Self-adaptive Systems Driven by Runtime Models

A Systematic Literature Review of Approaches

Dr. Marcello Thiry
Technological Science Center of Earth and Sea (CTTMar)
University of Vale do Itajaí (UNIVALI)
Florianópolis, Brazil
marcello.thiry@gmail.com

Roger Anderson Schmidt
Technological Science Center of Earth and Sea (CTTMar)
University of Vale do Itajaí (UNIVALI)
Florianópolis, Brazil
rasflp@gmail.com

***Abstract*— Model-Driven Software Engineering (MDSE) represents a promising research area with a variety of challenging issues open for discussion. Expanding the limits of the MDSE paradigm, runtime models keep abstract representations of the running system in order to trigger on-the-fly software reconfigurations. One of the most popular applications of runtime models are self-adaptive systems, since abstractions can be fine-tuned not only in the development phases, but also in runtime. As this kind of system needs to modify its behavior during execution, this can be achieved by means of high-level model interventions. The objective of this article is to present relevant approaches of self-adaptive systems driven by runtime models. This article can help practitioners to get an overall picture of current approaches, in terms of methods, techniques and tools. Researchers can also be inspired to create new or to extend current approaches, facing the challenges identified here. To that end, we conducted a rigorous Systematic Literature Review based on the guidelines proposed by Kitchenham. This paper provides answers for four research questions, based on 16 selected articles. In the conclusion, we present some considerations and challenges based on the results obtained from this review.**

Keywords-Self-Adaptive System; Runtime Model; Model-Driven Software Engineering; Systematic Literature Review

I. INTRODUCTION

Modern software execution environments have become increasingly decentralized, heterogeneous, uncertain and changing. Fully adequate for these scenarios, mobile sensor-based devices have been providing significant computational power in various domains. From their assorted sensors, a huge amount of data can be collected resulting in a rich environmental context. Reacting to changes in this context, mobile applications for remote environments, such as daily life objects from the Internet of Things (IoT) [1] or small devices connected to Wireless Sensor Networks [2] should dynamically adapt themselves to a new external configuration. Thanks to this, Context-Aware and Mobile (CAM) applications [3] that explore Pervasive Computing techniques [4] have been receiving special interest in the recent years.

This background has been leading the software engineering community to propose innovative ways for building, running and managing systems and services [5]. Besides that, the boundaries between design and runtime have to be changed, as designers can not anticipate all possible circumstances that might appear during the execution of an application [1].

In order to meet those demanding expectations, dynamic adaptive systems represent a turning point for responding to changes, as software becomes capable of reconfiguring itself, without the need of being rebuilt. In addition, systems should be able to adapt its structure and/or behavior in response to changes in the execution context and varying user needs. For this purpose, dynamic reconfiguration should be applied at runtime whenever it is needed [5] [6] [7].

However, a particularly important problem arises from the complexity to manage self-adaptive systems. A promising approach to deal with this complexity is to develop adaptation mechanisms that leverage software models, referred to as models@run.time, or runtime models [8]. Conceptually, a runtime model is defined as an abstraction of a running system that is being manipulated during its execution for a specific purpose. Runtime models are also a causally connected self-representation of the system that emphasizes the structure, behavior or goals from a problem space perspective [9].

Therefore, the combination of runtime models and self-adaptive systems opens the possibility of using abstractions at different levels to change the behavior of running systems. The "model-driven" approaches based on runtime models raise the importance of modeling activities. As identified by [9] [10] and several others, both topics present important problems to be addressed, which increases the relevancy of research in this field.

In this paper, a literary research was undertaken on the proposals of approaches for development, execution, change monitoring and reconfiguration of self-adaptive systems through runtime models. For the purpose of clarification, the understanding of an “approach” assumed by this paper is characterized in Sect. II.C. In order to identify, evaluate and interpret all the available papers relevant to our research questions, we choose the Systematic Literature Review (SLR) research method.

This article is structured as follows: Sect. II provides some background about the research topics. Sect. III presents some summarized information about the research method definition that guided this review. In Sect. IV, the outcomes from the research process execution are exposed. Sect. V shows the data extraction and synthesis. Then, the next section presents analysis and discussion based on the research questions. Finally, we conclude this review in Sect. VII.

DOI reference number: 10.18293/SEKE2017-168

II. BACKGROUND

A. Runtime Models inside the MDSE Context

In the literature, there is no consensus about the concept of MDSE. MDSE can be defined as a methodology for applying the advantages of modeling to software engineering activities, which comprises the following aspects: concepts, notations, process and rules, and tools [11]. MDSE is also defined as a family of development processes that focuses on the model as primary development artifact [12].

Traditionally, the MDSE area has primarily focused on using models at design, implementation and deployment phases of the software development life cycle. However, as systems become more adaptable, reconfigurable and self-manageable, runtime models are needed to tackle the complexity of dynamic adaptations by keeping an abstract model of the running system. It pushes the idea of reflection one-step further by synchronizing the abstract model with the actual system, so a change performed on the model is automatically accommodated by the system [1].

B. Self-Adaptive Systems (SAS)

A dynamically adaptive system should be able to adapt its structure and/or behavior in response to changes in the execution context and varying user needs. For this purpose, dynamic re-configuration should be applied at runtime whenever it is needed, in an anticipated or unanticipated form. This determines if the reconfiguration is caused by expected changes in requirements, so it can be considered and planned before being needed, otherwise it is impossible to predict [7].

In addition, reconfiguration actions can be either architectural or behavioral. Architectural reconfigurations consist of modifying the system structure such as add, remove, start, stop, replace and migrate components/connections. On the other hand, behavioral reconfigurations are limited to modify properties of components or connections [7] .

C. Characterization of an "Approach" for this SLR

This paper considers an approach the documented integration of the following components:

- *Process:* an overall workflow definition
- *Abstractions:* meta-meta-models, meta-models, models
- *Transformations:* models-to-text, text-to-models, final or intermediate code generation
- *Self-adaptive Infrastructure:* framework, middleware, or model-based approach [13] that provides low-level services for SAS development and execution
- *Tools*: developed or integrated supporting tools

III. DEFINITION OF THE SYSTEMATIC LITERATURE REVIEW

Our research methodology is inspired by Kitchenham et al. guidelines [14] and procedures [15], which are specifically proposed for Systematic Literature Reviews (SLR) in software engineering. To that end, three main phases are defined: 1. Planning the Review (identify the need for a review and develop a review protocol – Sect. III); 2. Conducting the Review (identify the research, select primary studies, assess the study quality, extract, monitor and synthesize the data – Sects. IV and V); and 3. Reporting the Review (present the results of the review and its dissemination to the interested parties – Sects. VI and VII). We started from the identification of the need for a review, as it follows.

A. Related Work

When starting the planning phase of our research methodology, we checked the need for a systematic review. Therefore, a search for SLRs on the main topics was conducted online through selected databases, from 2012 to 2017 (up to Jan 30). The basic terms in the search string for SLRs were "systematic literature review", "self-adaptive", "model-driven" and several other variations. As a result, we found 55 reviews. After grouping and removing the duplicates, 31 distinct SLRs remained. Then, the SLRs were filtered by metadata evaluation, a stage in which 28 were dismissed for being out of context (8) or too specific in topics such as, security (4), processes (3), DSLs (2), formal verifications (2), and another 9 different topics.

Finally, 3 SLRs remained for full text evaluation. Firstly, the review conducted by Svetits and Zdun [9] presented a comprehensive research on models at runtime literature, classifying the articles in terms of: objectives, architectures, techniques, and kinds of models. Secondly, the SLR conducted by Giachetti et al. [16] focused on interoperability in MDD processes, by evaluating five related features from selected approaches. Finally, Becker et al. [17] focused on model-driven performance engineering, classifying approaches into adaptation, architecture, performance analysis, and applicability criteria.

In contrast to the cited works, this review is exclusively focused on self-adaptive systems driven by runtime models, with the adoption of a well-defined criteria of what is considered an approach. Besides that, its covers articles published until 2017.

B. Research Questions

As suggested by [14] and [16], the PICO(C) criteria [18] was applied in order to frame and structure our research questions. The values of each criterion are the following:

- *Population:* Domain experts, software architects, and systems analysts. In short, model designers.
- *Intervention:* Approaches for dynamic reconfigurations of self-adaptive systems through runtime models.
- *Comparison:* evaluation of different approaches applying defined criteria.
- *Outcomes:* processes, methods and techniques, model types and transformations, reconfiguration strategies, context-awareness and consistency checking, etc.
- *Context:* development and execution phases of self-adaptive systems.

After taking into consideration the five viewpoints presented in the PICO(C) model, it became easier to identify the research questions that follow:

RQ1. What is the central point of each proposal in order to support dynamic reconfigurations of self-adaptive

systems through runtime models? What are the most predominant methods or techniques applied?

RQ2. What levels of abstraction are provided for the model designer? Do the studies suggest an assignment of modeling tasks to different roles according to their skills (e.g. domain expert, software engineer)?

RQ3. Do the studies present an overview of the suggested process? How are they composed in terms of metamodeling, model languages and transformations (code generation included) and related tools?

RQ4. What are the strategies for model changes monitoring and adaptive system dynamic reconfigurations? Does the adaptation engine implemented by or integrated with the studies support anticipated and unanticipated context changes?

C. Literature Selection Criteria

During the research protocol definition, we specified a reliable study selecting criteria in order to ensure that all primary studies provide direct evidence about the research question [14].

1) Inclusion criteria

- Articles published from January 1, 2012 to January 20, 2017 (around 5 years range)
- Articles found in selected electronic databases
- Articles published in peer-reviewed journals, conferences and workshops

2) Exclusion criteria

- Studies not reported in English
- Publications without abstracts
- Books, web sites, technical reports, and master thesis
- Publications in which the research topics are not clearly established and documented, or that explore the term "model" outside the context of software development

D. Data sources

The search strategy for this review included four electronic databases (TABLE I), selected in terms of relevancy in the research areas of Computer Science and Engineering.

TABLE I. Results from search on databases

Database	Total amount of records [a]	SLRs (the need of a review)	Primary Studies (initial search)
SCOPUS (Elsevier)	Over 60 million	21	111
Association for Computing Machinery (ACM) Guide to Computing Literature	2,625,656	17	71
IEEE Xplore Digital Library	4,145,171	10	33
ScienceDirect (Elsevier)	Over 14 million	7	9
Total		**55**	**224**

a. Data collected on January 30, 2017

E. Search String Composition

After the database selection, we started to compose the search string by defining its components and keywords. Besides, we had to explore thoroughly every engine mechanism in order to produce a consistent query for each database [19]. During this process, we were aware of the variations of the research concepts, as shown in TABLE II. After many testing iterations, our search string was defined by aggregating the key topics for this review.

TABLE II. Search keywords

Concept	Keyword and synonymous
P = Self-Adaptive Systems	"self-adaptive", "self adaptive", "adaptive system", "adaptive software"
Q = Model-Driven Software Engineering	"model-driven", "model driven", "mde", "mdse", "mdd"
S = Runtime Models	"models at runtime", "models@runtime", "models@run.time", "models for runtime", "runtime models"
Result	**(P ∨ Q ∨ S)**

IV. Execution of the SLR

A. Literature Search Process

The literature search process that guided our review is presented in Figure 1, which shows the outcomes from every activity of the process, highlighting the articles selected and dismissed due to related reasons. Initially, we submitted the final versions of them to each selected electronic database (1). Then the metadata of all returned articles (title, abstract and keywords) was exported to BibTeX format, which is accepted by the Mendeley import tool. Next, we proceeded with a careful metadata checking of every single article against the literature selection criteria (2). As a result, 150 articles were selected. After grouping and duplicates removal (3), 134 articles remained.

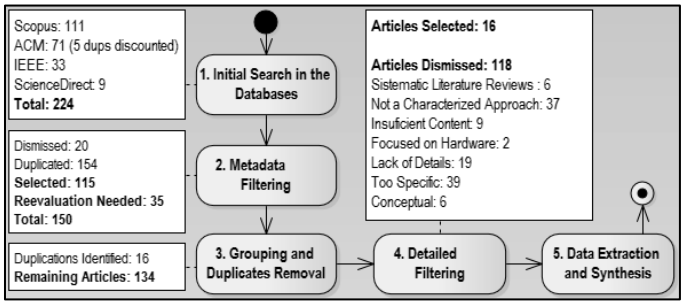

Figure 1. Literature Search Process

The final selection of articles was performed after full text reading and evaluation (4). In this stage, we especially focused on the compliance with the approach characterized in Sect. II.C. As a result, the 16 articles presented in TABLE III were selected for data extraction and synthesis (5).

TABLE III. Selected Studies

ID	Reference
S1	S. Loukil, S. Kallel, and M. Jmaiel, "An approach based on runtime models for developing dynamically adaptive systems," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 365–375, 2017.

ID	Reference
S2	F. Moyano, C. Fernandez-Gago, and J. Lopez, "A Model-driven Approach for Engineering Trust and Reputation into Software Services," *J. Netw. Comput. Appl.*, vol. 69, no. C, pp. 134–151, 2016.
S3	B. Djoudi, C. Bouanaka, and N. Zeghib, "A formal framework for context-aware systems specification and verification," *J. Syst. Softw.*, vol. 122, pp. 445–462, 2016.
S4	J. M. T. Portocarrero, F. C. Delicato, P. F. Pires, T. C. Rodrigues, and T. V. Batista, "SAMSON: Self-adaptive middleware for wireless sensor networks," in *Proceedings of the ACM Symposium on Applied Computing*, vol. April 04–08, pp. 1315–1322, 2016.
S5	M. Hussein, R. Nouacer, and A. Radermacher, "A Model-Driven Approach for Validating Safe Adaptive Behaviors," in *Proceedings of the 19th Euromicro Conference on Digital System Design*, pp. 75–81, 2016.
S6	J. Yu, Q. Sheng, J. K. Y. Swee, J. Han, C. Liu, and T. H. Noor, "Model-driven development of adaptive web service processes with aspects and rules," *J. Comput. Syst. Sci.*, vol. 81, no. 3, pp. 533–552, 2015.
S7	P. A. de S. Duarte, F. M. Barreto, F. A. de A. Gomes, W. V. de Carvalho, and F. A. M. Trinta, "CRITiCAL: A Configuration Tool for Context Aware and mobiLe Applications," in *Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference* – Volume 02, pp. 159–168, 2015.
S8	J. Bocanegra, J. Pavlich-Mariscal, and A. Carillo-Ramos, "MiDAS: A model-driven approach for adaptive software," in *Proceedings of the WEBIST 2015 – 11th International Conference on Web Information Systems and Technologies*, pp. 281–286, 2015.
S9	D. B. Abeywickrama, N. Hoch, and F. Zambonelli, "An integrated Eclipse plug-in for engineering and implementing self-adaptive systems," in *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 3–8, 2014.
S10	M. Hussein, J. Han, J. Yu, and A. Colman, "Enabling Runtime Evolution of Context-Aware Adaptive Services," in *Proceedings of the 2013 IEEE International Conference on Services Computing*, pp. 248–255, 2013.
S11	M. Luckey and G. Engels, "High-quality specification of self-adaptive software systems," in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pp. 143–152, 2013.
S12	N. Ferry, F. Chauvel, A. Rossini, B. Morin, and A. Solberg, "Managing multi-cloud systems with CloudMF," in *ACM International Conference Proceeding Series*, pp. 38–45, 2013.
S13	C. Ghezzi, L. S. Pinto, P. Spoletini, and G. Tamburrelli, "Managing non-functional uncertainty via model-driven adaptivity," in *Proceedings of International Conference on Software Engineering*, pp. 33–42, 2013.
S14	J. Floch, C. Frà, R. Fricke, K. Geihs, M. Wagner, J. Lorenzo, E. Soladana, S. Mehlhase, N. Paspallis, H. Rahnama, P. A. Ruiz, and U. Scholz, "Playing MUSIC - Building context-aware and self-adaptive mobile applications," *Softw. - Pract. Exp.*, vol. 43, no. 3, pp. 359–388, 2013.
S15	S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. A. Papadopoulos, "A development framework and methodology for self-adapting applications in ubiquitous computing environments," *J. Syst. Softw.*, vol. 85, no. 12, pp. 2840–2859, 2012.
S16	M. Amoui, M. Derakhshanmanesh, J. Ebert, and L. Tahvildari, "Achieving dynamic adaptation via management and interpretation of runtime models," *J. Syst. Softw.*, vol. 85, no. 12, pp. 2720–2737, 2012.

V. Data Extraction and Synthesis

In consonance with the research questions presented in Sect. III.B, relevant information was extracted from the selected studies. The overall answers to the first question (RQ1) are presented in Table IV. In relation to the third question (RQ3), Table V and Table VI present data extracted.

Table IV. Overview of the selected studies (RQ1)

ID	Central Point	Main Methods or Techniques
S1	Middleware responsible for monitoring the system and performing architectural reconfiguration by Aspect Oriented Software Development. Concurrency between reconfigurations is supported.	Architecture Description Language (ADL), AOSD
S2	Framework for trust and reputation that allows developers to implement different types of security models in a high level of abstraction thanks to the usage of metamodeling techniques.	Kevoree Distributed Dynamic Component Model Integration
S3	Framework for specification and verification of context-aware systems. A DSL is provided to allow designers to specify context entities, states and actions. It also provides a tool with several features.	CBSE, Formal Methods, Maude-based DSL
S4	Process to generate an instance of a Reference Architecture from its specification. Model-driven transformations are used to map elements and to generate the source code to be deployed in a WSN platform.	Reference Architecture (RA), Middleware Instantiation, MAPE-K
S5	Approach to facilitate the validation of the adaptive behavior of embedded software in the fully electric vehicles domain. Fault injection and monitoring techniques are applied on a virtual self-adaptive platform.	Architecture Description Language (ADL)
S6	Approach to support the development of dynamically adaptive WS-BPEL based systems. To that end, an aspect-oriented method is developed in order to perform runtime changes.	AOSD, WS-BPEL, Web Ontology Language (OWL)
S7	Approach for developing Context-Aware and Mobile (CAM) applications, by modelling contextual information and rule-based behaviour by using a visual notation.	OSGi, Middleware (LOCCAM), DSL
S8	Framework that provides a new language for requirements (with uncertainty support), a method to derive concrete implementations in specific architectures, besides a mechanism for traceability and sincronization.	DSL, Traceability
S9	An integrated Eclipse plug-in tool to architect, engineer and implement self-adaptive systems through a feedback loop-based approach. Also provides modeling, simulation and code generation features.	FCL-Based, MAPE-K
S10	Approach to enable runtime evolution of context-aware adaptive services in response to unanticipated changes in their environments or functionalities. Differences between running and its evolved model are computed.	SAS Runtime Evolution
S11	Method for software specification by using UML based concern-specific modeling language. It allows separated and explicit specification of self-adaptivity concerns.	SoC, MAPE-K
S12	Framework for modeling dynamically cloud-based adaptive systems by enabling adaptation at runtime. It consists of a tool-supporting DSL and a models@runtime environment.	Cloud Computing, DSL
S13	Framework to support the development and execution of software that tolerates manifestations of uncertainty, in order to satisfy certain non-functional requirements divided into two classes Threshold-based and Max/Min.	Probability Theory, Markov Decision Process (MDP)
S14	Description of typical context and adaptation features relevant for the development of context-aware and self-adaptive mobile applications, based on several demonstrations of the MUSIC adaptation framework.	Context-awareness, Self-adaptation, MUSIC framework
S15	Discussion of the motivation, technical approach, and results of the MUSIC project, which provides a software development framework for self-adaptive applications that operate in ubiquitous and dynamic environments.	OSGi, MUSIC framework, MAPE-K
S16	Approach for realizing fine-grained dynamic adaptation in software systems by managing and interpreting graph-based models of software at runtime. Includes a comprehesive case study presented in detail.	AOSD, MAPE-K, TGraph Approach

Table V. External tools applied (RQ3)

Name	Type	Studies
Ocarina	Distribute Applications Generator	S1
Kevoree	Distributed Reconfigurable Software Dev.	S2
Papyrus	Model-Based Engineering tool	S5
Acceleo	Transformation Tool	S3
UNISIM-VP	Virtual Platform for Simulator Environment	S5
Drools	Business Rules Engine	S6
BPEL	Execution Language Engine	S6
LOCCAM	Self-Adaptive Middleware	S7
JET	Template Engine for Code Generating	S9
ROAD	Apache Axis2 Extension for adapting services	S10
PRISM	Probabilistic Model Checker	S13
MUSIC	Self-Adaptive Framework	S14, S15
MOFScript	Transformation Builder	S14, S15
JGraLab	API for Processing TGraphs	S16

Table VI. Languages and Notations applied (RQ3)

Name	Purpose	Studies
UML	General-purpose Modeling	S5, S9, S13, S16
OWL	Semantic Web	S6, S14, S15
OCL	Object Constraint	S1, S3
Drools	Business Rule	S6, S10
AADL	Architecture Analysis and Design	S1
AO4AADL	Aspect Oriented Extension for AADL	S1
Schematron	Rule-based XML Validation	S1
Kevscript	Kevoree Reflection Layer Scripting	S2
Pi-ADL	Formal Description	S4
EAST-ADL	Architecture Description	S5
BPMN	Business Process Description	S6
UML AL	UML Action Profile	S9
DMM	Graph-transformation	S11
ATN	Automata Theory	S13
MUSIC	UML Profile for Adaptation Modeling	S15
GReQL	Graph Repository Query Language	S16
GReTL	Graph Repository Transformation Language	S16

VI. ANALYSIS AND DISCUSSION

In this section, we analyzed each research question with the purpose of achieving the defined goals.

RQ1. What is the central point of each proposal in order to support dynamic reconfigurations of self-adaptive systems through runtime models? What are the most predominant methods or techniques applied?

A recurring method found in the studies is the MAPE-K reference model, which served as basis for their autonomic feedback loop implementation (S4, S9, S11, S15, S16). It is also important to highlight that the Component-Based Software Engineering (CBSE), in general or specifically related to the OSGi dynamic component model, had significate representation among the studies (S2, S3, S6, S7, S15, S16). This approach has been commonly used to support the Separation of Concerns (SoC) design principle and to provide dynamic architectural reconfigurations in the proposed self-adaptive systems. Towards similar goals, Aspect Oriented Software Development (AOSD) is found in a representative number of studies as well (S1, S6, S16).

RQ2. What levels of abstraction are provided for the models designer? Do the studies suggest an assignment of modeling tasks to different roles according to their skills (e.g. domain expert, software engineer)?

Ten studies present a high-level business model for runtime interventions (S1-S8, S10, S13). In general, the studies propose a clear separation between business logic, context awareness and adaptation concerns, by using models (S1, S3, S5, S6, S7, S10, S15), API (S2), reference architecture (S4) and language (S11). Two studies present a specific model for invariants specification (S1, S3). Less than half of the studies suggest well-defined separated roles for each abstraction level (S1, S4-S8, S11). In relation to the number of architectural layers, studies present two (S1, S2, S5, S7, S13), three (S4, S6, S8, S10) and four (S3, S16).

RQ3. Do the studies present an overview of the suggested process? How are they composed in terms of metamodeling, model languages and transformations (code generation included) and related tools?

Self-adaptive systems driven by runtime models require at least a two-stage process, since there are not only development phases, but also runtime dynamics when the system's reconfiguration is expected to happen. In general, studies present a process in the form of an overall workflow (big picture discussed in detail) for both introduced stages (S1, S3-S7, S10, S11, S13, S16). In relation to the number of model transformations, the methods presented by the studies include; only one (S3, S5, S7, S8), two (S2) or three (S1, S6). Code generation, which represents a special kind of model transformation, is explored by the studies aiming to a platform/programming language, as follow: Java/Java (S1, S2, S5, S7, S9, S13-S16), AspectJ/Java (S1) and Contiki/C (S4). Among UML artifacts, Activity Diagram is the most popular representation applied in the studies as a source for code generation (S9, S13, S16). Another core concept of MDSE is metamodeling, which techniques are widely used by the studies (S1-S5, S7, S10-S12, S16). Some approaches implement Domain Specific Languages from scratch (S3, S6, S7, S8, S11). Besides that, several studies provide in-house developed tools for models, in concern of design (S1, S3, S6, S7, S8, S9, S10, S12, S14 and S15) and transformation (S3, S6, S8, S10, S13, S14, S15). To that end, most of these tools are Eclipse EMF-based. Finally, S1 and S3 present sound techniques for invariants specification.

RQ4. What are the strategies for model changes monitoring and adaptive system dynamic reconfigurations? Does the provided or suggested adaptation engine support anticipated and unanticipated context changes?

The most identified strategy for model change monitoring and adaptive system dynamic reconfigurations is the implementation of a middleware or framework (S1-S3, S5, S8, S12-S16). Formal methods are used in S3 and S13, but the latter generates an automaton from UML Activity Diagrams, while the former defines semantics for a DSL to formalize context-aware systems structure and behavior. In relation to reconfiguration actions, architectural (S1, S2, S3, S4, S7, S16) and behavioral (S3, S13, S6) were recognized. Finally, adaptation engines of S1, S3 and S10 can cope with unanticipated context changes.

A. Threats to Validity

Publication bias represents a serious threat to the validity of the conclusions produced by a SLR, since it is likely that published studies will have more 'positive' results, that is, failures are seldom reported. A common threat during the search process is not finding relevant studies. We minimized this risk by selecting four comprehensive databases. Besides that, we composed a search string with several variations, and we gave special attention to details of each query tool provided by the selected databases.

With respect to the influence of personal researchers' opinions during the selecting process, we defined a very clear literature selecting criteria during the SLR planning phase. Lastly, some information not explicitly described in the articles had to be inferred by the authors. To minimize this threat, we promoted review sessions, when conflicting or unclear interpretations were discussed cooperatively.

VII. Conclusion

This article presents an overview on approaches for development and execution of self-adaptive systems driven by runtime models. As a result of this Systematic Literature Review, 16 relevant articles were considered, after analyzing more than 220 initial studies.

From what we observed, the method of consistency checking between model and system at runtime is not clearly discussed in the majority of the analyzed articles. Regarding this problem, few studies demonstrated a rollback behavior when the change invalidated a constraint or invariant. Consistency checking is noted by some authors as an important challenge in this field.

Another challenge relies on using the adequate abstract models to define and perform dynamic reconfiguration. UML and its extension mechanisms have been widely used in this direction. Ontologies and Business Process Models have been explored also. In order to increase the model expressiveness, we considered the usage of different languages and notations extremely positive.

Finally, there is a limited range of options available when it comes to environments for distributed reconfigurable software development and execution. Furthermore, the existent ones need to evolve to support traceability, unanticipated changes (uncertainty levels), models reusability and low-level services for self-adaptive systems implementation and monitoring.

In reference to future work, we plan to consider more articles through the "snowballing" practice from references of the selected papers. Besides this, aligning the research protocol towards microservices-based approaches as a provider for architectural reconfigurations, is planned.

References

[1] F. Moyano, C. Fernandez-Gago, and J. Lopez, "A Model-driven Approach for Engineering Trust and Reputation into Software Services," *J. Netw. Comput. Appl.*, vol. 69, no. C, pp. 134–151, 2016.

[2] J. M. T. Portocarrero, F. C. Delicato, P. F. Pires, T. C. Rodrigues, and T. V. Batista, "SAMSON: Self-adaptive middleware for wireless sensor networks," in *Proceedings of the ACM Symposium on Applied Computing*, vol. 04–08, pp. 1315–1322, 2016.

[3] P. A. de S. Duarte, F. M. Barreto, F. A. de A. Gomes, W. V. de Carvalho, and F. A. M. Trinta, "CRITiCAL: A Configuration Tool for Context Aware and mobiLe Applications," in *Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference - Volume 02*, pp. 159–168, 2015.

[4] B. Djoudi, C. Bouanaka, and N. Zeghib, "A formal framework for context-aware systems specification and verification," *J. Syst. Softw.*, vol. 122, pp. 445–462, 2016.

[5] B. H. C. Cheng, R. De Lemos, H. Giese, P. Inverardi, and J. Magee, *Software Engineering for Self-Adaptive Systems*. 2009.

[6] S. Wätzoldt and H. Giese, "Classifying distributed self-∗ systems based on runtime models and their coupling," in *CEUR Workshop Proceedings*, 2014, vol. 1270, pp. 11–20.

[7] S. Loukil, S. Kallel, and M. Jmaiel, "An approach based on runtime models for developing dynamically adaptive systems," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 365–375, 2017.

[8] G. Blair, N. Bencomo, and R. B. France, "MODELS@RUN.TIME Introduction," *Computer (Long. Beach. Calif).*, vol. 42, no. 10, pp. 22–27, 2009.

[9] M. Szvetits and U. Zdun, "Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime," *Softw. Syst. Model.*, pp. 1–39, 2013.

[10] M. Goulão, V. Amaral, and M. Mernik, "Quality in model-driven engineering: a tertiary study," *Softw. Qual. J.*, vol. 24, no. 3, pp. 601–633, 2016.

[11] M. Brambilla, J. Cabot, and M. Wimmer, *Model-driven software engineering in practice*. Morgan & Claypool Publishers, 2012.

[12] N. Macedo, T. Jorge, and A. Cunha, "A Feature-based Classification of Model Repair Approaches," *IEEE Trans. Softw. Eng.*, vol. PP, no. 99, p. 1, 2016.

[13] F. Křikava, P. Collet, and R. B. France, "ACTRESS: Domain-specific modeling of self-adaptive software architectures," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 391–398, 2014.

[14] B. Kitchenham, D. Budgen, and P. Brereton, "Guidelines for performing systematic literature reviews in software engineering," *Int. Conf. Soft. Engin.*, vol. 45, no. 4ve, p. 1051, 2006.

[15] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, no. TR/SE-0401, p. 28, 2004.

[16] G. . Giachetti, F. Valverde, and B. Marín, "Interoperability for model-driven development: Current state and future challenges," in *Proceedings - International Conference on Research Challenges in Information Science*, 2012.

[17] M. Becker, M. Luckey, and S. Becker, "Model-driven performance engineering of self-adaptive systems: A survey," in *QoSA'12 - Proceedings of the 8th International ACM SIGSOFT Conference on the Quality of Software Architectures*, pp. 117–122, 2012.

[18] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences: A Practical Guide*. 2006.

[19] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *J. Syst. Softw.*, vol. 80, no. 4, pp. 571–583, 2007.