e-Informatica Software Engineering Journal, Volume 16, Issue 1, 2022, pages: 220101, DOI 10.37190/e-Inf220101

Self-Adaptation Driven by SysML and Goal Models – A Literature Review

Amal Ahmed Andaa*, Daniel Amyot*
*School of Electrical Engineering and Computer Science, University of Ottawa
aanda027@uottawa.ca, damyot@uottawa.ca

Abstract

Background: *Socio-cyber-physical systems* (SCPSs) are a type of cyber-physical systems with social concerns. Many SCPSs, such as smart homes, must be able to adapt to reach an optimal symbiosis with users and their contexts. The Systems Modeling Language (SysML) is frequently used to specify ordinary CPSs, whereas goal modeling is a requirements engineering approach used to describe and reason about social concerns.
Objective: This paper aims to assess existing modeling techniques that support adaptation in SCPSs, and in particular those that integrate SysML with goal modeling.
Method: A systematic literature review presents the main contributions of 52 English articles selected from five databases that use both SysML and goal models (17 techniques), SysML models only (11 techniques), or goal models only (8 techniques) for analysis and self-adaptation.
Result: Existing techniques have provided increasingly better modeling support for adaptation in a SCPS context, but overall analysis support remains weak. The techniques that combine SysML and goal modeling offer interesting benefits by tracing goals to SysML (requirements) diagrams and influencing the generation of predefined adaptation strategies for expected contexts, but few target adaptation explicitly and most still suffer from a partial coverage of important goal modeling concepts and of traceability management issues.

Keywords: adaptation, cyber-physical systems, goal modeling, socio-technical systems, SysML, traceability, uncertainty

1. Introduction

Cyber-physical systems (CPSs) are systems that tightly "integrate physical, software, and network aspects in a sometimes adverse physical environment" [1]. They are composed of hybrid components such as hardware (e.g., sensors, devices, and networks) and software, which can even be integrated at runtime. Horváth [2] 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 [2, 3].

Socio-cyber-physical systems (SCPSs) are a type of CPSs that are also socio-technical systems, 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

© 2022 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY license international.
Submitted: 26 Apr. 2021; Revised: 11 Aug. 2021; Accepted: 26 Sep. 2021; Available online: 15 Dec. 2021

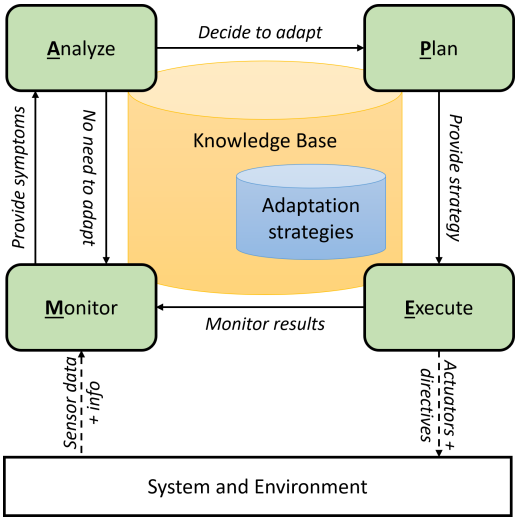

Figure 1. Self-adaptation activities: MAPE-cycle (adapted from [8])

to adapt to changing conditions in order to reach an optimal symbiosis with users (and other stakeholders) and their contexts [2]. Examples include existing systems such as air traffic control systems, and emerging ones such as smart homes/cities [4], human-oriented services exploiting the Internet of Things (IoT) [5], adaptive Systems of Systems (SoS) [6], and Industry 4.0 [7].

Many CPSs and SCPSs monitor their environments, which enables them to detect contexts where the system 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 behavior accordingly. Kephart and Chess [8] divided the adaptation process into four different activities (Monitor, Analyze, Plan, and Execute), collectively called *MAPE-cycle* and illustrated in Figure 1. These activities share some knowledge and interact with the rest of the system and its environment. The general functionality of each activity is as follows:

- *Monitor*: Gathers information about monitored system features and the environmental context.
- *Analyze*: Analyzes the information and determines whether to activate the planning process and what information should be passed on to it.
- *Plan*: Selects the most suitable adaptation strategy (some might be predefined) depending on the information provided by the analysis activity.
- *Execute*: Executes the selected adaptation strategy, with impact on the system and the environment that again must be monitored.

Some challenges coming with adaptive systems were identified and addressed by Bocanegra et al. [9], Muñoz-Fernández et al. [10], and Horváth [2] with Model-Driven Engineering (MDE) approaches. In particular, *goal modeling*, which enables the description of stakeholder and system goals together with their relationships, is used as part of many MDE approaches to facilitate traceability, deal with uncertainty, manage stakeholder objectives, and support requirements engineering at design time and at runtime. Bocanegra et al. [9] further stated that integrating MDE and goal-oriented requirements engineering is a promising way to solve many self-adaptation challenges. In addition, Muñoz-Fernández et al. suggest that traceability supports reasoning about system behavior and the changes or events that triggered a specific adaptation at runtime [10]. Although traceability in SCPSs was

identified by Bordeleau et al. as a challenge [11], ideally, for self-adaptive systems to be realizable, traceability should be managed synchronously from the beginning (goals) to the end of the system (code). Even if Bocanegra et al. cited the lack or loss of information while transforming (goal) models to code as one weakness of many MDE approaches [9], the benefits of goal modeling in this context tend to outweigh its drawbacks.

MDE is the basis of many Systems Engineering (SE) methods meant to deal with complex, technological obstacles and the heterogeneous nature of multidisciplinary systems [1, 12, 13]. One opportunity here is to include stakeholder goals into targeted systems via modeling. In the same context, the *Systems Modeling Language* (SysML) is a language standardized by the Object Management Group (OMG) [14, 15] and the International Organization for Standardization [16] to support SE methods. SysML reuses part of the Unified Modeling Language (UML), including use case, sequence, activity, and state machine diagrams, and modifies other types of diagrams to produce block and internal block diagrams. Moreover, SysML adds parametric and requirements diagrams to facilitate the connection between system components and their requirements [15]. SysML enables the modeling of software and hardware components as well as their relationships, in a way that simplifies their design [17] and reduces their complexity [18, 19].

SysML modelers can connect requirements to other model 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 [20]. Cross-disciplinary model fusion and flexible model integration are known to be challenging [21], but a multi-level modeling approach is still a promising avenue in contexts such as Systems of Systems and adaptive SCPS [1, 22, 23]. There exist many goal modeling languages that can help here, including KAOS [24, 25], *i** [26], and the Goal-oriented Requirement Language (GRL) [27], part of the User Requirements Notation (URN) [28]. GRL is discussed further here because this is the only internationally standardized goal modeling language so far and one of the few languages that supports indicators, which enable monitoring in an adaptive context. GRL has also been used in the modeling and design of large CPSs, some with a social aspect but without adaptation (e.g., for collaborative CPSs [29]), some that adapt but without a social aspect (e.g., for unmanned aircraft systems [30]), and some that model adaptive SCPSs (e.g., for smart homes [31]).

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 illustrates a GRL model of a simplified hybrid car's engine system and its related user's goals. The system needs to select which engine(s) to activate so that speed and distance from other cars are properly controlled while ensuring that the user's concerns (comfortable driving, measured via a vibration indicator, and costs minimized) are satisfied. GRL *actors* (illustrated as ellipses ⬭) are used to capture 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 convert this information into satisfaction levels. Intentions can also be AND/OR-decomposed (⟶), whereas an arrow (⟶) with a negative/positive weight (normalized to a value 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 0 and +100) indicate the current level

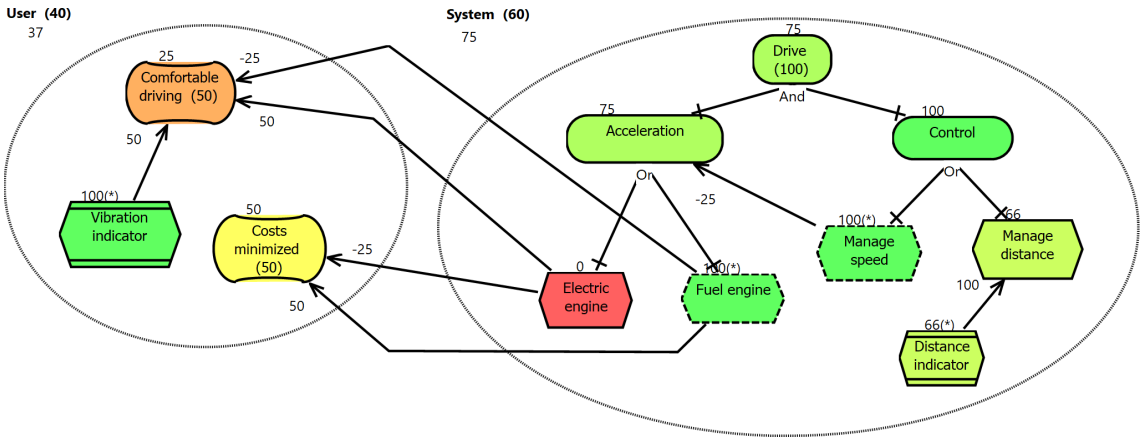

Figure 2. Simplified goal model (in GRL) of a hybrid car's engine

of satisfaction in a given context called a *strategy*, whose initial values (*) are propagated to the other elements and actors based on an automated propagation algorithm [32].

The other goal modeling languages also support distinctive goal and quality concepts, together with AND/OR decomposition, and assignment of goals to actors (or "agents" in other languages). For example, Figure 3 briefly highlights the syntax of the popular KAOS language [24, 25], also used in several adaptive SCPS approaches. Unlike GRL, KAOS neither supports indicators nor contributions, but it includes explicit concepts for *obstacles* and *threats*, akin to goals or tasks that would be the source of negative contributions in GRL.

GRL was initially created to support requirements engineering activities during development; however, it can also be used in a runtime adaptation context [33]. 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 [20].

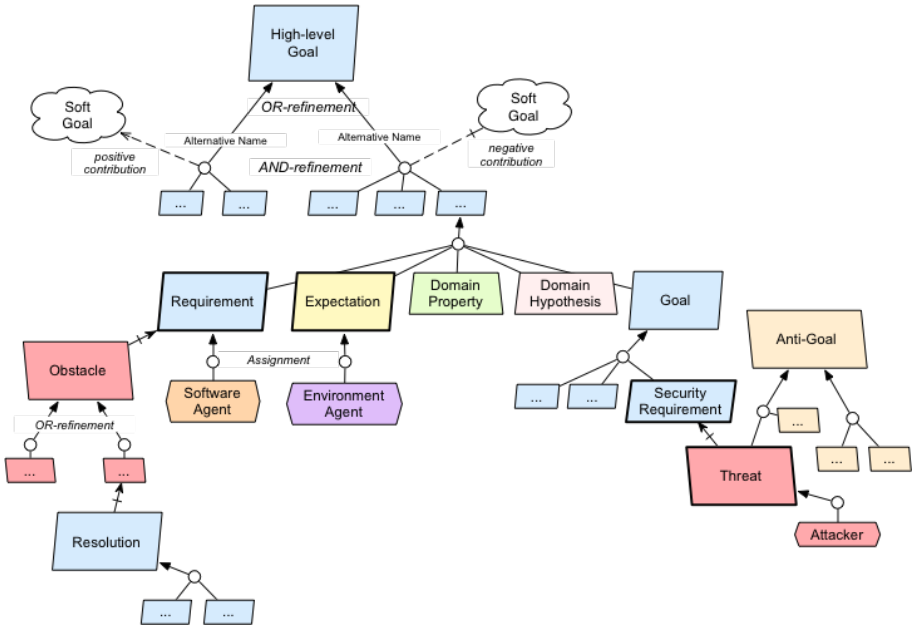

Figure 3. KAOS goal modeling concepts and syntax (from `https://kaos.info.ucl.ac.be/`)

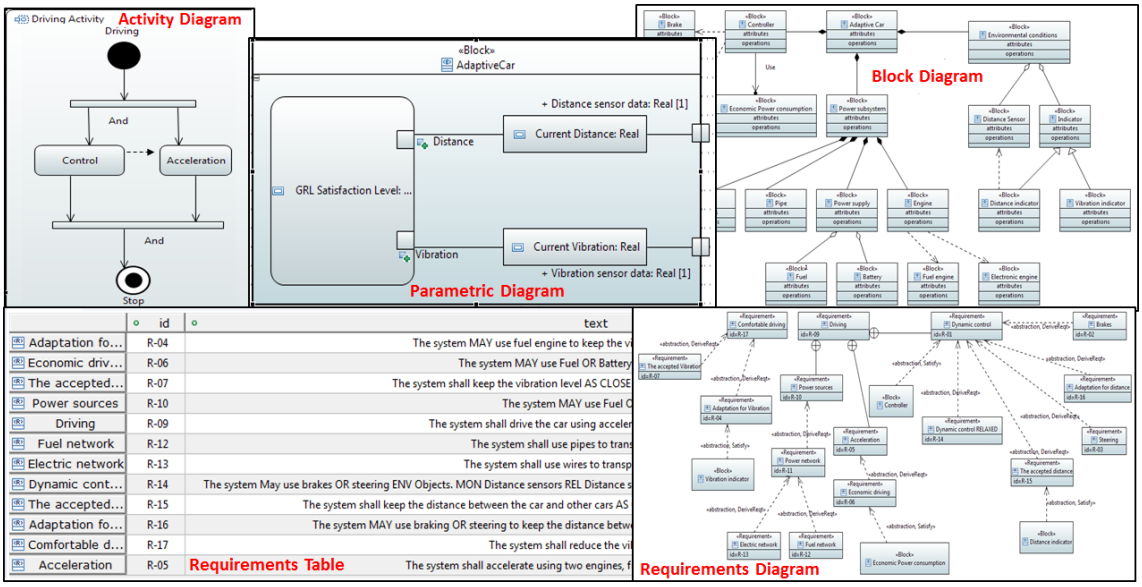

Figure 4. Some SysML diagrams of a simplified hybrid car's engine

To briefly illustrate SysML (more commonly known than goal modeling languages), Figure 4 highlights important types of diagrams and tables for adaptive systems, with content relevant to the simplified hybrid car's engine example. A requirements table contains natural language requirements (functional and non-functional) with their attributes. This information, together with traceability and other relationships (e.g., derivation or satisfaction), can be visualized in a requirements diagram. A block definition diagram shows the system components (software and hardware) as a static structure of blocks with their attributes, operations, constraints, interfaces, and relationships (e.g., containment or generalization). Parametric diagrams describe mathematical functions for constraint blocks. Activity diagrams present the dynamic flow of activities that describe behavioral aspects. SysML also supports other diagrammatic views for use cases, sequences, state machines, internal blocks, and packages.

In order to adjust requirements engineering and modeling techniques for modeling SCPSs, and especially adaptive ones, it is necessary to consider the integration between SysML (which supports system specification) and a goal modeling language that supports social concepts and comprehensive decision making [3, 20, 34, 35]. The objective of this paper is to survey and analyze articles that relate to goal and SysML modeling and their support for self-adaptive systems, especially SCPSs. The focus is on the techniques that combine *both* SysML and goal modeling in that context (and this literature review is quite exhaustive in that regard), but some key techniques that only use SysML or goal modeling are also discussed in order to enable useful comparisons.

The main contributions of this literature review are its consolidation of existing work and an outline of achievements and challenges related to existing techniques. This review is also original in that although there exist general literature surveys and mappings about adaptive systems [36–38], CPS modeling [39], goal modeling [40], and SysML [41, 42], none really goes deep into the combination of goal and SysML modeling for emerging types of adaptive socio-cyber-physical systems. This review will help academics understand what contributions and gaps exist in that research area. It will also raise the awareness of practitioners in the existing techniques for self-adaptation based on SysML and/or goal modeling, while providing guidance in the selection of appropriate techniques in their development context.

This paper is structured as follows. Section 2 describes the methodology used to plan, conduct, and report on the literature review. Section 3 provides the selected resources on Goal/SysML integration, as well as their evaluation methods, concepts, and objectives. In addition, the methods presented for self-adaptation are classified into different categories, and the self-adaptation concepts and dimensions are evaluated and extracted. Results are compared in order to provide insight into their aims, achievements, and challenges. Related literature reviews are discussed in Section 4. The limitations and threats to the validity of this review are explained in Section 5. Finally, Section 6 concludes the paper.

2. Methodology

Based on the systematic literature review approach of Kitchenham and Charters [43], we followed three common steps: planning, conducting, and reporting on the review.

2.1. 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.

2.1.1. Setting the study goal and research questions

SCPSs combine stakeholder goals, software, and hardware components. Some SCPSs may also be self-adaptive. In this context, the objective of this review 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?

2.1.2. 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 quasi-synonyms in our context. The automatic search was conducted in two phases, first with a focus on SysML/goal integration and second on SysML models and self-adaptation. Table 1 specifies each search conducted with the related query. Because Google Scholar retrieved thousands of papers (with many false positives), we eliminated adaptation/social terms from the second query to ensure papers only integrating goal and SysML models would be included in our dataset, as 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 (which retrieves thousands of papers with many false positives), 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. The number 60 was selected based on observing an increasingly high density of false positives as we went down the lists of results, especially after 40 results. The return on the time investment after 60 results was deemed ineffective.

Table 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"

Inclusion and Exclusion Criteria. We used inclusion and exclusion criteria to select which papers to keep. The inclusion criteria were:

1. The article is peer reviewed (no book, patent, tutorial, magazine, or gray literature).
2. The article is written in English.
3. For queries 1 and 2 in Table 1, the article provides or clarifies methods about Goal/SysML integration.
4. For queries 3 and 4 in Table 1, the article includes methods using SysML for self-adaptation support.

The exclusion criteria were:

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.

A paper satisfying one of the exclusion criteria or not satisfying all of the inclusion criteria was excluded. Some papers did discuss a combination of goal modeling with SysML modeling, but not their integration or self-adaptation. For example, Tueno Fotso et al. [44] integrate KAOS-like AND/OR goal models with a subset of SysML for the generation of Event-B specifications, but not for adaptive systems.

2.1.3. Quality assessment criteria

We used the checklist in Table 2 to provide a qualitative assessment of each study.

Table 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

2.1.4. Identifying data extraction items

Table 3 details the data items extracted from each selected paper, together with their related research questions from the planning stage.

Table 3. Data extracted from each paper

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

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

1. Are the goals integrated with SysML as a model, or as text/requirement?
2. What are the diagram types that were used in the integration?
3. Are common goal modeling concepts considered in the integration? These include goal types, qualitative/quantitative contributions between different types of goals, and goal dependencies.
4. Is goal analysis considered in the integration?
5. What is the purpose of the integration?
6. Is the integration fully automatic, semi-automatic, or manual?
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 different perspectives?
        i. When are the alternatives handled? (*closed*: at development phase; or *open*: at runtime).
        ii. 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 chooses between alternatives (analysis and selection processes) [37]. 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 [3]. 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 behaviors 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. (Cost-impact)
12. When does the adaptation happen before specific events (Proactive) or after specific events (Reactive)? (Temporal adaptation)
13. Does the system monitor specific features or does it monitor its environment continually using sensors? (Temporal monitoring)
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? (Evolution)

   b) Do system goals remain unchanged, change within constraints, or change without constraints? (Flexibility)
   c) How many goals are considered in the adaptation process? (Multiplicity)
   d) Are the goals dependent or independent of each other? (Dependency)
17. What is the reason for the adaptation? (Change)
   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? (Timeliness)
19. What was the reason for using a SysML model to specify self-adaptive systems?

2.2. Conducting the review

After having identified the queries and databases engines, the study was conducted along four steps: search, screening, data extraction, and quality assessment.

2.2.1. Search methods

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 irrelevant articles, we used the following strategy on this engine:
   a) The first query was applied to retrieve papers first using the "anywhere" option and second using the "in the title" option. The latter retrieved only 10 papers while the former retrieved 4,200 papers. We considered only the first 60 papers (which are ranked by relevance).
   b) To include further relevant papers, we conducted the second query using the "anywhere" option. This returned 660 papers, and again only the first 60 papers were considered.

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, "anywhere" and "in the title" separately. The first option led to 3,860 papers, and only the first 60 papers were included in our dataset. The second 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 (snowballing).

2.2.2. Screening

The papers retrieved through the previous step were screened for relevance. The exclusion and inclusion criteria were applied on each paper based on abstracts and conclusions. If the information was insufficient to decide whether a paper was relevant or not, its introduction

and method sections were read. If the information was still insufficient to decide, the full-text of the article also was read. Discussions between both authors were required for nine papers because it was difficult to decide whether they were relevant or not.

2.2.3. Result

Figure 5 illustrates the results of the screening process, with the numbers of results returned for each query available in Table 4. From the goals/SysML search, out of 444 papers returned by the search engines, 33 papers were deemed relevant whereas 411 were rejected, including 58 duplicates[1]. For the SysML and adaptation search, 334 papers (including 18 duplicates) were considered but only eleven articles met our criteria. Most of the papers were found by Scopus, with much duplication by the Web of Science, and the other engines added little value. In addition to these 44 papers, eight papers on goal modeling for adaptive systems

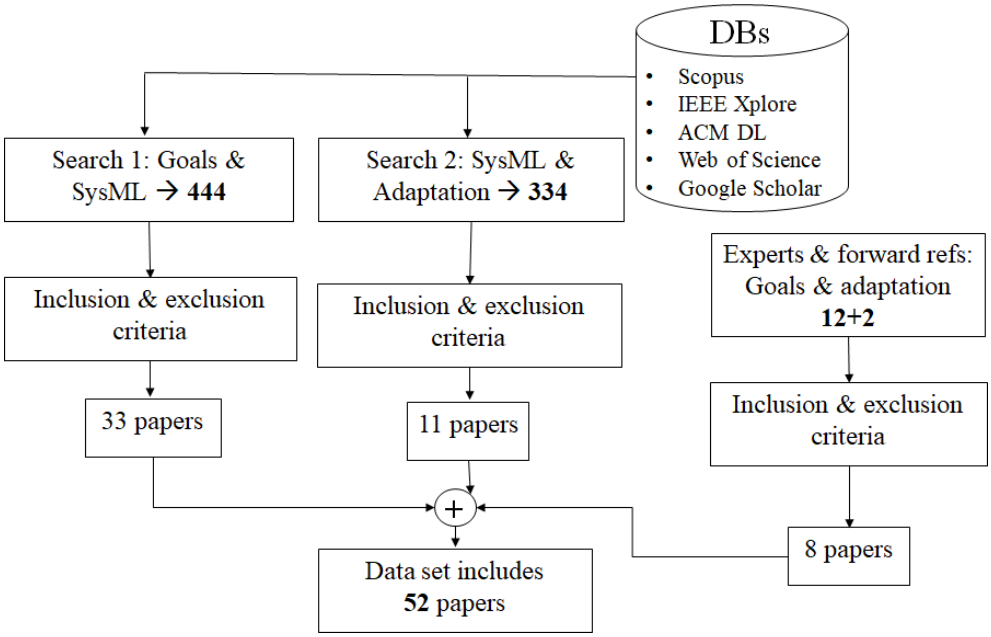

Figure 5. Result of the screening process

Table 4. Results of the searches per databases

Database	Goal and SysML	SysML and adaptation
Scopus	70	62
IEEE Xplore	60	67
Web of Science	28	17
ACM Digital library	23	62
Google Scholar (title only)	23	6
Google Scholar (any field)	60+60 (from 4,200)	60+60 (from 3,860)
Google Scholar (with goals-SysML)	60+60 (660)	-
Total (with duplicates)	**444**	**334**
Total (unique papers)	**386**	**316**

[1]A table with the accepted and rejected papers is available online at http://bit.ly/SysML-Goal-SLR

were included at the suggestion of an expert and with a simple forward search (we did not aim to be exhaustive here). This resulted in 52 papers that were eligible for the analysis.

2.2.4. Data extraction

For each study, we extracted the data items mentioned in Table 3. Extracting this data was done iteratively from the selected studies to accumulate information concerning our research questions.

2.2.5. Quality Assessment Process

The collected studies were compared against the criteria listed in Table 2. We did not evaluate how good the articles were (beyond ensuring they were not coming from a predatory source), but we did evaluate how useful each would be to the study. The result of the quality assessment against the criteria explained in Table 2 is provided in Tables A1 and A2 in Appendix A.

3. Discussion

We classify and present the selected studies in a way that will enable answering our research questions accurately. We split the discussion into five subsections: integration methods, adaptation support methods, adaptation assessment, and challenges.

3.1. 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 33 articles retrieved by the Goal/SysML search (Table A1). We classify the studies according to the applied methods and current objectives. A total of 17 methods, named M1 to M17 in Table 5, are proposed by these articles. The types of goal modeling languages and SysML diagrams used in each method are also listed in Table 5, and the main papers in each collection are highlighted in **bold**.

3.1.1. Languages and diagrams involved

For each method, we extracted the goal modeling language and SysML diagrams used (Table 5). Any additional model was considered out of the scope of the study.

The most commonly used SysML diagrams in the 17 proposed methods are requirements diagrams and block diagrams, in that order. All presented methods but three (M1, M11, M14) connected goals or goal models with requirements diagrams, while nearly half the methods (M1, M2, M9, M11, M12, M13, M17) used block diagrams in their integration.

From a goal model perspective, several different languages were used. The most popular languages in these methods are KAOS [24] (M5, M7, M8) and GRL [28] (M2, M4, M17). OMG's Business Motivation Model [76] was also mentioned once in M6 and RELAX [77] once in M8. Several methods only used textual goals or non-functional requirements (NFRs), with some integrating them more formally as SysML stereotypes (M15 and M16). Instead of integrating goal models themselves, Ingram et al. [45] (M1) used goal model analysis

Table 5. Selected studies and their methods used
(FG = Functional goals; NFG = Non-functional goals)

Research studies	Code	Goal language	SysML diagrams
Ingram et al. [45]	M1	Fault tolerance strategies	Block (and dependency relationships)
Amyot et al. [20]	M2	GRL	Requirements, block
Vanderperren and Dehaene [46]	M3	NFG (no specific notation)	Requirements, use case
Ozkaya [47]	M4	GRL	Requirements
Matoussi et al. [48], Laleau et al. [49]	M5	KAOS (FG)	Requirements
Cui and Paige [50]	M6	Business Motivation Model	Requirements
Gnaho et al. [51, 52], Mammar and Laleau [53], Bousse [54]	M7	KAOS (FG, NFG)	Requirements
Ahmad et al. [55], Ahmad et al. [56–58], Ahmad [59], Ahmad and Bruel [60, 61], Belloir et al. [62]	M8	KAOS (FG, NFG), RELAX	Requirements
Apvrille and Roudier [63, 64], Roudier and Apvrille [65]	M9	Textual goals/NFR	Requirements, block, state machine, parametric
Tsadimas et al. [66]	M10	NFR diagram	Requirements
Spyropoulos and Baras [67]	M11	Textual NFRs	Block, parametric
Badreddin et al. [68]	M12	Textual goals (based on GRL)	Requirements, block, use case
Fan et al. [69]	M13	Textual goals	Requirements, block, activity
Wang [70]	M14	Textual goals (with AND/OR decomposition)	Use case
Lee et al. [71]	M15	Requirements diagrams with goal stereotypes	Requirements
Maskani et al. [72]	M16	Requirements diagrams with goal stereotypes	Requirements
Anda and Amyot [73], **Anda and Amyot [74]**, Anda [75], **Anda and Amyot [31]**	M17	GRL	Requirements, block, internal block, parametric, state machine

results in their integration to increase the confidence of system designers when defining the system architecture. From another perspective, Anda and Amyot [74] (M17) used GRL models and enabled analysis through arithmetic expressions.

3.1.2. 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 6 reports on seven main objectives, together with their related methods and articles. The management of *uncertainty and adaptation* (which is concerned with whether the information being monitored is reliable enough to justify adaptation decision, and with what adaptation will help satisfy goals the best), has attracted the highest number of studies (14), with four different methods. However, the *architecture selection and modeling* objective (which is important at design time to find suitable trade-offs between various non-functional goals such as performance, cost, and reliability of systems and adaptations) is targeted by a more varied set of methods (6). These two important objectives are followed by *formal validation and verification* (to ensure safety, liveness, security, and other such properties), and *traceability* (to manage change effectively and to ensure coverage during quality assurance). Other objectives were mentioned only by one or two papers, namely *process improvement* (e.g., so goals are more explicitly considered), *requirements visualization* (e.g., to see how system requirements trace to or contribute to goals), and *impact assessment of non-functional requirements on functional requirements* (e.g., to consider trade-offs involving both types of requirements).

Table 6. Objectives of Goal/SysML integration and related methods

Objectives	Methods	Articles
Uncertainty & adaptation	M1, M2, M8, M17	[20, 45, 51, 55–62, 73–75]
Architecture selection & modeling	M1, M9, M10, M11, M13, M17	[45, 63–67, 69, 73, 74]
Formal V&V	M5, M7, M8, M15	[48, 49, 53, 54, 58, 71]
Traceability	M6, M12, M14, M16, M17	[31, 50, 68, 70, 72]
Development process improvement	M3, M17	[46, 75]
Requirements visualization	M4	[47]
Impact of NFRs on FRs	M7	[52]

3.1.3. Method characteristics

Integrating 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 7 includes information about each main study and the related data that explains the following:

1. Whether the method was automated;
2. Whether the method integrated goals as a model;

3. Whether the main goal concepts are used in the integration;
4. Whether goal analysis was supported; and
5. The method realization (usually through a profile).

Table 7. Extracted data on the integration dimensions
(F = fully automatic, S = Semi-automatic, M = Manual, ? = Unknown)

Research method	Code	Auto	Goal model	Goal concepts	Goal analysis	Method realization
Ingram et al. [45]	M1	?	M	M	S	Profile
Amyot et al. [20]	M2	S	F	S	S	Investigating
Vanderperren and Dehaene [46]	M3	?	M	S	M	Profile
Ozkaya [47]	M4	?	?	?	M	Investigating
Matoussi et al. [48]	M5	?	S	S	M	Profile
Cui and Paige [50]	M6	?	S	S	M	Profile
Gnaho et al. [51, 52]	M7	?	S	S	M	Profile
Ahmad et al. [55]	M8	S	S	S	M	Profile
Apvrille and Roudier [63, 64]	M9	S	M	S	S	Profile
Tsadimas et al. [66]	M10	S	M	S	M	Profile
Spyropoulos and Baras [67]	M11	S	M	S	S	Profile
Badreddin et al. [68]	M12	?	S	S	S	Textual syntax
Fan et al. [69]	M13	S	M	M	M	Profile
Wang [70]	M14	M	S	S	M	Mapping
Lee et al. [71]	M15	M	S	S	M	Profile
Maskani et al. [72]	M16	M	S	S	M	Profile
Anda and Amyot [31, 74]	M17	F	F	F	F	Math functions and RMS

To assess how far the methods go in their integration, Table 7 includes columns that are further explained below. Note however that some methods were still under development or investigating alternatives. As they did not provide sufficient details about their process, the level of automation and the method realization were 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 whether the goals or requirements were translated automatically or re-entered manually.

Automation. Does the method support an MDE (automated) approach? Several studies [9] 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 [9, 46]. As seen in Table 7, most selected methods used goals (partially) as a model with SysML requirements diagrams. In contrast, Badreddin et al. [68] proposed the only method (M12) that does not support a graphical MDE approach and presented a new language that combines the models using a textual syntax. In many studies, goals have actually been translated to a textual, hierarchical structure using a profiled SysML requirements diagram, (with various degrees of formalization). One method (M17) automatically translates GRL goal models to mathematical functions that can be embedded in SysML models. Some studies used SysML block and parametric diagrams with some goal model analysis such as trade-off analysis.

Goal Modeling Concepts. Were important goals modeling concepts (goals, softgoals, decompositions, actor importance, contribution weights, indicators, etc.) part of the integration with the SysML model? Anda and Amyot [31, 74] proposed the only method (M17),

named CGS4Adaptation, that includes all the elements of goal models in their integration. Goals were integrated with SysML requirements diagrams in most methods but not all goal modeling concepts were mapped. These methods extended requirements diagrams with goal types (functional and/or non-functional) and some goal relationships (mainly AND/OR decomposition). However, quantitative/qualitative contributions between goals, importance of goals to their containing actors, and indicators with parameters are seldom covered. For example, Cui and Paige [50] integrated goals model without considering the quantitative values of the contribution relationships between goals or indicator parameters, whereas Ahmad et al. [55] integrated all types of goals and their relationships except for contribution weights, importance levels, and indicators. This prevents modelers from quantitatively 1) performing goal analysis to guide the selection of alternatives (at design time) and 2) supporting dynamic adaptation at runtime according to user preferences [36].

When dealing with NFRs, the methods presented by Apvrille and Roudier [63, 64], Tsadimas et al. [66], and Spyropoulos and Baras [67] focused on the important role of goal-oriented techniques in system architecture and design selection. However, none actually transformed or linked goals to the design phase. Instead, they broke down system goals into non-functional requirements and linked them to design elements of SysML requirements diagrams. In contrast to these methods, Maskani et al. [72] expanded the requirements profile with security goals and requirements while the related stakeholders, goals, assets, and risks were added as attributes.

Goal Analysis. Trade-off analysis can be conducted through positive and negative contributions between goals during the decision-making process, e.g., to determine which actors will be satisfied or dissatisfied by a particular solution or adaptation strategy. 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 [55, 63, 64] or to include possible choices in the system implementation phase [45, 67]. However, these analyses are limited to static decisions and adaptions, often outside of the SysML model as well. To support goal-based design selection and runtime adaption in a way that is integrated with SysML, Anda and Amyot [73–75, 78] generate arithmetic functions from GRL models that can be inserted in SysML models for simulation and optimization, and in the system code for runtime adaptations.

Method Realization. As seen in Table 7, all but four studies used some level of SysML profiling to map goal concepts to SysML concepts (often using requirements diagrams as a basis). Badreddin et al. [68] however proposed (in M12) integrating both views through a new textual language (fSysML), whereas two other studies were still investigating this aspect. In M17, in addition embedding functions generated form goal models in the SysML models, the authors also support importing both the goal and SysML models into a third-party traceability tool (commonly called a Requirements Management System – RMS) to enable managing traceability links between the elements of the goal and SysML models (blocks and requirements diagrams and their relationships), hence also enabling impact analysis and consistency checks as models evolve [31].

3.2. 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. 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.

3.2.1. Self-adaptation properties

We classified the studies based on the four common *self-** properties of self-adaption [8, 35], namely self-healing (from failures and incorrect states), self-configuration (to changing contexts and resources), self-optimization (to best meet specific goals), and self-protection (to avoid system harm). This classification was done with the help of related quality attributes, as suggested by Mistrik et al. [79] and Salehie and Tahvildari [80]. We extracted the non-functional requirements (NFRs) cited in the 52 eligible studies before we related them to four self-* properties.

For the studies where an adaption rationale was available, we established a mapping to self-* types via NFRs. Table 8 details the results. Sixteen methods support systems in adapting themselves while running by responding to changes that could be external (environmental) or internal (the system itself) [81]. Only four of them [45, 55, 67, 74] integrate goal and SysML models for both system design and self-adaptation. SysML also was hired by another 7 methods to support self-adaptation.

In terms of adaptation approaches that use goal models but not SysML, we find several methods such as those from Morandini et al. [82, 83], Qian et al. [94], Ramnath et al. [95], Baresi et al. [86], and Baresi and Pasquale [87, 88]. Additional diversity is brought by pattern-based and case-based approaches [90, 94].

Table 8. Distribution of self-* properties and non-functional requirements among the studies

Self-*	NFRs	Goal/SysML	Goal	SysML
Self-Healing	Fault diagnosing tolerance	Ingramet al.[45]	Morandini et al. [82, 83]	Bareiß et al. [84], Parri et al. [85]
Self-Configuration	Adaptability, Integrity and Availability	Ahmad [59], Ahmad et al. [55–58], Ahmad and Bruel [60, 61]		
	Adaptability	Anda and Amyot [73, 74], Anda [75]	Baresi et al. [86], Baresi and Pasquale [87, 88]	Hussein et al. [89], Meacham [90]
	Reliability			Ribeiro et al. [91]
Self-Optimization	Resource utilization	Spyropoulos and Baras [67]		Lopes et al. [92], Souza et al. [93]
	Time behavior		Qian et al. [94]	
Self-Protection	Security	Belloir et al. [62]	Ramnath et al. [95]	

3.2.2. Adaptation phase and development

Support for the development of adaptive systems is provided at different levels. Some studies provide analysis and design methods for such systems, but *without* explicit adaptation support. Others that come *with* adaptation support do so either for design-time adaptation or for 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 [80, 90].

Table 9 shows that most of the studies that integrate goal and SysML models target the development adaptive systems for different reasons (i.e., uncertainty reduction, complexity simplification, system validation and verification) other than for adaptation, while most of the methods that target self-adaptation trough SysML models or goal models separately implement their adaption strategies, mechanisms, and decisions at design time (design-time adaptation). Interestingly, runtime adaption in SysML is currently lacking contributions.

Table 9. Distribution of the studies related to development of adaptive systems

Study Category	Without Adaptation Support	With Adaptation Support	
	Analysis and Design Only	Design-Time Adaptation	Runtime Adaptation
Goal/SysML	[31, 46–54, 63–66, 68–72]	[20, 45, 55–62, 67]	[73–75]
SysML and adaptation	[18, 19, 96]	[17, 84, 85, 89–93]	
Goals and Adaptation	[97]	[82, 83, 86–88, 95]	[94]

3.2.3. Adaptation approaches

Tables 10 and 11 summarize the approaches each method applies to meet its objectives. Three methods (in four articles) used the *i** goal modeling language, and one (in four articles) used GRL, a language that originates from *i**. Four methods used the KAOS language (in 11 articles), and RELAX [77] was used in a few instances. Please note that the methods in Tables 10 and 11 are different and independent from the integration methods described in Table 5.

Table 10. Methods using combined Goal/SysML models to represent self-adaptive systems

Method	Overview
Ingram et al. [45]	Employed conditions and roles of a fault tolerance study to choose the best strategy for managing traffic problems.
Ahmad et al. [55]	Used SysML, KAOS and RELAX to manage uncertainty at runtime.
Spyropoulos and Baras [67]	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.
Anda and Amyot [31, 73, 74], Anda [75]	Transformed GRL and feature models into mathematical functions that can be executed outside of goal modeling tools including SysML, simulation, optimization, and implementation tools. Also, goal and SysML models are imported into an RMS to manage traceability and consistency as models evolve.

Table 11. Methods using SysML models or goal models separately to represent self-adaptive systems

Method	Overview	Using
Morandini et al. [82, 83]	Unified goal model (i*), failure model, and environmental model to support self-adaptation.	Goals
Qian et al. [94]	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.	Goals
Ramnath et al. [95]	Linked strategies for attack and protection at the design layers of the proposed architecture.	Goals
Meacham [90]	Combined pattern-based with case-based reasoning approaches where repeated falls were collected and analyzed to identify their patterns, leading to solutions as plans.	SysML
Ribeiro et al. [91]	Modeled real time requirements and managed traceability through extending SysML requirements diagram with relationships and properties. Synchronized relationships were used to represent parallel real-time requirements.	SysML
Bareiß et al. [84]	Modified the SysML meta-model to create a SysML4Pack profile that combines SysML model, OCL [98] and the state machines of OMAC to represent predictable faults of automatic production systems.	SysML
Lopes et al. [92]	Integrated SysML models with trade-off analysis and techno-economical cost-benefit analysis to optimize electricity management, generation, and distribution among customers.	SysML
Soyler and Sala-Diakanda [17]	Included disaster management strategies in a SysML system architecture with continuous feedback from the last disaster data.	SysML
Akbas and Karwowski [18]	Combined dynamic models with agent-based models that were extracted from system design using SysML models.	SysML
Souza et al. [93]	Created a SmartCitySysML profile that extends the profiles of requirement and block in SysML to represent smart city elements.	SysML
Horkoff et al. [97]	Integrated goal models i* with the MAVO framework of [99] to iterate over the analysis process for early uncertainty reduction.	Goals
Baresi et al. [86]	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) [100] plus fuzzy temporal operations such as <, >, <=, and *approximately* to express the fuzzy goals.	Goals
Parri et al. [85]	Combined system configurations derived from SysML block definition diagrams (BDD) metadata with a failure model derived from fault tree via digital twins and data analysis agents.	SysML
Baresi and Pasquale [88]	Used service composition based on the Business Process Execution Language (BPEL) [101] to transform the FLAGS/KAOS model in Baresi et al. [86] to membership functions and abstract processes, semi-automatically. These functions trigger the adaptation strategies using Boolean conditions.	Goals
Baresi and Pasquale [87]	Added operators from RELAX Language to the FLAGS language in Baresi and Pasquale [88] 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.	Goals

Several integrations (and, at times, extensions) were also done with goal models or SysML models separately, hence answering the sub-question SQ2.2.

Goal Model Integrations with Languages Other than SysML

1. In order to support dynamic adaptive systems, Morandini et al. [82, 83] integrated models of goals, failures, and the environment.
2. To deal with unpredicted changes at runtime, Qian et al. [94] integrated goal models with case-based reasoning.
3. Horkoff et al. [97] integrated goal models with the MAVO framework to reduce uncertainty early.
4. Baresi et al. [86] and Baresi and Pasquale [87, 88] described goals using a formal linear temporal logic (LTL) language and the RELAX language for usage at runtime.
5. Anda and Amyot [31] adapted a model import method [102] to import and trace GRL models into an RMS (IBM Rational DOORS [103]). They also generate functions from goal and feature models that can be embedded in SysML models.

SysML Model Integrations with Non-Goal-Oriented Languages

1. Meacham [90] did an integration of SysML with UML to specify cases of presented patterns, while Soyler and Sala-Diakanda [17] also supported an integration with UML, but this time to represent the structure and behavior of systems in one single environment.
2. Additional relationships and properties were added to SysML requirements diagrams by Ribeiro et al. [91] for representing runtime requirements in a hierarchical way and for managing requirements traceability for system validation and verification.
3. Bareiß et al. [84] used an integration with OMAC state machines, ISA-88 physical models, and OCL constraints for transforming models to code.
4. Lopes et al. [92] 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. [19] and Akbas and Karwowski [18] for minimizing system complexity and specifying system agents in a hierarchical structure.
6. Smart city elements including different types of requirements, solutions, processes, stakeholders, problems, and dimensions have been added to a SysML profile by Souza et al. [93] to support a domain-specific modeling process.
7. Real configuration items from SysML BDD properties and diagnostic, predictive, and prescriptive analytics derived from fault tree are integrated by Parri et al. [85] to discover alternative configurations when a runtime violation is detected.
8. Ginigeme and Fabregas [96] derived configuration parameters from the stakeholder's requirements and system design in SysML to be used by a discrete-event simulation (Arena) tool to evaluate the design configurations.
9. SysML BDDs and requirements diagram are imported in an RMS (DOORS) by Anda and Amyot [31] to support consistency and completeness checks (against imported GRL models) as well as more common impact analysis and change management processes.

3.3. Adaptation assessment

In order to answer questions SQ2.1 and SQ2.2 on adaptation methods, we extracted information that identifies terms inspired from existing adaptation taxonomies [37, 80] and modeling dimensions of self-adaptation [81, 104]. Using these terms was helpful in inferring

correct indicators that specify how each method supports self-adaptation and what roles each model plays in this adaptation.

Among the articles collected, some were eliminated from this assessment because their adaptation methods were redundant or not described in sufficient detail. In particular, Akbas and Karwowski [18] and Horkoff et al. [97] designed self-adaptive systems for reducing uncertainty and system complexity. They supported the use of self-adaptive systems but not the use of adaptation where the system re-configures itself to become more usable. In addition, Baresi et al. [86] and Baresi and Pasquale [87, 88] expressed the same methods with different emphases, so we considered them as one method represented by the most detailed paper [87]. Finally, Spyropoulos and Baras [67] provided information about their dynamic decision-making process but not about the adaptation strategies and properties; this paper was excluded from the adaptation properties and dimensions assessment, but kept for the decision-making criterion. As a result, 14 methods are discussed here.

3.3.1. Adaptation terms

The selected terms were defined in Section 2.2.4 on data extraction. Table 12 illustrates the assessment of each adaptation term against related methods and studies. The color coding reflects how positive a result is (green = positive, yellow = neutral, red = negative, and white = unknown or inappropriate).

3.3.2. Adaptation modeling dimensions

Three types of modeling dimensions (goal, change, and mechanisms), proposed by Andersson et al. [81], are used to specify self-adaptive properties. Some of these properties are overlapping with the adaptation taxonomy previously mentioned. Some of the methods, such as those presented by Ahmad et al. [55], Anda and Amyot [74], and Baresi and Pasquale [87], 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 13.

3.3.3. Assessment results

The surveyed methods handled self-adaptation from several perspectives: adaptation terms and modeling dimensions, early management of uncertainty, the use of different languages to deal with adaptation, frameworks for developing self-adaptive systems, adaptation strategies, and finally decision-making and strategy selection processes. This section provides further assessment of the methods along these six perspectives. A comparison of the methods according to the used models is also provided to highlight the contribution of these models to the ability of systems to self-adapt.

Adaptation Terms and Modeling Dimensions. From Tables 12 and 13, several observations can be made:

- Most of the collected methods realized a *closed* approach of adaptation by including their strategies with system design. Only three were clearly *open*, i.e., more amenable to adaptation to unforeseen situations and contexts.
- All of the collected methods supported their adaptation approach at design time, and they do not enhance or change them at runtime using a learning technique (e.g., based on machine learning).

Table 12. 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)

	Goals and SysML			Goals only				SysML only						
Adaptation terms	Ingram et al. [45]	Anda and Amyot [74]	Ahmad et al. [55]	Baresi and Pasquale [87]	Morandini et al. [83]	Qian et al. [94]	Ramnath et al. [95]	Meacham [90]	Ribeiro et al. [91]	Bareiß et al. [84]	Lopes et al. [92]	Soyler and Sala-Diakanda [17]	Souza et al. [93]	Parri et al. [85]
Adaptation Type	C	O	?	C	C	O	C	C	C	C	?	O	C	C
Model-based	Y	Y	Y	Y	Y	P	Y	Y	Y	Y	Y	Y	Y	Y
Decision (Analyze/Selection process)	?	Dy	Sta	Dy/ Sta	Sta	Dy	Sta	Sta	?	Sta	?	?	?	Dy/ Sta
Adaptation approach	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Layer	Md	Ap	Ap	Ap	Ap	Ap	Ap	Ap	?	?	Md	Ap	Ap	Md
Artifact	Sr/S	Sr	Sr	Sr	Sr	Sr	Sr	Sr	?	St	Sr /St	Sr	Sr	St
Cost-impact	W	W	W	W	W	W	W	W	?	W	W	W	W	W
Temporal adaptation	Rt	Pt /Rt	Rt	Rt	Pt/ Rt	Rt	Rt	Rt	?	Rt	Rt	Rt	Pt/ Rt	Pt/ Rt
Temporal monitoring	Ad	Co	Co	Co	Co	Co	?	Co	?	Co	Co	Co	Co	Co
Human intervention	N	N	N	P	P	N	N	N	?	P	?	?	N	P

– According to their effected layers and artifacts, they supported only a weak adaptation (i.e., no change to the architecture at runtime).
– 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 13.
– Most methods that used goals as a model managed 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 two methods. Qian et al. [94] used 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

Table 13. Modeling dimension of the selected methods
(Sta=Static, Dy=Dynamic, Rgd=Rigid, Cns=Constrained, Ncn = Unconstrained,
Mlti = Multiple, S = Single, E = External, I = Internal, Nfr = Non-functional requirement,
Gt = guaranteed, NGt = Not guaranteed, ? = Unknown)

	Goals and SysML			Goals only				SysML only						
Adaptation terms	Ingram et al. [45]	Anda and Amyot [74]	Ahmad et al. [55]	Baresi and Pasquale [87]	Morandini et al. [83]	Qian et al. [94]	Ramnath et al. [95]	Meacham [90]	Ribeiro et al. [91]	Bareiß et al. [84]	Lopes et al. [92]	Soyler and Sala-Diakanda [17]	Souza et al. [93]	Parri et al. [85]
Goal														
Evolution	Sta	Sta	Sta	Dy	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta	Sta
Flexibility	Rgd	Ncn	Cns	Cns	Cns	Ncn	Rgd	Rgd	Rgd	Rgd	Rgd	Rgd	Cns	Rgd
Multiplicity	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	S	Mlti	Mlti	Mlti	Mlti
Dependency	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	Mlti	S	Mlti	Mlti	Mlti	Mlti
Change														
Source	E	E& I	E	E&I	E& I	I	E	E	?	I	E& I	E	E	I
Type	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	NFR	?	NFR	NFR
Mechanisms														
Timeliness	Gt	Gt	?	NGt	Gt	NGt	Gt	Gt	?	Gt	Gt	?	Gt	Gt

new solutions could be unsuitable for the current problem and consequently lead to non-guaranteed adaptation timeliness, as shown in Table 13. On the contrary, Anda and Amyot [73, 74] have used a goal reasoning method (without constraints or conditions on its choices) to generate on the fly the best solutions when unforeseen circumstances are encountered at runtime. Since the used mathematical functions include the impact of the current environmental condition on all elements of the goal models and are restricted by the mathematical function of feature models, the created solutions are feasible and the best (the functionality and quality of the system satisfy its stakeholders' objectives) for the current environmental condition.

– Baresi and Pasquale [87] 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.
– Most methods (except three) included work or comments 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.

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. [97] presented a formal iterative goal analysis process with a tool that integrated *i** goal models with the MAVO framework [99] to remove unnecessary requirements alternatives. The treatment of uncertainty in general goal modeling is further explored by Alwidian et al. [105].

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 often too strict that context. This need was addressed in by Ahmad et al. [55], who used the RELAX language [77] as a more formal representation of this idea for monitoring environmental conditions and detecting violations. In addition, the formal language called FLAGS [87, 88] formalizes the KAOS goal modeling language through LTL. In order to represent fuzzy goals with uncertainty, LTL is accompanied by fuzzy temporal operators based on RELAX [87]. This language was used to keep tracking and using the goal model from requirements elicitation to the implementation phase. Anda and Amyot [31, 73, 74] generate arithmetic functions in common languages (including C, Java, and Python) from GRL and feature models, which enable analysis and implementation to be done with a wide range of development tools.

Frameworks for Designing Self-Adaptive Systems. Several approaches and frameworks were presented to design and select an appropriate architecture for self-adaptive systems (SAS). Morandini et al. [83] extended the Tropos framework [106] 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. 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.

To support system reliability, flexibility, and runtime recoverability, Parri et al. [85] proposed a software/hardware framework, called JARVIS, for developing CPSs and Systems of Systems. JARVIS adopts SysML BDDs and fault trees to discover configuration alternatives using digital twins and data analytic agents.

Security strategies can also affect user privacy and cost. For this reason, Ramnath et al. [95] proposed a non-functional framework to deal with adaptive security analysis. The goal model is linked from and to dynamic behavior 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 [17] 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, Akbas and Karwowski [18] 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 12) manage uncertainty through rigid or constrained goals. From the Goal/SysML integration methods, Ingram et al. [45] used fault tolerance analysis and rules to deal with errors. Without considering goal models, Ribeiro et al. [91], Parri et al. [85], Bareiß et al. [84], Souza et al.[93], and Soyler and Sala-Diakanda [17] triggered their embedded strategies, configuration, or plans to respond to internal or environmental changes. Similarly, Morandini et al. [82, 83] 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 characteristic common to 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 a given contextual change (see Tables 12 and 13).
- *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 [17] (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. [94] 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 [107]). In contrast, Meacham [90] 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. [94] did. To enhance overall system performance, new strategies or configurations can be issued by the optimization method of Anda and Amyot [31, 73], which deals with unexpected conditions at runtime.

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). 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 [37, 108]. These methods realized their decision-making processes as follows.

- *Action 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) [108]. Meacham [90] used a pattern analysis algorithm to trigger the adaptation while Morandini et al. [83] used a goal modeling approach and several types of conditions that trigger the adaptation process. In order to select a suitable recovery strategy, Bareiß et al. [84] 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 in practice [10, 108].

– *Utility and goal policies:* In order to select an optimal adaptation strategy, experts are needed to identify the control variables required by the utility policy approach. Such 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 monitored data at runtime [108].
Ramnath et al. [95] 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 [67] used trade-off analysis to get the optimal solution for power allocation in their Microgrid system. In their approach, Lopes et al. [92] 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 [87] 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, Parri et al. [85] used Fault Tree analysis to detect and predict failures while the suitable configurations were associated to the tree via digital twins. Also, Qian et al. [94] used case-based reasoning in all MAPE activities (see Figure 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 still required [10, 108]. On the other hand, goal and feature models transformed to mathematical functions by Anda and Amyot [73, 74, 78] are used in MAPE activities to monitor, analyze, and select the suitable strategies at design time and during runtime adaptation.

3.3.4. Self-adaptation, goals, and SysML

The collected articles display distinguished features when classified according to the three categories (Figure 5) initially used to search the literature: goal models (without SysML), SysML models (without goal models), and goal models combined with SysML models.

From Section 3.2.3, goal models are used to reduce uncertainty early and provide adaptation rationale and alternatives. Also, based on Table 9, goals are involved into all the methods that enhance their adaptation solutions at runtime while runtime adaption is not well supported in SysML. In Table 12, two methods used goal models to generate new solutions/strategies when facing unknown conditions at runtime while only one SysML-based method used feedback to create new strategies, but without much detail. Similarly, three methods out of four that provided dynamic decisions exploited goal models while only one method (Parri et. al. [85]) supports dynamic analysis with a strategy selection process using SysML. From a modeling dimension perspective, SysML is involved in all the inflexible methods that use fixed goals in their adaptation approaches. On the other hand, all the goal-based methods manage multiple goals and their dependencies defined in goal models.

Also from Section 3.2.3, SysML provides a suitable environment that reduces the complexity of self-adaptive systems and represents them in one single environment through profiles. Such profiles are used to strengthen domain-specific modeling by adding new terms for new types of systems such as smart cities [93]. However, these profiles do not represent goal model elements and analysis together, and valuable information is lost during the mapping of goal-related concepts to SysML concepts, which reduces the flexibility of theses methods (and leads to rigid or constrained goals instead of goal reasoning enabling selections and trade-offs among goals). Only one method [74] integrates goal model analysis (including goals, softgoals, actors, tasks, indicators, contributions, relationships, and their importance) with SysML models without using profiles. This integration involves mathematical expressions generated from goal models, which enables a flexible method with open adaptation together with dynamic decision in analysis and mapping mechanisms.

3.4. Challenges

The studies have faced several challenges while employing their methods, and some are further explored below.

Usability of Integration. The integration processes are often characterized by remodeling goals with design tools (duplication of work), which not only causes risks of information loss and inconsistencies, but also consumes much development effort and time. One reason is that requirements, goal models, and SysML design artifacts have different and specific environments and tools that deal with their creation, management, and analysis needs. Representing a goal model using another tool with a different purpose (such as s SysML design tool) 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.

Goal Models and MAPE Activities. The MAPE activities (monitor, analyze, plan, and execute, see Figure 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. However, in these studies, the scalability of the proposed methods is rarely formally assessed.

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 linksweights) 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 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 contexts.

Unmanageable Traceability. High-level goals are usually more stable than low-level ones, and they help guide the evolution of requirements from elicitation to runtime adaptation [9, 61]. However, to truly unlock the benefits of goal-orientation (including consistency/completeness, conflict, trade-off, and impact analyses), SysML system design components should be linked to goals at all levels [9, 54, 55]. Yet, it is difficult to manage

traceability and consistency between goal and SysML models. Embedding goal models in SysML tools can help, but amount to redeveloping goal-based analysis in such tools, and none of the existing methods really does this. Overall, although establishing links between goals/requirements, system design, and the implementation was an objective of most selected methods, most fail from providing sufficient and practical traceability support except one [31], which uses an external RMS to do so but with low usability due the high number of tools involved (goal modeling environment, SysML tool, and RMS). SysML tool vendors should consider better integrating goal modeling and analysis capabilities in their solutions.

4. Related work

Although a literature review is already about collecting and assessing related work, it is also important to situate it among other literature reviews on related topics.

Zahid et al. [39] have recently published a systematic mapping of semi-formal and formal methods in requirements engineering of (industrial) Cyber-Physical Systems. Although SysML is mentioned on a few occasions, adaptive CPSs are not covered. Surprisingly, goal modeling is not discussed in their review.

There are also generic language-oriented systematic mappings on goal-oriented modeling (e.g., from Horkoff et al. [40]) and SysML (e.g., from Wolny [41, 42]) but they are superficial in their treatment of (self-)adaptive systems.

In contrast, there are several literature surveys and mappings on adaptive systems, whether they are cyber-physical or not. In particular:

- de Lemos et al. [109] provided an important roadmap for software engineering research on self-adaptive systems, which emphasized the identification and representation of goals, the management of the design space, and the validation of models (without mentioning SysML or goal-oriented modeling however) as important challenges. Many of the methods addressed in our review provide contributions in those areas.
- Macías-Escrivá et al. [36] also provided a survey with research challenges, but without details on modeling aspects.
- Krupitzer et al. [37] reviewed software engineering approaches for self-adaptive systems and discuss some goal-oriented methods, but no SysML-based ones, and not to the depth of our own review.
- Yang et al.[110] provided a review of requirements modeling and analysis for self-adaptive systems, where they identified 16 methods, some of which involving goal modeling. No SysML-based method was identified at the time this review was published (2014). Their main assessment was related to the coverage of important modeling and analysis concepts by the methods and the languages they used. CPSs are not mentioned.
- More recently, Porter et al. [38] explored the types of questions that are researched in the literature in relation to self-adaptive systems, instead of methods.

Our literature review is unique in that it is fairly exhaustive in its coverage of goal/SysML integrations and of SysML methods targeting adaptation, with partial coverage of some of the main goal-oriented methods for adaptive systems. It also provides a deeper analysis of multiple research questions and facets of these methods. It finally positions these methods in the CPS domain, with a specific emphasis on emerging types of adaptive socio-cyber-physical systems.

5. Limitations and threats to validity

As highlighted by Feldt and Magazinius [111], the validity of any study depends on the degree of correctness of its conclusions, including threats related to bias and over-generalization. We applied some strategies to mitigate common threats to validity, but several remain, as discussed below.

5.1. Internal validity

The first author selected and reviewed the papers, and extracted the raw data, with supervision and informal consultations and checks from the second author. In addition, one of the methods studied here (M17) also comes from the authors of this literature review. There is hence a risk of bias here. To mitigate this threat, we consulted several experts, including the authors of some of the selected papers, to increase the level of confidence in our assessment of their contributions. We have also used existing assessment criteria from the literature whenever they were available (e.g., from [37, 80, 81, 104]). However, the authors of the many papers reviewed here 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 both authors were involved in the selection in cases where we were unsure about relevance. Yet, one remaining threat here is that the selected literature was limited to the English language.

We tried to be exhaustive for papers combining goal and SysML modeling, as well as for papers about SysML for self-adaptation. However, we manually selected primary articles (proposed by experts based on citations and reputation, as there were too many such papers) that support adaptation using goal models (Figure 5). 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, as a comparison point and as an indication of future opportunities.

5.2. External validity

This type of validity is related to the generalization of the results outside of the study's scope [111, 112]. The number of studies that focus on the integration of goal models with SysML models is rather small. If we consider the method granularity, only 17 methods were presented and four of them were specifically targeting adaptation. This is also why we focused on a descriptive presentation of our results, without trying to discuss statistical significance in the answers to our research questions.

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. 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, again as comparison points. Still, general

conclusions about the use of goal modeling for adaption (without an integration with SysML) or about the integration of SysML and goal models outside of a CPS context should not be inferred from this literature review.

6. Conclusion

The number, complexity, and importance of socio-cyber-physical systems (SCPSs), which consider the goals of their stakeholders at design time and at runtime, is increasing in our societies [23]. In some SCPSs, the need for adaptability driven by stakeholder goals was partially addressed in the peer-reviewed scientific literature. This paper reviewed 52 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 5, which presents a total of 17 methods, labeled M1 to M17, extracted from 33 studies. KAOS and GRL are the most frequently mentioned goal modeling languages in that context.

- **SQ1.1.** *Why have these integrations been proposed?* The objective of each study was presented in Table 6, 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 7 and its explanation in Section 3.1.3, which concluded that mapping parts of goal models to SysML requirements diagrams via profiles (formal or not) is by far the most often used approach through all 17 methods.

RQ2. *What are the collected methods that support self-adaptation?* By classifying the collected methods using NFRs, self-* properties (Table 8), and adaptation phases (Table 9), methods that support self-adaptation are listed and described in Table 10 (for the four approaches that integrate goals and SysML models) and Table 11 (for a sample of 15 approaches that use either SysML or goal models).

- **SQ2.1.** *How do the methods support self-adaptive systems?* This question was answered by Tables 12 and 13, which respectively identify terms inspired from the adaptation taxonomies and modeling dimensions of self-adaptation. The discussion around these tables (Section 3.3) 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 3.2.3, and by discussing the adaptation assessment criteria in Section 3.3.

Although there was much improvement in the last decade, the main results show that 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) or an absence of information (e.g., indicators sensing external contexts)

that play key roles 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 effort, most of the proposed methods were unable to implement goal-based reasoning in all activities. This consequently leads to situations where incorrect adaptation solutions are produced and used, and in time constraints that cannot be guaranteed. In fact, although modeling goal and SysML models in a single tool could help solve traceability problems and support adaptation, achieving this integration with existing design and analysis tools remains a challenge, as highlighted in Section 3.4.

To address many of the challenges and limitations observed throughout this review, we identify the following research directions.

- Developing and evolving adaption methods for SCPS where the goal models exploit important quantitative information such as contribution weights, importance levels to stakeholders, and indicators that measure different facets of the context. Such information is often necessary in models for data-centric systems and is very important for non-trivial adaptive SCPSs [113]. Such methods exist, but they are seldom integrated with SysML design activities.
- Ensuring that methods exploit the goal models through the MAPE cycle to their fullest extent, especially during runtime adaptation for unforeseen contexts (open approaches). Again, several opportunities have been explored in the goal modeling community but they are yet to be exploited in a SysML modeling and analysis context.
- As most integrated methods only support weak adaptation, there are opportunities to investigate goal-oriented, strong adaptations of component structures and architectures at runtime in an SysML context.
- Improving the usability and scalability of goal/SysML integrations for adaptive systems, with proper tool support, especially as the models grow in size and are frequently modified.
- Enabling (machine) learning during adaptation in integrated goal/SysML methods. None of the current work currently exploits this opportunity.

Despite many observed gaps and challenges, we believe the benefits of goal modeling (potential or actual) combined with SysML for adaptive SCPSs outweigh the identified drawbacks, and that further research will bring innovative and practical solutions in the near future.

Acknowledgment

Amal Ahmed Anda is supported by a scholarship from the Libyan Ministry of Education. We are thankful to the Natural Science and Engineering Research Council of Canada (Discovery program) for their support.

References

[1] B. Tekinerdogan, D. Blouin, H. Vangheluwe, M. Goulão, P. Carreira et al., *Multi-Paradigm Modelling Approaches for Cyber-Physical Systems*. Elsevier Science, 2020.

[2] I. Horváth, "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.

[3] I.J. Jureta, A. Borgida, N.A. Ernst, and J. Mylopoulos, "The requirements problem for adaptive systems," *ACM Transactions on Management Information Systems (TMIS)*, Vol. 5, No. 3, 2015, p. 17.

[4] A. Smirnov, A. Kashevnik, and A. Ponomarev, "Multi-level self-organization in cyber-physical--social systems: Smart home cleaning scenario," *Procedia CIRP*, Vol. 30, 2015, pp. 329–334, 7th Industrial Product-Service Systems Conference – PSS, industry transformation for sustainability and business.

[5] F. Zambonelli, "Towards a general software engineering methodology for the internet of things," *CoRR*, Vol. abs/1601.05569, 2016. [Online]. http://arxiv.org/abs/1601.05569

[6] E. Cavalcante, T. Batista, N. Bencomo, and P. Sawyer, "Revisiting goal-oriented models for self-aware systems-of-systems," in *2015 IEEE International Conference on Autonomic Computing (ICAC)*, July 2015, pp. 231–234.

[7] M. Sanchez, E. Exposito, and J. Aguilar, "Autonomic computing in manufacturing process coordination in industry 4.0 context," *Journal of Industrial Information Integration*, Vol. 19, 2020, p. 100159. [Online]. https://www.sciencedirect.com/science/article/pii/S2452414X20300340

[8] J.O. Kephart and D.M. Chess, "The vision of autonomic computing," *Computer*, Vol. 36, No. 1, 2003, pp. 41–50.

[9] J. Bocanegra, J. Pavlich-Mariscal, and A. Carrillo-Ramos, "On the role of model-driven engineering in adaptive systems," in *Computing Conference (CCC), 2016 IEEE 11th Colombian.* IEEE, 2016, pp. 1–8.

[10] J.C. Muńoz-Fernández, R. Mazo, C. Salinesi, and G. Tamura, "10 challenges for the specification of self-adaptive software," in *12th International Conference on Research Challenges in Information Science (RCIS)*, May 2018, pp. 1–12.

[11] F. Bordeleau, B. Combemale, R. Eramo, M. van den Brand, and M. Wimmer, "Tool-support of socio-technical coordination in the context of heterogeneous modeling," in *6th Int. Workshop on the Globalization of Modeling Languages (GEMOC), MODELS 2018 Workshops*, 2018, pp. 1–3.

[12] BKCASE Governing Board, "Guide to the Systems Engineering Body of Knowledge (SEBoK) v. 1.9.1," 2014, p. 945. [Online]. https://bit.ly/2PWwxFJ

[13] T. Huldt and I. Stenius, "State-of-practice survey of model-based systems engineering," *Systems Engineering*, 2018, pp. 1–12 (online first).

[14] OMG, "OMG Systems Modeling Language (SysML), Version 1.6," Object Management Group, 2019. [Online]. https://www.omg.org/spec/SysML/

[15] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[16] ISO, "ISO/IEC 19514:2017 – Information technology – Object management group systems modeling language (OMG SysML)," International Organization for Standardization, 2017. [Online]. https://www.omg.org/spec/SysML/

[17] A. Soyler and S. Sala-Diakanda, "A model-based systems engineering approach to capturing disaster management systems," in *2010 IEEE International Systems Conference*, apr 2010, pp. 283–287.

[18] A.S. Akbas and W. Karwowski, "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.

[19] A.S. Akbas, K. Mykoniatis, A. Angelopoulou, and W. Karwowski, "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*, DEVS '14. San Diego, CA, USA: Society for Computer Simulation International, 2014, pp. 31:1–31:6. [Online]. http://dl.acm.org/citation.cfm?id=2665008.2665039

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

[21] G. Mussbacher, D. Amyot, R. Breu, J.M. Bruel, B.H.C. Cheng et al., "The relevance of model-driven engineering thirty years from now," in *Model-Driven Engineering Languages and Systems*, J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfran, Eds. Cham: Springer International Publishing, 2014, pp. 183–200.

[22] J.A. Lane and T. Bohn, "Using SysML modeling to understand and evolve systems of systems," *Systems Engineering*, Vol. 16, No. 1, 2013, pp. 87–98.

[23] C. Ncube and S.L. Lim, "On systems of systems engineering: A requirements engineering perspective and research agenda," in *26th International Requirements Engineering Conference (RE)*. IEEE CS, Aug 2018, pp. 112–123.

[24] A. Van Lamsweerde, *Requirements engineering: From system goals to UML models to software*. Chichester, UK: John Wiley & Sons, 2009, Vol. 10.

[25] S. Woldeamlak, A. Diabat, and D. Svetinovic, "Goal-oriented requirements engineering for research-intensive complex systems: A case study," *Systems Engineering*, Vol. 19, No. 4, 2016, pp. 322–333.

[26] E.S.K. Yu, "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.

[27] D. Amyot and G. Mussbacher, "User Requirements Notation: the first ten years, the next ten years," *JSW*, Vol. 6, No. 5, 2011, pp. 747–768.

[28] ITU-T, "Recommendation Z.151 (10/18): User Requirements Notation (URN) – Language Definition," 2018. [Online]. http://www.itu.int/rec/T-REC-Z.151/en

[29] M. Daun, J. Brings, L. Krajinski, V. Stenkova, and T. Bandyszak, "A GRL-compliant iStar extension for collaborative cyber-physical systems," *Requirements Engineering*, Vol. 26, No. 4, 2021, pp. 325–370.

[30] K. Neace, R. Roncace, and P. Fomin, "Goal model analysis of autonomy requirements for unmanned aircraft systems," *Requirements Engineering*, Vol. 23, No. 4, 2018, pp. 509–555.

[31] A.A. Anda and D. Amyot, "Traceability management of GRL and SysML models," in *SAM'20: 12th System Analysis and Modelling Conference*. ACM, 2020, pp. 117–126.

[32] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton et al., "Evaluating goal models within the Goal-oriented Requirement Language," *International Journal of Intelligent Systems*, Vol. 25, No. 8, 2010, pp. 841–877.

[33] D. Amyot, H. Becha, R. Bræk, and J.E. Rossebø, "Next generation service engineering," in *First ITU-T Kaleidoscope Academic Conference – Innovations in NGN: Future Network and Services*, 2008, pp. 195–202.

[34] M. Alenazi, N. Niu, W. Wang, and J. Savolainen, "Using obstacle analysis to support SysML-based model testing for cyber physical systems," in *8th Int. Model-Driven Requirements Engineering Workshop (MODRE)*. IEEE CS, 2018, pp. 46–55.

[35] G. Blair, N. Bencomo, and R.B. France, "Models@ run. time," *Computer*, Vol. 42, No. 10, 2009.

[36] F.D. Macías-Escrivá, R. Haber, R. del Toro, and V. Hernandez, "Self-adaptive systems: A survey of current approaches, research challenges and applications," *Expert Systems with Applications*, Vol. 40, No. 18, 2013, pp. 7267–7279.

[37] C. Krupitzer, F.M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive and Mobile Computing*, Vol. 17, 2015, pp. 184–206.

[38] B. Porter, R.R. Filho, and P. Dean, "A survey of methodology in self-adaptive systems research," in *International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS 2020)*. IEEE, 2020, pp. 168–177.

[39] F. Zahid, A. Tanveer, M.M. Kuo, and R. Sinha, "A systematic mapping of semi-formal and formal methods in requirements engineering of industrial cyber-physical systems," *Journal of Intelligent Manufacturing*, 2021, pp. 1–36.

[40] J. Horkoff, F.B. Aydemir, E. Cardoso, T. Li, A. Maté et al., "Goal-oriented requirements engineering: an extended systematic mapping study," *Requirements Engineering*, Vol. 24, No. 2, 2019, pp. 133–160.

[41] W. Wang, N. Niu, M. Alenazi, and L. Da Xu, "In-place traceability for automated production systems: A survey of PLC and SysML tools," *IEEE Transactions on Industrial Informatics*, Vol. 15, No. 6, 2018, pp. 3155–3162.

[42] S. Wolny, A. Mazak, C. Carpella, V. Geist, and M. Wimmer, "Thirteen years of SysML: a systematic mapping study," *Software & Systems Modeling*, Vol. 19, No. 1, 2020, pp. 111–169.
[43] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.
[44] S.J. Tueno Fotso, M. Frappier, R. Laleau, A. Mammar, and M. Leuschel, "Formalisation of SysML/KAOS goal assignments with B system component decompositions," in *Integrated Formal Methods*, C.A. Furia and K. Winter, Eds. Cham: Springer International Publishing, 2018, pp. 377–397.
[45] C. Ingram, Z. Andrews, R. Payne, and N. Plat, "SysML fault modelling in a traffic management system of systems," in *System of Systems Engineering (SOSE), 2014 9th International Conference on*. IEEE, 2014, pp. 124–129.
[46] Y. Vanderperren and W. Dehaene, "SysML and systems engineering applied to UML-based SoC design," in *Proc. of the 2nd UML-SoC Workshop at 42nd DAC, USA*, 2005.
[47] I. Ozkaya, "Representing requirement relationships," in *First International Workshop on Visualization in Requirements Engineering, REV 2006*, 2007.
[48] A. Matoussi, F. Gervais, and R. Laleau, "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*. IEEE, 2011, pp. 139–148.
[49] R. Laleau, F. Semmak, A. Matoussi, D. Petit, A. Hammad et al., "A first attempt to combine SysML requirements diagrams and B," *Innovations in Systems and Software Engineering*, Vol. 6, No. 1, 2010, pp. 47–54.
[50] X. Cui and R. Paige, "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.
[51] C. Gnaho, R. Laleau, F. Semmak, and J.M. Bruel, "bCMS requirements modelling using SysML/KAOS," 2013. [Online]. https://goo.gl/QU9Tgn
[52] C. Gnaho, F. Semmak, and R. Laleau, "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.
[53] A. Mammar and R. Laleau, "On the use of domain and system knowledge modeling in goal-based Event-B specifications," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9952 LNCS, 2016, pp. 325–339.
[54] E. Bousse, "Requirements management led by formal verification," Master's thesis, Master's thesis, Computer Science, University of Rennes, France, 2012.
[55] M. Ahmad, N. Belloir, and J.M. Bruel, "Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems," *Journal of Systems and Software*, Vol. 107, 2015, pp. 50–70.
[56] M. Ahmad, J.M. Bruel, R. Laleau, and C. Gnaho, "Using RELAX, SysML and KAOS for ambient systems requirements modeling," in *Procedia Computer Science*, Vol. 10, 2012, pp. 474–481.
[57] M. Ahmad, J. Araújo, N. Belloir, J.M. Bruel, C. Gnaho et al., "Self-adaptive systems requirements modelling: Four related approaches comparison," in *Comparing Requirements Modeling Approaches Workshop (CMA@ RE), 2013 International*. IEEE, 2013, pp. 37–42.
[58] M. Ahmad, I. Dragomir, J.M. Bruel, I. Ober, and N. Belloir, "Early analysis of ambient systems SysML properties using Omega2-IFX," in *SIMULTECH 2013*, 2013.
[59] M. Ahmad, "First step towards a domain specific language for self-adaptive systems," in *New Technologies of Distributed Systems (NOTERE), 2010 10th Annual International Conference on*. IEEE, 2010, pp. 285–290.
[60] M. Ahmad and J.M. Bruel, "bCMS requirements modelling using RELAX/SysML/ KAOS," in *3rd CMA Workshop at RE'2013*, 2013.

[61] M. Ahmad and J.M. Bruel, "A comparative study of RELAX and SysML/KAOS," Institut de Recherche en Informatique de Toulouse, University Toulouse II Le Mirail, France, Tech. Rep., 2014.

[62] N. Belloir, V. Chiprianov, M. Ahmad, M. Munier, L. Gallon et al., "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*. ACM, 2014, p. 32.

[63] L. Apvrille and Y. Roudier, "SysML-Sec: A SysML environment for the design and development of secure embedded systems," *APCOSEC, Asia-Pacific Council on Systems Engineering*, 2013, pp. 8–11.

[64] L. Apvrille and Y. Roudier, "Designing safe and secure embedded and cyber-physical systems with SysML-Sec," in *International Conference on Model-Driven Engineering and Software Development*. Springer, 2015, pp. 293–308.

[65] Y. Roudier and L. Apvrille, "SysML-Sec: A model driven approach for designing safe and secure systems," in *Model-Driven Engineering and Software Development (MODELSWARD), 2015 3rd International Conference on*. IEEE, 2015, pp. 655–664.

[66] A. Tsadimas, M. Nikolaidou, and D. Anagnostopoulos, "Extending SysML to explore non-functional requirements: the case of information system design," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 2012, pp. 1057–1062.

[67] D. Spyropoulos and J.S. Baras, "Extending Design Capabilities of SysML with Trade-off Analysis: Electrical Microgrid Case Study," *Procedia Computer Science*, Vol. 16, 2013, pp. 108–117.

[68] O. Badreddin, V. Abdelzad, T.C. Lethbridge, and M. Elaasar, "FSysML: Foundational executable SysML for cyber-physical system modeling," in *CEUR Workshop Proceedings*, Vol. 1731, 2016, pp. 38–51.

[69] Z. Fan, T. Yue, and L. Zhang, "SAMM: an architecture modeling methodology for ship command and control systems," *Software and Systems Modeling*, Vol. 15, No. 1, 2016, pp. 71–118.

[70] H. Wang, "Multi-Level Requirement Model and Its Implementation For Medical Device," Master's thesis, Master's thesis, Mechanical and Energy Engineering, Purdue University, United States, 2018.

[71] S. Lee, S. Park, and Y.B. Park, "Self-adaptive system verification based on SysML," in *2019 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE CS, 2019, pp. 1–3.

[72] I. Maskani, J. Boutahar, and S. El Ghazi El Houssaïni, "Modeling telemedicine security requirements using a SysML security extension," in *2018 6th International Conference on Multimedia Computing and Systems*, 2018, pp. 1–6.

[73] A. Anda and D. Amyot, "An optimization modeling method for adaptive systems based on goal and feature models," in *2020 IEEE Tenth International Model-Driven Requirements Engineering (MoDRE)*. IEEE, 2020, pp. 11–20.

[74] A.A. Anda and D. Amyot, "Arithmetic semantics of feature and goal models for adaptive cyber-physical systems," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 245–256.

[75] A.A. Anda, "Modeling adaptive socio-cyber-physical systems with goals and SysML," in *26th International Requirements Engineering Conference (RE)*. IEEE CS, 2018, pp. 442–447.

[76] OMG, "Business Motivation Model (BMM), Version 1.3," Object Management Group, 2015. [Online]. https://www.omg.org/spec/BMM/

[77] J. Whittle, P. Sawyer, N. Bencomo, B.H.C. Cheng, and J.M. Bruel, "RELAX: A language to address uncertainty in self-adaptive systems requirement," *Requirements Engineering*, Vol. 15, No. 2, 2010, pp. 177–196.

[78] Y. Fan, A.A. Anda, and D. Amyot, "An arithmetic semantics for GRL goal models with function generation," in *System Analysis and Modeling. Languages, Methods, and Tools for Systems Engineering*, F. Khendek and R. Gotzhein, Eds. Cham: Springer International Publishing, 2018, pp. 144–162.

[79] I. Mistrik, N. Ali, R. Kazman, J. Grundy, and B. Schmerl, *Managing Trade-offs in Adaptable Software Architectures*. Morgan Kaufmann, 2016.

[80] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM transactions on autonomous and adaptive systems (TAAS)*, Vol. 4, No. 2, 2009, p. 14.

[81] J. Andersson, R. De Lemos, S. Malek, and D. Weyns, "Modeling dimensions of self-adaptive software systems," *Software engineering for self-adaptive systems*, 2009, pp. 27–47.

[82] M. Morandini, L. Penserini, and A. Perini, "Automated mapping from goal models to self-adaptive systems," in *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2008, pp. 485–486.

[83] M. Morandini, L. Penserini, A. Perini, and A. Marchetto, "Engineering requirements for adaptive systems," *Requirements Engineering*, Vol. 22, No. 1, 2017, pp. 77–103.

[84] P. Bareiß, D. Schütz, R. Priego, M. Marcos, and B. Vogel-Heuser, "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)*, sep 2016, pp. 1–7.

[85] J. Parri, F. Patara, S. Sampietro, and E. Vicario, "A framework for model-driven engineering of resilient software-controlled systems," *Computing*, 2020, pp. 1–24.

[86] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy goals for requirements-driven adaptation," in *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 2010, pp. 125–134.

[87] L. Baresi and L. Pasquale, "Adaptive goals for self-adaptive service compositions," in *Web Services (ICWS), 2010 IEEE international conference on*. IEEE, 2010, pp. 353–360.

[88] L. Baresi and L. Pasquale, "Live goals for adaptive service compositions," *Proceedings of the 2010 ICSE Workshop on Software*, 2010.

[89] M. Hussein, S. Li, and A. Radermacher, "Model-driven development of adaptive iot systems." in *MODELS (Satellite Events)*, 2017, pp. 17–23.

[90] S. Meacham, "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.

[91] F.G.C. Ribeiro, S. Misra, and M.S. Soares, "Application of an extended SysML requirements diagram to model real-time control systems," in *International Conference on Computational Science and Its Applications*. Springer, 2013, pp. 70–81.

[92] A.J. Lopes, R. Lezama, and R. Pineda, "Model Based Systems Engineering for Smart Grids as systems of systems," in *Procedia Computer Science*, Vol. 6, 2011, pp. 441–450.

[93] L.S. Souza, S. Misra, and M.S. Soares, "SmartCitySysML: A SysML Profile for Smart Cities Applications," in *Computational Science and Its Applications – ICCSA 2020*. LNCS 12254, Springer, 2020, pp. 383–397.

[94] W. Qian, X. Peng, B. Chen, J. Mylopoulos, H. Wang et al., "Rationalism with a dose of empiricism: combining goal reasoning and case-based reasoning for self-adaptive software systems," *Requirements Engineering*, Vol. 20, No. 3, 2015, pp. 233–252.

[95] R. Ramnath, V. Gupta, and J. Ramanathan, "RED-Transaction and Goal-Model Based Analysis of Layered Security of Physical Spaces," in *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*. IEEE, 2008, pp. 679–685.

[96] O. Ginigeme and A. Fabregas, "Model based systems engineering high level design of a sustainable electric vehicle charging and swapping station using discrete event simulation," in *2018 Annual IEEE International Systems Conference (SysCon)*. IEEE, 2018, pp. 1–6.

[97] J. Horkoff, R. Salay, M. Chechik, and A. Di Sandro, "Supporting early decision-making in the presence of uncertainty," in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*. IEEE, 2014, pp. 33–42.

[98] J.B. Warmer and A.G. Kleppe, *The object constraint language: Precise modeling with UML (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 1998.

[99] R. Salay, M. Famelis, and M. Chechik, "Language independent refinement using partial modeling," in *Fundamental Approaches to Software Engineering*, J. de Lara and A. Zisman, Eds. Springer Berlin Heidelberg, 2012, pp. 224–239.

[100] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.

[101] W. Emmerich, B. Butchart, L. Chen, B. Wassermann, and S. Price, "Grid service orchestration using the business process execution language (bpel)," *Journal of Grid Computing*, Vol. 3, No. 3-4, 2005, pp. 283–304, cited By 104.

[102] A. Rahman and D. Amyot, "A DSL for importing models in a requirements management system," in *4th Int. Model-Driven Requirements Engineering Workshop (MoDRE)*. IEEE CS, 2014, pp. 37–46.

[103] IBM, "Rational DOORS v9.6.1," 2018. [Online]. http://goo.gl/yGWpze

[104] B.H.C. Cheng, R. De Lemos, H. Giese, P. Inverardi, and J. Magee et al., "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software engineering for self-adaptive systems*, Vol. LNCS 5525. Springer, 2009, pp. 1–26.

[105] S.A. Alwidian, M. Dhaouadi, and M. Famelis, "A vision towards a conceptual basis for the systematic treatment of uncertainty in goal modelling," in *SAM'20: 12th System Analysis and Modelling Conference*, A. Gherbi, W. Hamou-Lhadj, and A. Bali, Eds. ACM, 2020, pp. 139–142.

[106] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, Vol. 8, No. 3, 2004, pp. 203–236.

[107] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with goal models," in *International Conference on Conceptual Modeling*. Springer, 2002, pp. 167–181.

[108] J.O. Kephart and W.E. Walsh, "An artificial intelligence perspective on autonomic computing policies," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*. IEEE, 2004, pp. 3–12.

[109] R. de Lemos, H. Giese, H.A. Müller, M. Shaw, J. Andersson 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.

[110] Z. Yang, Z. Li, Z. Jin, and Y. Chen, "A systematic literature review of requirements modeling and analysis for self-adaptive systems," in *Requirements Engineering: Foundation for Software Quality*, C. Salinesi and I. van de Weerd, Eds. Springer, 2014, pp. 55–71.

[111] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research – An initial survey," in *SEKE*, 2010, pp. 374–379.

[112] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Verbeek, and A. Chatzigeorgiou, "Identifying, categorizing and mitigating threats to validity in software engineering secondary studies," *Information and Software Technology*, Vol. 106, 2019, pp. 201–230.

[113] B. Combemale, J.A. Kienzle, and G. Mussbacher et al., "A hitchhiker's guide to model-driven engineering for data-centric systems," *IEEE Software*, Vol. 38, No. 4, 2021, pp. 71–84.

A. Quality Assessment

This appendix complements Section 2.2.4 by presenting, in Tables A1 and A2, the result of the quality assessment against the criteria explained in Table 2. The color coding reflects how positive a result is (green is positive, yellow is neutral, and red is negative).

Table A1. 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 [20]	Y	Y	Y	N	N	Y	Y	Y	N
Ahmad 2010 [59]	Y	Y	Y	Y	N	N	?	Y	P
Ahmad et al. 2012 [56]	Y	Y	N	Y	Y	Y	Y	Y	Y
Ahmad et al. 2013 [57]	Y	?	?	?	?	Y	?	P	N
Ahmad et al. 2013 [58]	Y	Y	N	Y	Y	Y	Y	P	P
Ahmad et al. 2015 [55]	Y	Y	Y	Y	Y	Y	Y	Y	Y
Ahmad and Bruel 2013 [60]	Y	Y	N	Y	Y	Y	Y	P	P
Ahmad and Bruel 2014 [61]	Y	?	?	?	?	N	?	Y	Y
Anda and Amyot 2020 [31]	Y	Y	Y	Y	Y	Y	Y	N	N
Anda and Amyot 2020 [73]	Y	Y	Y	Y	Y	N	?	Y	Y
Anda 2018 [75]	Y	Y	Y	Y	Y	N	?	Y	Y
Anda and Amyot 2019 [74]	Y	Y	Y	Y	Y	N	?	Y	Y
Apvrille and Roudier 2013 [63]	Y	Y	Y	Y	Y	N	?	N	N
Apvrille and Roudier 2015 [64]	Y	Y	N	Y	Y	Y	Y	N	N
Badreddin et al. 2016 [68]	Y	Y	Y	P	P	Y	Y	Y	N
Belloir et al. 2014 [62]	Y	Y	N	Y	Y	Y	Y	Y	P
Bousse 2012 [54]	Y	P	P	N	N	N	?	N	N
Cui and Paige 2012 [50]	Y	Y	Y	Y	Y	Y	P	N	N
Fan et al. 2016 [69]	Y	Y	Y	Y	Y	Y	Y	N	N
Gnaho et al. 2013 [51]	Y	Y	N	Y	Y	Y	Y	N	N
Gnaho et al. 2013 [52]	Y	Y	P	Y	Y	N	?	N	N
Ingram et al. 2014 [45]	Y	Y	Y	Y	Y	Y	Y	Y	Y
Laleau et al. 2014 [49]	Y	Y	Y	Y	Y	Y	Y	N	N
Lee et al. 2019 [71]	Y	Y	Y	Y	Y	N	?	N	N
Mammar and Laleau 2016 [53]	Y	Y	N	Y	Y	Y	P	N	N
Maskani el al. 2018 [72]	Y	Y	Y	Y	Y	Y	Y	N	N
Matoussi et al. 2011 [48]	Y	Y	N	Y	Y	N	?	N	N
Ozkaya 2007 [47]	Y	P	Y	N	N	N	?	N	N
Roudier and Apvrille 2015 [65]	Y	Y	N	Y	Y	Y	Y	N	N
Spyropoulos and Baras [67]	Y	Y	Y	Y	Y	Y	Y	P	N
Tsadimas et al. 2012 [66]	Y	Y	Y	Y	Y	Y	Y	N	N
Vanderperren and Dehaene 2005 [46]	Y	P	Y	P	N	N	?	P	N
Wang 2018 [70]	Y	Y	Y	Y	Y	Y	Y	N	N

Table A2. 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 [86]	Y	Y	Y	P	Y	Y	Y	Y	P
Baresi and Pasquale 2010 [87]	Y	Y	P	Y	Y	Y	P	Y	Y
Baresi and Pasquale 2010 [88]	Y	Y	P	P	Y	Y	Y	Y	P
Horkoff et al. 2014 [97]	Y	Y	Y	Y	Y	N	?	N	N
Morandini et al. 2008 [82]	Y	Y	N	N	Y	Y	Y	P	N
Morandini et al. 2017 [83]	Y	Y	Y	Y	Y	Y	Y	Y	Y
Qian et al. 2015 [94]	Y	Y	Y	Y	Y	N	?	Y	Y
Ramnath et al. 2008 [95]	Y	Y	Y	Y	Y	Y	Y	Y	Y
SysML and Adaptation									
Akbas and Karwowski 2013 [18]	Y	Y	N	Y	Y	Y	Y	N	N
Akbas et al. 2014 [19]	Y	Y	Y	Y	Y	Y	Y	N	N
Bareiß et al. 2016 [84]	Y	Y	Y	Y	Y	Y	Y	Y	Y
Ginigeme and Fabregas 2018 [96]	Y	Y	Y	Y	Y	N	?	N	N
Hussein et al. 2017 [89]	Y	Y	Y	Y	Y	Y	Y	N	N
Lopes et al. 2011 [92]	Y	Y	Y	Y	Y	Y	P	P	P
Meacham 2017 [90]	Y	Y	Y	Y	Y	Y	Y	Y	N
Parri et al. 2020 [85]	Y	Y	Y	Y	Y	Y	Y	Y	P
Ribeiro et al. 2013 [91]	Y	Y	Y	Y	Y	Y	Y	P	N
Soyler and Sala-Diakanda 2010 [17]	Y	Y	Y	P	P	Y	Y	P	P
Souza et al. 2020 [93]	Y	Y	Y	P	P	Y	P	P	N