowS)) /
    Max(1,Close_windowS + Open_windowS)*100.0,(Max(Off_dish_washeR,
    On_dish_washeR)) / Max(1,Off_dish_washeR + On_dish_washeR)*100.0),Min(
    Min((Max(On_air_ventilatoR,Off_air_ventilatoR)) / Max(1,
    On_air_ventilatoR + Off_air_ventilatoR)*100.0,Min((Max(On_water_heateR
    ,Off_water_heateR)) / Max(1,On_water_heateR + Off_water_heateR)*100.0,
    Piecewise((100,Max(Water_heater_remote_accesS,Water_heater_schedulE)
    <=0),(((((Water_heater_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Water_heater_schedulE) / Max(1 ,Water_heater_remote_accesS +
    Water_heater_schedulE)*100.0,True)))),Min(Min((Max(Min(Min(0,On_heateR
     - On_air_conditioneR),(((On_heateR + Min(Close_windowS,
    Off_air_ventilatoR)) / 200.0 )* 100.0 )),Off_heateR)) / Max(1,
    On_heateR + Off_heateR)*100.0,Piecewise((100,Max(Heater_remote_accesS,
    Heater_schedulE)<=0),(((((Heater_remote_accesS + InterneT) / 200.0 )*
    100.0 ) + Heater_schedulE) / Max(1 ,Heater_remote_accesS +
    Heater_schedulE)*100.0,True))),Min((Max(Off_air_conditioneR,Min(Min(0,
    On_air_conditioneR - On_heateR),(((On_air_conditioneR + Min(
    Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 )))) / Max(1,
    Off_air_conditioneR + On_air_conditioneR)*100.0,Piecewise((100,Max(
    Air_conditioner_remote_accesS,Air_conditioner_schedulE)<=0),(((((
    Air_conditioner_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Air_conditioner_schedulE) / Max(1 ,Air_conditioner_remote_accesS +
    Air_conditioner_schedulE)*100.0,True))))))
Windows_controlleR= (Max(Close_windowS,Open_windowS)) / Max(1,
    Close_windowS + Open_windowS)*100.0
Heater_switcH= (Max(Min(Min(0,On_heateR - On_air_conditioneR),(((
    On_heateR + Min(Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 ))
    ,Off_heateR)) / Max(1,On_heateR + Off_heateR)*100.0
Water_heater_switcH= (Max(On_water_heateR,Off_water_heateR)) / Max(1,
    On_water_heateR + Off_water_heateR)*100.0
Air_conditioner_switcH= (Max(Off_air_conditioneR,Min(Min(0,
    On_air_conditioneR - On_heateR),(((On_air_conditioneR + Min(
    Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 )))) / Max(1,
    Off_air_conditioneR + On_air_conditioneR)*100.0
SensorS= Min(Min(Wind_sensorS,Smoke_detectoR),Min(Light_sensorS,Min(Min(
    Temperature_settinG,Piecewise((100,Temperature_remote_accesS<=0),((((
    Temperature_remote_accesS + InterneT) / 200.0 )* 100.0 ) / Max(1 ,
    Temperature_remote_accesS)*100.0,True))),Min(100,(
    Smart_security_camerA + Motion_sensorS) / Max(1 ,Smart_security_camerA
     + Motion_sensorS)*100.0))))
Smart_thermostaT= Min(Temperature_settinG,Piecewise((100,
    Temperature_remote_accesS<=0),((((Temperature_remote_accesS + InterneT
    ) / 200.0 )* 100.0 ) / Max(1 ,Temperature_remote_accesS)*100.0,True)))
Security_sensoR= Min(100,(Smart_security_camerA + Motion_sensorS) / Max(1
     ,Smart_security_camerA + Motion_sensorS)*100.0)
Internet_connectioN= Min(Cyber_attack_detectioN,InterneT)
#The function of Model
ShsfeatureS= Min(Min(Min(WifI,Min(100,(Smart_griD + Wind_turbineS +
    Solar_paneL) / Max(1 ,Smart_griD + Wind_turbineS + Solar_paneL)*100.0)
    ),Min(Min(Min(Wind_sensorS,Smoke_detectoR),Min(Light_sensorS,Min(Min(
    Temperature_settinG,Piecewise((100,Temperature_remote_accesS<=0),((((
```

```
    Temperature_remote_accesS + InterneT) / 200.0 )* 100.0 ) / Max(1 ,
    Temperature_remote_accesS)*100.0,True))),Min(100,(
    Smart_security_camerA + Motion_sensorS) / Max(1 ,Smart_security_camerA
     + Motion_sensorS)*100.0)))),Min(Min((Max(Close_windowS,Open_windowS))
     / Max(1,Close_windowS + Open_windowS)*100.0,(Max(Off_dish_washeR,
    On_dish_washeR)) / Max(1,Off_dish_washeR + On_dish_washeR)*100.0),Min(
    Min((Max(On_air_ventilatoR,Off_air_ventilatoR)) / Max(1,
    On_air_ventilatoR + Off_air_ventilatoR)*100.0,Min((Max(On_water_heateR
    ,Off_water_heateR)) / Max(1,On_water_heateR + Off_water_heateR)*100.0,
    Piecewise((100,Max(Water_heater_remote_accesS,Water_heater_schedulE)
    <=0),(((((Water_heater_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Water_heater_schedulE) / Max(1 ,Water_heater_remote_accesS +
    Water_heater_schedulE)*100.0,True)))),Min(Min((Max(Min(Min(0,On_heateR
     - On_air_conditioneR),(((On_heateR + Min(Close_windowS,
    Off_air_ventilatoR)) / 200.0 )* 100.0 )),Off_heateR)) / Max(1,
    On_heateR + Off_heateR)*100.0,Piecewise((100,Max(Heater_remote_accesS,
    Heater_schedulE)<=0),(((((Heater_remote_accesS + InterneT) / 200.0 )*
    100.0 ) + Heater_schedulE) / Max(1 ,Heater_remote_accesS +
    Heater_schedulE)*100.0,True))),Min((Max(Off_air_conditioneR,Min(Min(0,
    On_air_conditioneR - On_heateR),(((On_air_conditioneR + Min(
    Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 )))) / Max(1,
    Off_air_conditioneR + On_air_conditioneR)*100.0,Piecewise((100,Max(
    Air_conditioner_remote_accesS,Air_conditioner_schedulE)<=0),(((((
    Air_conditioner_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
    Air_conditioner_schedulE) / Max(1 ,Air_conditioner_remote_accesS +
    Air_conditioner_schedulE)*100.0,True)))))))),Piecewise((100,Max(
    Cyber_attack_detectioN,Max(Learning_techniquE,InterneT))<=0),((Min(
    Cyber_attack_detectioN,InterneT) + Learning_techniquE) / Max(1 ,Max(
    Cyber_attack_detectioN,InterneT) + Learning_techniquE)*100.0,True)))
FMDiagramName =  'ShsfeatureS'

#variable list
List= ['Water_heater_schedulE','Motion_sensorS','Off_dish_washeR','
    Learning_techniquE','Solar_paneL','On_water_heateR','
    Off_air_ventilatoR','Temperature_settinG','Wind_turbineS','
    Heater_schedulE','Heater_remote_accesS','Water_heater_remote_accesS','
    On_heateR','On_air_ventilatoR','Smart_griD','Wind_sensorS','
    On_air_conditioneR','WifI','InterneT','Off_air_conditioneR','
    Off_water_heateR','Air_conditioner_schedulE','
    Temperature_remote_accesS','Air_conditioner_remote_accesS','
    Cyber_attack_detectioN','Close_windowS','Smoke_detectoR','
    Light_sensorS','Smart_security_camerA','Open_windowS','On_dish_washeR'
    ,'Infrared_sensoR','Off_heateR']
LANG = []
langList = ['python','c','c++','java',"javascript",'matlab','r','cp']

List= ['Water_heater_schedulE','Motion_sensorS','Off_dish_washeR','
    Learning_techniquE','Solar_paneL','On_water_heateR','
    Off_air_ventilatoR','Temperature_settinG','Wind_turbineS','
    Heater_schedulE','Heater_remote_accesS','Water_heater_remote_accesS','
    On_heateR','On_air_ventilatoR','Smart_griD','Wind_sensorS','
    On_air_conditioneR','WifI','InterneT','Off_air_conditioneR','
    Off_water_heateR','Air_conditioner_schedulE','
    Temperature_remote_accesS','Air_conditioner_remote_accesS','
```

```
    Cyber_attack_detectioN','Close_windowS','Smoke_detectoR','
    Light_sensorS','Smart_security_camerA','Open_windowS','On_dish_washeR'
    ,'Infrared_sensoR','Off_heateR']
def allPrint():
    Translate('Min(Min(Min(WifI,Min(100,(Smart_griD + Wind_turbineS +
        Solar_paneL) / Max(1 ,Smart_griD + Wind_turbineS + Solar_paneL)
        *100.0)),Min(Min(Min(Wind_sensorS,Smoke_detectoR),Min(
        Light_sensorS,Min(Min(Temperature_settinG,Piecewise((100,
        Temperature_remote_accesS<=0),((((Temperature_remote_accesS +
        InterneT) / 200.0 )* 100.0 ) / Max(1 ,Temperature_remote_accesS)
        *100.0,True))),Min(100,(Smart_security_camerA + Motion_sensorS) /
        Max(1 ,Smart_security_camerA + Motion_sensorS)*100.0)))),Min(Min((
        Max(Close_windowS,Open_windowS)) / Max(1,Close_windowS +
        Open_windowS)*100.0,(Max(Off_dish_washeR,On_dish_washeR)) / Max(1,
        Off_dish_washeR + On_dish_washeR)*100.0),Min(Min((Max(
        On_air_ventilatoR,Off_air_ventilatoR)) / Max(1,On_air_ventilatoR +
         Off_air_ventilatoR)*100.0,Min((Max(On_water_heateR,
        Off_water_heateR)) / Max(1,On_water_heateR + Off_water_heateR)
        *100.0,Piecewise((100,Max(Water_heater_remote_accesS,
        Water_heater_schedulE)<=0),(((((Water_heater_remote_accesS +
        InterneT) / 200.0 )* 100.0 ) + Water_heater_schedulE) / Max(1 ,
        Water_heater_remote_accesS + Water_heater_schedulE)*100.0,True))))
        ,Min(Min((Max(Min(Min(0,On_heateR - On_air_conditioneR),(((
        On_heateR + Min(Close_windowS,Off_air_ventilatoR)) / 200.0 )*
        100.0 )),Off_heateR)) / Max(1,On_heateR + Off_heateR)*100.0,
        Piecewise((100,Max(Heater_remote_accesS,Heater_schedulE)<=0),(((((
        Heater_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
        Heater_schedulE) / Max(1 ,Heater_remote_accesS + Heater_schedulE)
        *100.0,True))),Min((Max(Off_air_conditioneR,Min(Min(0,
        On_air_conditioneR - On_heateR),(((On_air_conditioneR + Min(
        Close_windowS,Off_air_ventilatoR)) / 200.0 )* 100.0 )))) / Max(1,
        Off_air_conditioneR + On_air_conditioneR)*100.0,Piecewise((100,Max
        (Air_conditioner_remote_accesS,Air_conditioner_schedulE)<=0),(((((
        Air_conditioner_remote_accesS + InterneT) / 200.0 )* 100.0 ) +
        Air_conditioner_schedulE) / Max(1 ,Air_conditioner_remote_accesS +
         Air_conditioner_schedulE)*100.0,True)))))))),Piecewise((100,Max(
        Cyber_attack_detectioN,Max(Learning_techniquE,InterneT))<=0),((Min
        (Cyber_attack_detectioN,InterneT) + Learning_techniquE) / Max(1 ,
        Max(Cyber_attack_detectioN,InterneT) + Learning_techniquE)*100.0,
        True)))',FMDiagramName,List,LANG,2)
if(len(sys.argv)==1):
  LANG = langList
  allPrint()
else:
  for i in sys.argv:
    if(sys.argv.index(i)==0):continue
    if  (i.lower() not in langList):
      LANG = langList
      break
    else:
      LANG.append(str(i.lower()))
  allPrint()
```

B.3 Optimization Model of SHMS

Listing B.2 provides the final optimization model of the SHMS. It first defines the ancillary variables (coming from GRL indicators) and the decision variables whose values are selected by the optimizer. In addition to the feature and GRL functions (defined as "decision expressions" here, with the prefix `dexpr`), the model indicates what to maximize (the `SmartHome` GRL model) and additional constraints on some variables.

Listing B.2: Model optimization code for the Smart Home case study, for IBM CPLEX' CP Optimizer

```
using CP;

// Ancillary variables
float Electricity_pricES=...;
int Cooling_habitSS=...;
int Cooling_timES=...;
int FirE=...;
int FulL=...;
int Heating_habitSS=...;
int Heating_timES=...;
int Heating_water_timES=...;
int LighT=...;
float TemperaturES=...;
int No_suspicious_behaviour_detecteDS=...;
int Remote_coolinGS=...;
int Remote_water_heatinGS=...;
float Response_timES=...;
int Smoke_detectioNS=...;
int Suitable_weatheR=...;
int User_permissioN=...;
float Wind_speeDS=...;
int Remote_heatinGS=...;
int Enable_water_heater_accesS=...;
int Enable_water_heater_schedulE=...;
int Enable_temperature_accesS=...;
int Enable_air_conditioner_accesS=...;
int Enable_heater_schedulE=...;
int Enable_heater_accesS=...;
int Enable_air_conditioner_schedulE=...;

// Decision variables
dvar int Learning_techniquE in 0..100;
dvar int Off_air_ventilatoR in 0..100;
dvar int Temperature_settinG in 0..100;
dvar int Wind_sensorS in 0..100;
dvar int Off_water_heateR in 0..100;
dvar int Cyber_attack_detectioN in 0..100;
dvar int Light_sensorS in 0..100;
dvar int Off_dish_washeR in 0..100;
dvar int Motion_sensorS in 0..100;
dvar int Smart_griD in 0..100;
```

```
dvar int WifI in 0..100;
dvar int Close_windowS in 0..100;
dvar int Smart_security_camerA in 0..100;
dvar int Infrared_sensoR in 0..100;
dvar int Smoke_detectoR in 0..100;
dvar int Off_heateR in 0..100;
dvar int Off_air_conditioneR in 0..100;


// Features expressions
 int Response_timE= Response_timES<=2 ? 100: 0;
dexpr int No_suspicious_behaviour_detecteD =
   No_suspicious_behaviour_detecteDS;
dexpr int InterneT = minl(No_suspicious_behaviour_detecteD ,
   Response_timE) ;
dexpr int Water_heater_remote_accesSGRL=Enable_water_heater_accesS;
dexpr int Water_heater_remote_accesS=(((Water_heater_remote_accesSGRL +
   InterneT) / 200.0 )* 100.0 )< 100? 0 :100;
dexpr int Water_heater_schedulE=Enable_water_heater_schedulE;
dexpr int Temperature_remote_accesSGRL=Enable_temperature_accesS;
dexpr int Temperature_remote_accesS=(((Temperature_remote_accesSGRL +
   InterneT) / 200.0 )* 100.0 )<100?0:100;
dexpr int Air_conditioner_remote_accesSGRL=Enable_air_conditioner_accesS;
dexpr int Air_conditioner_remote_accesS=(((
   Air_conditioner_remote_accesSGRL + InterneT) / 200.0 )* 100.0 )< 100?
   0 :100;
dexpr int Heater_schedulE=Enable_heater_schedulE;
dexpr int Heater_remote_accesSGRL=Enable_heater_accesS;
dexpr int Heater_remote_accesS=(((Heater_remote_accesSGRL + InterneT) /
   200.0 )* 100.0 )<100?0:100;
dexpr int Air_conditioner_schedulE=Enable_air_conditioner_schedulE;
dexpr int Cooling_habitS=minl(Learning_techniquE, Cooling_habitSS);
dexpr int Cooling_timE=minl(Air_conditioner_schedulE, Cooling_timES);
dexpr int Heating_habitS=minl(Learning_techniquE, Heating_habitSS);
dexpr int Heating_timE=minl(Heater_schedulE, Heating_timES);
dexpr int Heating_water_timE=minl(Water_heater_schedulE,
   Heating_water_timES);
dexpr float High_temperaturE=minl(maxl(Temperature_settinG ,
   Enable_temperature_accesS),TemperaturES>=27?100:0);
dexpr float Low_temperaturE=minl(maxl(Temperature_settinG ,
   Enable_temperature_accesS),TemperaturES<=15?100:0);
dexpr float Remote_coolinG=minl(Air_conditioner_remote_accesS,
   Remote_coolinGS);
dexpr int Remote_heatinG=minl(Heater_remote_accesS, Remote_heatinGS);
dexpr int Remote_water_heatinG=minl(Water_heater_remote_accesS,
   Remote_water_heatinGS);
dexpr float Electricity_pricE = (Electricity_pricES <= 6.4) ? ( 100): 0;
dexpr float On_dish_washeR = minl(Electricity_pricE , FulL);
dexpr int Smoke_detectioN=minl(Smoke_detectoR, Smoke_detectioNS);
dexpr int Wind_speeD=(Wind_speeDS>=12)&&(Wind_speeDS<90)? 100:0;
dexpr int Wind_turbines = Wind_speeD;
dexpr int Solar_paneL = LighT;
dexpr int On_air_ventilatoR=Smoke_detectioN ;
dexpr float Open_windowS = maxl(0.0 , minl(100.0 , (100*FirE+minl(
```

```
    Suitable_weatheR,minl(No_suspicious_behaviour_detecteD,User_permissioN
    ))*100.0) / 100.0)) ;
dexpr int On_water_heateR = maxl(Remote_water_heatinG ,
    Heating_water_timE) ;

// From GRL function
dexpr float On_heateRGRL = minl(Low_temperaturE,maxl(0.0 , minl(100.0 , (
    maxl(Remote_heatinG,maxl(Heating_habitS,Heating_timE))*100.0) / 100.0)
    ));
// From feature functions
dexpr float On_heateR=(((On_heateRGRL + minl(Close_windowS,
    Off_air_ventilatoR)) / 200.0 )* 100.0 )< 100? 0 :On_heateRGRL;
// From GRL function
dexpr float On_air_conditionerGRL = minl(High_temperaturE,maxl(0.0 , minl
    (100.0 , (maxl(Remote_coolinG,maxl(Cooling_habitS,Cooling_timE))
    *100.0) / 100.0))) ;
// From feature functions
dexpr float On_air_conditioneR=minl(maxl(0,On_air_conditionerGRL -
    On_heateR),(((On_air_conditionerGRL + minl(Close_windowS,
    Off_air_ventilatoR)) / 200.0 )* 100.0 ))< 100? 0 :
    On_air_conditionerGRL;

// Smart Home GRL function
dexpr float  SmartHome=(1/32)*maxl(0.0, minl(100.000000000000, 1*(
    Solar_paneL + Wind_turbines))) + (1/8)*maxl(0.0, minl
    (100.000000000000, -0.25*InterneT + 0.25*Temperature_settinG + 0.75*
    maxl(Cyber_attack_detectioN, maxl(Motion_sensorS,
    Smart_security_camerA)))) + (5/32)*maxl(0.0, minl(100.000000000000,
    0.25*(Air_conditioner_schedulE + Heater_schedulE - InterneT +
    On_air_ventilatoR - Open_windowS + Water_heater_schedulE))) + (3/32)*
    maxl(0.0, minl(100.000000000000, 0.25*Air_conditioner_schedulE + 0.25*
    Heater_schedulE + 0.5*InterneT + 0.75*Learning_techniquE - 0.25*
    On_air_ventilatoR + 0.25*Open_windowS + 0.25*Water_heater_schedulE)) +
     (3/32)*maxl(0.0, minl(100.000000000000, 0.25*Air_conditioner_schedulE
     + 0.25*Heater_schedulE + 0.5*InterneT + 0.75*Learning_techniquE +
    0.25*Temperature_remote_accesS + 0.25*Water_heater_schedulE - 0.25*
    maxl(0.0, minl(100.000000000000, -0.25*InterneT + 0.25*
    Temperature_settinG + 0.75*maxl(Cyber_attack_detectioN, maxl(
    Motion_sensorS, Smart_security_camerA)))))) + (1/2)*minl(
    Infrared_sensoR, minl(Light_sensorS, minl(Smoke_detectoR, minl(
    Wind_sensorS, minl(maxl(Temperature_remote_accesS, Temperature_settinG
    ), minl(maxl(Cyber_attack_detectioN, Motion_sensorS,
    Smart_security_camerA), minl(maxl(Smart_griD, Solar_paneL,
    Wind_turbines), minl(maxl(Off_dish_washeR, Off_water_heateR,
    On_dish_washeR, On_water_heateR), minl(maxl(Close_windowS,
    Off_air_conditioneR, Off_air_ventilatoR, Off_heateR,
    On_air_conditioneR, On_air_ventilatoR, On_heateR, Open_windowS), maxl(
    Air_conditioner_remote_accesS, maxl(Air_conditioner_schedulE, maxl(
    Heater_remote_accesS, maxl(Heater_schedulE, maxl(InterneT, maxl(
    Learning_techniquE, maxl(Water_heater_remote_accesS, maxl(
    Water_heater_schedulE, WifI))))))))))))))));
// Features model function
dexpr float FeatureModel=minl(Light_sensorS, minl(Smoke_detectoR, minl(
    Temperature_settinG, minl(WifI, minl(Wind_sensorS, minl(100.0*(
```

```
    Motion_sensorS + Smart_security_camerA)/maxl(1, Motion_sensorS +
    Smart_security_camerA), minl(100.0*maxl(Close_windowS, Open_windowS)/
    maxl(1, Close_windowS + Open_windowS), minl(100.0*maxl(
    Off_air_ventilatoR, On_air_ventilatoR)/maxl(1, Off_air_ventilatoR +
    On_air_ventilatoR), minl(100.0*maxl(Off_dish_washeR, On_dish_washeRF)/
    maxl(1, Off_dish_washeR + On_dish_washeR), minl(100.0*maxl(
    Off_water_heateR, On_water_heateR)/maxl(1, Off_water_heateR +
    On_water_heateR), minl(100.0*(Smart_griD + Solar_paneL + Wind_turbines
    )/maxl(1, Smart_griD + Solar_paneL + Wind_turbines), minl(((
    Temperature_remote_accesS <= 0) ? (
    100
)
: (
    50.0*(InterneT + Temperature_remote_accesS)/maxl(1,
        Temperature_remote_accesS)
)), minl(100.0*maxl(Off_heateR, On_heateR)/maxl(1, Off_heateR + On_heateR
    ), minl(((maxl(Cyber_attack_detectioN, maxl(InterneT,
    Learning_techniquE)) <= 0) ? (
    100
)
: (
    100.0*(Learning_techniquE + minl(Cyber_attack_detectioN, InterneT))/
        maxl(1, Learning_techniquE + maxl(Cyber_attack_detectioN, InterneT)
        )
)), minl(100.0*maxl(Off_air_conditioneR, On_air_conditioneR )/maxl(1,
    Off_air_conditioneR + On_air_conditioneR), minl(((maxl(
    Air_conditioner_remote_accesS, Air_conditioner_schedulE) <= 0) ? (
    100
)
: (
    (50.0*Air_conditioner_remote_accesS + 100.0*Air_conditioner_schedulE +
         50.0*InterneT)/maxl(1, Air_conditioner_remote_accesS +
        Air_conditioner_schedulE)
)), minl(((maxl(Heater_remote_accesS, Heater_schedulE) <= 0) ? (
    100
)
: (
    (50.0*Heater_remote_accesS + 100.0*Heater_schedulE + 50.0*InterneT)/
        maxl(1, Heater_remote_accesS + Heater_schedulE)
)), ((maxl(Water_heater_remote_accesS, Water_heater_schedulE) <= 0) ? (
    100
)
: (
    (50.0*InterneT + 50.0*Water_heater_remote_accesS + 100.0*
        Water_heater_schedulE)/maxl(1, Water_heater_remote_accesS +
        Water_heater_schedulE)
)))))))))))))))));

maximize SmartHome;

subject to {
FeatureModel==100;
decisionVar:
```

```
 Learning_techniquE==0 || Learning_techniquE==100;
 Off_air_ventilatoR==0 || Off_air_ventilatoR==100;
 Temperature_settinG==0 || Temperature_settinG==100;
 Wind_sensorS==0 || Wind_sensorS==100;
 Off_water_heateR==0 || Off_water_heateR==100;
 Cyber_attack_detectioN==0 || Cyber_attack_detectioN==100;
 Light_sensorS==0 || Light_sensorS==100;
 Off_dish_washeR==0 || Off_dish_washeR==100;
 Motion_sensorS==0 || Motion_sensorS==100;
 Smart_griD==0 || Smart_griD==100;
 WifI==0 || WifI==100;
 Close_windowS==0 || Close_windowS==100;
 Smart_security_camerA==0 || Smart_security_camerA==100;
 Infrared_sensoR==0 || Infrared_sensoR==100;
 Smoke_detectoR==0 || Smoke_detectoR==100;
 Off_heateR==0 || Off_heateR==100;
 Off_air_conditioneR==100 || Off_air_conditioneR==0 ;
}
```

Appendix C

Model Traceability Management, with Application to the Smart Home

This appendix provides the reusable template to export SysML models created using Apache'S *Velocity Template Language* (VTL) [24] and executed by No Magic's Cameo Systems Modeler tool (Section 5.1.2). It also includes the generated DXL script that describes the SysML model of the Smart Home Management System (SHMS) introduced in Section 8.1. The reusable consistency and completeness rules coded in DXL are finally presented.

C.1 Report Template for Exporting SysML Models

Listing C.1 details the report template that we created to generate DXL descriptions of SysML requirements and block definition diagrams created by No Magic's Cameo Systems Modeler tool.

Listing C.1: VTL report template for producing DXL scripts from SysML models

```
#set($var1="#")
$var1 include "addins/DSL/lib/Utilities.dxl"
pragma runLim, 0
beginImport("${project.name}")

#* Read and print diagrams' information *#
#foreach ($diagram in $Diagram)
//$diagram.diagramType
#if($diagram.diagramType == "SysML Block Definition Diagram")
blockDefinitionDiagram("$diagram.elementID","$diagram.name", "", "$
    diagram.image","$diagram.diagramType of Smart Home Case study" )
#set($eList = $report.getDiagramElements($diagram))
#foreach($e in $sorter.sort($eList,"elementType"))
#if($e.elementType == "block" || $e.elementType == "constraintblock")
blockToDiagram("$e.elementID$diagram.elementID","ContainedBy","","$e.
    elementID","$diagram.elementID")
```

```
#end
#end
#else
#if($diagram.diagramType == "Requirement Diagram")
requirementsDiagram("$diagram.elementID","$diagram.name", "", "$diagram.
    image","diagram.diagramType of Smart Home Case study" )
#set($rList = $report.getDiagramElements($diagram))
#foreach($r in $sorter.sort($rList,"elementType"))
#if($r.elementType == "requirement")
requirementToDiagram("$r.elementID$diagram.elementID","ContainedBy","","$
    r.elementID","$diagram.elementID")
#end
#end
#end
#end
#end

// Blocks
#foreach($e in $Block)
block("$e.elementID","$e.name","$e.documentation","Block")
#set($varlast= "")
#foreach($Satss in $sorter.sort($report.filterElement($e.clientDependency
    , ["Satisfy"]),"supplier"))
#set($s=$report.getSupplierElement($Satss))
#if($varlast != $s.elementID)
#set($varlast = $s.elementID)
requirementToBlock("$Satss.elementID","SatisfiedBy","","$s.elementID","$e
    .elementID")
#end
#end

#* Generalization relationships*#
#foreach($parttc in $sorter.sort($report.filterElement($e.ownedElement, [
    "Generalization"]),"general"))
Blockparent("$parttc.elementID","ChildOf","","$e.elementID","$parttc.
    general.elementID")
#end
#set($varlast= "")
#set($varlastnone= "")

#* Aggregation relationships*#
#foreach($parttr in $sorter.sort($report.filterElement($e.ownedAttribute,
     ["Reference Property"]),"type"))
#if($parttr.Aggregation =="shared")
#if($varlast != $parttr.type.elementID)
#set($varlast = $parttr.type.elementID)
Blockaggregation("$parttr.elementID","IsSharedBy","","$parttr.type.
    elementID","$e.elementID")
#end
#end
#if($parttr.Aggregation =="none" && $varlastnone != $parttr.type.
    elementID)
#set($varlastnone= $parttr.type.elementID)
blocklink("$parttr.elementID","$parttr.association.name","","$e.elementID
```

```
    ","$parttr.type.elementID")
#end
#end
#set($varlast= "")

#* DependOn relationships*#
#foreach($parttc in $sorter.sort($report.filterElement($e.
    supplierDependency, ["Usage"]),"client"))
#set($c=$report.getClientElement($parttc))
#if($varlast != $c.elementID)
#set($varlast = $c.elementID)
BlockDependency("$parttc.elementID","DependOn","","$c.elementID","$e.
    elementID")
#end
#end

#* Block's parts *#
#foreach($partt in $e.ownedAttribute)
#if((! $report.isNull($partt.type)) && ($partt.Aggregation =="composite")
    )
   blockparts("$partt.elementID","PartOf","","$partt.type.elementID","$e.
       elementID")
#else
#if($partt.elementType=="partproperty")
blockparts("$partt.elementID","PartOf","","$partt.elementID","$e.
    elementID")
block("$partt.elementID","$partt.name","$partt.documentation","Block")

#* Satisfy relationships*#
#set($varlast1= "")
#foreach($Satss in $sorter.sort($report.filterElement($partt.
    clientDependency, ["Satisfy"]),"supplier"))
#set($s=$report.getSupplierElement($Satss))
#if($varlast1 != $s.elementID)
#set($varlast1 = $s.elementID)
requirementToBlock("$Satss.elementID","SatisfiedBy","","$s.elementID","$
    partt.elementID")
#end
#end

#* Parts' dependency *#
#set($varlast1= "")
#foreach($parttc in $sorter.sort($report.filterElement($partt.
    supplierDependency, ["Usage"]),"client"))
#set($c=$report.getClientElement($parttc))
#if($varlast1 != $c.elementID)
#set($varlast1 = $c.elementID)
BlockDependency("$parttc.elementID","DependOn","","$c.elementID","$partt.
    elementID")
#end
#end
// End of nested parts

#end
```

```
#end
#end
#end

#* ConstraintBlock *#
#foreach($e in $ConstraintBlock)
block("$e.elementID","$e.name","$e.documentation","ConstraintBlock")
#set($varlast= "")
#foreach($Satss in $sorter.sort($report.filterElement($e.clientDependency
    , ["Satisfy"]),"supplier"))
#set($s=$report.getSupplierElement($Satss))
#if($varlast != $s.elementID)
#set($varlast = $s.elementID)
requirementToBlock("$Satss.elementID","SatisfiedBy","","$s.elementID","$e
    .elementID")
#end
#end

#* Generalization relationships of constraint blocks *#
#foreach($parttc in $sorter.sort($report.filterElement($e.ownedElement, [
    "Generalization"]),"general"))
Blockparent("$parttc.elementID","ChildOf","","$e.elementID","$parttc.
    general.elementID")
#end

#* Aggregation relationships of constraint block *#
#set($varlast= "")
#set($varlastnone= "")
#foreach($parttr in $sorter.sort($report.filterElement($e.ownedAttribute,
     ["Reference Property"]),"type"))
#if($parttr.Aggregation =="shared")
#if($varlast != $parttr.type.elementID)
#set($varlast = $parttr.type.elementID)
Blockaggregation("$parttr.elementID","IsSharedBy","","$parttr.type.
    elementID","$e.elementID")
#end
#end
#if($parttr.Aggregation =="none" && $varlastnone != $parttr.type.
    elementID)
#set($varlastnone= $parttr.type.elementID)
blocklink("$parttr.elementID","$parttr.association.name","","$e.elementID
    ","$parttr.type.elementID")
#end
#end

#* DependOn relationships of constraint blocks *#
#set($varlast= "")
#foreach($parttc in $sorter.sort($report.filterElement($e.
    supplierDependency, ["Usage"]),"client"))
#set($c=$report.getClientElement($parttc))
#if($varlast != $c.elementID)
#set($varlast = $c.elementID)
BlockDependency("$parttc.elementID","DependOn","","$c.elementID","$e.
    elementID")
```

```
#end
#end

#* Parts of constraint blocks *#
#foreach($partt in $report.filterElement($e.ownedAttribute, ["Part
    Property"]))
#if($partt.elementType=="partproperty")
#if($report.isNull($partt.type))
blockparts("$partt.elementID","PartOf","","$partt.elementID","$e.
    elementID")
block("$partt.elementID","$partt.name","$partt.documentation","Block")
#else
blockparts("$partt.elementID","PartOf","","$partt.type.elementID","$e.
    elementID")
#end
#end
#end
#end

// Requirements
#foreach($r in $Requirement)
#if($r.elementType == "requirement")
requirement("$r.elementID","$r.name","$r.text","$r.id")
#foreach($con in $r.nestedClassifier)
subrequirement("$con.elementID$r.elementID","ContainedBy","","$con.
    elementID","$r.elementID")
#end
#set($varlast= "")
#foreach($parttc in $sorter.sort($report.filterElement($r.
    supplierDependency, ["DeriveReqt"]),"client"))
#set($c=$report.getClientElement($parttc))
#if($varlast != $c.elementID)
#set($varlast = $c.elementID)
requirementlink("$parttc.elementID","DerivedFrom","","$c.elementID","$r.
    elementID")
#end
#end
#end
#end

endImport
```

C.2 DXL Script of the SHMS SysML Model

Listing C.2 shows the generated DXL descriptions of the SHMS SysML model described in Chapter 7. This script was produced by using the report template in Listing C.1 with No Magic's Cameo Systems Modeler tool.

Listing C.2: DXL script of the SysML model of SHMS in Report.txt

```
# include "addins/DSL/lib/Utilities.dxl"
pragma runLim, 0
beginImport("Smart_HomeModel")

// SysML Block Definition Diagram
blockDefinitionDiagram("_19_0_1_3ed01a0_1556488933151_218962_41629","Smart home", "", "file:/C:/
    CaseStudy/Report_files/_19_0_1_3ed01a0_1556488933151_218962_41629.jpg","SysML Block Definition
    Diagram of Smart Home Case study" )
blockToDiagram("_19_0_2_5f801fb_1577719967217_702345_42272_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_2_5f801fb_1577719967217_702345_42272","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_1_3ed01a0_1556492760396_507721_41712_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_1_3ed01a0_1556492760396_507721_41712","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_2_5f801fb_1577316023401_419606_44790_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_2_5f801fb_1577316023401_419606_44790","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_2_5f801fb_1577316008752_797941_44745_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_2_5f801fb_1577316008752_797941_44745","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_2_5f801fb_1577719829903_504408_42226_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_2_5f801fb_1577719829903_504408_42226","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_1_3ed01a0_1556488960061_314183_41664_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_1_3ed01a0_1556488960061_314183_41664","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_1_3ed01a0_1557653152129_996648_41998_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_1_3ed01a0_1557653152129_996648_41998","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_1_3ed01a0_1556493245441_397422_41802_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_1_3ed01a0_1556493245441_397422_41802","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_2_5f801fb_1577725403297_875277_42675_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_2_5f801fb_1577725403297_875277_42675","
    _19_0_1_3ed01a0_1556488933151_218962_41629")
blockToDiagram("_19_0_1_3ed01a0_1556494897831_576536_41895_19_0_1_3ed01a0_1556488933151_218962_41629
    ","ContainedBy","","_19_0_1_3ed01a0_1556494897831_576536_41895","
    _19_0_1_3ed01a0_1556488933151_218962_41629")

// SysML Block Definition Diagram
blockDefinitionDiagram("_19_0_1_3ed01a0_1557280989232_10149_42072","Appliance", "", "file:/C:/
    CaseStudy/Report_files/_19_0_1_3ed01a0_1557280989232_10149_42072.jpg","SysML Block Definition
    Diagram of Smart Home Case study" )
blockToDiagram("_19_0_2_5f801fb_1577309819323_228579_43573_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577309819323_228579_43573","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577296462612_990181_42335_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577296462612_990181_42335","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577298166442_772973_42959_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577298166442_772973_42959","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577298113692_753529_42824_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577298113692_753529_42824","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577298253341_75561_43049_19_0_1_3ed01a0_1557280989232_10149_42072",
    "ContainedBy","","_19_0_2_5f801fb_1577298253341_75561_43049","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577296513628_30107_42425_19_0_1_3ed01a0_1557280989232_10149_42072",
    "ContainedBy","","_19_0_2_5f801fb_1577296513628_30107_42425","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577296421097_27138_42245_19_0_1_3ed01a0_1557280989232_10149_42072",
    "ContainedBy","","_19_0_2_5f801fb_1577296421097_27138_42245","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_1_3ed01a0_1556493245441_397422_41802_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_1_3ed01a0_1556493245441_397422_41802","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577725403297_875277_42675_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577725403297_875277_42675","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577298149529_988792_42914_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577298149529_988792_42914","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577296439888_254934_42290_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577296439888_254934_42290","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577296335448_164770_42200_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577296335448_164770_42200","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577298128702_302366_42869_19_0_1_3ed01a0_1557280989232_10149_42072"
    ,"ContainedBy","","_19_0_2_5f801fb_1577298128702_302366_42869","
    _19_0_1_3ed01a0_1557280989232_10149_42072")
blockToDiagram("_19_0_2_5f801fb_1577296486581_87921_42380_19_0_1_3ed01a0_1557280989232_10149_42072",
    "ContainedBy","","_19_0_2_5f801fb_1577296486581_87921_42380","
    _19_0_1_3ed01a0_1557280989232_10149_42072")

// Requirements Diagram
requirementsDiagram("_19_0_1_3ed01a0_1556488714506_800572_41597","Smart Home Requirements", "", "
    file:/C:/CaseStudy/Report_files/_19_0_1_3ed01a0_1556488714506_800572_41597.jpg","diagram.
    diagramType of Smart Home Case study" )
requirementToDiagram("
```

```
      _19_0_2_5f801fb_1577717654485_241642_42189_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577717654485_241642_42189","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_1_3ed01a0_1556505449360_339815_42138_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_1_3ed01a0_1556505449360_339815_42138","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_1_3ed01a0_1557247827853_665093_42038_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_1_3ed01a0_1557247827853_665093_42038","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577762436696_581641_42351_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577762436696_581641_42351","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_1_3ed01a0_1556505347654_420954_42095_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_1_3ed01a0_1556505347654_420954_42095","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_1_3ed01a0_1556759570816_345133_41755_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_1_3ed01a0_1556759570816_345133_41755","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577762642929_263788_42403_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577762642929_263788_42403","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577762320087_573213_42342_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577762320087_573213_42342","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577762675944_55200_42412_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577762675944_55200_42412","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577761938994_450219_42234_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577761938994_450219_42234","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577825810947_416118_43677_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577825810947_416118_43677","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577819040506_475885_43021_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577819040506_475885_43021","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577762769175_699539_42421_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577762769175_699539_42421","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577818860636_983262_42888_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577818860636_983262_42888","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577762584541_170356_42369_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577762584541_170356_42369","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577818913027_109047_42931_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577818913027_109047_42931","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577819029972_864756_42978_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577819029972_864756_42978","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577761862440_951449_42191_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577761862440_951449_42191","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_1_3ed01a0_1557691990274_110187_42455_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_1_3ed01a0_1557691990274_110187_42455","
      _19_0_1_3ed01a0_1556488714506_800572_41597")
requirementToDiagram("
      _19_0_2_5f801fb_1577762835986_935540_42430_19_0_1_3ed01a0_1556488714506_800572_41597","
      ContainedBy","","_19_0_2_5f801fb_1577762835986_935540_42430","
      _19_0_1_3ed01a0_1556488714506_800572_41597")

// SysML Block Definition Diagram
blockDefinitionDiagram("_19_0_2_5f801fb_1577325420209_498574_44969","Monitoring", "", "file:/C:/
      CaseStudy/Report_files/_19_0_2_5f801fb_1577325420209_498574_44969.jpg","SysML Block Definition
      Diagram of Smart Home Case study" )
blockToDiagram("_19_0_2_5f801fb_1579314646156_228960_42246_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1579314646156_228960_42246","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_1_3ed01a0_1556494897831_576536_41895_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_1_3ed01a0_1556494897831_576536_41895","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327875928_803538_46289_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327875928_803538_46289","
```

```
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327826392_311144_45860_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327826392_311144_45860","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327811175_439055_45665_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327811175_439055_45665","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327851008_819294_46094_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327851008_819294_46094","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327889448_436692_46406_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327889448_436692_46406","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327900704_434303_46523_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327900704_434303_46523","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327817213_750406_45743_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327817213_750406_45743","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327893800_204695_46445_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327893800_204695_46445","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327813544_912565_45704_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327813544_912565_45704","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327863256_270017_46211_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327863256_270017_46211","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327837936_643658_45938_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327837936_643658_45938","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327840861_860881_45977_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327840861_860881_45977","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327835504_276967_45899_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327835504_276967_45899","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327823048_822183_45821_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327823048_822183_45821","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327879632_857017_46328_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327879632_857017_46328","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327820255_551671_45782_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327820255_551671_45782","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327908296_710205_46601_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327908296_710205_46601","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327843673_515599_46016_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327843673_515599_46016","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327867985_639474_46250_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327867985_639474_46250","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327884391_667651_46367_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327884391_667651_46367","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327855079_680504_46133_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327855079_680504_46133","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327904395_111499_46562_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327904395_111499_46562","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327846520_969016_46055_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327846520_969016_46055","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327897552_243103_46484_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327897552_243103_46484","
      _19_0_2_5f801fb_1577325420209_498574_44969")
blockToDiagram("_19_0_2_5f801fb_1577327859648_843376_46172_19_0_2_5f801fb_1577325420209_498574_44969
      ","ContainedBy","","_19_0_2_5f801fb_1577327859648_843376_46172","
      _19_0_2_5f801fb_1577325420209_498574_44969")

// Blocks
block("_19_0_2_5f801fb_1577719829903_504408_42226","Normalize temperature","","Block")
requirementToBlock("_19_0_2_5f801fb_1577822214975_271466_43309","SatisfiedBy","","
      _19_0_2_5f801fb_1577762769175_699539_42421","_19_0_2_5f801fb_1577719829903_504408_42226")


block("_19_0_1_3ed01a0_1557653152129_996648_41998","Smart grid","","Block")


block("_19_0_1_3ed01a0_1556493245441_397422_41802","Appliance controller","","Block")
requirementToBlock("_19_0_2_5f801fb_1577813455859_808454_42811","SatisfiedBy","","
      _19_0_2_5f801fb_1577762584541_170356_42369","_19_0_1_3ed01a0_1556493245441_397422_41802")
requirementToBlock("_19_0_2_5f801fb_1577813475722_703840_42821","SatisfiedBy","","
      _19_0_1_3ed01a0_1557247827853_665093_42038","_19_0_1_3ed01a0_1556493245441_397422_41802")
requirementToBlock("_19_0_2_5f801fb_1577822208375_780631_43297","SatisfiedBy","","
      _19_0_2_5f801fb_1577762769175_699539_42421","_19_0_1_3ed01a0_1556493245441_397422_41802")

Blockaggregation("_19_0_2_5f801fb_1577597735109_239503_42303","IsSharedBy","","
```

```
      _19_0_2_5f801fb_1577296335448_164770_42200","_19_0_1_3ed01a0_1556493245441_397422_41802")
Blockaggregation("_19_0_2_5f801fb_1577597796056_215672_42320","IsSharedBy","","
      _19_0_2_5f801fb_1577296421097_27138_42245","_19_0_1_3ed01a0_1556493245441_397422_41802")
Blockaggregation("_19_0_2_5f801fb_1577597887045_823599_42371","IsSharedBy","","
      _19_0_2_5f801fb_1577296439888_254934_42290","_19_0_1_3ed01a0_1556493245441_397422_41802")
Blockaggregation("_19_0_2_5f801fb_1577308742157_7022_43328","IsSharedBy","","
      _19_0_1_3ed01a0_1556494897831_576536_41895","_19_0_1_3ed01a0_1556493245441_397422_41802")
Blockaggregation("_19_0_2_5f801fb_1577597829592_68994_42337","IsSharedBy","","
      _19_0_2_5f801fb_1577296486581_87921_42380","_19_0_1_3ed01a0_1556493245441_397422_41802")
Blockaggregation("_19_0_2_5f801fb_1577597861112_683449_42354","IsSharedBy","","
      _19_0_2_5f801fb_1577296462612_990181_42335","_19_0_1_3ed01a0_1556493245441_397422_41802")
Blockaggregation("_19_0_2_5f801fb_1577597929080_372989_42388","IsSharedBy","","
      _19_0_2_5f801fb_1577296513628_30107_42425","_19_0_1_3ed01a0_1556493245441_397422_41802")
BlockDependency("_19_0_2_5f801fb_1577720576434_527971_42347","DependOn","","
      _19_0_2_5f801fb_1577719829903_504408_42226","_19_0_1_3ed01a0_1556493245441_397422_41802")

block("_19_0_2_5f801fb_1577296439888_254934_42290","Heater controller","","Block")

blocklink("_19_0_2_5f801fb_1577309553430_456374_43480","on/off","","
      _19_0_2_5f801fb_1577296439888_254934_42290","_19_0_2_5f801fb_1577298113692_753529_42824")

block("_19_0_1_3ed01a0_1556494897831_576536_41895","Monitor","","Block")

BlockDependency("_19_0_2_5f801fb_1577722015789_722922_42453","DependOn","","
      _19_0_2_5f801fb_1577719829903_504408_42226","_19_0_1_3ed01a0_1556494897831_576536_41895")
BlockDependency("_19_0_2_5f801fb_1577330626316_976928_46641","DependOn","","
      _19_0_1_3ed01a0_1556492760396_507721_41712","_19_0_1_3ed01a0_1556494897831_576536_41895")

blockparts("_19_0_2_5f801fb_1577313409141_92851_44560","PartOf","","
      _19_0_2_5f801fb_1577313409141_92851_44560","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313409141_92851_44560","motion sensors","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577334187506_745637_47065","DependOn","","
      _19_0_2_5f801fb_1577327851008_819294_46094","_19_0_2_5f801fb_1577313409141_92851_44560")
// End of nested parts

blockparts("_19_0_2_5f801fb_1577313430645_529581_44564","PartOf","","
      _19_0_2_5f801fb_1577313430645_529581_44564","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313430645_529581_44564","wind sensors","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577334739322_846214_47117","DependOn","","
      _19_0_2_5f801fb_1577327837936_643658_45938","_19_0_2_5f801fb_1577313430645_529581_44564")
// End of nested parts

blockparts("_19_0_2_5f801fb_1577313448637_817253_44568","PartOf","","
      _19_0_2_5f801fb_1577313448637_817253_44568","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313448637_817253_44568","light sensors","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577333619795_878461_46934","DependOn","","
      _19_0_2_5f801fb_1577327820255_551671_45782","_19_0_2_5f801fb_1577313448637_817253_44568")
// End of nested parts

blockparts("_19_0_2_5f801fb_1577313464604_743054_44572","PartOf","","
      _19_0_2_5f801fb_1577313464604_743054_44572","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313464604_743054_44572","infared sensors","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577334700122_862861_47111","DependOn","","
      _19_0_2_5f801fb_1577327817213_750406_45743","_19_0_2_5f801fb_1577313464604_743054_44572")
// End of nested parts

blockparts("_19_0_2_5f801fb_1577313490964_930214_44576","PartOf","","
      _19_0_2_5f801fb_1577313490964_930214_44576","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313490964_930214_44576","smart security camera","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577334212075_553640_47077","DependOn","","
      _19_0_2_5f801fb_1577327851008_819294_46094","_19_0_2_5f801fb_1577313490964_930214_44576")
// End of nested parts

blockparts("_19_0_2_5f801fb_1577313514053_559727_44580","PartOf","","
      _19_0_2_5f801fb_1577313514053_559727_44580","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313514053_559727_44580","fire detectors","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577334641578_733917_47093","DependOn","","
      _19_0_2_5f801fb_1577327811175_439055_45665","_19_0_2_5f801fb_1577313514053_559727_44580")
// End of nested parts

blockparts("_19_0_2_5f801fb_1577313549377_708521_44584","PartOf","","
      _19_0_2_5f801fb_1577313549377_708521_44584","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313549377_708521_44584","smok detectors","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577334680905_478556_47105","DependOn","","
      _19_0_2_5f801fb_1577327826392_311144_45860","_19_0_2_5f801fb_1577313549377_708521_44584")
//End of nested parts

blockparts("_19_0_2_5f801fb_1577313719686_164099_44588","PartOf","","
```

```
      _19_0_2_5f801fb_1577313719686_164099_44588","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577313719686_164099_44588","cyber attack detector","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577334201346_345847_47071","DependOn","","
      _19_0_2_5f801fb_1577327851008_819294_46094","_19_0_2_5f801fb_1577313719686_164099_44588")
// End of nested parts

blockparts("_19_0_2_5f801fb_1577315441742_654700_44648","PartOf","","
      _19_0_2_5f801fb_1577315441742_654700_44648","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577315441742_654700_44648","smart thermostates","","Block")

// Nested part
BlockDependency("_19_0_2_5f801fb_1577333849852_239500_47011","DependOn","","
      _19_0_2_5f801fb_1577327908296_710205_46601","_19_0_2_5f801fb_1577315441742_654700_44648")
//End of nested parts

  blockparts("_19_0_2_5f801fb_1577334836918_365062_47133","PartOf","","
      _19_0_2_5f801fb_1577327823048_822183_45821","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335245329_886643_47154","PartOf","","
      _19_0_2_5f801fb_1577327884391_667651_46367","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335257478_753355_47188","PartOf","","
      _19_0_2_5f801fb_1577327843673_515599_46016","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335261319_910974_47205","PartOf","","
      _19_0_2_5f801fb_1577327893800_204695_46445","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335267469_28834_47222","PartOf","","
      _19_0_2_5f801fb_1577327863256_270017_46211","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335277482_998153_47239","PartOf","","
      _19_0_2_5f801fb_1577327908296_710205_46601","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335284310_357509_47256","PartOf","","
      _19_0_2_5f801fb_1577327837936_643658_45938","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335297633_904759_47290","PartOf","","
      _19_0_2_5f801fb_1577327817213_750406_45743","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335309998_650082_47307","PartOf","","
      _19_0_2_5f801fb_1577327811175_439055_45665","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335316373_516361_47324","PartOf","","
      _19_0_2_5f801fb_1577327851008_819294_46094","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335349408_410996_47392","PartOf","","
      _19_0_2_5f801fb_1577327879632_857017_46328","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335361014_877456_47426","PartOf","","
      _19_0_2_5f801fb_1577327840861_860881_45977","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335364686_859689_47443","PartOf","","
      _19_0_2_5f801fb_1577327897552_243103_46484","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335370740_983951_47460","PartOf","","
      _19_0_2_5f801fb_1577327904395_111499_46562","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335376262_987111_47477","PartOf","","
      _19_0_2_5f801fb_1577327900704_434303_46523","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335380754_102740_47494","PartOf","","
      _19_0_2_5f801fb_1577327867985_639474_46250","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335386194_686119_47511","PartOf","","
      _19_0_2_5f801fb_1577327846520_969016_46055","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335390537_454297_47528","PartOf","","
      _19_0_2_5f801fb_1577327875928_803538_46289","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335400334_498464_47545","PartOf","","
      _19_0_2_5f801fb_1577327813544_912565_45704","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335408754_305776_47562","PartOf","","
      _19_0_2_5f801fb_1577327889448_436692_46406","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335416798_875487_47579","PartOf","","
      _19_0_2_5f801fb_1577327835504_276967_45899","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335425950_762017_47596","PartOf","","
      _19_0_2_5f801fb_1577327855079_680504_46133","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335446885_679010_47613","PartOf","","
      _19_0_2_5f801fb_1577327859648_843376_46172","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1577335846646_946480_47630","PartOf","","
      _19_0_2_5f801fb_1577327826392_311144_45860","_19_0_1_3ed01a0_1556494897831_576536_41895")
  blockparts("_19_0_2_5f801fb_1579319628890_574125_42420","PartOf","","
      _19_0_2_5f801fb_1577327820255_551671_45782","_19_0_1_3ed01a0_1556494897831_576536_41895")
block("_19_0_2_5f801fb_1577298113692_753529_42824","Smart heater","","Block")

Blockparent("_19_0_2_5f801fb_1577727393152_778017_42826","ChildOf","","
     _19_0_2_5f801fb_1577298113692_753529_42824","_19_0_2_5f801fb_1577725403297_875277_42675")

block("_19_0_2_5f801fb_1577316008752_797941_44745","Wind turbines","","Block")


block("_19_0_2_5f801fb_1577309819323_228579_43573","Smart dishwasher","","Block")

Blockparent("_19_0_2_5f801fb_1577727277574_49140_42799","ChildOf","","
     _19_0_2_5f801fb_1577309819323_228579_43573","_19_0_2_5f801fb_1577725403297_875277_42675")

block("_19_0_2_5f801fb_1577296462612_990181_42335","Water heater controller","","Block")

blocklink("_19_0_2_5f801fb_1577309598528_550469_43499","on/off","","
     _19_0_2_5f801fb_1577296462612_990181_42335","_19_0_2_5f801fb_1577298149529_988792_42914")

block("_19_0_2_5f801fb_1577298149529_988792_42914","Smart water heater","","Block")

Blockparent("_19_0_2_5f801fb_1577727379257_797392_42822","ChildOf","","
     _19_0_2_5f801fb_1577298149529_988792_42914","_19_0_2_5f801fb_1577725403297_875277_42675")

block("_19_0_2_5f801fb_1577296335448_164770_42200","Air conditioner controller","","Block")
```

```
blocklink("_19_0_2_5f801fb_1577309684791_353682_43518","on/off","","
     _19_0_2_5f801fb_1577296335448_164770_42200","_19_0_2_5f801fb_1577298128702_302366_42869")

block("_19_0_2_5f801fb_1577298166442_772973_42959","Smart window","","Block")

Blockparent("_19_0_2_5f801fb_1577727258227_380124_42793","ChildOf","","
     _19_0_2_5f801fb_1577298166442_772973_42959","_19_0_2_5f801fb_1577725403297_875277_42675")

block("_19_0_2_5f801fb_1579314646156_228960_42246","wind sensors","","Block")


block("_19_0_1_3ed01a0_1556492760396_507721_41712","Energy controller","","Block")
requirementToBlock("_19_0_1_3ed01a0_1556507267181_725525_42309","SatisfiedBy","","
     _19_0_1_3ed01a0_1556505347654_420954_42095","_19_0_1_3ed01a0_1556492760396_507721_41712")

blocklink("_19_0_2_5f801fb_1577316213543_910649_44885","on/off","","
     _19_0_1_3ed01a0_1556492760396_507721_41712","_19_0_1_3ed01a0_1557653152129_996648_41998")
blocklink("_19_0_2_5f801fb_1577316263039_613020_44904","on/off ","","
     _19_0_1_3ed01a0_1556492760396_507721_41712","_19_0_2_5f801fb_1577316023401_419606_44790")
blocklink("_19_0_2_5f801fb_1577316289360_457636_44923","on/off","","
     _19_0_1_3ed01a0_1556492760396_507721_41712","_19_0_2_5f801fb_1577316008752_797941_44745")

block("_19_0_2_5f801fb_1577296513628_30107_42425","Windows controller","","Block")

blocklink("_19_0_2_5f801fb_1577309757920_936390_43556","on/off","","
     _19_0_2_5f801fb_1577296513628_30107_42425","_19_0_2_5f801fb_1577298166442_772973_42959")

block("_19_0_2_5f801fb_1577725403297_875277_42675","Smart device","","Block")
requirementToBlock("_19_0_2_5f801fb_1577822417974_744779_43378","SatisfiedBy","","
     _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577725403297_875277_42675")


block("_19_0_1_3ed01a0_1556488960061_314183_41664","Smart Home","","Block")

Blockaggregation("_19_0_2_5f801fb_1577308168742_851720_43096","IsSharedBy","","
     _19_0_1_3ed01a0_1557653152129_996648_41998","_19_0_1_3ed01a0_1556488960061_314183_41664")

  blockparts("_19_0_1_3ed01a0_1556555430317_702515_42319","PartOf","","
       _11_5EAPbeta_be00301_1147431819399_50461_1671","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577308175855_751190_43112","PartOf","","
       _19_0_1_3ed01a0_1556494897831_576536_41895","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577720438434_223671_42335","PartOf","","
       _19_0_2_5f801fb_1577719829903_504408_42226","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577721033570_659179_42359","PartOf","","
       _19_0_1_3ed01a0_1556492760396_507721_41712","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577721071791_482784_42375","PartOf","","
       _19_0_1_3ed01a0_1556493245441_397422_41802","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577721088374_815057_42391","PartOf","","
       _19_0_2_5f801fb_1577719967217_702345_42272","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577721132434_845668_42407","PartOf","","
       _19_0_2_5f801fb_1577316023401_419606_44790","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577721145174_724238_42423","PartOf","","
       _19_0_2_5f801fb_1577316008752_797941_44745","_19_0_1_3ed01a0_1556488960061_314183_41664")
  blockparts("_19_0_2_5f801fb_1577725426503_529976_42722","PartOf","","
       _19_0_2_5f801fb_1577725403297_875277_42675","_19_0_1_3ed01a0_1556488960061_314183_41664")
block("_19_0_2_5f801fb_1577719967217_702345_42272","Provid services","","Block")
requirementToBlock("_19_0_2_5f801fb_1577828145225_235293_43738","SatisfiedBy","","
     _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577719967217_702345_42272")

BlockDependency("_19_0_2_5f801fb_1577721238900_458493_42435","DependOn","","
     _19_0_1_3ed01a0_1556493245441_397422_41802","_19_0_2_5f801fb_1577719967217_702345_42272")

blockparts("_19_0_2_5f801fb_1577725638738_307355_42737","PartOf","","
     _19_0_2_5f801fb_1577725638738_307355_42737","_19_0_2_5f801fb_1577719967217_702345_42272")
block("_19_0_2_5f801fb_1577725638738_307355_42737","Internet","","Block")
// Nested part

// End of nested parts
blockparts("_19_0_2_5f801fb_1577725655945_87785_42741","PartOf","","
     _19_0_2_5f801fb_1577725655945_87785_42741","_19_0_2_5f801fb_1577719967217_702345_42272")
block("_19_0_2_5f801fb_1577725655945_87785_42741","WiFi","","Block")
// Nested part

// End of nested parts
blockparts("_19_0_2_5f801fb_1577725671967_294752_42744","PartOf","","
     _19_0_2_5f801fb_1577725671967_294752_42744","_19_0_2_5f801fb_1577719967217_702345_42272")
block("_19_0_2_5f801fb_1577725671967_294752_42744","Learning technique","","Block")

// Nested part
requirementToBlock("_19_0_2_5f801fb_1577821752007_965787_43256","SatisfiedBy","","
     _19_0_2_5f801fb_1577762675944_55200_42412","_19_0_2_5f801fb_1577725671967_294752_42744")
// End of nested parts

block("_19_0_2_5f801fb_1577296421097_27138_42245","Dishwasher controller","","Block")

blocklink("_19_0_2_5f801fb_1577309863374_946025_43620","on/off","","
     _19_0_2_5f801fb_1577296421097_27138_42245","_19_0_2_5f801fb_1577309819323_228579_43573")

block("_19_0_2_5f801fb_1577296486581_87921_42380","Ventilator controller","","Block")

blocklink("_19_0_2_5f801fb_1577309728202_834012_43537","on/off","","
     _19_0_2_5f801fb_1577296486581_87921_42380","_19_0_2_5f801fb_1577298253341_75561_43049")
```

```
block("_19_0_2_5f801fb_1577298128702_302366_42869","Smart air conditioner","","Block")

Blockparent("_19_0_2_5f801fb_1577727268933_170261_42796","ChildOf","","
     _19_0_2_5f801fb_1577298128702_302366_42869","_19_0_2_5f801fb_1577725403297_875277_42675")

block("_19_0_2_5f801fb_1577316023401_419606_44790","Solar panels","","Block")


block("_19_0_2_5f801fb_1577298253341_75561_43049","Smart ventilator","","Block")

Blockparent("_19_0_2_5f801fb_1577727306216_703025_42818","ChildOf","","
     _19_0_2_5f801fb_1577298253341_75561_43049","_19_0_2_5f801fb_1577725403297_875277_42675")

// ConstraintBlock
block("_19_0_2_5f801fb_1577327813544_912565_45704","User permission","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327863256_270017_46211","Heating time","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327875928_803538_46289","Enable heater access","","ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577822826709_993300_43489","SatisfiedBy","","
     _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577327875928_803538_46289")

BlockDependency("_19_0_2_5f801fb_1577333986659_921843_47035","DependOn","","
     _19_0_2_5f801fb_1577327846520_969016_46055","_19_0_2_5f801fb_1577327875928_803538_46289")
block("_19_0_2_5f801fb_1577327811175_439055_45665","Fire","","ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577812915131_505322_42722","SatisfiedBy","","
     _19_0_2_5f801fb_1577762320087_573213_42342","_19_0_2_5f801fb_1577327811175_439055_45665")

block("_19_0_2_5f801fb_1577327851008_819294_46094","No suspicious behaviour detected","","
     ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577820504378_760566_43146","SatisfiedBy","","
     _19_0_2_5f801fb_1577819029972_864756_42978","_19_0_2_5f801fb_1577327851008_819294_46094")

block("_19_0_2_5f801fb_1577327817213_750406_45743","Full","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327846520_969016_46055","Remote heating","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327879632_857017_46328","Remote water heating","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327884391_667651_46367","Water heating time","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327889448_436692_46406","Heating habits","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327893800_204695_46445","Enable air conditioner access","","
     ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577822702805_967030_43469","SatisfiedBy","","
     _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577327893800_204695_46445")

BlockDependency("_19_0_2_5f801fb_1577333878739_980133_47017","DependOn","","
     _19_0_2_5f801fb_1577327843673_515599_46016","_19_0_2_5f801fb_1577327893800_204695_46445")
block("_19_0_2_5f801fb_1577327900704_434303_46523","Enable water heater access","","ConstraintBlock"
     )
requirementToBlock("_19_0_2_5f801fb_1577822609751_793171_43459","SatisfiedBy","","
     _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577327900704_434303_46523")

BlockDependency("_19_0_2_5f801fb_1577333836834_646310_47005","DependOn","","
     _19_0_2_5f801fb_1577327879632_857017_46328","_19_0_2_5f801fb_1577327900704_434303_46523")
block("_19_0_2_5f801fb_1577327820255_551671_45782","Light","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327904395_111499_46562","Enable water heater schedule","","
     ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577822557694_370203_43429","SatisfiedBy","","
     _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577327904395_111499_46562")

BlockDependency("_19_0_2_5f801fb_1577333782953_87963_46999","DependOn","","
     _19_0_2_5f801fb_1577327884391_667651_46367","_19_0_2_5f801fb_1577327904395_111499_46562")
block("_19_0_2_5f801fb_1577327826392_311144_45860","Smoke detection","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327840861_860881_45977","Cooling time","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327908296_710205_46601","Thermostate setting","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327843673_515599_46016","Remote cooling","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327835504_276967_45899","Cooling habits","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327867985_639474_46250","Enable heater schedule","","ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577822756877_157763_43479","SatisfiedBy","","
     _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577327867985_639474_46250")

BlockDependency("_19_0_2_5f801fb_1577333954931_654999_47029","DependOn","","
     _19_0_2_5f801fb_1577327863256_270017_46211","_19_0_2_5f801fb_1577327867985_639474_46250")
block("_19_0_2_5f801fb_1577327837936_643658_45938","Wind speed","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327859648_843376_46172","Suitable weather","","ConstraintBlock")

block("_19_0_2_5f801fb_1577327823048_822183_45821","Electricity price","","ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577821573247_915559_43203","SatisfiedBy","","
     _19_0_1_3ed01a0_1557247827853_665093_42038","_19_0_2_5f801fb_1577327823048_822183_45821")

block("_19_0_2_5f801fb_1577327897552_243103_46484","Enable air conditioner schedulE","","
```

```
    ConstraintBlock")
requirementToBlock("_19_0_2_5f801fb_1577822896356_680691_43499","SatisfiedBy","","
    _19_0_2_5f801fb_1577762835986_935540_42430","_19_0_2_5f801fb_1577327897552_243103_46484")

BlockDependency("_19_0_2_5f801fb_1577333897668_607205_47023","DependOn","","
    _19_0_2_5f801fb_1577327840861_860881_45977","_19_0_2_5f801fb_1577327897552_243103_46484")
block("_19_0_2_5f801fb_1577327855079_680504_46133","Response time","","ConstraintBlock")

// Requirements
requirement("_19_0_1_3ed01a0_1556759570816_345133_41755","annual consumption management","The system
     must ensure that annual consumption must be less than local regulations.","30.1.2")
requirementlink("_19_0_1_3ed01a0_1557718341747_630474_42667","DerivedFrom","","
    _19_0_1_3ed01a0_1557247827853_665093_42038","_19_0_1_3ed01a0_1556759570816_345133_41755")

requirement("_19_0_2_5f801fb_1577819040506_475885_43021","Cyber intrusion prevention","The Smart
    Home must support prevention of unauthorized computer intrusion.","21.2.2")

requirement("_19_0_2_5f801fb_1577818860636_983262_42888","Physical intrusion prevention","The Smart
    Home must support prevention of unauthorized physical intrusion.","21.8.1")

requirement("_19_0_2_5f801fb_1577825810947_416118_43677","SmartHomeSystem","Smart home must provide
    accommodation for its inhabitants and it must be secure, safe, and economic","30")
subrequirement("_19_0_1_3ed01a0_1557691990274_110187_42455_19_0_2_5f801fb_1577825810947_416118_43677
    ","ContainedBy","","_19_0_1_3ed01a0_1557691990274_110187_42455","
    _19_0_2_5f801fb_1577825810947_416118_43677")
subrequirement("_19_0_2_5f801fb_1577717654485_241642_42189_19_0_2_5f801fb_1577825810947_416118_43677
    ","ContainedBy","","_19_0_2_5f801fb_1577717654485_241642_42189","
    _19_0_2_5f801fb_1577825810947_416118_43677")
subrequirement("_19_0_2_5f801fb_1577762584541_170356_42369_19_0_2_5f801fb_1577825810947_416118_43677
    ","ContainedBy","","_19_0_2_5f801fb_1577762584541_170356_42369","
    _19_0_2_5f801fb_1577825810947_416118_43677")

requirement("_19_0_2_5f801fb_1577762642929_263788_42403","Assistance and support","The Smart Home
    must accommodate physical, medical, and mental conditions of the inhabitants","30.3.1")
requirementlink("_19_0_2_5f801fb_1577821684351_812837_43213","DerivedFrom","","
    _19_0_2_5f801fb_1577762769175_699539_42421","_19_0_2_5f801fb_1577762642929_263788_42403")

requirement("_19_0_2_5f801fb_1577761938994_450219_42234","Cyber intrusion","The Smart Home must
    support prevention and detection of unauthorized computer intrusion.","30.2.1")
subrequirement("_19_0_2_5f801fb_1577819029972_864756_42978_19_0_2_5f801fb_1577761938994_450219_42234
    ","ContainedBy","","_19_0_2_5f801fb_1577819029972_864756_42978","
    _19_0_2_5f801fb_1577761938994_450219_42234")
subrequirement("_19_0_2_5f801fb_1577819040506_475885_43021_19_0_2_5f801fb_1577761938994_450219_42234
    ","ContainedBy","","_19_0_2_5f801fb_1577819040506_475885_43021","
    _19_0_2_5f801fb_1577761938994_450219_42234")

requirement("_19_0_2_5f801fb_1577819029972_864756_42978","Cyber intrusion detection","The Smart Home
     must support detection of unauthorized computer intrusion.","21.2.1")

requirement("_19_0_2_5f801fb_1577762769175_699539_42421","Daily activities","The Smart Home must
    assist inhabitants with certain everyday tasks","30.3.3")

requirement("_19_0_2_5f801fb_1577762675944_55200_42412","Learning","The Smart Home must learn from
    the behaviour of inhabitants.","30.3.2")

requirement("_19_0_2_5f801fb_1577762436696_581641_42351","Emergency","The Smart Home must allow
    physical access to emergency services (firemen, hospital service, etc.)","30.2.3")

requirement("_19_0_2_5f801fb_1577818913027_109047_42931","Physical intrusion detection","The Smart
    Home must support detection of unauthorized physical intrusion.","21.8.2")

requirement("_19_0_1_3ed01a0_1557691990274_110187_42455","Economy","The system must save energy.","
    30.1")
subrequirement("_19_0_1_3ed01a0_1556505449360_339815_42138_19_0_1_3ed01a0_1557691990274_110187_42455
    ","ContainedBy","","_19_0_1_3ed01a0_1556505449360_339815_42138","
    _19_0_1_3ed01a0_1557691990274_110187_42455")
subrequirement("_19_0_1_3ed01a0_1556759570816_345133_41755_19_0_1_3ed01a0_1557691990274_110187_42455
    ","ContainedBy","","_19_0_1_3ed01a0_1556759570816_345133_41755","
    _19_0_1_3ed01a0_1557691990274_110187_42455")

requirement("_19_0_1_3ed01a0_1556505449360_339815_42138","energy saving","The Smart Home must be
    energy efficient","30.1.1")
requirementlink("_19_0_1_3ed01a0_1557718336094_38254_42653","DerivedFrom","","
    _19_0_1_3ed01a0_1556505347654_420954_42095","_19_0_1_3ed01a0_1556505449360_339815_42138")

requirement("_19_0_2_5f801fb_1577762320087_573213_42342","Fire detection","The Smart Home must be
    able to detect signs of fire","30.2.2")

requirement("_19_0_2_5f801fb_1577762584541_170356_42369","Accommodation","The system must provide
    accommodation services to its inhabitants","30.3")
subrequirement("_19_0_2_5f801fb_1577762642929_263788_42403_19_0_2_5f801fb_1577762584541_170356_42369
    ","ContainedBy","","_19_0_2_5f801fb_1577762642929_263788_42403","
    _19_0_2_5f801fb_1577762584541_170356_42369")
subrequirement("_19_0_2_5f801fb_1577762675944_55200_42412_19_0_2_5f801fb_1577762584541_170356_42369"
    ,"ContainedBy","","_19_0_2_5f801fb_1577762675944_55200_42412","
    _19_0_2_5f801fb_1577762584541_170356_42369")
subrequirement("_19_0_2_5f801fb_1577762769175_699539_42421_19_0_2_5f801fb_1577762584541_170356_42369
    ","ContainedBy","","_19_0_2_5f801fb_1577762769175_699539_42421","
    _19_0_2_5f801fb_1577762584541_170356_42369")
subrequirement("_19_0_2_5f801fb_1577762835986_935540_42430_19_0_2_5f801fb_1577762584541_170356_42369
    ","ContainedBy","","_19_0_2_5f801fb_1577762835986_935540_42430","
    _19_0_2_5f801fb_1577762584541_170356_42369")
```

```
requirement("_19_0_1_3ed01a0_1557247827853_665093_42038","Appliances management","the system shall
    automatically schedule the operation of appliances with high energy needs to 'off-peak' times
    with lower electricity rates","10")

requirement("_19_0_2_5f801fb_1577717654485_241642_42189","SecurityAndSafety","The system must be
    safe and secure","30.2")
subrequirement("_19_0_2_5f801fb_1577761938994_450219_42234_19_0_2_5f801fb_1577717654485_241642_42189
    ","ContainedBy","","_19_0_2_5f801fb_1577761938994_450219_42234","
    _19_0_2_5f801fb_1577717654485_241642_42189")
subrequirement("_19_0_2_5f801fb_1577762320087_573213_42342_19_0_2_5f801fb_1577717654485_241642_42189
    ","ContainedBy","","_19_0_2_5f801fb_1577762320087_573213_42342","
    _19_0_2_5f801fb_1577717654485_241642_42189")
subrequirement("_19_0_2_5f801fb_1577762436696_581641_42351_19_0_2_5f801fb_1577717654485_241642_42189
    ","ContainedBy","","_19_0_2_5f801fb_1577762436696_581641_42351","
    _19_0_2_5f801fb_1577717654485_241642_42189")
subrequirement("_19_0_2_5f801fb_1577761862440_951449_42191_19_0_2_5f801fb_1577717654485_241642_42189
    ","ContainedBy","","_19_0_2_5f801fb_1577761862440_951449_42191","
    _19_0_2_5f801fb_1577717654485_241642_42189")

requirement("_19_0_2_5f801fb_1577762835986_935540_42430","User preferences","The Smart Home must
    accommodate specific preferences entered by the inhabitants","30.3.4")

requirement("_19_0_1_3ed01a0_1556505347654_420954_42095"," self energy priority management ","The
    system must prioritize self-produced energy","ECH03030")

requirement("_19_0_2_5f801fb_1577761862440_951449_42191","Physical intrusion","The Smart Home must
    support prevention and detection of unauthorized physical intrusion.","30.2.5")
subrequirement("_19_0_2_5f801fb_1577818860636_983262_42888_19_0_2_5f801fb_1577761862440_951449_42191
    ","ContainedBy","","_19_0_2_5f801fb_1577818860636_983262_42888","
    _19_0_2_5f801fb_1577761862440_951449_42191")
subrequirement("_19_0_2_5f801fb_1577818913027_109047_42931_19_0_2_5f801fb_1577761862440_951449_42191
    ","ContainedBy","","_19_0_2_5f801fb_1577818913027_109047_42931","
    _19_0_2_5f801fb_1577761862440_951449_42191")

endImport
```

C.3 Consistency and Completeness Rules in DXL

Listing C.3 provides the DXL code of the consistency and completeness rules defined in Section 4.2.3 and checked against the SHMS modules in the DOORS database in Section 8.2.2.

The DXL function checkModule in Listing C.3 goes through each object in the identified module (parameter 1, which can be blocks, requirements, intentional elements, or actors module) and checks the out/in links based on the TIM defined in Fig. 4.4. This function searches for the desired links of specific linksets (parameters 2 and 3) and the related modules (parameters 4 and 5) according to the rules identified in Section 4.2.3. The function checkModule calls two recursive function, inL and out, to search for the in and out links in the identified module via the specified linkset. In these methods, if the current object has no link to/from the specified module via the specified linkset, the findparentff function is called to check if the parents have the desired links (Rule 7). Also, the intra-model links imported from the original models are exploited by checkModule to cover implicit links and track the parents' chain of objects via the findparentff method. Finally, if the inL and/or out methods did not return links, a consistency and completeness problem is recorded in the corresponding column of DOORS module being checked.

Listing C.3: Consistency and completeness checks in DXL

```
pragma runLim, 0

// Generic function to find a DOORS object based on its model identifier
```

```
Object findObject( string ID, Module theModule ) {
  Object parentObject=null

  for parentObject in theModule do {
    if ( parentObject."ID" "" == ID ) {
      return parentObject
    }
  }
  return null
}

// Search link loop
bool ContainedReq(Object obj, string t2){
  Link l = null
  ModName_ otherMod = null
  Object otherol=null
  for l in all (obj -> "*") do
   {
    ModuleVersion otherVersion = targetVersion(l)
    otherMod = module(otherVersion)
    if (null otherMod || isDeleted otherMod) continue
    if (name(otherMod) != t2) return false
   }
    return true
}

// Search for oulink has specific linkset (ts)
Object outLink(Object obj, string t2){
  Link l = null
  ModName_ otherMod = null
  Object otherol=null
  for l in all (obj -> "*") do
   {
    ModuleVersion otherVersion = targetVersion(l)
    otherMod = module(otherVersion)
    if (null otherMod || isDeleted otherMod) continue
    if (name(otherMod) != t2) continue
    otherol = target l
    if (null otherol)
    {
        load(otherVersion,false)
    }
    otherol = target l
    if (null otherol) continue
    if (isDeleted otherol) continue
    delete(otherVersion)
    return otherol
   }
  return null

 }

// This function returns the parent of a specific object
Object findparentff( Object obC ,  Module currentModule ) {
```

```
    Object pO=null
    Module pM=null
    string sM
   if (name currentModule "" == "Blocks") {
     sM = "/SysMLSmartHomeModel/Block Relationships"
     pM = read(sM, false)
     for pO in pM do {
       if ( pO."Name_" "" != "PartOf")  continue
        if (pO."Source ID" "" == obC."id" "" ) {

          return findObject(pO."Destination ID" "" , currentModule)
                }
        }
       }

   if (name currentModule "" == "Intentional Elements") {
     sM = "/URNSmartHome/Intentional Element Associations"
     pM = read(sM, false)
       for pO in pM do {

       if (pO."Type" "" != "decomposition") continue


        if (pO."Source ID" "" == obC."id" "" ) {

          return findObject(pO."Destination ID" "" , currentModule)
                }
        }
       }

    if (name currentModule "" == "Requirements") {
      sM = "/SysMLSmartHomeModel/Requirements Relationships"
      Object tmp=null
      pM = read(sM, false)
       for pO in pM do {

       if (pO."Name_" "" != "ContainedBy") continue


        if (pO."Source ID" "" == obC."id" "" ) {
            if (ContainedReq(pO, "Requirements")) return findObject(pO."
               Destination ID" "" , currentModule)
                }
        }
       }

       return null
}


// Search of links from the module to another module
Object out(Object obj, string LM, string t2, Module currentModule){
  Object other=null
```

```
  if (null obj) {
    return null
   }

  other = outLink(obj,t2)
  if (! null other) return other
  return out( findparentff( obj,  currentModule), LM, t2, currentModule)
 }


// Find incoming links
Object findInLinks(Object obj, string LM, string t3){
 Link l = null
 ModName_ otherMod = null
 Object othero=null
 LinkRef lr = null
 for lr in all(obj<-"*") do
 {
  if (name(module lr) != LM) continue
        // print LM " After if !=" name(module lr) "\n"
  ModuleVersion mvSource = sourceVersion(lr)
  otherMod = module(mvSource)
        if (name(otherMod) != t3) continue
  if (!null otherMod)
  {
    if ((!isDeleted otherMod) && (null data(mvSource)))
    {
      load((mvSource), false)
    }
  }
  delete(mvSource)
  }

 for l in all(obj<-"*") do {
    if (name(module l) != LM) continue
  ModuleVersion otherVersion = sourceVersion(l)
  otherMod = module(otherVersion)
  if (null otherMod || isDeleted otherMod) continue
      if (name(otherMod) != t3) continue
  othero = source(l)
  if (null othero)
  {
    load(otherVersion, false)
  }
  othero = source(l)
  if (null othero) continue
  if (isDeleted othero) continue
       return othero
    }
return null
}

// Search for links coming to the module
Object inL(Object obj, string LM, string t3,  Module currentModule){
```

```
 Object othero=null
 if (null obj) {
 return null
 }
 othero= findInLinks(obj, LM, t3)
 if (! null othero) return othero
 return inL( findparentff( obj,  currentModule), LM, t3, currentModule)
 }

// Assign default values to the analysis variables
void SetVarAll(Object objM,string ss, string valb){
   if (ss=="Blocks") {objM."Requirements"=valb
      objM."Intentional Elements"=valb}
   if (ss=="Requirements") {objM."Blocks"=valb
      objM."Intentional Elements"=valb}
   if (ss=="Intentional Elements") {objM."Blocks"=valb
     objM."Requirements"=valb}
   if (ss=="Actors") {objM."Blocks"=valb}
}

// Update the object by the id of the linked objects,
void updateTraceVar(Object upObject, string upChoice, string id){
  if (upChoice == "Requirements")  {upObject."Requirements"= id}
                    else { if (upChoice == "Intentional Elements")  {
                       upObject."Intentional Elements"= id}
                          else  {upObject."Blocks"= id}
                        }
}

// Check all the in and out links
void checkModule(Module currentModule, string LinkName1,string LinkName2
    , string t2 , string t3, Module Module2) {
  Object goal = null
  Object req = null
  Object Block=null
  bool x= false
  string ss=""
  Object objM= null
  ss= currentModule."name"
  bool updated

 for objM in currentModule do {
   x= objM."ignore"
   if (x  ) continue
   // Search for out links
   if (t2 !="") {
      req=null
      updated=false
    // Search in the imported relationships
      if ( ss == "Blocks") {
        if ( t2 == "Requirements") {
            req=inL(objM, LinkName1, "Requirements Relationships",
               currentModule)
      if ( ! null req) {
```

```
        updateTraceVar(objM,  t2, req."Source ID" "")
        updated=true
                              }
                    }
                }
  if ( null req){req=out(objM,LinkName1 ,t2, currentModule)}
  if ( null req){
      print name currentModule " " objM."Name_" " is not linked to " t2 "
          \n"
      objM."ConsistencyAndCompleteness" = objM."
          ConsistencyAndCompleteness" "not linked to" t2 "\n"
      }
  else {if (updated != true) {updateTraceVar(objM,  t2, req."ID" "")}}

 }

// Search for in links
 if (t3 !="") {
   updated=false
   goal=null
   if ( ss == "Requirements") {
        if ( t3 == "Blocks") {
            goal=inL(objM, LinkName2, "Requirements Relationships",
                currentModule)
        if ( ! null goal) {
                updateTraceVar(objM,  t3,goal."Destination ID" "")
        updated=true
                            }
                    }
                }
   if ( null goal) {goal=inL(objM,LinkName2, t3, currentModule)}
   if ( null goal) {
        print objM."Name_" " is not linked to " t3 "\n"
        objM."ConsistencyAndCompleteness" = objM."
            ConsistencyAndCompleteness" "not linked to" t3 "\n"
                    }
    else {if (updated != true) updateTraceVar(objM,  t3,goal."ID" "")}
       }
    }
 }

// Clear the analysis variables before starting
void clearModule(Module currentModule){
  Object cObj
  for cObj in currentModule do {
   cObj."ConsistencyAndCompleteness" = ""
   cObj."Implicit Link"=""
   SetVarAll(cObj,name(currentModule) "","")
   }
}

// This function brings the missed link between three linked objects, if
    any. For example, if we have a block linked to a requirement but not
    to intentional elements in a block module, the function returns the id
```

```
    of the intentional element linked to this requirement from the
    requirements module.
string bringObject(Object objM,string ss, Module ModelB, Module ModelR,
    Module ModelI){
  Object relatedO=null
  if (ss=="Blocks") {if (objM."Requirements" "" !="" ) {
    if ( objM."Intentional Elements" "" =="") {relatedO=findObject(objM."
        Requirements" "", ModelR)
                                              return relatedO."Intentional
                                                  Elements" "" }}
  if ( objM."Requirements" "" =="" ) {
    if ( objM."Intentional Elements" "" !="" ) {relatedO=findObject(objM.
        "Intentional Elements" "", ModelI)
                                              return relatedO."Requirements
                                                  " ""}}
   }
  if (ss=="Requirements") {
    if ( objM."Blocks" "" !="" ) {
       if ( objM."Intentional Elements" "" =="") { relatedO=findObject(
          objM."Blocks" "", ModelB)
                                                 return relatedO."
                                                     Intentional Elements" "
                                                     "}}
    if ( objM."Blocks" "" =="" ) {
       if ( objM."Intentional Elements" "" !="") {relatedO=findObject(
          objM."Intentional Elements" "", ModelI)
                                              return relatedO."Blocks" ""}}
  }
   if (ss=="Intentional Elements") {
     if (objM."Blocks" "" !="" ) {
       if ( objM."Requirements" ""=="") { relatedO=findObject(objM."
          Blocks" "", ModelB)
                                         return relatedO."Requirements" ""
                                            }}
     if ( objM."Blocks" "" =="") {
       if (objM."Requirements" "" !="") {relatedO=findObject(objM."
          Requirements" "", ModelR)
                                          return relatedO."Blocks" "" }}
    }
   return ""
 }

// Search for implicit links
void checkImplicitLink(Module CurrentModule, Module BlockModule,Module
    RequirementsModule,Module IntentionalModule) {
  Object obj=null
  string implicitL=""
  for obj in CurrentModule do {
    implicitL=bringObject(obj,name CurrentModule "", BlockModule,
       RequirementsModule,IntentionalModule)
    if (implicitL == "") continue
       obj."Implicit Link"="id: " implicitL
  }
}
```

```
// Main program
Module currentModule, BlockModule, RequirementsModule, IntentionalModule,
    ActorModule

string m = "/SysMLSmartHomeModel/Blocks"
BlockModule = edit(m, false)
clearModule(BlockModule)
m = "/SysMLSmartHomeModel/Requirements"
RequirementsModule = edit(m, false)
clearModule(RequirementsModule)
m = "/URNSmartHome/Intentional Elements"
IntentionalModule = edit(m, false)
clearModule(IntentionalModule)
m = "/URNSmartHome/Actors"
ActorModule = edit(m, false)
clearModule(ActorModule)
print "Module: " name BlockModule "\n"

// Check Rules number 2, 5 and 7. Each block/its parents shall satisfy
    requirement/s and intentional element/s
checkModule(BlockModule, "Satisfy","Satisfy", "Requirements", "
    Intentional Elements", RequirementsModule)
// Check Rules number 4, 6 and 7. Each Requirement/its parents shall
    trace intentional element/s and satisfied by block/s
print "Module: "  name RequirementsModule "\n\n"
checkModule(RequirementsModule,"Traced By","Satisfy", "Intentional
    Elements","Blocks",BlockModule)
// Check Rules number 1, 3 and 7. Each intentional element/its parents
    shall be traced to requirement/s and satisfied by block/s
print name IntentionalModule "\n\n"
checkModule(IntentionalModule,"Satisfy","Traced By","Blocks","
    Requirements",BlockModule)
// Check Rules number 1, Actors shall be satisfied by Block/s
print name ActorModule "\n\n"
checkModule(ActorModule,"","Satisfy","","Blocks", BlockModule)
// Implicitly check
print "implicitly check"
checkImplicitLink(BlockModule,BlockModule,RequirementsModule,
    IntentionalModule)
checkImplicitLink(RequirementsModule,BlockModule,RequirementsModule,
    IntentionalModule)
checkImplicitLink(IntentionalModule,BlockModule,RequirementsModule,
    IntentionalModule)

save BlockModule
close( BlockModule, false )
save RequirementsModule
close( RequirementsModule, false )
save IntentionalModule
close( IntentionalModule, false )
save ActorModule
close( ActorModule, false )
```

Appendix D

Optimization Modeling Tips

This appendix provides some tips that can help modelers get important benefits in using the optimization modeling method of CGS4Adaptation.These tips result from the experience gained in the illustrative example in Section 6.2.5, in the case study in Section 7.6.2, and in the performance evaluation of Chapter 9).

1. Distribute the constraints between GRL and feature models when modeling. For instance, model the conflict between features in the feature model using Xor relationships, or Exclude/Include constraints as URN links. As these links reduce the complexity of GRL function (this GRL function is the objective function, and reducing its complexity and size facilitates its usage as well as enhances the overall system performance, see Chapter 9), their formulas also can be used at runtime to solve conflicts caused by unforeseen contexts.

2. When modeling Exclude constraints, start the URNLink from the less important features, so that in some unforeseen contexts, the system disables these features when the more important features become selected (100) instead of returning unresolved or low-quality solutions. However, remember that getting no solution also helps detect unforeseen contexts and enables providing suitable solutions during the modeling stage. In this case, do not use formulas of leaf features to validate the selected features by indicators before optimizing the model.

3. For higher control of a specific situation (such as fire, smoke, and physical attack detection in the smart home case study), use *positive* contribution links to activate desired features, and *negative* contributions to deactivate unwanted features when the relationships are not between features only (the result of indicators make the situation and there is no conflict between the related features such as in case of fire, open the windows and switch the electricity off). Then, let the feature function inform you whether the model provides a valid solution for each context or whether it contains conflicting solutions.

4. To validate a solution in any situation according to different contexts (such as fire and smoke detection, low power, low wind speed, cyber or physical threats):

(a) Use the feature function as an objective function of the optimization model (with GRL formulas of the features associated with indicators), which should return optimal solutions equal to 100. You can test different conditions to check that you have a well-designed model that contains at least one valid solution for each expected context.

(b) Use GRL functions as objective functions and feature functions as constraints to determine optimal solutions. Then, you can improve your solutions by updating the model to increase overall system quality in this context.

5. Use the GRL formula of a *specific goal or actor* as an objective function and feature function as a constraint together with GRL functions of the features associated with indicators to determine the optimal solutions for the selected model element and evaluate its satisfaction level within a specific context.