# A Systematic Literature Review of UML-based Domain-Specific Modeling Languages for Self-adaptive Systems

João Pablo S. da Silva
Institute of Informatics, Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil
jpsilva@inf.ufrgs.br

Miguel Ecar
Campus Alegrete, Federal University of Pampa
Alegrete, RS, Brazil
miguel@ecarsm.com

Marcelo S. Pimenta
Institute of Informatics, Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil
mpimenta@inf.ufrgs.br

Gilleanes T. A. Guedes
Campus Alegrete, Federal University of Pampa
Alegrete, RS, Brazil
gilleanesguedes@unipampa.edu.br

Luiz Paulo Franz
Campus Alegrete, Federal University of Pampa
Alegrete, RS, Brazil
luizpaulofranz@gmail.com

Luciano Marchezan
Campus Alegrete, Federal University of Pampa
Alegrete, RS, Brazil
lucianomarchezan94@gmail.com

## ABSTRACT

Self-adaptive Systems (SaSs) operate under uncertainty conditions and have intrinsic properties that make their modeling a non-trivial activity. This complexity can be minimized by using Domain-Specific Modeling Languages (DSMLs), which may be created by extending Unified Modeling Language (UML). In face of this, we propose investigating how the UML has been customized to create DSMLs that provide proper support for SaSs modeling. To achieve this, we performed a Systematic Literature Review (SRL) by retrieving studies with snowballing technique, selecting studies according to inclusion and exclusion criteria, and extracting and analyzing data to answer our research questions. As the outcome, we retrieved 786 studies and selected 16 primary studies published between 2005 and 2017. The results reveal that the class diagram has been customized through the profile-based mechanism to provide proper support to analysis and design of context-awareness and self-adaptiveness properties.

## CCS CONCEPTS

• **Software and its engineering → Designing software**; **Unified Modeling Language (UML)**;

## KEYWORDS

Systematic Literature Review (SLR); Snowballing Technique; Self-adaptive Systems (SaS); Domain-Specific Modeling Language (DSML); Unified Modeling Language (UML)

**ACM Reference Format:**
João Pablo S. da Silva, Miguel Ecar, Marcelo S. Pimenta, Gilleanes T. A. Guedes, Luiz Paulo Franz, and Luciano Marchezan. 2018. A Systematic Literature Review of UML-based Domain-Specific Modeling Languages for Self-adaptive Systems. In *SEAMS '18: SEAMS '18: 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3194133.3194136

## 1 INTRODUCTION

Self-adaptive Systems (SaSs) can autonomously decide how to adapt their behavior at runtime in response to contextual changes [3, 11, 12]. In the last years, software engineering research agendas have been formulated to address challenges of developing and maintaining SaSs [26]. In this work, we are particularly concerned with issues related to SaSs modeling. SaSs have intrinsic properties [33] and operate under uncertainty conditions [22], which make their modeling a non-trivial activity.

SaSs modeling complexity can be minimized by using Domain-Specific Modeling Languages (DSMLs). A DSML provides primitives to express higher-level abstractions and constraints of a targeted domain [13]. A way for creating a DSML is by extending the Unified Modeling Language (UML) metamodel [38]. In face of foregoing, we enunciated the following main question: **How the UML has been customized to create DSMLs that provide proper support for SaSs modeling?**

To answer this question, we first searched in scientific literature secondary studies[1] that investigated UML-based DSMLs for SaSs. Van Velsen et al. [39] carried out a literature review to discover how the user-centered evaluations of personalized systems have been conducted and how they can be improved. Weyns et al. [43] performed a study to understand and characterize the use of formal methods in SaSs. Patikirikorala et al. [32] investigated the design of SaSs using control engineering approaches. Becker et al. [5] provided a literature review of the state-of-the-art in performance engineering for SaSs.

Mohabbati et al. [29] aimed to identify and characterize the existing research on service-orientation and software product line engineering. Weyns and Ahmad [42] provided a comprehensive study that identifies claims and evidence for architecture-based self-adaptation. Yang et al. [45] carried out a literature review of

[1]According to Jalali and Wohlin [18], "research literature may be divided into primary studies (new studies on a specific topic) or secondary studies (summarizing or synthesizing the current state of research on a specific topic)".

requirements modeling and analysis for SaSs. Yuan et al. [46] performed a systematic survey of the state-of-the-art of self-protecting software systems using a classifying taxonomy. Brings et al. [10] proposed a search strategy for a systematic review on uncertainty. Siqueira et al. [36] performed a literature review to characterize the challenges for testing adaptive systems.

Analyzing the set of secondary studies mentioned above, we noticed that most of them have relevant contributions. However, we also perceived that they did not answer the main question enunciated before. For this reason, we propose a Systematic Literature Review (SLR) to answer this question by retrieving studies with snowballing technique, selecting studies according to inclusion and exclusion criteria, and extracting and analyzing data to answer our research questions.

As the outcome, we retrieved 786 studies and selected 16 primary studies published between 2005 and 2017. The SLR results reveal that the class diagram has been customized through the profile-based mechanism to provide proper support to analysis and design of context-awareness and self-adaptiveness properties.

The remainder of this paper is organized as follows. Section 2 provides a brief conceptual background. Section 3 presents the protocol that guided this SLR. Section 4 reports the execution and results of this SLR, besides answer the research questions. Section 5 discusses some threats to validity of our work. Finally, Section 6 presents our conclusions and future works.

## 2 CONCEPTUAL BACKGROUND

For a better understanding of this work, we present below an overview of self-adaptive systems, UML-based domain-specific modeling languages, and the snowballing technique.

### 2.1 Self-adaptive Systems

Self-adaptive Systems (SaSs) have several definitions in specialized literature [26]. However, the term "self-adaptive" is usually employed to characterize systems that can autonomously adapt their behavior at runtime [3, 31]. This type of systems is able to perceive contextual changes and (re)organize its features or services in response to these changes [11, 12]. As can be seen in Figure 1, SaSs have a set of intrinsic characteristics organized in three levels:

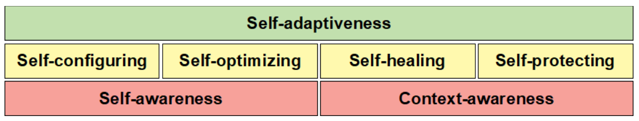

**Figure 1: SaSs intrinsic properties [33].**

- **primitive level** (bottom) contains the properties of awareness of itself and its context;
- **major level** (middle) contains the four autonomic properties (self-configuring, self-healing, self-optimizing, and self-protecting [20]);
- **general level** (top) contains the global property of self-adaptiveness [33].

### 2.2 UML-based DSML

Domain-Specific Modeling Languages (DSMLs) provide primitives to express higher-level abstractions and constraints of a targeted domain. They promote modeling productivity because software engineers may reuse concepts instead of specify them. Moreover, they contribute to quality because concepts include implicit integrity constraints, which prevent construction of nonsensical models [13]. DSMLs have been used for expressing concepts of many domains like Automotive, Embedded, Enterprise Applications, Industrial, Mobile, Telecommunication, User Interface Design, and Testing [28].

A way for creating a DSML is by extending the Unified Modeling Language (UML) metamodel [38]. UML is a graphical general-purpose modeling language broadly used by both industry and academy [8]. It provides two extension approaches that allows creating DSMLs:

- **metamodel-based** extends the UML metamodel by applying a language of next higher-metalevel, for example, the Meta Object Facility (MOF);
- **profile-based** extends the UML metamodel by applying mechanisms defined by UML itself, such as, stereotypes, tagged values, and constraints [38].

Metamodel-based extensions allow to modify the UML syntax and semantics [30], for example, it is possible to define a new syntax to represent views in a Mode-View-Controller (MVC) architecture or modify the attributes semantics to represent primary and foreign keys in entity–relationship model. Whereas, profile-based extensions allow to extend UML with concepts coming from a particular platform or domain [27], such as embedded systems domain or Enterprise JavaBeans (EJB) platform, but it is not allowed to modify existing metamodels [30].

### 2.3 Snowballing Technique

The snowballing technique refers to using study references and citations to identify additional studies [44]. According to Jalali and Wohlin [18], snowballing is able to yield similar results to database searches. Additionally, it may be more efficient than database searches because reduces the amount of false-positives. The first snowballing step is to define the initial studies list, which must be formed of relevant studies for review goals [40]. From the initial studies list, two iterative cycles are performed:

- **backward snowballing** consists in to investigate all studies cited by a study from the initial studies list;
- **forward snowballing** consists in to investigate all studies that cite a study from the initial studies list [40].

The backward and forward snowballing should be performed for each paper at once, until there is no more candidates for inclusion. It is important to execute each iteration at a time to get traceability. It is also important to decide on either inclusion or exclusion before starting to use a new study for snowballing. If there is a mistake in including a paper, the whole process is compromised and it should be rolled back [44].

## 3 REVIEW PROTOCOL

We performed this SLR according to guidelines presented by Kitchenham and Charters [21]. We present next, the research questions, searching process, selection process, and data extraction strategy that guided our work.

### 3.1 Research Questions

The research questions are important because they conduct all review process [21]. Our main objective is to investigate how UML has been customized to create DSMLs that provide proper support for SaSs modeling. Hence, we enunciated the following research questions:

**RQ-1** What modeling issues have motivated UML customization?

**RQ-2** How UML has been customized to support SaSs modeling?

Regarding the first question, we want to understand the challenges related to SaS modeling and how they have been addressed. Concerning the second question, we want to understand the customization processes and what UML primitives have been extended.

### 3.2 Search Process

In this SLR, we applied the snowballing technique to discover studies. According to this technique, first, we should establish an initial studies list, which is used as starting point for snowballing. To achieve it, we retrieved all papers published in International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). We chose SEAMS as the source because it is one of the main scientific events of the Software Engineering for Self-Adaptive Systems Community. After retrieving studies from SEAMS, we defined the initial studies list by applying the inclusion criteria (see Subsection 3.3).

Since the initial studies list has been defined, we performed the backward and the forward steps according to the snowballing technique. For each initial list study, we retrieved all papers cited by it (backward snowballing) and all papers that cite it (forward snowballing). The retrieved studies were verified to eliminate redundancies and ensure information completeness. We reapplied inclusion criteria in the retrieved papers to select new studies for the list. These steps (backward and forward snowballing) were iterated until no more new studies were found. Hence, in each iteration (also called round) we selected new relevant studies from the retrieved studies list.

### 3.3 Selection Process

We selected the relevant studies by applying the inclusion and exclusion criteria in the retrieved studies list. The **inclusion criteria** acted as the first filter, accepting studies directly related to the research objective. As stated previously, the inclusion criteria was applied in each snowballing round by a peer-reviewing process. We accepted studies that received “Yes” in all of the following questions:

- Is there in the abstract any mention to UML extensions?
- Is there in the abstract any mention to SaSs properties?

The **exclusion criteria** acted as the second filter, excluding studies that did not meet some general requirements. Differently from inclusion criteria, the exclusion criteria were applied, also by a peer-reviewing process, in the backward and the forward output. We rejected studies that received “No” in at least one of the following criteria:

- Is the study a full primary study?
- Is the study written in English?
- Does the study have at least six pages?
- Is not the study a repeated publication?

We noted that, in some cases, a study has more than one version (conference and journal, for example). In these cases, we considered the study as a repeated publication and selected the more complete version.

### 3.4 Data Extraction Strategy

The data extraction was performed by a peer-reviewing process, in which the reviewers read the selected studies and collected the following data:

- SaSs properties addressed;
- engineering models approached;
- UML diagrams customized;
- extension mechanisms applied.

After extraction, we tabulated the data to summarize the basic information about each study, ordering by publication year. The data were analyzed to answer the research questions based on the following simple strategy:

- to answer the RQ-1, we relate SaSs properties with engineering models;
- to answer the RQ-2, we relate UML diagrams with extension mechanisms.

## 4 EXECUTION AND RESULTS

According to the review protocol (see Section 3), we applied the snowballing technique to find studies. This activity was carried out on January 8-10, 2018. Table 1 presents the snowballing execution history. **Round 0** refers to the initial studies list definition, which is composed of all papers published in SEAMS between 2006 and 2017. **Rounds 1 to 6** refer to the execution of backward and forward snowballing. As the search process result, we retrieved 786 studies, of which 26 were accepted by applying the inclusion criteria.

**Table 1: Snowballing execution history.**

| Round | Retrieved | Selected |
|---|---|---|
| Round 0 | 222 | 2 |
| Round 1 | 59 | 4 |
| Round 2 | 69 | 2 |
| Round 3 | 31 | 6 |
| Round 4 | 254 | 9 |
| Round 5 | 100 | 3 |
| Round 6 | 51 | 0 |
| | 786 | 26 |

After finishing the backward and the forward execution, we concluded the selection process by applying the exclusion criteria.

As the output, we rejected 10 of 26 studies previously accepted. Hence, we accepted 16 primary studies[2] for this SLR, which are presented in Table 2.

Since the relevant studies list has been defined, we performed data extraction, as stated in the review protocol (see Section 3). Lastly, we analyzed the results to answer the research questions, as can be seen in next subsections.

## 4.1 What modeling issues have motivated UML customization?

Our objective with this question is to understand the challenges related to SaS modeling and how they have been addressed. To answer it, we extracted from selected studies the relation between SaSs properties and engineering models.

SaSs have intrinsic properties organized in a three level hierarchy. Primitive level has self-awareness and context-awareness properties. Major level has the four autonomic properties: self-configuring, self-healing, self-optimizing, and self-protecting. General level has self-adaptiveness property. Table 3 relates the SaSs properties with the selected studies, where we observe that 9 studies address context-awareness property, 1 study address self-configuring property, 1 study address self-protecting property, and 6 studies address self-adaptiveness property.

Models are abstractions of reality [8]. They help to cope with complexity, focusing on a determined perspective and hiding non relevant details. During development, several models are created to express aspects of the software in different perspectives [37]. We highlight the following models to develop a software: requirements, analysis, design, coding, testing, and deployment. Table 4 relates the engineering models with the selected studies, where we note that 4 studies address requirements models, 12 studies address analysis models, 13 studies address design models, and 2 studies address coding models.

Figure 2 presents a cross analysis that relates the SaSs properties with engineering models. We observe that each study can count more than once. From SaSs properties angle, we notice that the most addressed properties are context-awareness and self-adaptiveness. From engineering models perspective, we observe that the most addressed engineering models are analysis and design models. The cross analysis shows that context-awareness analysis and design, followed by self-adaptiveness design and analysis are widely exploited in the studies.

Therefore, according to this SLR results, we notice that issues related to context-awareness and self-adaptiveness analysis and design have motivated the creation of UML-based DSMLs for SaSs. We also highlight that there are gaps in relations between SaSs properties and engineering models. We did not found studies that address issues related to self-healing or self-protecting (from SaSs properties perspective) and testing or deployment (from engineering models perspective). Moreover, there are gaps for self-configuring and self-protecting requirements, self-configuring analysis, and context-awareness, self-configuring, and self-protecting coding.

[2]All retrieved studies, with their respective status, are available at https://goo.gl/xkVjTi.

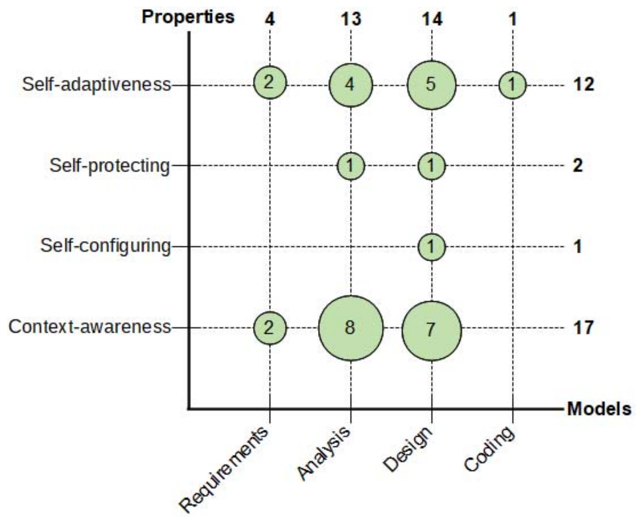

**Figure 2: SaSs properties *versus* engineering models.**

## 4.2 How UML has been customized to support SaSs modeling?

We intend with this question to understand the customization processes and what UML primitives have been extended. To accomplish it, we extracted the relation between customized diagrams and extension mechanisms from the selected studies.

Diagrams are graphical presentations of a set of elements [8]. There is in UML 2.5, a set of fourteen diagrams organized into two groups. Structural diagrams (class, component, composite structure, deployment, object, package, and profile) allow to represent the static system structure. Behavioral diagrams (activity, communication, interaction overview, sequence, state machine, timing, and use cases) allow to represent the dynamic object behavior in a system [30]. Table 5 relates the UML diagrams with the selected studies, where we observe that 10 studies customize class diagram, 3 studies customize component diagram, 4 studies customize use case diagram, 5 studies customize activity diagram, and 5 studies customize sequence diagram.

As mentioned before, there are two mechanisms to extend UML: metamodel-based and profile-based. Table 6 relates the UML extension mechanisms with the selected studies, where we note that 5 studies report metamodel-based extensions and 11 studies report profile-based extensions.

Figure 3 presents a cross analysis that relates the customized diagrams with extension mechanisms. We note that each study can count more than once. From UML diagrams perspective, we notice that the most customized structural diagram is class diagram and the most customized behavioral diagrams are activity and sequence diagrams. From extension mechanisms angle, we observe that the most used mechanism is profile-based. The cross analysis shows that the class diagram customization through the profile-based mechanism has been widely used.

Therefore, according this SLR results, we notice that the profile-based mechanism has been used to customize the class diagram, providing proper support to SaSs modeling. We point out that

Table 2: SLR selected studies.

| # | Title | First Author |
|---|---|---|
| 01 | ContextUML: a UML-based modeling language for model-driven development of context-aware Web services | Sheng [34] |
| 02 | MDD Approach for the Development of Context-Aware Applications | Ayed [4] |
| 03 | Modeling context in mobile distributed systems with the UML | Simons [35] |
| 04 | Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications | Fuentes [14] |
| 05 | PCP: Privacy-aware Context Profile Towards Context-aware Application Development | Kapitsaki [19] |
| 06 | Making control loops explicit when architecting self-adaptive systems | Hebig [16] |
| 07 | Adapt Cases: Extending Use Cases for Adaptive Systems | Luckey [25] |
| 08 | Extended UML for the Development of Context-Aware Applications | Benselim [6] |
| 09 | Extending UML to model Web 2.0-based context-aware applications | Hsu [17] |
| 10 | Specifying security requirements of context aware system using UML | Almutairi [2] |
| 11 | High-Quality Specification of Self-Adaptive Software Systems | Luckey [24] |
| 12 | Modelling context-aware RBAC models for mobile business processes | Schefer-wenzl [41] |
| 13 | An Extension of UML Activity Diagram to Model the Behaviour of Context-Aware Systems | Al-Alshuhai [1] |
| 14 | FAME: A UML-based framework for modeling fuzzy self-adaptive software | Han [15] |
| 15 | Heavyweight extension to the UML class diagram metamodel for modeling context aware systems in ubiquitous computing | Boudjemline [9] |
| 16 | Towards A UML Profile for Context-Awareness Domain | Benselim [7] |

Table 3: SaS properties addressed in the studies.

| | Context-awareness | Self-configuring | Self-protecting | Self-adaptiveness |
|---|---|---|---|---|
| Sheng [34] | ✓ | | | |
| Ayed [4] | ✓ | | | |
| Simons[35] | ✓ | | | |
| Fuentes [14] | | ✓ | | |
| Kapitsaki [19] | | | ✓ | |
| Hebig [16] | | | | ✓ |
| Luckey [25] | | | | ✓ |
| Benselim [6] | ✓ | | | |
| Hsu [17] | ✓ | | | |
| Almutairi [2] | ✓ | | | |
| Luckey [24] | | | | ✓ |
| Schefer-wenzl [41] | ✓ | | | |
| Al-Alshuhai [1] | | | | ✓ |
| Han [15] | | | | ✓ |
| Boudjemline [9] | ✓ | | | |
| Benselim [7] | ✓ | | | |

Table 4: Engineering models addressed in the studies.

| | Requirements | Analysis | Design | Coding |
|---|---|---|---|---|
| Sheng [34] | | ✓ | ✓ | |
| Ayed [4] | | ✓ | ✓ | |
| Simons[35] | | ✓ | ✓ | |
| Fuentes [14] | | | ✓ | ✓ |
| Kapitsaki [19] | | ✓ | ✓ | |
| Hebig [16] | | | ✓ | |
| Luckey [25] | ✓ | ✓ | ✓ | |
| Benselim [6] | | ✓ | ✓ | |
| Hsu [17] | | ✓ | ✓ | |
| Almutairi [2] | ✓ | | | |
| Luckey [24] | | | ✓ | ✓ |
| Schefer-wenzl [41] | | ✓ | | |
| Al-Alshuhai [1] | | ✓ | | |
| Han [15] | ✓ | ✓ | ✓ | |
| Boudjemline [9] | | ✓ | ✓ | |
| Benselim [7] | ✓ | ✓ | ✓ | |

none of the studies propose customizations for composite structure, deployment, object, or package structural diagrams and communication, interaction overview, state machine, or timing behavioral diagrams.

## 5 THREATS TO VALIDITY

Firstly, we highlight a risk related to our searching process. The ability to find relevant studies directly impacts the SLR results. In this SLR, we used the snowballing technique, which had its effectiveness demonstrated by Jalali and Wohlin [18]. Nevertheless, there is the possibility we have had unrecognized some relevant study in our searching process.

Another risk to be considered is related to the selection process. In this step, researchers apply inclusion, exclusion, and quality

Table 5: UML diagrams customized in the studies.

| | Class | Component | Use Case | Activity | Sequence |
|---|---|---|---|---|---|
| Sheng [34] | ✓ | | | | |
| Ayed [4] | ✓ | | | | ✓ |
| Simons[35] | ✓ | | | | |
| Fuentes [14] | ✓ | ✓ | | ✓ | ✓ |
| Kapitsaki [19] | ✓ | | | | |
| Hebig [16] | | ✓ | | | |
| Luckey [25] | ✓ | | ✓ | | ✓ |
| Benselim [6] | ✓ | | | | |
| Hsu [17] | ✓ | | | | |
| Almutairi [2] | | | ✓ | | |
| Luckey [24] | | ✓ | | ✓ | |
| Schefer-wenzl [41] | | | | ✓ | |
| Al-Alshuhai [1] | | | | ✓ | |
| Han [15] | ✓ | | ✓ | | ✓ |
| Boudjemline [9] | ✓ | | | | |
| Benselim [7] | | | ✓ | ✓ | ✓ |

Table 6: Mechanisms used to extend UML in the studies.

| | Metamodel-based | Profile-based |
|---|---|---|
| *Sheng [34] | | ✓ |
| Ayed [4] | | ✓ |
| Simons[35] | | ✓ |
| Fuentes [14] | | ✓ |
| Kapitsaki [19] | | ✓ |
| Hebig [16] | | ✓ |
| Luckey [25] | | ✓ |
| Benselim [6] | | ✓ |
| Hsu [17] | | ✓ |
| Almutairi [2] | ✓ | |
| Luckey [24] | ✓ | |
| Schefer-wenzl [41] | ✓ | |
| Al-Alshuhai [1] | ✓ | |
| Han [15] | | ✓ |
| Boudjemline [9] | ✓ | |
| Benselim [7] | | ✓ |

* Defined according to UML 1.x.

criteria to obtain the relevant studies. At this moment, the papers are not evaluated in depth. Therefore, the selection process may produce false-positives and false-negatives. To mitigate the risk related to screening of papers, we performed a peer review process, i.e., each paper was screened by two researchers.

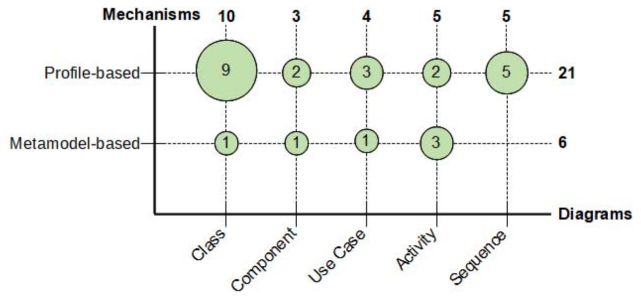

**Figure 3: Customized diagrams *versus* extension mechanism.**

At last, we note a risk related to data extraction. At this point, the researchers perform a qualitative analysis to extract the data that will answer the research questions. Conceptual background and interpretation capability may affect the research decisions and compromise the results. To mitigate this, we applied Delphi method [23] to obtain consensus about the extracted data.

## 6 CONCLUSIONS

In this paper, we report the outcomes of a SLR that investigated how the UML has been customized to create DSMLs that provide proper support for SaSs modeling. This SLR was carried out by retrieving studies with the snowballing technique, selecting studies according to inclusion and exclusion criteria, and extracting and analyzing data to answer our research questions. As the output, we retrieved 786 studies and selected 16 primary studies published between 2005 and 2017.

Our first research question asks what modeling issues have motivated UML customization. Cross analysis of SaSs properties and engineering models shows that context-awareness analysis and design, followed by self-adaptiveness design and analysis are widely exploited. Our second research question asks how UML has been customized to support SaSs modeling. Cross analysis of customized diagrams and extension mechanisms shows that the class diagram customization through the profile-based mechanism has been widely used.

This SRL reveals that class diagram has been customized through the profile-based mechanism to provide proper support to analysis and design of context-awareness and self-adaptiveness properties. It also reveals that SaSs major level properties (Self-configuring, Self-healing, Self-optimizing, and Self-protecting) are not properly supported by a UML-based DSML. We intend to address this question in future works.

## ACKNOWLEDGMENTS

This work is supported by the Postgraduate Program in Computing (PPGC) of the Federal University of Rio Grande do Sul (UFRGS) and the Academic Development Program (PDA) of Federal University of Pampa (Unipampa).

## REFERENCES

[1] A. Al-alshuhai and F. Siewe. An Extension of UML Activity Diagram to Model the Behaviour of Context-Aware Systems. In *2015 IEEE International Conference*

*on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 431–437, Liverpool, GB, oct 2015. IEEE.

[2] S. Almutairi, G. Bella, and A. Abu-Samaha. Specifying security requirements of context aware system using UML. In *Seventh International Conference on Digital Information Management (ICDIM 2012)*, pages 259–265. IEEE, aug 2012.

[3] J. Andersson, R. de Lemos, S. Malek, and D. Weyns. Modeling Dimensions of Self-Adaptive Software Systems. In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 27–47. Springer Berlin Heidelberg, Berlin, DE, 2009.

[4] D. Ayed, D. Delanote, and Y. Berbers. MDD Approach for the Development of Context-Aware Applications. In *Modeling and Using Context*, Lecture Notes in Computer Science, pages 15–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[5] M. Becker, M. Luckey, and S. Becker. Model-Driven Performance Engineering of Self-Adaptive Systems: A Survey. In *Proceedings of the 8th international ACM SIGSOFT conference on Quality of Software Architectures - QoSA '12*, page 117, New York, New York, USA, 2012. ACM Press.

[6] M. S. Benselim and H. Seridi-Bouchelaghem. Extended UML for the Development of Context-Aware Applications. In *Networked Digital Technologies*, volume 293 of *Communications in Computer and Information Science*, pages 33–43. Springer, Berlin, DE, 2012.

[7] M.-S. Benselim and H. Seridi-Bouchelaghem. Towards A UML Profile for Context-Awareness Domain. *International Arab Journal of Information Technology*, 14(June):195–207, 2017.

[8] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Boston, MA, US, 2 edition, 2005.

[9] H. Boudjemline, M. Touahria, A. Boubetra, and H. Kaabeche. Heavyweight extension to the UML class diagram metamodel for modeling context aware systems in ubiquitous computing. *International Journal of Pervasive Computing and Communications*, 13(3):238–251, sep 2017.

[10] J. Brings, A. Salmon, and S. Saritas. Context uncertainty in requirements engineering: Definition of a search strategy for a systematic review and preliminary results. In *CEUR Workshop Proceedings*, volume 1342, pages 171–178, Essen, DE, 2015. CEUR.

[11] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw. Engineering Self-Adaptive Systems through Feedback Loops. In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 48–70. Springer Berlin Heidelberg, Berlin, DE, 2009.

[12] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Di Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin Heidelberg, Berlin, DE, 2009.

[13] U. Frank. Some guidelines for the conception of domain-specific modelling languages. In *Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures*, pages 93–106, Bonn, DE, 2011. Gesellschaft für Informatik.

[14] L. Fuentes, N. Gamez, and P. Sanchez. Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications. In *2008 5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*, number Mompes, pages 34–43. IEEE, apr 2008.

[15] D. Han, Q. Yang, J. Xing, J. Li, and H. Wang. FAME: A UML-based framework for modeling fuzzy self-adaptive software. *Information and Software Technology*, 76:118–134, aug 2016.

[16] R. Hebig, H. Giese, and B. Becker. Making control loops explicit when architecting self-adaptive systems. In *Proceeding of the second international workshop on Self-organizing architectures - SOAR '10*, page 21, New York, New York, USA, 2010. ACM Press.

[17] I.-C. Hsu. Extending UML to model Web 2.0-based context-aware applications. *Software: Practice and Experience*, 42(10):1211–1227, oct 2012.

[18] S. Jalali and C. Wohlin. Systematic Literature Studies: Database Searches vs. Backward Snowballing. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '12*, page 29, New York, New York, USA, 2012. ACM Press.

[19] G. M. Kapitsaki and I. S. Venieris. PCP: Privacy-aware Context Profile towards Context- aware Application Development. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services - iiWAS '08*, page 104, New York, New York, USA, 2008. ACM Press.

[20] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41 – 50, 2003.

[21] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature reviews in Software Engineering (Version 2.3). Technical report, Keele University and Durham University Joint Report, Durham, GB, 2007.

[22] 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*, 17(Part B):184–206, feb 2015.

[23] H. A. Linstone and M. Turoff. *The Delphi Method: Techniques and Applications*. Addison-Wesley, Reading, MA, US, 1975.

[24] M. Luckey and G. Engels. High-quality specification of self-adaptive software systems. In *2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 143–152, Los Alamitos, CA, USA, may 2013. IEEE.

[25] M. Luckey, B. Nagel, C. Gerth, and G. Engels. Adapt Cases: Extending Use Cases for Adaptive Systems Markus. In *Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11*, page 30, New York, New York, USA, 2011. ACM Press.

[26] 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*, 40(18):7267–7279, dec 2013.

[27] I. Malavolta, H. Muccini, and M. Sebastiani. Automatically Bridging UML Profiles to MOF Metamodels. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 259 – 266, Funchal, 2015. IEEE.

[28] MetaCase. DSML Examples, 2017.

[29] B. Mohabbati, M. Asadi, D. Gašević, M. Hatala, and H. A. Müller. Combining service-orientation and software product line engineering: A systematic mapping study. *Information and Software Technology*, 55(11):1845–1859, nov 2013.

[30] O. M. G. OMG. Unified Modeling Language (UML), 2015.

[31] P. Oreizy, M. Gorlick, R. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, and A. Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14(3):54–62, may 1999.

[32] T. Patikirikorala, A. Colman, J. Han, and L. Wang. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 33–42, Zurich, CH, jun 2012. IEEE.

[33] M. Salehie and L. Tahvildari. Self-Adaptive Software: Landscape and Research Challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):1–42, may 2009.

[34] Q. Sheng and B. Benatallah. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services. In *International Conference on Mobile Business (ICMB'05)*, pages 206–212, Sydney, NSW, AU, 2005. IEEE.

[35] C. Simons and G. Wirtz. Modeling context in mobile distributed systems with the UML. *Journal of Visual Languages & Computing*, 18(4):420–439, aug 2007.

[36] B. R. Siqueira, F. C. Ferrari, M. A. Serikawa, R. Menotti, and V. V. de Camargo. Characterisation of Challenges for Testing of Adaptive Systems. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing - SAST*, pages 1–10, New York, New York, USA, 2016. ACM Press.

[37] I. Sommerville. *Software Engineering*. Addison Wesley, Boston, MA, US, 9 edition, 2010.

[38] T. Stahl, M. Völter, J. Bettin, A. Haase, and S. Helsen. *Model-Driven Software Development: Techonolgy, Engineering, Management.* John Wiley and Sons, Chichester, GB, 2006.

[39] L. VAN VELSEN, T. VAN DER GEEST, R. KLAASSEN, and M. STEEHOUDER. User-centered evaluation of adaptive and adaptable systems: a literature review. *The Knowledge Engineering Review*, 23(03):261–281, sep 2008.

[40] J. Webster and R. T. Watson. Analyzing the Past To Prepare for the Future : Writing a Review. *MIS Quarterly*, 26(2):13–23, 2002.

[41] S. S. Wenzl and M. Strembeck. Modelling context-aware RBAC models for mobile business processes. *International Journal of Wireless and Mobile Computing*, 6(5):448, 2013.

[42] D. Weyns and T. Ahmad. Claims and Evidence for Architecture-Based Self-adaptation: A Systematic Literature Review. In *Software Architecture*, volume 7957 of *Lecture Notes in Computer Science*, pages 249–265. Springer Berlin Heidelberg, Berlin, DE, 2013.

[43] D. Weyns, M. U. Iftikhar, D. G. de la Iglesia, and T. Ahmad. A survey of formal methods in self-adaptive systems. In *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering - C3S2E '12*, pages 67–79, New York, New York, USA, 2012. ACM Press.

[44] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, pages 1–10, New York, New York, USA, 2014. ACM Press.

[45] 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*, volume 8396 of *Lecture Notes in Computer Science*, pages 55–71. Springer International Publishing, Cham, CH, 2014.

[46] E. Yuan, N. Esfahani, and S. Malek. A Systematic Survey of Self-Protecting Software Systems. *ACM Transactions on Autonomous and Adaptive Systems*, 8(4):1–41, jan 2014.