2018 IEEE 26th International Requirements Engineering Conference

Modeling Adaptive Socio-Cyber-Physical Systems with Goals and SysML

Amal Ahmed Anda
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, ON, Canada K1N6N5
Email: aanda027@uottawa.ca

***Abstract*—Socio-cyber-physical systems (SCPSs) are cyber-physical systems (CPSs) with a socio-technical system (STS) aspect. Several SCPSs need to adapt dynamically to changing situations in order to reach an optimal symbiosis with users in their contexts. Tailoring requirements engineering activities and modeling techniques is needed for developing SCPSs and supporting their runtime adaptability. The proposed thesis aims to combine a common way of modeling STSs (i.e., goal modeling with the Goal-oriented Requirement Language – GRL) to a conventional way of modeling CPSs (i.e., SysML and feature models) in order to integrate social concepts early in SCPS requirements, design, simulation, and implementation activities. To help guarantee system quality and compliance during both design time and runtime adaptations, the thesis proposes to translate goal and feature models to mathematical functions used to validate possible design and adaptation alternatives both during simulations at design time and adaptations at runtime. These functions can be used outside goal modeling tools and be combined to SysML models, simulations, problem solvers, and implementation tools. Furthermore, an integration between GRL and SysML models via a third-party requirements management system is proposed in order to strengthen system traceability and help ensure that stakeholder goals were considered properly during the SCPS development process.**

I. Introduction

Socio-cyber-physical systems (SCPSs) have emerged when social concerns, typical of socio-technical systems (STSs), became part of cyber-physical systems (CPSs). Common examples of SCPSs include air traffic control systems, intelligent transportation systems, smart cities, smart homes, and adaptive Systems of Systems (SOSs). These systems consider human concerns in addition to more conventional cyber, software and hardware concerns and components. Due to environmental and situational changes, some SCPSs may also need to adapt dynamically to reach an optimal symbiosis between users and their contexts [1]. Several challenges related to these kind of systems were addressed in the literature [1], [2] where model-driven engineering (MDE) approaches and goal-oriented requirements engineering (GORE) were both presented as solutions. Traceability should also be managed between many types of artifacts such as goals, requirements, design, and code [3]. Traceability is important to support system consistency/completeness assessment, validation, verification, change management, and impact analysis. Moreover, in order to help guarantee system quality and compliance, system and stakeholder goals should be considered during system evolution at design time but also at runtime while adapting to new purposes.

Lace and Kirikova [4] as well as Botanegra et al. [5] observed that current requirements engineering (RE) activities and languages need improvement in order to satisfy the emergent needs of SCPSs and adaptive systems. The integration between MDE and GORE shows potential as a solution here. Such integration could be particularly realized with two standard languages: 1) the *Systems Modeling Language* (SysML) for software and hardware elements [6], and 2) the *Goal-oriented Requirement Language* (GRL) for stakeholder goals and adaptation support [7].

SysML is a profile of UML that supports modeling systems with hardware and software components, such as CPSs and SOSs. SysML reuses part of UML, including use case, sequence, activity, and state machine diagrams, and replaces class diagrams with block and internal block diagrams. Moreover, SysML adds two new types of diagrams (parametric and requirements), which facilitate the connection between system components and their requirements [6]. SysML aims to enable the modeling of software and hardware components as well as their relationships within a single environment while simplifying their design and reducing their complexity [6]. Although SysML modelers can connect requirements to different design elements such as use cases, test cases, and blocks, SysML lacks important social modeling concepts for SCPS such as stakeholders and their goals/objectives.

GRL, part of the User Requirements Notation (URN) standard [8], is used to model and analyze stakeholder and system objectives as well as their relationships. GRL supports quantitative and qualitative trade-off and what-if analyses [7] usable at design time but also in an adaptation context [9]. Note that GRL also supports Key Performance Indicators (KPIs) to monitor contextual information [9] and support evidence-based decision making. A KPI assesses a current observable value against target, threshold, and worst value parameters, and outputs a satisfaction level that can be propagated to tasks, goals, and softgoals in the model. KPIs can be used 1) at design time via evaluation strategies or external data sources (database, web server, Excel sheet, etc.) and 2) at runtime using monitoring sensors and real-time data [10]. GRL is an excellent candidate for integration with SysML because it supports not only goals but also indicators (for the monitoring

2332-6441/18/$31.00 ©2018 IEEE
DOI 10.1109/RE.2018.00059

RE 2018, Banff, Canada
Doctoral Symposium Paper

of contexts in an adaptation situation), it is a standardized language, and it has relevant types of tool-supported analyzes.

A GRL model can be used to evaluate different strategies composed of initial satisfaction values attached to goals and other elements. Strategies can correspond to configurations of alternatives during design and adaptation [10]. However, the validity of these configurations and strategies should be determined by the system features variability, possibly formalized as a feature model where features and qualities (e.g., GRL softgoals) can be monitored [11]. In order to develop a SCPS able to adapt to its users concerns while monitoring quality and compliance, we propose a modeling approach that integrates GRL, SysML and feature models. By linking stakeholders goals and social concerns to a SCPS's design and implementation, we aim to support developers in producing rational system architectures and adaptation solutions.

II. Motivation

Development process of SCPSs faces several challenges such as modeling user goals beside hardware and software elements, managing a traceability between models, dealing with emergent properties, managing a high level of uncertainty, reducing the complexity as well as adapting according to external and internal changes. However, traditional requirements engineering processes have trouble dealing with these needs, and further improvements are required [4], [12]. Such improvements have been explored in literature with a particular focus on SCPSs that suffer from most of the above challenges [4]. Moreover, in order to support the development process of adaptive systems, the need for integrating goals and their evaluation criteria with SysML was identified by the SysML 2.0 request for proposals [13].

III. Problem Statement

SysML and goal models can be integrated in many different ways to support the development of SCPSs. For example, Amyot et al. [7] highlighted different types of integrations at the language level (e.g., by bringing goals into SysML, integrating tools, or using third-party tools for traceability or requirements management), and our recent literature review [14] surveyed existing approaches, including those related to adaptation. Yet, existing integration solutions show severe limitations in terms of traceability, usability, analysis, and support for adaptation at design time and run-time. In this context, our research questions are:

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

In order to support SCPS evolution during development and runtime adaptation, the focus will be on improving requirements engineering activities through several integrations between GRL, feature models and SysML during system design, simulation, and implementation.

IV. Related Work

The **integration** between SysML and goal models has been explored in several pieces of research, for self-adaptation support and other reasons [14]. Ahmad et al. [12] integrated the RELAX [15] language with KAOS [16] and SysML models for relaxing requirements that are varying at runtime depending on environmental conditions, in order to model uncertainty and adaptation. A profile has been used to extend SysML requirements diagrams to represent functional and non-functional goals. However, the presented method has not focused on system dynamic behavior and important information was transferred neither to the system design nor to the system implementation, including contribution relationships (with weights) between goals. Similarly, for system tracing and reasoning, Cui and Paige [17] integrated OMG's Business Motivation Model [18] with SysML using profiles. Requirement diagrams were extended by different levels of business motivation and their relationships, except the weights of the contribution relationships and indicators. In fact, remodeling goals with SysML design tools through profiles causes information loss, inconsistent models and duplication of work. Furthermore, this approach hurts traceability with 1) a lack of model synchronization, 2) low usability in terms of model import and export, and 3) low scalability of the SysML requirements diagram [7]. This consequently leads to unmanageable traceability that hurts model consistency, the change management process, impact analysis, as well as completeness and consistency checks.

In addition, managing goal models at runtime is the solution for conducting trade-off analysis and selecting the best adaptation strategy using real-time data. Although some research approaches proposed translating goal models to the system implementation, they have not exploited goal analysis in all relevant steps of a self-adaptation process (Monitor, Analyze, and Plan) while executing. For example, Qian et al. [19] used goal model reasoning only when their adaptation method failed in finding a solution, and this model was not used during monitoring, analysis, or strategy selection. Morandini et al. [20] extended goal models with different conditions and used goals as agents with conditions to trigger the first successful adaptation plan. Again, even when a goal's condition is violated, goal reasoning is not used during monitoring, analysis or planning. On the other hand, Baresi and Pasquale [21] employed goal-based reasoning in the monitoring and analysis steps but not in the planing process, where they used conditions attached to system operations. Overall, trade-off analysis and selecting the best adaptation solution at runtime are functionalities negatively affected by all these methods because the goal model is not mapped rigorously/completely in the system design and used effectively in the implementation. Hence, important information with a key role in self-adaptation is lost or ignored during this process.

Finally, to support the design selection process through simulation and runtime adaptation, some approaches attempted to transform goal models and/or feature models into **mathematical functions**. Ramirez and Cheng [22] proposed the Athena approach, which uses the KAOS and RELAX goal languages. In order to monitor the surrounding environment, Athena generates fuzzy functions automatically for softgoals while using templates that return Boolean values for functional goals. The values of the generated functions are propagated to calculate the overall satisfaction level and determine whether the goals are violated or not. However, the approach does not cover contribution relationships, contribution weights, and the relative importance of each goal in the analysis process. Moreover, the proposed approach does not support trade-off analysis or selection between alternatives to provide the most-suited adaptations.

Because of the difficulty of conducting goal-based reasoning of a large goal model, Chitra et al. [23] used a multi-objective optimization based on the satisfaction levels of the goal model. They identified the weights of the leaf softgoals automatically. Next, these weights were propagated to calculate the satisfaction level of the top softgoal. The satisfaction level of each leaf softgoal was used as argument of the main multi-objective function to select the solution that maximized the satisfaction level of the softgoals. However, this function could generate unfeasible solutions caused by invalid combinations of alternatives. Moreover, functional goals, relationships between goals, indicators and importance values of model elements were not involved in the analysis.

In contrast, in order to support the software product lines (SPL) and product reuse, Noorian et al. [24] considered goals, softgoals and features in selecting a product using an optimization model. The required goals and softgoals of the new product were selected by a user and an objective function was then built by the summation of the impact of each feature on the selected softgoals and goals separately. Three types of constraints (features, goals, and their integration) were used as rules with the proposed utility function to eliminate the invalid configurations. However, only part of the goal model was involved in the optimization model and the utility function does not represent goal-based reasoning in which the softgoals, the goals and their relationships are involved.

To conclude, although goal/feature models were transformed to mathematical functions to deal effectively with the complexity and scalability of large models, in reality, the proposed functions were often incomplete or imprecise, and sometimes they were not representing goal-based reasoning.

V. Solution Overview

We plan to address the unmanageable traceability and related issues that were mentioned in the above section by integrating GRL and SysML models using a third-party traceability system (e.g., a requirements management system – RMS – such as IBM DOORS [25]). Moreover, in order to support self-adaptation, the goal model will be transformed to a complete mathematical function for effective and efficient goal-based reasoning in SysML models and in implementations. In addition, a feature model will be used to constrain design and adaptation alternatives in the goal model to configurations that are valid globally. The feature model will hence influence the generation of the arithmetic function. With a particular focus on SCPSs, these goal-based functions will be transferred and employed in the right places (Monitor, Analyze and Plan activities) to strengthen the effectiveness and benefits of goal-based reasoning.

VI. Expected Contributions

Answering the research questions will contribute in adapting some requirements engineering activities to become suitable for emergent SCPSs using existing tools. The expected contributions of the thesis include:

1) Integrated social concerns with:
   a) Design activities, with a traceability that will be managed by a third-party RMS for: 1) system validation, 2) rationale documentation, 3) goal-design consistency and completeness check, and 4) change management and impact analysis.
   b) Simulation activities to evaluate design and adaptation alternatives by 1) verifying the developed SCPS models and 2) selecting the right design architecture and alternatives that are able to adapt according to both user preferences and conflicts.
   c) System implementation, to support runtime self-adaptation based on monitored data, user preferences, and performance objectives.
2) Automated transformation of goal models to mathematical functions. This will provide a way to support runtime system adaptation, but also research on goal-based reasoning/analysis outside goal modeling tools (such as SysML simulation tools). Without mapping or remodeling the goals, these functions can be transformed to several languages such as Java, Matlab, and C++, ready to be integrated in operational models and runnable adaptation code.
3) Automated transformation of feature models to mathematical functions that can be used with goal models as constraints to control system quality and compliance while designing, executing, and adapting the system dynamically.
4) A practical method for SCPS modeling that takes advantage of existing tools (e.g., jUCMNav for GRL/feature modeling/analysis [26], Papyrus [27] for SysML modeling/simulation, and IBM DOORS as the RMS).

VII. Originality

The originality of the proposed approach comes from two contributions:

1) A new integration of GRL, feature, and SysML models to support managing traceability without duplication of work while eliminating the negative impacts of information loss. Such integration is an emergent need of the modeling of SCPSs [4] and self-adaptive systems [13].

2) An automated transformation from goal and feature models to complete arithmetic functions in multiple programming languages. These functions can be used in several contexts such as system simulation (in SysML) to select system design, and goal-based reasoning at runtime to support self-adaption while handling the complexity of real-world systems.

VIII. Proposed Approach

The successful development of SCPSs requires engineers to strengthen the link between stakeholder goals and system artifacts at development time and runtime while minimizing development effort and information loss. This was the inspiration for our approach, which focuses on improving requirements engineering activities by injecting social concerns into SCPS design, simulation, and implementation activities. This leads to three main areas of interest: traceability, design time adaptation through simulation, and runtime adaptation.

The input of our approach is composed of SysML, GRL, and feature models, together with connections between them. The GRL model will be transformed in two ways: 1) to DOORS objects for traceability, through the use of MI-DSL transformations [28] and 2) to mathematical functions for simulation (e.g., as part of SysML models) and runtime adaptation (e.g., combined to implementation code). SysML models will also be imported in DOORS for traceability and used for simulation and code generation.

The overall approach is described in Figure 1 where the involved RE activities are: 1) traceability management between goal and SysML model elements, 2) simulation supporting GRL and feature models as mathematical functions, and 3) system code implementation injected by GRL and feature models as mathematical functions. The description of the modeling process is given in the following subsections.

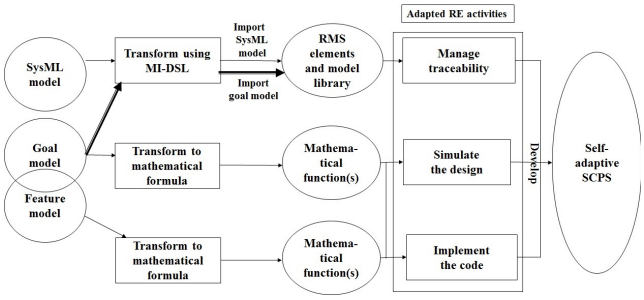

Fig. 1. Proposed approach to integrate SysML, GRL, and feature models for SCPS development.

A. Managing traceability at design time

A SCPS development process particularly needs a comprehensive traceability that follows as well as illustrates the requirements in both directions, stakeholders' goals, and system design and implementation [3]. This approach supports system developers in managing the problem space as well as validating and verifying the selected architecture. As this kind of traceability is needed for SCPSs development, it is also required in software product lines (SPL) and system reuse [29]. Based on the result of Amyot et al. [7], which explored the efficiency of means to manage traceability between goal and SysML models, and based on the results of the comparison of existing approaches integrating goal and SysML models for self-adaptation and other reasons [14], the suggested approach will integrate SysML models with GRL/feature models through a third-party repository (i.e., a commercial RMS, namely DOORS to begin with). As this method promotes the synchronization of models and their consistency, and as DOORS is a mature product used in many companies that model with SysML, this approach should also minimize the efforts needed for the integration. In the same context, GRL models have already been integrated with the DOORS RMS [28] for traceability purposes through MI-DSL. This process is visualized with bold lines in Figure 1. SysML models can also be integrated with DOORS by extending a tool such as Papyrus with a simple export mechanism invoking the MI-DSL library [28]. As a result, the elements of both the GRL model and the SysML model will be represented in the RMS repository, as shown in Figure 1. At this point, we can take advantage of the RMS to define and exploit traceability between models. This approach will help answer our first research question. There are, however, challenges in deciding what traceability information to track across the goal and SysML models, but different combinations can be explored by configuring MI-DSL descriptions accordingly.

B. Goal models for simulations and runtime adaptation

Self-adaptation support should be considered early while selecting system design and architecture, and later while implementing the SCPS. Therefore, reusing the goal model (or part thereof) in a simulation context and at runtime is desirable. We aim to support designers when choosing between design alternatives by transforming goal and feature models to mathematical functions that can be executed outside goal modeling tools through SysML, simulation tools, or constraint solver tools such as JaCoP [30]. This approach can help designers consider user preferences/objectives as well as conflicts when choosing architectures and selecting among alternative designs. Similarity, having the goal model runnable in a SysML-based simulation can help explore design-time adaptation by considering and prioritizing the embedded adaptation strategies according to user preferences and context. In addition to simulation support, we are interested in solving goal models (in GRL) at run-time while a system is executing. This method is of particular interest to adaptive systems, where goals are often used to react/adapt to a monitored context through indicators, sensors, and other information sources [14]. The complete goal model (actors, resources, goals, soft goals, tasks, indicators, relationships, weights, etc.) needs to be encoded in the application's programming language (Java, C, etc.). A transformation from GRL models to mathematical functions is desirable here. However, typically, a goal model alone may not encode all the restrictions needed to ensure the validity

of global solutions [11]. The validity of these configurations (and corresponding GRL strategies) can be determined by a companion feature model where the features correspond to the leaves of a GRL model.

A secondary problem is that as the number of target languages for specifying functions can be high, we would like to minimize the effort required for supporting many languages. Ideally, an intermediate representation for mathematical functions can be used as a target for GRL/feature model transformations, with additional (but small, and hopefully already existing) transformations to specific programming languages.

Using this approach, we hope to answer the second research question and support an effective and consistent goal-based reasoning during runtime adaptation activities (monitoring, analysis, and planning). The produced SCPSs will be characterized by: 1) being rational about system requirements and design elements early at design time, 2) paying attention to their stakeholders' goals and their qualities when selecting design alternatives, 3) being able to adapt dynamically to their user preferences and monitored environment, and 4) being able to validate generated (feature) configurations and (goal model) strategies both at design time and at runtime.

C. Illustrative example

To illustrate the approach, we use a much simplified GRL model of a hybrid car's engine system. As shown in Figure 2, an optimal decision needs to balance the actors' goals (i.e., comfortable driving and acceleration). The environment is monitored using a vibration sensor/indicator (target=0, threshold=10, worst=20, units=Hz). Calculating the overall satisfaction of both actors (user and system) enables deciding whether to change the currently selected tasks. If yes, the system selects the combination of tasks (electric engine and/or gas engine) that satisfies the acceleration goal while maximizing driving comfort, depending on current environmental conditions. The following code snippet illustrates the C++ mathematical function representing this calculation.

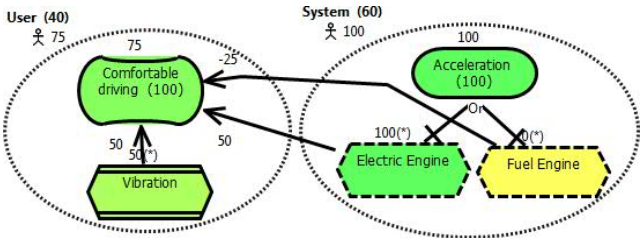

Fig. 2. The simplified GRL model of a hybrid car's engine system.

```
#include <iostream>
using namespace std;
double Adaptivecar(double Vibration,double
    Electric_engine,double Fuel_engine){
  double CarSatisfaction, Vib_Satisfaction;
  if (Vibration <= 0.0) { Vib_Satisfaction=100.0;
    }
  else if (Vibration > 0 && Vibration <= 10.0)
    { Vib_Satisfaction=std::fabs(0.1*VibratioN -
        1.0)*50 + 50.0; }
    else if (Vibration > 10.0 && Vibration < 20.0)
      { Vib_Satisfaction= -std::fabs(0.1*VibratioN
          - 1.0)*50 + 50.0; }
      else { Vib_Satisfaction=0; }
  CarSatisfaction = 0.40*std::max(0, std::min
      (100.0, 0.50*Electric_engine - 0.25*
      Fuel_engine + 0.50*Vib_Satisfaction +0.60*
      std::max(Electric_engine, Fuel_engine)));
  return CarSatisfaction;
}
```

IX. Methodology

As this research aims to produce a method artifact with partial tool support in a context where new requirements are discovered as the research evolves, the *Design Science Methodology* (DSM) [31] was selected. DSM proposes iterative steps to reach feasible and efficient solutions for research needs leading to artifacts that increase the abilities of humans and organizations.

According to DSM, the identified artifacts are developed and evaluated based on rigorous methods following seven guidelines described by Hevner et al. [31]. In DSM, the developed artifacts (i.e., tools, methods, models, and/or their instantiations) are developed iteratively using the feedback of the evaluation process.

I instantiated DSM to suit my research context and artifacts. I plan on following the DSM steps in general but, because I am designing two related artifacts (an integration and development method for SCPSs and a transformation tool from GRL to functions in programming languages), the DSM's iteration cycle is adapted to include 4 steps: further develop the method, further develop the tool, evaluate the method and evaluate the tool. The method will be evaluated iteratively using an ongoing illustrative example (a hybrid car) and two case studies. The first case study will be done in the context of IoT-based intelligent transportation, in collaboration with the *Socit de Transport de l'Outaouais* (the public transportation system in Gatineau, Canada) whereas we expect the second case study to be done in collaboration with another industrial partner (likely Thales in Ottawa).

X. Progress

We have conducted a systematic literature review to explore the contributions of English articles that integrate goal and SysML models and that support comprehensive requirements engineering, modeling techniques, and self-adaptation [14]. Articles that used goal models or SysML models separately for adaptation were further investigated, but less systematically.

We have explored different criteria and options for the integration of GRL with SysML, with or without an RMS for traceability management [7].

In addition, we have prototyped a transformation from GRL models to mathematical functions using the jUCMNav modeling tool for URN [26]. Instead of creating separate transformations for different programing languages, one intermediate representation was targeted: *SymPy* [32]. SymPy uses a Python library for symbolic mathematics that has code generators for different programming languages. Using

this tool, we can already generate arithmetic and runnable representations of goal models in Java, JavaScript, C, C++, R, Python and Matlab.

So far, the above points led to one publication and two recent paper submissions where I am first author or co-author.

XI. Conclusion

Socio-cyber-physical systems consider human goals as a part of their concerns. Hence, these concerns should be considered while configuring these systems at design time and at runtime. Although mapping or remodeling goals at design time to realize goal and SysML models integration is the most common approach in the literature, it is also time consuming and leads to information losses. Despite the fact that combining goal and SysML modeling into one tool is a desirable solution for solving traceability issues, achieving such an integration with current design tools remains a challenge. Because adjusting requirements engineering activities to follow the adaptive SCPSs is an urgent need, we proposed an approach to integrate social and goal-oriented concepts with system design, simulation, and implementation activities while monitoring system quality and compliance. In the proposed approach, users goals will be considered early when choosing a system architecture and design alternatives as well as when supporting dynamic adaptation at runtime. As this approach will be realized without work repetition or information loss through the generation of functions from goal/feature models to be embedded in SysML models and implementations, the existing tools will also be used to minimize coding errors and the effort needed for these adjustments.

Acknowledgment

I would like to thank the Libyan Ministry of Education for its financial support. I am thankful to Prof. Daniel Amyot for his support and help. I am also thankful to Yuxuan Fan for having prototyped a SymPy formula generator for GRL models in the jUCMNav tool.

References

[1] 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.

[2] 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, p. 17, 2015.

[3] K. Welsh, N. Bencomo, P. Sawyer, and J. Whittle, "Self-explanation in adaptive systems based on runtime goal-based models," in *Transactions on Computational Collective Intelligence XVI*. Springer, 2014, pp. 122–145.

[4] K. Lace and M. Kirikova, "Required changes in requirements engineering approaches for socio-cyber-physical systems," in *CRE+FIRE, REFSQ-JP 2018*. CEUR-WS, Vol-2075, 2018.

[5] 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.

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

[7] 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.

[8] ITU-T, *Recommendation Z.151 (10/12): User Requirements Notation (URN) - Language Definition*, Std., 2012, http://www.itu.int/rec/T-REC-Z.151/en.

[9] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating goal models within the goal-oriented requirement language," *International Journal of Intelligent Systems*, vol. 25, no. 8, pp. 841–877, 2010.

[10] A. Pourshahid, I. Johari, G. Richards, D. Amyot, and O. S. Akhigbe, "A goal-oriented, business intelligence-supported decision-making methodology," *Decision Analytics*, vol. 1, no. 1, p. 9, 2014.

[11] M. B. Duran and G. Mussbacher, "Investigation of feature run-time conflicts on goal model-based reuse," *Information Systems Frontiers*, vol. 18, no. 5, pp. 855–875, 2016.

[12] 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, pp. 50–70, 2015.

[13] OMG, *Systems Modeling Language (SysML) v2 Request For Proposal (RFP)*, Std., 2017, http://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-2.

[14] A. A. Anda and D. Amyot, "Self-Adaptation Driven by SysML and Goal Models: A Literature Review," *Systems Engineering*, 2018, (submitted).

[15] 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, pp. 177–196, 2010.

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

[17] 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.

[18] OMG, *Business Motivation Model (BMM), version 1.3*, Std., 2015, https://www.omg.org/spec/BMM/1.3/.

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

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

[21] 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.

[22] A. J. Ramirez and B. H. C. Cheng, "Automatic Derivation of Utility Functions for Monitoring Software Requirements," in *Model Driven Engineering Languages and Systems (MODELS)*. Springer Berlin Heidelberg, 2011, pp. 501–516.

[23] M. Chitra, Subramanian, A. Krishna, and A. Kaur, "Optimal goal programming of softgoals in goal-oriented requirements engineering." in *PACIS*, 2016, p. 202.

[24] M. Noorian, E. Bagheri, and W. Du, "Toward automated qualitycentric product line configuration using intentional variability," *Journal of Software: Evolution and Process*, vol. 29, no. 9, p. e1870, 2017.

[25] IBM, *Rational DOORS V9.6.1*, Std., 2018, goo.gl/yGWpze.

[26] D. Amyot, A. Shamsaei, J. Kealey, E. Tremblay, A. Miga, G. Mussbacher, M. Alhaj, R. Tawhid, E. Braun, and N. Cartwright, "Towards Advanced Goal Model Analysis with jUCMNav," in *Advances in Conceptual Modeling*. Springer Berlin Heidelberg, 2012, pp. 201–210.

[27] S. Gérard, C. Dumoulin, P. Tessier, and B. Selic, "Papyrus: A UML2 Tool for Domain-Specific Language Modeling," in *Model-Based Engineering of Embedded Real-Time Systems*, vol. LNCS 6100. Springer Berlin Heidelberg, 2010, pp. 361–368.

[28] 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, 2014, pp. 37–46.

[29] A. Palmieri, P. Collet, and D. Amyot, "Handling regulatory goal model families as software product lines," in *Int. Conf. on Advanced Information Systems Engineering*. Springer, 2015, pp. 181–196.

[30] K. Kuchcinski and R. Szymanek, "JaCoP: Java Constraint Programming Library," 2018, http://jacop.cs.lth.se/.

[31] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Q.*, vol. 28, no. 1, pp. 75–105, March 2004.

[32] SymPy Development Team, "SymPy," 2018, http://www.sympy.org/.