Software & Systems Modeling (2019) 18:3049–3082
https://doi.org/10.1007/s10270-018-00712-x

REGULAR PAPER

Models@run.time: a guided tour of the state of the art and research challenges

Nelly Bencomo[1] · Sebastian Götz[2] · Hui Song[3]

Received: 9 February 2018 / Revised: 23 October 2018 / Accepted: 17 December 2018 / Published online: 9 January 2019

Abstract

More than a decade ago, the research topic models@run.time was coined. Since then, the research area has received increasing attention. Given the prolific results during these years, the current outcomes need to be sorted and classified. Furthermore, many gaps need to be categorized in order to further develop the research topic by experts of the research area but also newcomers. Accordingly, the paper discusses the principles and requirements of models@run.time and the state of the art of the research line. To make the discussion more concrete, a taxonomy is defined and used to compare the main approaches and research outcomes in the area during the last decade and including ancestor research initiatives. We identified and classified 275 papers on models@run.time, which allowed us to identify the underlying research gaps and to elaborate on the corresponding research challenges. Finally, we also facilitate sustainability of the survey over time by offering tool support to add, correct and visualize data.

Keywords Models@run.time · Self-reflection · Causal connection · Systematic literature review

1 Introduction

Model-driven engineering (MDE) highlights the importance of models to engineer systems. Models are artefacts with key roles during the software development process and have successfully been used for long time for communication between stakeholders, documentation, code generation, deployment and traceability between development stages, among other uses [35,81]. In contrast to those traditional uses of models during development time, more recent research initiatives have shown the utility of models during runtime [30,37]. Runtime models are envisioned to provide intelligent support to software during execution [16] as the distinction between software development and execution blurs. Runtime models can be used by the system itself, other systems or humans to help coping with the challenges posed by self-adaptive systems [56] (systems able to adapt themselves even to unanticipated changes) and the so-called eternal systems [133] (i.e., software systems that are required to survive fluctuations in their execution environment without or with only little human intervention). Runtime models can support reasoning and decision-making based on knowledge that may emerge at runtime but was not foreseen before execution [29].

A key principle underlying models@run.time is computational reflection as described by Pattie Maes, who defines "computational reflection to be the behaviour exhibited by a reflective system, where a reflective system is a computational system which is about itself in a causally connected way" [123]. In other words, runtime models represent a reflection layer, which is causally connected with the underlying system so that every change in the runtime model leads to a change in the reflected system and vice versa. Or, in Pattie Maes words: "A system is said to be causally connected to its domain if the internal structures and the domain they represent are linked in such a way that if one of them changes, this leads to a corresponding effect upon the other" [123]. Models@run.time combines the principles of computational reflection with model-driven engineering.

Communicated by Professor Yves Le Traon.

✉ Sebastian Götz
sebastian.goetz@acm.org

Nelly Bencomo
nelly@acm.org

Hui Song
hui.song@sintef.no

[1] Aston University, Birmingham, UK

[2] Technische Universität Dresden, Dresden, Germany

[3] Stiftelsen SINTEF, Trondheim, Norway

Based on the above, a runtime model is defined as a causally connected self-representation of the associated system that emphasizes the structure, behaviour or goals of the system and which can be manipulated at runtime for specific purposes [35]. Models@run.time can be used as a catalyst for the creation of future software that inevitably needs to be long-lived while coping with ever-changing environmental conditions, which are only partially known at development time [10].

Since the term was introduced, a plethora of approaches applying or supporting models@run.time have been developed [30,35,37,95]. The need to sort out and classify the prolific results is natural and urgent. Researchers have so far focused on applications and fundamentals of runtime models according to different specific research interests and goals. As a consequence, the research results are scattered across several different research communities and domains (such as robotics, embedded systems and MDE itself). On the one hand, different approaches in several domains show similarities, and approaches from different research communities could be rather complementary. The key point is that some of those approaches have been proposed without being aware of each other. This hinders the development of models@run.time towards a mature research field. On the other hand, the gaps between the core models@run.time technology and the diverse domains certainly need to be identified and categorized in order to further develop the research topic by researchers of the area.

Based on a systematic literature review, this paper provides a historical perspective on the development of the research topic models@run.time. Such a study is relevant at this point not only to appreciate the substantial body of work that already exists in the area of models@run.time, but also to step back, to understand current trends, and to anticipate future needs to evolve the research area in a more meaningful way. Hence, a gateway to new models@run.time approaches can be opened combining different experiences and new ideas to tackle new research challenges.

The main aims of this survey are to (i) evaluate the field by its outcomes, (ii) provide support for researchers to situate themselves in the research area for different purposes, including evaluation of their work and, (iii) discuss how the models@run.time paradigm is useful to build software of the future.

Research approach In order to pursue our aims, we are following these specific objectives:

- Present a taxonomy to allow the classification of existing models@run.time approaches.
- Based on the taxonomy, present an overview of the current state of the art in the research area of models@run.time.
- Elaborate on the current trends, research initiatives, research gaps and corresponding challenges, and propose relevant research directions.

The survey includes 275 papers from multiple research domains published in different venues. The timeline of the papers analyzed covers mainly work since 2006, but also some selected work (5 papers) before 2006, which can be seen as influential predecessor work of models@run.time and that laid the basis for the research line accordingly. First, we derived a taxonomy that defines different dimensions identified by a first content analysis. A dimension refers to an aspect of the research topic models@run.time to be studied. More concretely, based on an initial set of dimensions defined according to our own experience in the topic and having clarified the inclusion and exclusion criteria, we have assembled a collection of papers which we have iteratively analyzed. This analysis in turn refines the dimensions of the taxonomy. After that, the final taxonomy was the tool to be used to classify the papers, and therefore provided a thorough overview of the current state of research in models@run.time. The results were used to perform an analysis cross-cutting the different dimensions of the taxonomy to derive gaps in the form of research challenges and potential directions for future research efforts in the area.

By now, only one other survey on models@run.time has been published so far [172]. This survey is based on a keyword search and, thus, in contrast to this article, does not take into account, for example, the present knowledge of the models@run.time research community, e.g., by directly including papers from the workshop series on models@run.time or by studying seminal ancestor research initiatives. Further, we explicitly came up with a taxonomy that allows the classification of the outcomes studied. This taxonomy includes aspects studied in the survey by [172] such as the objectives, techniques and kind of models. However, our survey also includes the modelled artefacts, the domain to which the work has been applied and the study of intersecting research areas. Further, the dimension covering the modelled artefacts has allowed us to focus not only on the concept of architecture, but also on other aspects such as the concept of the user interface of the system under analysis, the study of the granularity of the architecture, the timeline of the life cycle (e.g., design time and runtime) and more. We studied at which level of abstraction which types of runtime models were used, which model-driven techniques have been applied at runtime and the relationship to the purpose of applying models@run.time. Different from other studies, we also emphasize the difference of fundamental research, i.e., research performed to develop approaches that follow the models@run.time paradigm and applied research of models@run.time, i.e., research efforts that use the runtime models paradigm.

A novelty, we introduce with this survey is that we provide means to the community to keep the data of the survey updated over time and, by this, meet the prerequisite for a self-sustainable survey. For this, we have developed an open-source toolkit [94], which enables the community to easily add the data about new papers, to correct the classification of existing papers and to visualize the survey data using different types of charts and tables.

Organization This paper is structured as follows: Section 2 describes the research method applied for the systematic literature review. Next, in Sect. 3 we present the first contribution of this paper, a taxonomy for the classification of models@run.time approaches based on several dimensions and which has been derived from an iterative and detailed analysis of the existing research literature. Section 4 discusses threats to validity of our survey. The second contribution, a classification of the surveyed models@run.time approaches according to the taxonomy with a thorough cross-dimension analysis between the different categories, is discussed in Sect. 5 using bubble matrix charts. Section 6 identifies challenges and future research directions based on the results of the survey. In Sect. 7, the paper concludes.

2 Research method applied

The review of the state of the art developed in this paper follows the principles described by Kitchenham in [113]. Based on those principles, a review is structured in three phases: *planning*, *conducting* and *reporting the review*. Accordingly, we explain in this section details about the planning applied, i.e., the research questions we posed. In particular, Sect. 2.1 describes the research questions that have driven the survey, Sect. 2.2 describes the procedure and the steps undertaken, and Sect. 2.3 details the process of the literature search explaining the *inclusion* and *exclusion* criteria used.

2.1 Research questions

In each of the three phases (*planning, conducting and reporting*), we followed the principles described in [113] adapting them to the context given by the research topic studied and the expertise of the authors in the topics. The initial activity of the planning phase was to identify and critically reflect upon the need for the survey to be conducted. The rational of this discourse is the motivation for this work as described in Sect. 1. Based on the aims and specific objectives described in Sect. 1, we have formulated the following research questions, which state the basis of the survey:

- **RQ1** How can existing research on models@run.time be classified?
- **RQ2** What is the state of the art of models@run.time research with respect to the classification?
- **RQ3** What can be inferred from the results associated with RQ2 that will lead to timely, relevant research directions for further investigation?

2.2 General procedure

Figure 1 depicts in detail the general procedure we have undertaken. As a starting point, in the *initialization step*, we specified the protocol used to conduct the survey. This protocol describes (i) which data sources to use, (ii) which criteria have to be met by a paper to be included in the survey, (iii) the exclusion criteria and (iv) an initial taxonomy to be used for the classification of papers, which qualified for the survey.

Based on this protocol, we defined our search strategy to be twofold: (i) we use several conferences, workshops, journals and books as primary data sources, and (ii) we use our knowledge about people relevantly active in the research community to identify further venues as primary data source, namely those where these people have published. Based on the papers collected using these primary data sources, we brainstormed and derived an initial set of dimensions for classification, which we iteratively refined during the process of mapping the papers to classes of these dimensions. Using the final set of dimensions, we derived the total collection of papers for closer review, where we (i) filtered the papers based on inclusion and exclusion criteria (cf. Sect. 2.3) and (ii) added further papers subject to the filtering by following citations with respect to the initial set of papers.

The outcome of the *initialization step* was, hence, a list of dimensions and a list of papers qualified for inclusion to the survey. Using these two artefacts, in the *second step*, we constructed the taxonomy. Again, we conducted this process iteratively, starting with an initial first draft of the taxonomy, which was refined while reviewing the papers. The partial result was the quasi-final taxonomy (the second draft). Due to distributed work by the authors, some identified classes in the taxonomy received different (but similar) names; however, they meant the same concept. Therefore, such classes were merged together to obtain more manageable graphs to be analyzed. The current draft of the taxonomy was cleansed accordingly to obtain the final version of the taxonomy to answer *RQ1*. The final version of the taxonomy is presented in Sect. 3.

In the *third step*, using the final taxonomy from the previous step and the list of papers subject to closer review, we mapped all qualified papers to the identified classes of the taxonomy, checked the consistency of the taxonomy and analyzed the resulting mapping, as an answer to *RQ2*. The focus of the analysis was on how balanced the work across the classes of the taxonomy is. By this, classes that have rarely or not been addressed at all could be identified. Similarly, new trends in the area were identified and analyzed accordingly.

Based on the analysis results, we finally proceeded to realize the *fourth step* of collecting a set of future research directions to give an answer to *RQ3*.

Throughout the whole process, we used a novel tool, which has been developed alongside our study in student projects at Technische Universität Dresden: the systematic literature review toolkit publicly available and ready to use for Windows, Linux and Mac-based systems.[1] The EPL-1.0 licensed open-source tool is under development since October 2014 and supports researchers conducting literature reviews in all before mentioned phases of a study, but, due to the lack of free-to-use search APIs, with the exception of collecting relevant literature from the publishers. The tool is still under active development. Also, the data presented in this paper (i.e., bibtex entries for all included papers, the taxonomy and the paper classification) are available as an example project of the toolkit online.[2]

The procedure we followed, as described above, is depicted in Fig. 1. The following subsection provides details about the selected literature.

2.3 Process of collecting relevant papers

A very important decision for literature surveys is which data sources shall be used to find as much existing work as possible. A typical approach is to use scholarly search engines such as Google Scholar or Microsoft Academic Search. Another opportunity is to collect papers directly from the websites of publishers such as ACM, IEEE and Springer. Both approaches are based on a good selection of keywords for the respective search. However, the list of keywords usually leads to a very large amount of results while still being incomplete. The reason for this is the lack of a common vocabulary in different research communities, especially when the field of research, subject to investigation, is young and changing, as it is in our case. In consequence, many papers, which actually belong to the corpus of literature to be investigated, are not found in those searches, because they used another terminology.

To ameliorate this problem, we decided to follow an alternative way, using conferences, workshops, books and journals that interface with models@run.time, as primary sources of papers for this survey, using our own knowledge on the communities and these venues. The premium selections were the models@run.time workshops [95], two special issues of journals [35,37] and one book on models@runtime [30]. These venues tend to contain frontier work and ideas that connect to other relevant research venues and therefore papers. At the same time, we looked at the top conferences on generic software engineering and its major branches, such as model-driven engineering (MODELS), requirement engineering (RE), component-based software (CBSE), and self-adaptive systems (SEAMS), among others. Qualified papers from these venues show the acceptance of models@run.time by a wider software engineering community. We also performed venue analysis on the development of the research topic models@run.time. As an example of interesting results, we found that in the area of software product lines (SPLC), no qualified papers were found even though it is a topic of interest. Models@run.time is cited in SPLC keynotes [19], keynotes of related venues [29,98,102] and also included in the call for papers of SPLC workshops [136].

Table 1 lists all used sources and, for each source, the total number of papers evaluated against inclusion and exclusion criteria and the number of papers, which qualified for inclusion. The table also provides the reader with the number of papers per venue.

As a starting point, we evaluated 95 papers published in the models@run.time workshop series, which is running since 2006. Interestingly, only 80 papers qualified for inclusion in the survey. The reason for the existence of papers published in models@run.time workshop proceedings, which neither contain fundamental nor applied research on models@run.time, can be found in the fact that the research topic has been evolving during the last decade. Especially, in early editions of the workshop, several papers on executable models have been presented at the workshop, to discuss the relation between executable and runtime models. However, eventually, the community agreed upon the need for a causal connection between a runtime model and a system for work to be considered as a models@run.time approach and that is not necessarily the case for executable model approaches.

In addition to these workshop papers, two special issues on models@run.time (in Springer Computing [37] and IEEE Computer [35], respectively) comprising nine (9) qualifying papers in total, and a book on models@run.time [30] containing nine (9) qualifying papers have been used as key source of information.

A further source for papers was the MODELS conference, which annually hosts the models@run.time workshop, and, for several years also had separate tracks on models@run.time. A second close conference to the research topic models@run.time is SEAMS, which also has shown separate tracks on models@run.time, and, hence, has been included as data source.

As models@run.time interfaces with many other research areas, we decided to include the following major conferences who offer intersecting areas: ICAC and SASO representing the autonomous computing community; CompArch, ECSA and WICSA representing the software architecture

[1] https://github.com/sebastiangoetz/slr-toolkit/releases.

[2] https://github.com/sebastiangoetz/slr-toolkit/tree/master/examples/mrt.

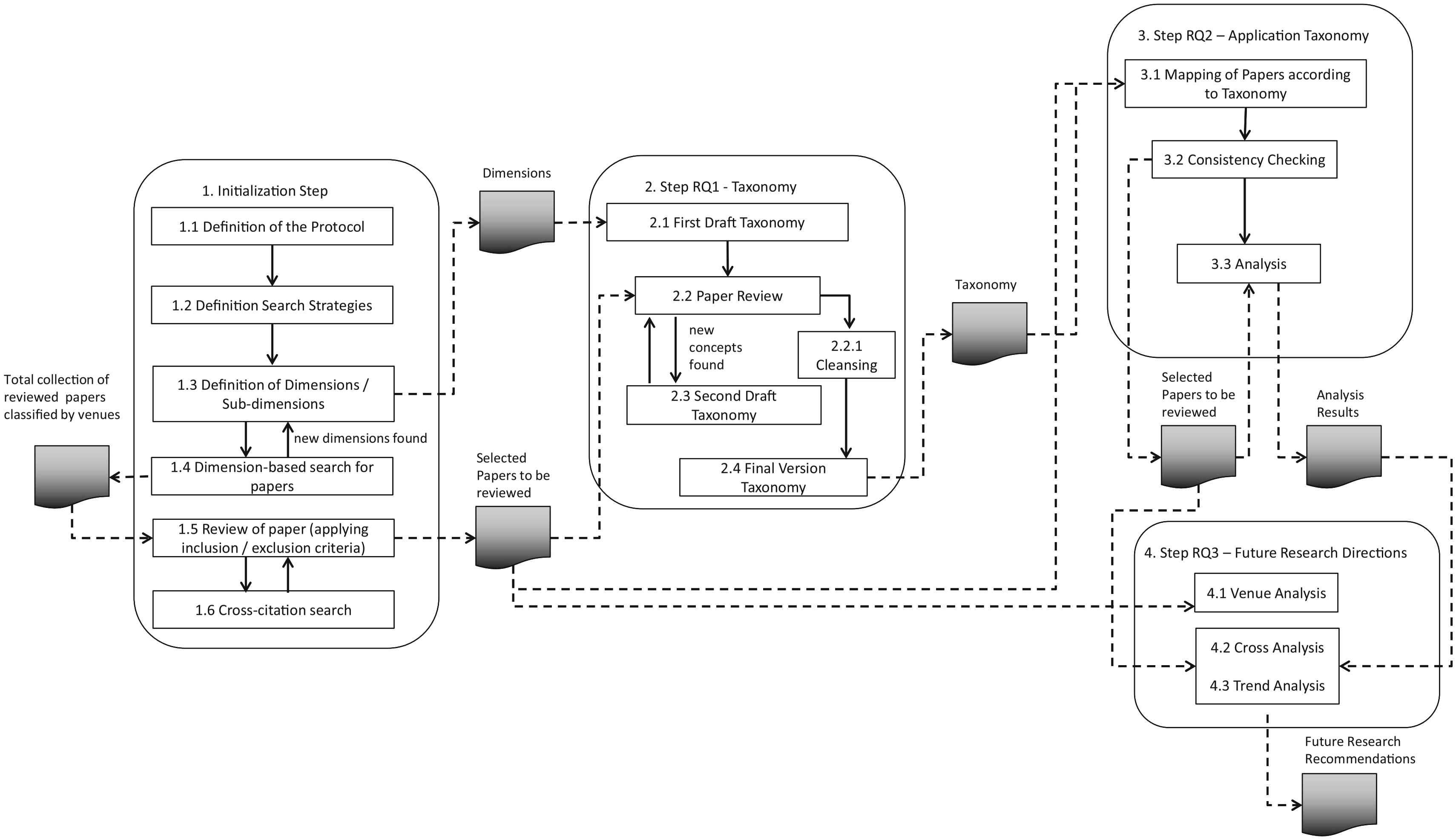

1. Initialization Step
1.1 Definition of the Protocol
1.2 Definition Search Strategies
1.3 Definition of Dimensions / Sub-dimensions
new dimensions found
1.4 Dimension-based search for papers
1.5 Review of paper (applying inclusion / exclusion criteria)
1.6 Cross-citation search
Total collection of reviewed papers classified by venues
Dimensions
Selected Papers to be reviewed
2. Step RQ1 - Taxonomy
2.1 First Draft Taxonomy
2.2 Paper Review
new concepts found
2.2.1 Cleansing
2.3 Second Draft Taxonomy
2.4 Final Version Taxonomy
Taxonomy
3. Step RQ2 – Application Taxonomy
3.1 Mapping of Papers according to Taxonomy
3.2 Consistency Checking
3.3 Analysis
Selected Papers to be reviewed
Analysis Results
4. Step RQ3 – Future Research Directions
4.1 Venue Analysis
4.2 Cross Analysis
4.3 Trend Analysis
Future Research Recommendations

Fig. 1 Procedure followed in the research method applied for the survey presented

Table 1 Quantified overview of included papers in this survey

Name	Type	#All papers	#Included
Models@run.time	Workshop	95	81
Requirements@run.time	Workshop	13	11
RAM-SE	Workshop	61	6
Ref. [30]	Book	11	9
MODELS	Conference	545	23
SEAMS	Conference	182	29
ICAC	Conference	366	31
SASO	Conference	385	11
CompArch	Conference	330	7
ECSA/WICSA	Conference	334	8
RE	Conference	528	6
SPLC	Conference	120	2
ICSE	Conference	640	6
Refs. [35,37]	Journal	25	9
SoSyM	Journal	n/a	3
JSS	Journal	n/a	2
TOSEM	Journal	n/a	1
TSE	Journal	n/a	4
TAAS	Journal	n/a	4
Google Scholar Search	n/a	n/a	24
Total		3635	275

community; RE representing the requirements engineering community; SPLC representing the software product line community; and ICSE as a general conference spanning all topics of software engineering. We also included in the search the following journals: SoSyM, TOSEM, TSE, TAAC, JSS, TAAS.

The inclusion of conferences and journals as sources of information for this evaluation aims at identifying mature work on models@run.time. To also include work on models@run.time which is at an early stage, we additionally included the following workshops: the workshop on adaptive and reflective middleware (ARM), which originally inspired the models@run.time workshop; the workshop on reflection, AOP and metadata for software evolution (RAM-SE); the European workshop on software architecture (EWSA); the requirements@run.time workshop and the workshop on dynamic software product lines (DSPL).

In order to ensure the inclusion of relevant work, we additionally performed a thorough search through a big spectrum of published papers with Google Scholar, using the keywords "runtime model" and "models@run.time". We indeed found additional papers that had not been classified before.

2.3.1 Inclusion criteria

We used the following criteria to filter the papers taken from the data sources described above, based on their title and abstract. For a paper to be included in this survey, it has to meet the following key requirement:

- The paper covers research where a model, which reflects the state of a system, should be causally connected with that system.

We consider every abstract representation of a system for a given purpose as a model, following the definition by [149]. By this, we do not only include structural representations of the system, but also models focusing on a certain aspect of the system, such as performance or variability.

For the causal connection, we follow the definition by Maes: "A system is said to be causally connected to its domain if the internal structures and the domain they represent are linked in such a way that if one of them changes, this leads to a corresponding effect upon the other" [123, p. 2]. Notably, this does not require that changes in the runtime model can be directly mapped to changes in the system, but includes approaches, where the effects are computed by (more complex) reasoning (e.g., performance models).

To ease the detection of papers to be included, we especially emphasized on the fact that at least one of the following characteristics is approached in the paper:

- The paper addresses runtime models or explicitly uses the term models@run.time.

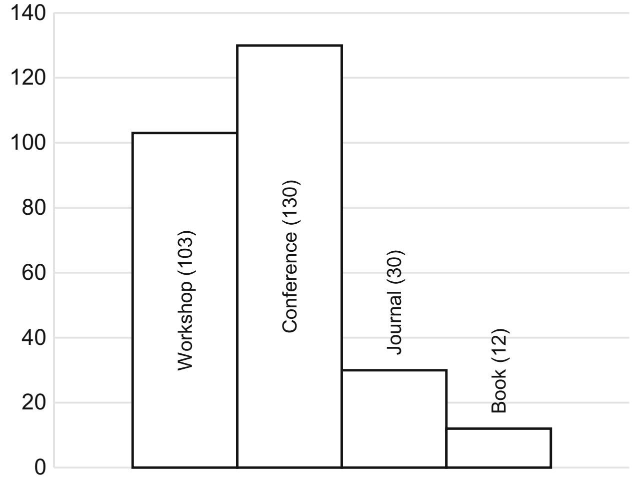


Fig. 2 Distribution of papers by venue types

- The paper uses self-representation, reflection or self-modelling.

While the first characteristic can only be met by papers after 2006, when the term models@run.time was coined for the first time, the second criteria can be met by papers published before 2006. As explained earlier, papers published before 2006 have been included as well, as they have also contributed in a significant way to the development of the research topic.

2.3.2 Exclusion criteria

Approaches on executable models are not to be considered models@run.time approaches, if they lack the causal connection to the system, but are the actual system. The survey only includes papers published until December 2017.

2.4 Overview of all included papers

In total, our survey covers 275 papers. As a first interesting observation, the distribution of these papers among the venue types where they have been published shows that models@run.time indeed has matured: most papers are published on conferences as illustrated in Fig. 2.

Another observation is that more and more work on models@run.time is not applying models@run.time for some specific purpose, but addresses fundamental research questions of models@run.time itself. Figure 3 illustrates that one-fifth of all papers captured by this survey addresses fundamental research questions.

A diagrammatic overview of the number of models@run.time papers published over the years is illustrated in Fig. 4. It can be seen that after 2007 the number drastically increased and, except for 2016, moves between 21 and 33 papers per year. The seemingly declining trend starting after 2014 started to reverse last year.

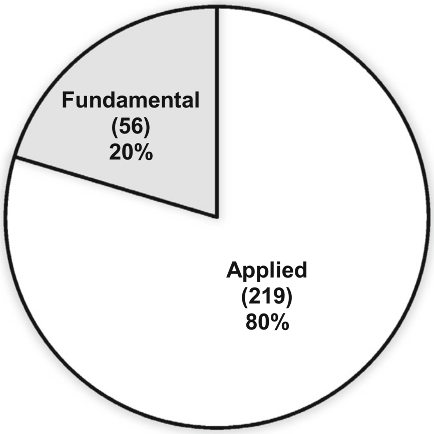


Fig. 3 Applied versus fundamental research

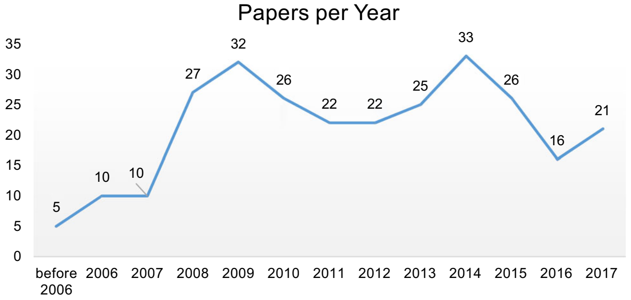


Fig. 4 Publication trend over the years

3 A taxonomy for research on models@run.time

In this section, we provide a general description of our taxonomy and a detailed description and analysis of its five major dimensions: applied research, fundamental research, application domains, intersecting research areas and supporting research initiatives.

3.1 Conceiving the taxonomy

A first draft of the taxonomy was identified according to 7 general questions, which are in line with seven (7) of the dimensions shown in the taxonomy proposed. The different nature of the first 4 and last 3 questions, was already visible at this point in time, as the questions have been grouped into primary and secondary questions.

The primary questions used as basis for the taxonomy were:

1. What software engineering artefact is modelled? (e.g., code or architecture).
2. What types of runtime models have been employed? (e.g., structural or behavioral models).
3. What are the purposes of the runtime model? (e.g., to build self-adaptive systems).
4. Which techniques from Model-driven Software Development have been used? (e.g., model comparison).

The secondary questions did not refer to a runtime model or concrete technique, but were more general:

5. To which application domain does the paper refer (e.g., automotive software engineering or ambient assisted living).
6. Which research areas are intersecting with the paper? (e.g., software product lines or model checking).
7. Which research initiatives funded or supported the conducted work?

The final quality of the results of undertaking a literature review process highly depends on the taxonomy applied, and therefore, the questions above are relevant. The taxonomy has driven the process and the understanding and has also allowed us to record each research initiative reported. As a prerequisite for the development of a taxonomy, an initial set of dimensions cross-cutting the research area were identified. Based upon our experience and reviewing papers from the various data sources described previously, we derived the following seven (7) dimensions.

- Modelled Artefacts
- Types of Used Models
- Purposes of Runtime Models
- Applied Techniques
- Domain of Application
- Intersecting Research Areas
- Related Research Initiatives

After an initial taxonomy was identified, we employed an iterative process to refine the taxonomy dimensions. While reviewing the collected papers with the aim to assign the papers to the dimensions described above, another general dimension was identified: the type of research, which is partitioned into *applied* and *fundamental* research. The first four dimensions are specific to applied research, while the remaining three are separate dimensions besides the type of research.

Based on the above, the final taxonomy was derived and is depicted in Fig. 5.

We have conceived the survey with tool support to update the data over time. We have developed a toolkit [94] that supports four main features: importing bibliographies, the specification of the taxonomy, classification of the literature and generation of analysis diagrams. The tool as well as the data of this survey is available open-source online.[3] We foresee that independent researchers can download the toolkit and contribute to the survey by supporting its evolution over the years, i.e., by placing themselves in the process described in Fig. 1. By this, the toolkit forms the basis for the sustainability of the survey.

In the following, we discuss in detail each of the fundamental questions described above to define the final taxonomy. For each question, we discuss the values of the corresponding dimension. That is, each dimension has a given number of possible values used to characterize the studied models@run.time approaches. As an example, the dimension *Modelled Artefacts* has at least the following values *Goals/Aims/Requirements*, *Architecture*, *Components* and *Code*. In the following, we briefly discuss these values. For each value we found, we show in braces how many papers fall into it. Note that some papers fall into multiple values per dimension, for example, an approach combining goal and architectural models. For each class, we provide three exemplary references, if there are more than three papers classified with it. We selected the three most recent publications while putting preference on journals first, conference papers second, workshop publications third and book chapters last.

3.2 Applied research on models@run.time

In the following, the four dimensions to classify applied research on models@run.time are discussed in detail, which correspond to the first level in the dimensions shown in Fig. 5, i.e., the *modelled artefacts*, the *types of runtime models* applied, the *purposes* for which the runtime models have been used and the *techniques* from model-driven software development that have been used.

3.2.1 Modelled artefacts

Abstraction is the neglection of unnecessary detail and an inherent characteristic of models by definition, as models are “abstractions of a subject for a given purpose” [149]. Models@run.time is specialized on abstracting the runtime state of a system. This runtime state can be observed from different levels of abstraction.

The most coarse-grain view onto a system is its architecture, i.e., the top-level components and how they are connected. Here, details of how the components work are neglected. Instead, the focus is on how the top-level components interact and which components depend on each other. For example, a traffic management system which focuses on streets and vehicles.

[3] https://github.com/sebastiangoetz/slr-toolkit.

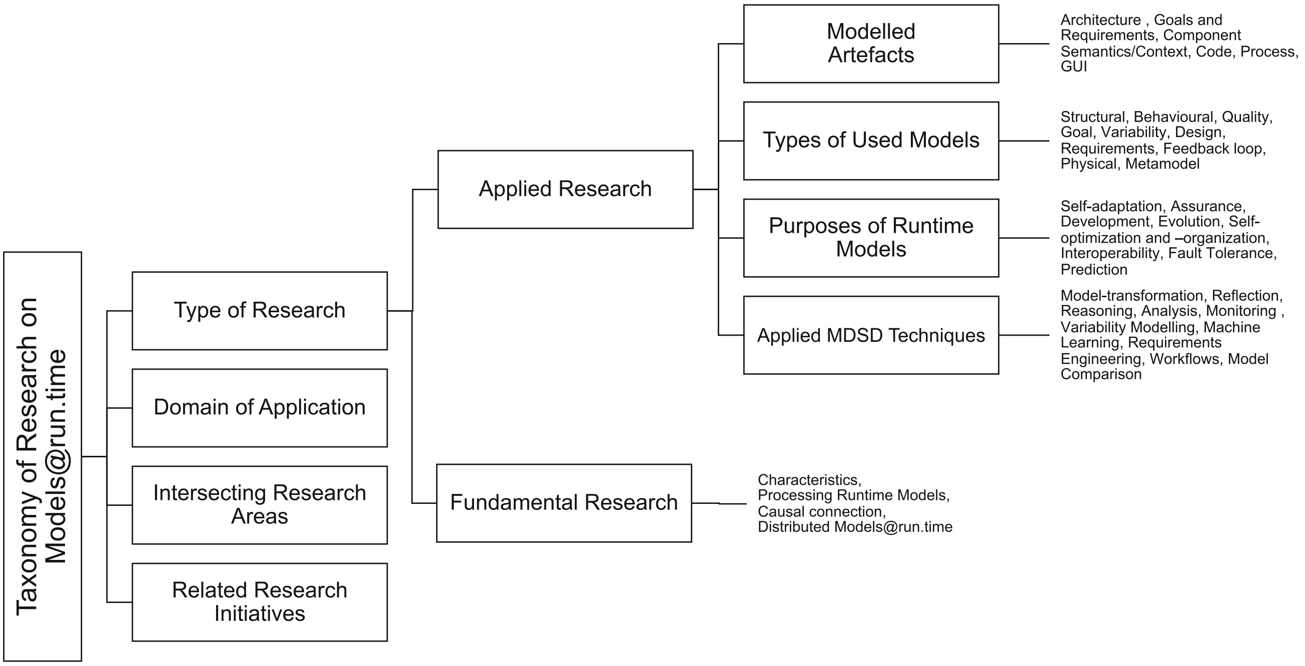

Taxonomy of Research on Models@run.time
Type of Research
Domain of Application
Intersecting Research Areas
Related Research Initiatives
Applied Research
Fundamental Research
Modelled Artefacts
Types of Used Models
Purposes of Runtime Models
Applied MDSD Techniques
Architecture , Goals and Requirements, Component Semantics/Context, Code, Process, GUI
Structural, Behavioural, Quality, Goal, Variability, Design, Requirements, Feedback loop, Physical, Metamodel
Self-adaptation, Assurance, Development, Evolution, Self-optimization and –organization, Interoperability, Fault Tolerance, Prediction
Model-transformation, Reflection, Reasoning, Analysis, Monitoring , Variability Modelling, Machine Learning, Requirements Engineering, Workflows, Model Comparison
Characteristics, Processing Runtime Models, Causal connection, Distributed Models@run.time

Fig. 5 Taxonomy of research on Models@run.time

A finer-grained view is, if individual components are modelled to capture, how they provide the offered functionality. Here, models of a traffic light's or a vehicle's behaviour serve as an example.

At the lowest level, software can be abstracted on code level, where each individual statement is considered important. A typical notation for code-level models is abstract syntax trees. Among the various purposes to keep and work on abstract syntax trees at system runtime are well-formed code-composition [112] and dynamic software updating [135].

In addition to modelling the system itself, runtime models are also constructed from some specific perspectives of abstraction, such as the semantics and context of the system's domain and external environment, the status of system with respect to its goals and requirements, the status of the system's process and its GUI. These abstraction perspectives may cross different levels of architecture, component and code. Hence, this dimension in general covers different perspectives of abstraction related to modelled artefacts and not just the perspectives of levels of abstraction.

The modelled artefact is often not applicable to *fundamental* approaches of models@run.time, i.e., approaches focusing on how runtime models should be represented and processed. This includes, in particular, meta-modelling for runtime models, model transformations (incl. megamodel processing) and techniques to evaluate runtime models.

The artefacts we found during our study are listed below. For each type of artefact, the number of occurrences is given in braces. The list is sorted according to the number of these occurrences, except for the value "none", which is always at the bottom of the list.

- **Architecture (131)** denotes a runtime model representing the current state of the *static* structure of the system's components and connectors (e.g., [71,101,132]).
- **Goals and requirements (32)** denotes a runtime model representing the state of the goals a system aims to achieve and the fulfilment of its requirements (occurs especially for self-optimizing systems) (e.g., [46,48,68]).
- **Component (26)** denotes a runtime model per component, not taking the connections between components into account (e.g., [38,75,126]).
- **Semantics/context (20)** denotes the current state of the system's domain instead of the system itself (e.g., [12, 82,86]).
- **Code (16)** denotes a runtime model representing the static (e.g., an abstract syntax tree) or dynamic (e.g., a state chart) structure of source code or code in an intermediate representation (e.g., [60,89,127]).
- **Process (12)** denotes the current state of a system's dynamic structure, i.e., which components exist and when or in which order they interact with each other (e.g., [15,77,183]).
- **GUI (1)** denotes a description of the current state of a system's graphical user interface ([53]).
- **None (37)** is used for fundamental approaches, which do not refer to a specific modelled artefact.

A comparative discussion on these findings is given in Sect. 5.1 for model types, Sect. 5.2 for purposes and Sect. 5.3 for modelling techniques.

3.2.2 Types of models@run.time

Besides the artefacts captured in runtime models, we also investigated which types of models were used at runtime. We identified the following types of causally connected runtime models in our study. For each model type, the number of occurrences is given in braces. Please note that some papers presented approaches with more than one type of runtime model.

- **Structural (131)** denotes a runtime model capturing the system constituents and their state (e.g., [71,101,132]).
- **Behavioural (45)** denotes a runtime model capturing the dynamics of the systems, i.e., what the system can or will do based on its current state (e.g., [15,38,77]).
- **Quality (23)** denotes models describing the current values of quality properties (i.e., non-functional properties) of a system or its constituents (e.g., [4,48,96]).
- **Goal (14)** denotes a runtime model capturing the current state of the system's goals (e.g., if they are currently fulfilled or violated) (e.g., [42,47,150]).
- **Variability (10)** denotes a runtime model capturing possible variants of the system or it's constituents and which variant is currently in use (e.g., [46,130,187]).
- **Design (7)** denotes design-time decisions, which are continuously synchronized with an evolving running system (used, e.g, for eternal system approaches) (e.g., [75,153, 170]).
- **Requirements (6)** denotes models representing the current set of requirements a system has to meet (e.g., [63, 68,178]).
- **Feedback loop (6)** denotes models describing one or more feedback loops, their connections and current state (i.e., are a special type of behavioural model) (e.g., [118,179,183]).
- **Physical (4)** denotes models describing the dynamics and current state of physical (i.e., continuous) phenomena (e.g., in Simulink) (e.g., [109,111,175]).
- **Metamodel (3)** denotes runtime models that are used to specify how the system or its environment are modelled (e.g., [119,147,163]).
- **None (26)** some fundamental approaches did not focus on a particular type of runtime model.

A comparative discussion on these findings is given in Sect. 5.1 for modelled artefacts, Sect. 5.4 for purposes and Sect. 5.5 for modelling techniques.

3.2.3 Purposes

An important aspect of applied research on models@run.time is the purpose for which runtime models are actually used. We, hence, investigated our body of literature in this regard and found the following purposes for using models@run.time.

- **Self-adaptation (123)** denotes the application of runtime models to build and/or operate self-adaptive systems (e.g., [48,71,75]).
- **Assurance (41)** denotes the application of runtime models to assure selected non-functional properties of a running system (e.g., [4,132,159]).
- **Development (32)** denotes the combination of runtime models with models from the development process to enable the usage of design-time knowledge at runtime (e.g., [6,49,126]).
- **Evolution (17)** denotes the application of runtime models to ease or enable the evolution of a software product (e.g., [5,44,68]).
- **Self-optimization and -organization (18)** denotes the application of runtime models to build and/or operate self-optimizing or -organizing systems[4] (e.g., [87,96, 101]).
- **Interoperability (9)** denotes the application of runtime models to bridge architectural mismatches between individual systems (e.g., [28,33,38]).
- **Fault tolerance (7)** denotes the application of runtime models to increase the fault tolerance of systems (e.g., [2, 76,169]).
- **Prediction (6)** denotes the application of runtime models to predict the behaviour of a system under observation (e.g., [65,97,161]).
- **None (22)** denotes approaches, which belong to fundamental research, where no particular purpose can be identified.

A comparative discussion on these findings is given in Sect. 5.2 for modelled artefacts, Sect. 5.4 for model types and Sect. 5.6 for modelling techniques.

3.2.4 Techniques

The fourth dimension denotes the model-driven techniques used in our body of literature on models@run.time. The intention of having this dimension is to enable an analysis of which techniques from model-driven software develop-

[4] Self-optimizing systems are a special subclass of self-adaptive systems [151]. Approaches of this class are not included in class *self-adaptation* to enable a separate investigation. Other subclasses of self-adaptive systems did not reveal to be significant.

ment, which have traditionally been designed to be used at design time, are now transferred to be used at runtime. The following list summarizes the techniques that we found are now applied at runtime:

- **Model-transformation (54)** denotes all techniques which realize a transformation of one model into another (e.g., [38,49,54]).
- **Reflection (42)** denotes techniques used to realize a causal connection between a subject and its model (e.g., [1,89,97]).
- **Reasoning (33)** denotes techniques to reason about multiple models. For example, to enable self-adaptation and optimization (e.g., [2,87,132]).
- **Analysis (30)** denotes techniques to gain novel insights from models. For example, to check their consistency (e.g., [15,77,101]).
- **Monitoring (24)** denotes techniques to capturing observations of a running system in a model (e.g., [125,126, 162]).
- **Variability modelling (22)** denotes techniques to model variants of a system (e.g., [3,68,82]).
- **Machine learning (15)** denotes techniques to give "computers the ability to learn without being explicitly programmed" [152] (e.g., [48,75,99]) and to capture this knowledge in models.
- **Requirements engineering (10)** denotes techniques to capture and assess the requirements of a system in models (e.g., [17,173,189]).
- **Workflows (6)** denotes techniques to model and execute complex processes (e.g., [110,117,131]).
- **Model comparison (4)** denotes techniques comparing models with each other (e.g., [128,129,137]).
- **None (34)** denotes approaches, which belong to fundamental research, where no particular technique can be identified.

A comparative discussion on these findings is given in Sect. 5.3 for modelled artefacts, Sect. 5.5 for model types and 5.6 for purposes.

3.3 Fundamental research on models@run.time

Besides approaches which make use of runtime models, we also identified 56 approaches, which aim at improving models@run.time as such fundamentally, i.e., fundamental approaches for models@run.time. For such approaches, most often the dimensions of applied research for models@run.time are not applicable. For example, an approach focusing on extending the expressiveness of runtime models, e.g., to capture a history of how each value changed over time, does not have a particular purpose for which the runtime model is applied. Instead, applied approaches can make use of fundamental approaches.

Our study revealed four main types of fundamental research on models@run.time which are explained in the following:

- **Characteristics (29)** of runtime models, i.e., approaches extending the expressiveness of runtime models to capture, e.g., temporal, continuous or spatial characteristics (e.g., [71,99,132]).
- **Processing runtime models (11)** Approaches investigating novel ways to utilize runtime models, e.g., approaches to model workflows of individual processing steps of a runtime model (e.g., [54,89,100]).
- **Causal connection (9)** Approaches trying to improve the way a system and its model are kept in synchronization, e.g., approaches introducing the concept of transactions to the causal connection (e.g., [13,88,143]).
- **Distributed Models@run.time (7)** Approaches investigating the effects and new challenges when multiple systems, each having their own runtime model, have to work together (e.g., [58,179,183]).

An interesting observation is that most fundamental work focused on novel characteristics for runtime models. But, the remaining three types of fundamental research are likely to be addressed more heavily in future work, due to the effects of the Internet of Things, for example.

3.4 Application domains

Alongside the type of research, we also classified our body of literature with respect to the domain to which the approach has been applied. Typically, the application domain denotes the origin of the case studies used to evaluate the respective approach. A surprising observation is that most approaches have been evaluated in the domain of enterprise software. Additionally, it can be observed that the list of domains where models@run.time has been applied is rather long (25 domains), showing the general applicability of models@run.time. On the contrary, almost half of all approaches (127) have not been applied to a specific domain. We refer the interested reader to the appendix for the complete list of the domains.

3.5 Intersecting research areas

While the application domain discussed in the previous subsection focuses on the domain in which case studies have been conducted to evaluate the respective approach, the dimension of intersecting research areas denotes which other research disciplines are addressed. In other words, the intersecting research areas denote areas to which the approach is contributing instead of just using it as a domain for evaluation.

In the following, we list all interfacing research areas we found by classifying our body of literature.

- Self-adaptive Software (73)
- Model-driven Software Development (34)
- Software Architecture (27)
- Distributed Systems (14)
- Formal Methods (12)
- Resource Management (11)
- Cloud Computing (9)
- Software Evolution (9)
- Security (9)
- Requirements Engineering (9)
- Fault Tolerance (8)
- Programming Languages (7)
- Software Product Lines (7)
- Autonomic Computing (5)
- Aspect-oriented Programming (5)
- Interoperability (5)
- Database Engineering (4)
- Multi-agent Systems (4)
- Performance Engineering (4)
- Business Process Engineering (3)
- Embedded Systems (3)
- Human–Computer Interaction (3)
- Optimization (3)
- Safety Engineering (3)
- Social Sciences (2)
- None (26)

Interestingly, the list of interfacing research areas is long (25). This emphasizes the high degree of interdisciplinarity of models@run.time as a research area. It can also be seen that there is a strong overlap with the research areas self-adaptive systems and model-driven software development, which is not surprising as models@run.time originated from these two research areas.

3.6 Supporting research initiatives

Finally, we investigated the research initiatives in which work on models@run.time has been conducted. We focused our search on research projects and found that there has been a large variety of research projects and funding agencies supporting work on models@run.time, but (except for very few projects, such as DiVA, MORISIA and M@TURE) a minority explicitly focuses on models@run.time. The above shows that models@run.time has also become an underlying technology. However, the small number of projects explicitly focusing on models@run.time hinders its further development with respect to the fundamental dimension. A structured list of all identified research projects can be found in the appendix of this paper.

4 Threats to validity

As in any survey, there are several threats to the validity of our study. In the following, we discuss the different aspects of these threats.

Research questions The research questions defined might not provide complete coverage of the current research field. To address this threat, we had several discussions, collected initial feedback from the community in the discussion sessions of our annual workshop on models@run.time and followed a thorough reflection process to validate the questions. The research questions were in line with the dimensions of the taxonomy identified and refined.

Set of dimensions An initial set of dimensions cross-cutting the research area of models@runtime were identified. We cannot guarantee that all relevant dimensions were identified. It is possible that other dimensions were missed or the current dimensions were biased. Also, the initial draft of the taxonomy was based on the experience of the authors, although collected by a constant exchange with the community over several years. (The study was conducted over a period of 3 years.) The latter may have steered the revisions of the taxonomy in a restricted way. In consequence, it may be the case that possibly not all classes were identified due to the restricted set of initial dimensions. We mitigated this characteristic threat to the extent possible by receiving early feedback and with discussions during our annual workshop. Also, we underwent an initial review of seminal papers from the various data sources following a thorough reflection process that ended in a refined set of dimensions.

Publication bias As in any study, we show no guarantee that all relevant studies were selected. It is possible that some relevant studies were not chosen throughout the search process. However, we think we have mitigated this threat. We decided against a keyword-based search to gather an initial set of papers subject to review. Instead, we initially targeted several relevant and well-known conferences, workshops and journals as the initial data source. The venues have a strong focus on software engineering. Although, using this approach, it is likely to find more relevant papers compared to the keyword-based search, there is still the possibility to miss relevant papers, which have been published in conferences, workshops, journals or books, which were not included as a data source. The latter was mitigated by double checking with a thorough search using Google Scholar with specific keywords (e.g., "runtime model" and "models@runtime") to allow the search through a big spectrum of publications (including IEEE and ACM publications and others).

Search conducted To mitigate the bias included in any survey when performing the search, we complemented our venue-based search with a keyword-based search. First, our search was based on an initial list of relevant venues, which was then extended by searches performed in digital databases. By this, other publications that were not considered initially were included too, as they were not published in the traditional venues where models@run.time and software engineering publications are usually hold, but are at the same time known in the community. The result is that our survey potentially covers publication venues from different domains, too. For example, several papers published at ICAC would not have been found by a keyword-based search, because they do not explicitly point out the usage of a runtime model. Instead, the performance models described in these papers are naturally runtime models, and the authors had no need to point this out explicitly.

Data extraction During the extraction process, the analysis was conducted based on our own judgement. However, despite double checking, some studies may have been classified incorrectly. In order to mitigate this threat, the classification process was performed and double checked by more than one researcher. Also, undergraduate students helped during the process by continuously developing an open-source toolkit[5] and therefore avoid potential pitfalls.

5 Cross-dimensional analysis of the taxonomy

In the following, we describe a cross-dimensional analysis of the work that has been conducted for all six combinations of the four applied research dimensions, i.e., the *modelled artefacts*, the *type of runtime model*, the *purpose of using a runtime model* and the *modelling techniques* transferred to runtime. The aim of this analysis is to identify gaps in the research landscape of models@run.time. The analysis highlights value pairs, which either have been extensively addressed (e.g., the use of structural models for self-adaptation) or have not been addressed at all (e.g., variability models for assurance). For this, we illustrate and analyze for each combination the number of approaches we have found during the survey.

The figures in the following sections list the values of the two dimensions under comparison as a bubble matrix chart. In all figures for each dimension, the value *none* has been omitted for clarity.[6] For each value, a number is provided in braces before its name. This number denotes how many papers have been classified with this value and any value of the other dimension. Notably, this excludes papers which have not been classified in the other dimension (i.e., are classified as "none"). Accordingly, the numbers shown here can be smaller than those shown in Sect. 3.2. For example, in Fig. 6, the value "architecture" is accounted as 130, while in Sect. 3.2.1, it is accounted as 131. The missing paper is of fundamental nature, has a clear focus on architecture, but does not focus on any particular type of runtime model. Each bubble in the bubble matrix charts depicts the number of approaches, which have been classified with respect to the two dimensions at this intersection. This number is visualized by differently sized circles. For the combination with the most approaches, the circle spans the full size of the cell. For other combinations, the size is scaled with respect to this reference number. For each combination with at least 10 approaches, the number is shown additionally.

In this section, we draw conclusions based on the data we have found, but do not derive challenges yet. The resulting challenges will be explained in the succeeding section.

5.1 Comparing the modelled artefacts and types of runtime models

Figure 6 shows that the most common combination is the use of structural runtime models at the architectural level (97 approaches). All remaining combinations have at most 13 approaches. Both categories are also clearly the most dominant in their respective dimension. However, besides this clear difference, further conclusions can be drawn from Fig. 6. Namely, and unsurprisingly, most work has been done on artefacts at higher levels of abstraction.

Notably, meta-models are only used in fundamental approaches [119,147,164]. As such approaches typically do not focus on a particular artefact, no approach can be found for any type of modelled artefact, which makes use of meta-models.

The sparsity of combinations in Fig. 6 indicates that many types of runtime models have only been investigated for few artefacts. For example, it seems that the use of variability models is still to be investigated on all levels except for the goal and architectural level. On the other hand, some runtime model types, like behaviour and quality models, have been investigated at almost all levels.

Finally, the graphical user interface (GUI) is a peculiar artefact. In our study, we only found one approach for this case [53], which uses structural runtime models. It can also be concluded that quality, goal and requirement models could be promising further types of runtime models to be investigated in this context.

[5] https://github.com/sebastiangoetz/slr-toolkit.

[6] Most of the 56 approaches classified as fundamental work would else be shown as dominating axes in the bubble matrix charts, which distracts from the investigation of applied approaches.

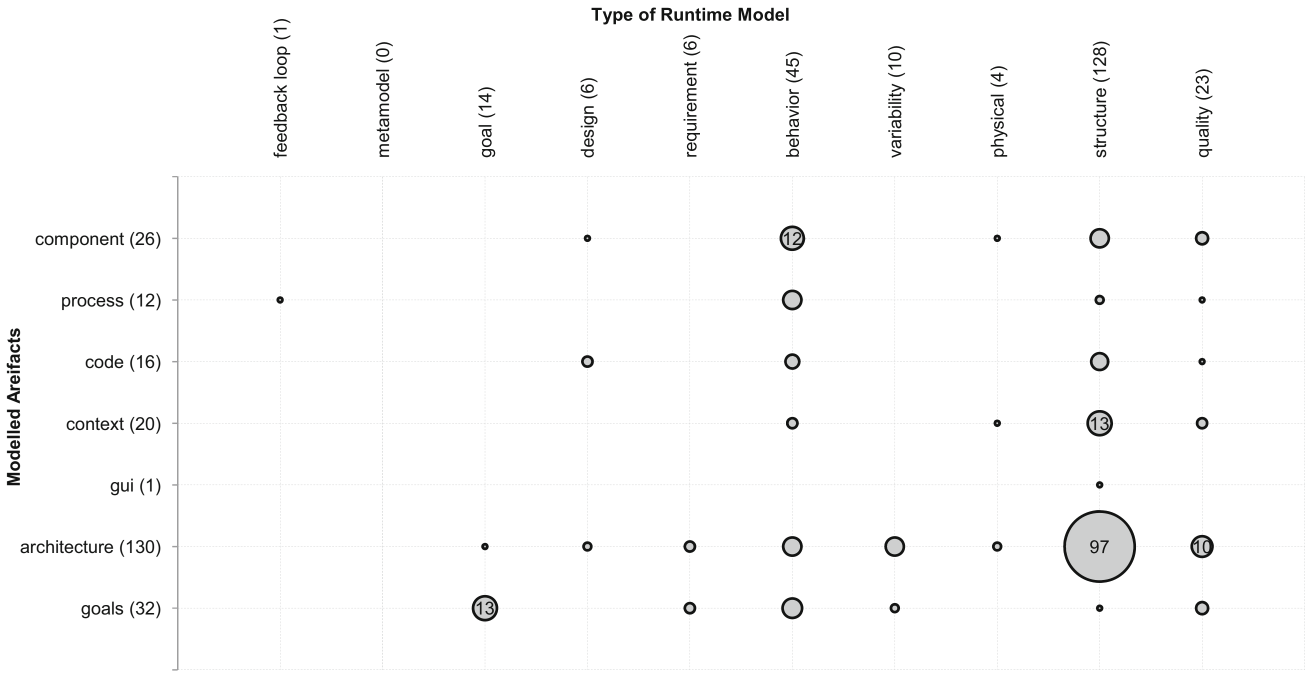


Fig. 6 Modelled artefact versus type of runtime model

5.2 Comparing the modelled artefacts and purpose of runtime model

Figure 7 shows how models@run.time for different *modelled artefacts* are used for different *purposes*.

From Fig. 7, it is evident that the combination of architecture and self-adaptation is dominant. Although this is consistent with the dominant positions of the two concepts in their own dimensions, it still indicates that an architecture-level runtime model does provide strong support for self-adaptation. In fact, modern software systems usually provide a flexible architecture in order to be able to adapt to the ever-changing environment at runtime. Models@run.time is a natural choice to enhance such an architectural level of adaptation and provides the necessary semantic basis for the system to achieve self-adaptation. A runtime model at a high abstraction level with a global view appears to be useful for self-adaptation. There are many approaches using a higher-level goal-based runtime model for self-adaptation [18,23,42]. On the contrary, there are relatively few self-adaptation approaches using runtime models on component or at source code level. Actually, we expect to see more work using context-level models at runtime to support self-adaptation, because it is still at a high abstraction level and the context changes are a main driving force behind self-adaptation.

Besides self-adaptation, there is also a significant number of approaches using architectural level models@run.time on development and assurance. Assurance used to be the main purpose of using static architecture models [51,74,90,132]. By analysis and validation of the system architecture, designers can have an early view on the high-level properties before starting the development. Therefore, it is not a surprise to see there are still many approaches using runtime information carried by an architecture model to support high-level assurance. However, what is interesting to see is that by connecting an architecture model with the running system, more researchers seek for a full-loop self-adaptation rather than only analysis and assurance of the system. An architectural runtime model is either used for developers as a reference [141], or by the program itself as a context at runtime [168].

The usage of goal-level runtime models is also concentrated on self-adaptation purposes [18,23,42]. Models@run.time at the levels of context, component and code is used as well [11,61,67]. At the context level, and besides self-adaptation, we expected more approaches on self-optimization and prediction. There are approaches attempting to use component and code-level models@run.time for almost every purpose, but so far, there is not a dominant purpose for either level revealing the real strength of models@runtime.

Finally, at the level of processes and GUIs, there are only few approaches using models@run.time, and only for few selected purposes.

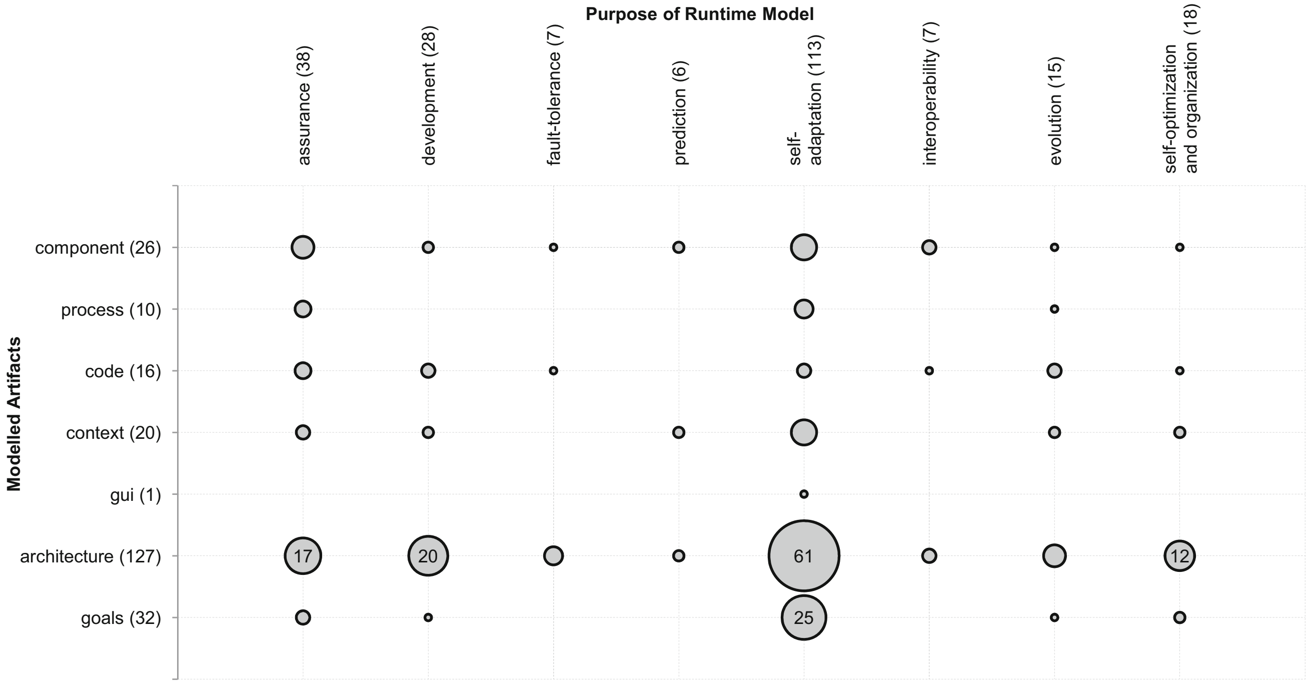


Fig. 7 Modelled artefact versus purpose of runtime model

5.3 Comparing the modelled artefacts and applied model-driven techniques

In Fig. 8, the *modelled artefacts* are mapped onto *applied model-driven techniques*.

Specifically, Fig. 8 shows that a big part of the research efforts has been towards the application of *applied model-driven techniques* at architectural levels. Specifically, at this level of abstraction, the most popular model-driven techniques are model-transformations [47,52,114,166,191,192] and reflection [97,120] followed by reasoning [11] and variability modelling [129] as well as analysis [159]. Similarly, it is also shown that there have been efforts in the application of model-driven techniques at the abstraction level of components. We concluded that it is, up to a point, related to the work performed at the architecture level of abstraction.

There is a list of natural pairs which are visible in Fig. 8, such as the application of workflow-based techniques [131,146] at the abstraction level of processes. The same applies to both, the pair of goal-based abstraction level with requirements engineering techniques [189] and goal-based abstraction level with reasoning techniques [17,42, 144,193].

The model-driven techniques reflection [14,59], model-transformations [106], monitoring [66] and analysis [60] have been virtually applied to all artefacts with the exception of the GUI, which has been less active. In fact, we only found one paper, where it has been approached with model-transformation techniques [53].

The model-driven techniques reasoning [39,57,144] and variability modelling [24,46,121,134] have been applied to different artefacts. However, there is no representation at the level of processes or GUIs.

5.4 Comparing the type of runtime model and their purpose

In Fig. 9, the *type of runtime model* is mapped onto the *purpose of the runtime model* dimension.

Among the surveyed approaches, the main driver behind models@run.time research appears to be the usage of structural models for self-adaptation [73,79,86]. A structural runtime model supports self-adaptation with a high-level, holistic view of the running system, in such a way that the self-adaptation engines can use the model to analyze the runtime phenomenon and enact the system directly. A related purpose is assurance [90], which involves mainly the analysis part of self-adaptation. In addition, structural models are also widely used in development and evolution [80,92].

Behavioural runtime models are mainly used for self-adaptation [131] and assurance [132], exposing the behaviour of the running system to the adaptation engines based on runtime models. Goal models are also manipulated at runtime, providing a high-level reference for self-adaptation [138]. Although self-adaptation is often built on top of control theory, there are not many approaches directly using a feedback loop to construct runtime models [183]. Instead, they

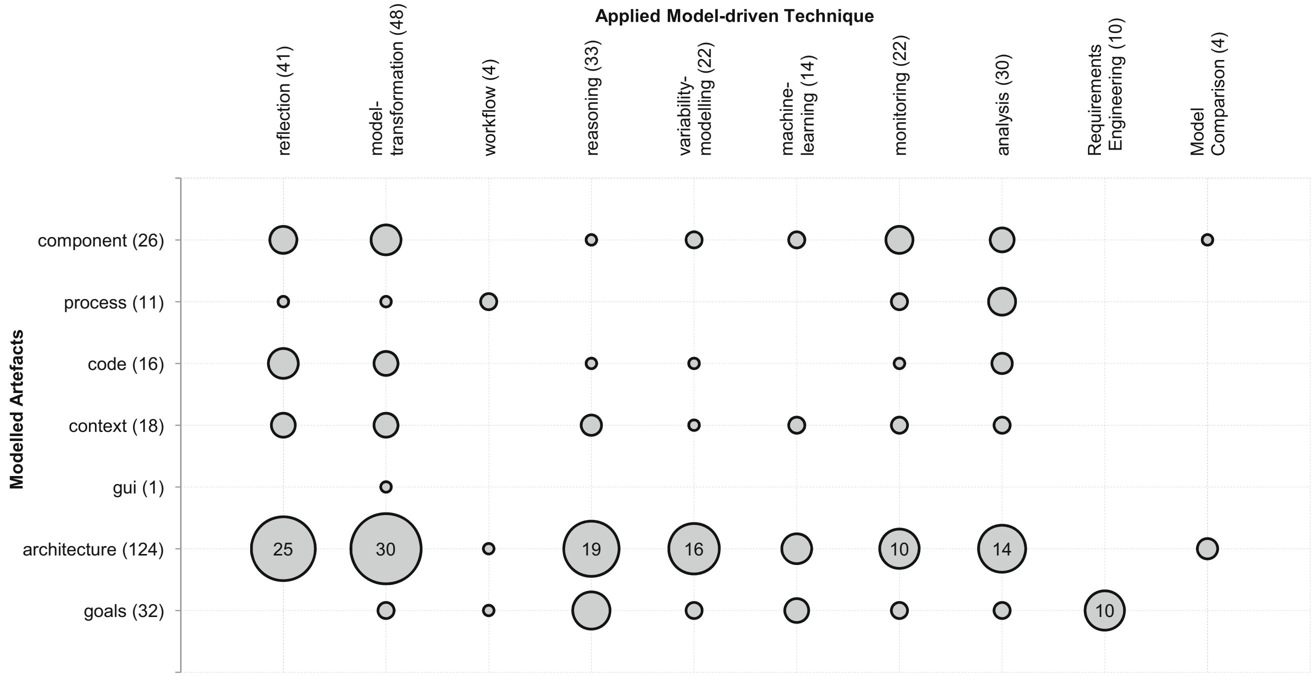


Fig. 8 Modelled artefact versus applied model-driven techniques

rather use a feedback loop on top of a structure or behaviour model [167].

Quality models are widely used at design and development time for assurance and fault-tolerance purposes, but they are not widely used as a way to construct models@run.time. A similar conclusion can be drawn for variability models. Although we admit that there is an abstraction gap between the system's runtime phenomenon and the quality or variability model, we still expect more approaches investigating the usage of these two types of models at runtime.

5.5 Comparing the type of runtime model and the applied model-driven techniques

Figure 10 depicts which *model-driven techniques* have been transferred or applied to which *types of runtime models*. Noteworthy, in this comparison, the model-driven techniques can be classified into major, middle and minor techniques in terms of the numbers of approaches we found in each category.

Not surprisingly, the major techniques are model transformations and reflection. Especially, reflection is an expected technique to appear, as it is part of the definition of models@run.time. On the other hand, model transformations are a natural prerequisite for models@run.time in order to connect the runtime model and the system it reflects. Middle used techniques are analysis, reasoning, monitoring and variability modelling. This, again, is an expected result, as all four techniques are natural means to achieve the different purposes for which runtime models are used (cmp. Sect. 5.6). Minor techniques found are model comparison, machine learning, requirements engineering and workflow modelling.

An analogous categorization can be done for the types of models. Major types are structural and behavioural models. Middle types are quality and goal models, and the remaining types can be considered minor types.

It is not surprising that the major techniques are applied to major types of models. However, an interesting exception is that model transformations and reflection, as major techniques, are applied on a relatively low level on behaviour models, compared to structural models. In contrast, analysis is the most popular technique for behavioural models. This reveals the complexity of behavioural models at runtime, and as a result the current research is more in a stage of directly observing runtime behaviours rather than utilizing them together with other models.

An interesting observation from Fig. 10 is the lack of techniques applied to runtime physical models, meta-models, feedback loops and requirement models.

5.6 Comparing applied model-driven techniques and the purpose of runtime models

Figure 11 depicts a comparison between *applied model-driven techniques* versus *purpose of runtime model*.

Among the surveyed applied model-driven techniques, self-adaptation [2,11,23] as a purpose appears as the main driver. Further, it has had a strong emphasis on the use of

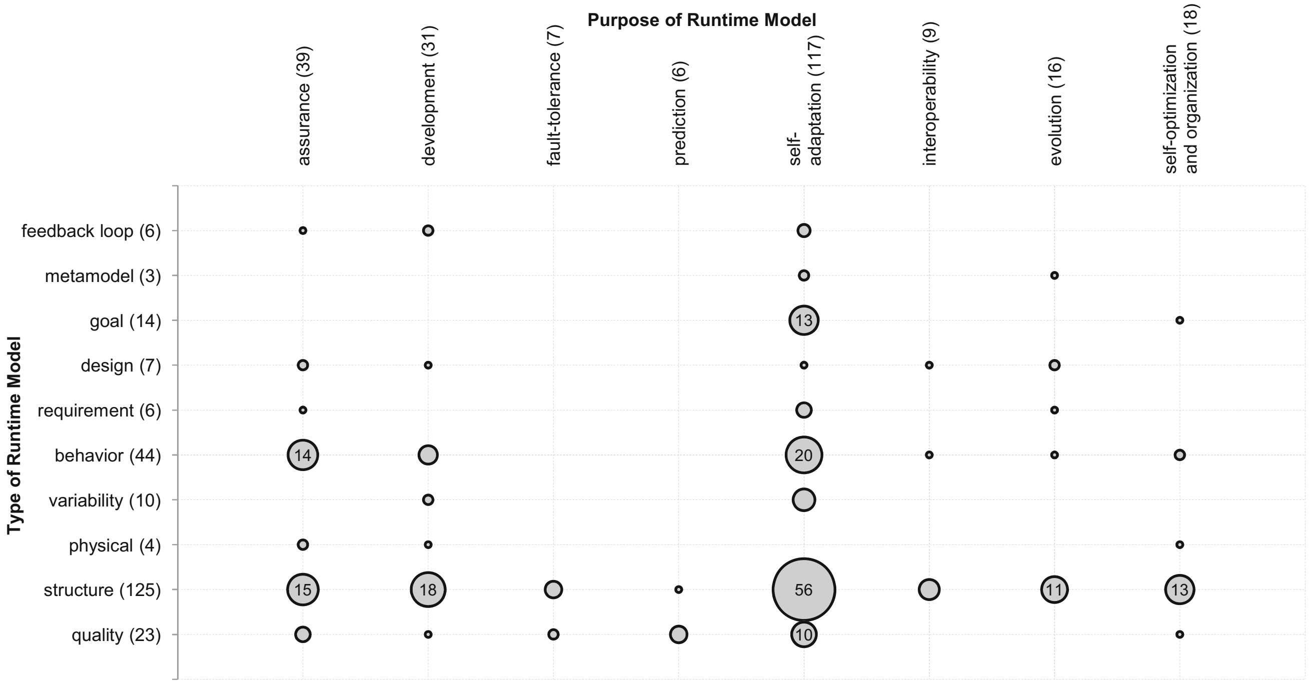


Fig. 9 Type of runtime model versus purpose of the runtime model

the techniques model-transformation [52], reasoning [55] and reflection [9,72] followed by variability modelling [45], requirements engineering techniques, analysis [77] and monitoring [78,107]. However, even if smaller, efforts have been made using techniques such as model comparison [7,130] and machine learning [28,71,75]. Techniques such as model-transformation have been applied for different purposes apart from self-adaptation such as development, evolution and interoperability.

Unsurprisingly, the technique reflection has again found to be used in all the surveyed purposes. It may be considered as predictable, as reflection is a technique that supports the implementation of models@run.time at any level of abstraction.

An observation from Fig. 11 is the limited use of model-driven techniques for prediction and fault tolerance followed by evolution and interoperability. The research efforts, in terms of the application of model-driven techniques, are related to the purposes self-adaptation, assurance and development, followed by self-optimization and interoperability.

6 Research challenges

Based on the data collected and the analysis performed and shown earlier, in this section we discuss the research challenges and the foreseen efforts needed to bring forward the state of the art up to the envisioned future of models@run.time research. We aim to classify the fundamental challenges based on the taxonomy proposed in Fig. 5. This strategically enables us to portrait the challenges in the context provided by the concepts previously discussed, and the analysis performed in Sect. 5.

In general, the analysis applied using the proposed taxonomy reveals that the research in the models@run.time community has concentrated on a set of dominant topics such as architectural runtime models, which have targeted particular purposes such as self-adaptation. As a result, potentially relevant topics, as well as their combination, have been overlooked by researchers. We hope that this survey helps to reveal some of these potentially useful combinations.

In this section, we summarize the areas where we consider that the state of the art of models@run.time can be improved. We also discuss research topics that have the potential to push forward the state of the art if explored further.

6.1 Challenges based on the modelled artefacts

The survey has shown that a big part of research efforts has been focused on the architectural level of abstraction, exploiting structural runtime models for running systems. Further, most of these approaches have targeted the particular purpose of self-adaptation. There has not been a strong focus on the use of models@run.time for neither lower levels of abstractions than those offered at the architecture level, i.e., code or components, nor higher levels of abstraction, such as systems of systems and the use of contextual situations. This section summarizes the challenges expected, both applying

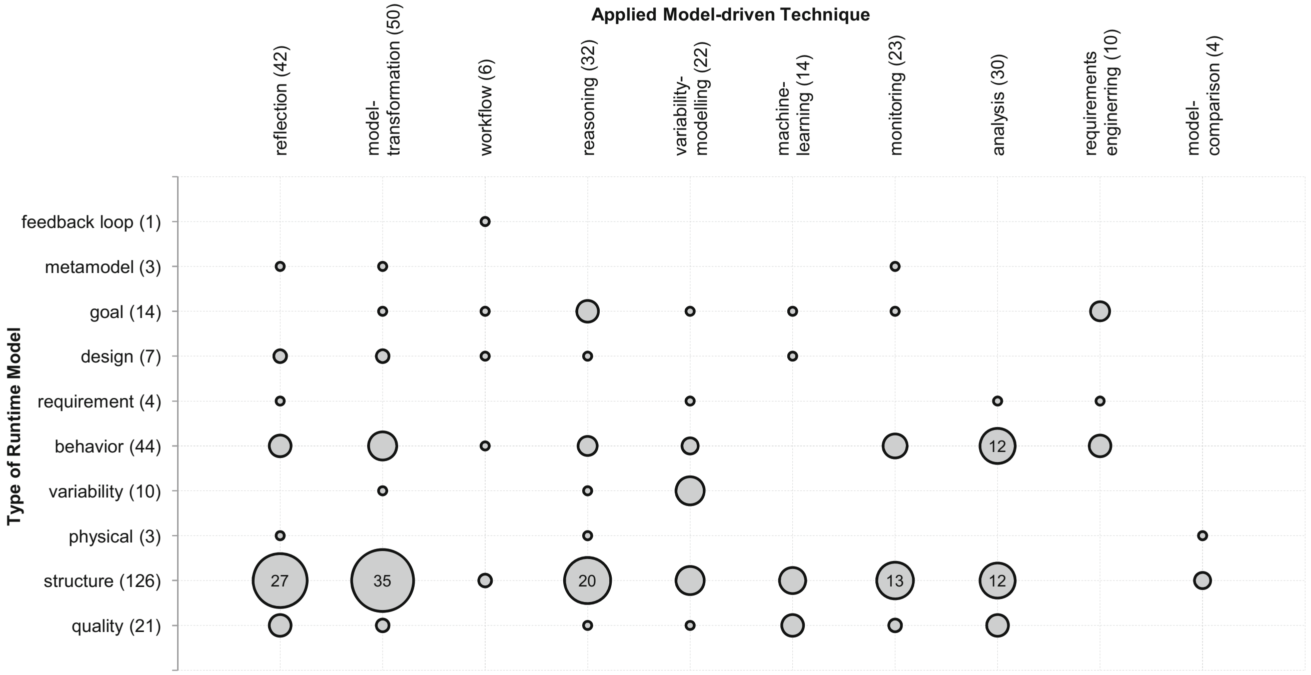


Fig. 10 Type of runtime model versus applied model-driven technique

models@run.time at lower and higher levels than architecture.

The need to apply models@run.time at lower levels than architecture Technology is complex and rapidly changing and, therefore, software engineering researchers also need to provide approaches with more direct support to developers [35] who have to deal with very specific technical knowledge. Therefore, there is the need to focus not just on architectural models, but on artefacts at lower levels of abstraction and finer-grained models, such as configurations of components and source code. For instance, authors in [43] explain initial ideas about how runtime models can be used to deal with what they call software changes in-the-small, i.e., changes at the code level, as opposed to software changes in-the-large, i.e., changes at the component or component configuration levels.

It is relevant to understand the runtime phenomenon of software systems from lower levels of abstraction than the macro-architectural level, such as the status inside individual components, or the behaviour of a particular piece of code while, at the same time, being able to exploit the use of abstractions and models. We especially foresee potential in approaches that deal with structural and behavioural runtime models used at the level of code (e.g., [5,160]). In fact, various models of code have been investigated by the compiler construction community (e.g., control flow graphs representing behavioural models or abstract syntax trees representing structural models). Traditionally, models used in compilers are not available at runtime, i.e., after the compilation process. The rising need for dynamic software updates with zero downtime (see, e.g., [135]) poses the challenge to recompile programs while they run and, hence, to keep the compiler models at runtime. As a direct consequence, models used in such compilers can be enriched with knowledge only available at system runtime. The effectiveness of combining model-driven techniques with code-level techniques from the domain of compiler construction has just recently been promoted at the MODELS 2017 conference in a keynote by Ira Baxter [20].

To support lower-level runtime models, novel techniques are required to realize the causal connection between the running system and such models. In order to keep a code-level runtime model (e.g., an abstract syntax tree) in synchronization with a running application subject to dynamic program rewriting or dynamic code analysis, current approaches focusing on architectural models cannot be directly applied (e.g., compare [160] with [165]). Current approaches do not offer enough support, and therefore, further research efforts are needed.

The need to apply models@run.time at higher level of abstractions than systems architectures The external context of systems and their interactions with other systems as is the case in systems-of-systems [124] have not yet been exhaustively investigated in the context of models@run.time. Indeed, we have not found any work on models@run.time for systems-of-systems, but some work focusing on context-adaptive systems (e.g., [105,184]). In [40], an overview of the current state of the art in self-aware computing systems [116],

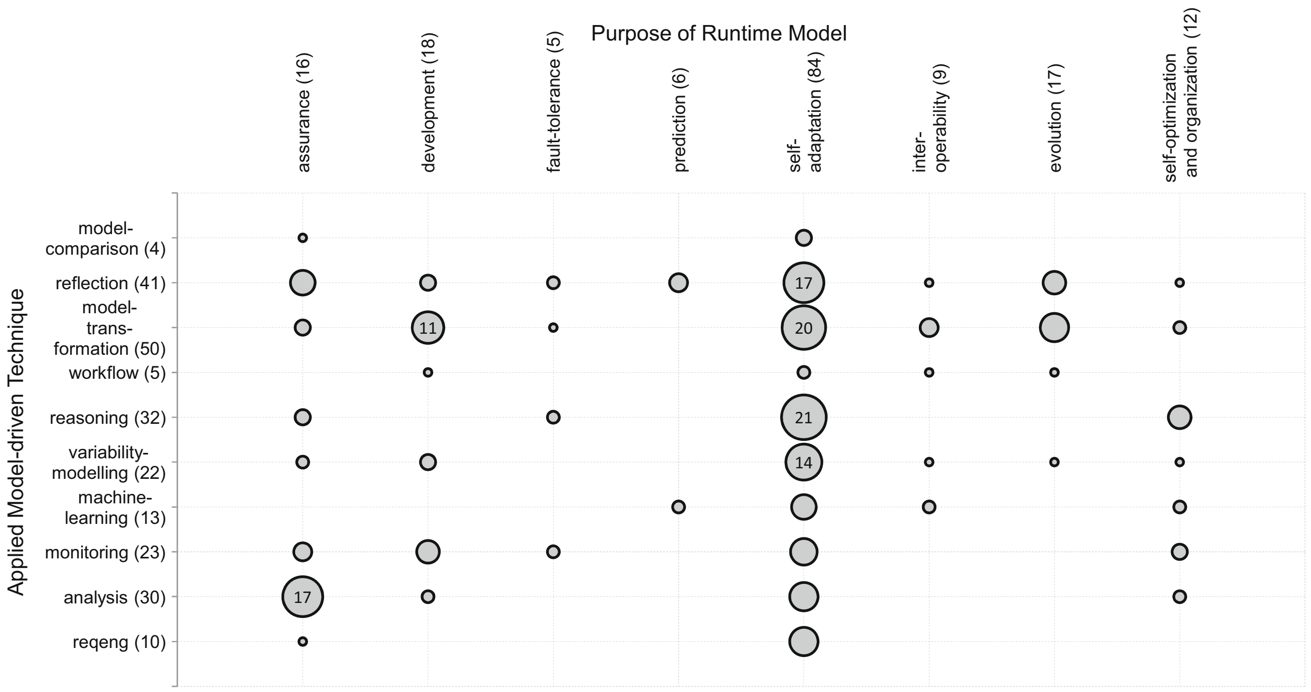


Fig. 11 Applied model-driven technique versus purpose of runtime model

i.e., systems which are aware of themselves, their context and their collaboration partners (i.e., systems-of-systems), is given. As we have shown in the previous sections, most work on models@run.time focused on self-adaptive software systems, which are—in contrast to self-aware computing systems—mainly aware of themselves, but typically neither on their environmental context nor their interaction with other systems. Current and most future types of software systems are inherently embedded into complex environments (e.g., autonomous vehicles including cars, trains and ships; or wearables like fitness trackers and health monitors) that work in different contexts. Hence, more work is required to investigate the application of models@run.time to capture and reason about the context of and interactions between these systems. Thus, we consider the application of the models@run.time paradigm to context-aware systems [103] and systems-of-systems [124] as a highly relevant research topic.

6.2 Challenges based on the runtime model type

Regarding the model types, it was observed that the ample majority of the surveyed papers represent the running system using structural models. Structural models tend to focus on how the software is composed, for example, in terms of components and their connections (i.e., architecture); or aspects and their patterns of composition. In contrast, behavioural models emphasize how the system executes, e.g., in terms of flows of events through the system. The following are the challenges we have identified with respect to the types of runtime models.

Goal models at runtime to address uncertainty According to our findings, runtime representations of goal-based models have not been studied exhaustively. (Just 14 papers were found, cf. Sect. 3.2.2.) Goal-based models allow mirroring the domain problem in declarative ways in contrast to procedural ways. Their use has opened possibilities to tackle decision-making using the support of machine learning (e.g., [23,140]) in order to tackle uncertainty, making them relevant in the design and development of future software systems. The emergence of the need to deal with uncertainty adds to the significance of the need of runtime goal models and models that in general offer support for reasoning about the domain problem in more declarative ways [25,155]. A particular aspect that needs to be addressed by modern software with respect to uncertainty is self-explanation [25]. Goal-based models have already been used to support self-explanation [188], but—as our analysis shows—have not, yet, been investigated extensively at runtime.

Towards variability models at runtime Surprisingly, only few research initiatives have investigated variability models at runtime (10 papers). Further, they have been applied only at the level of goals and architecture, which contrasts to the needs claimed by the variability community (i.e., SPLC and DSPL) [19,29,36,98,102,136]. Especially, the vision of dynamic software product lines, i.e., the ability to support runtime updates with zero downtime for products of a software product line, denotes an obvious link between variability models being used at runtime to adapt the respective programs. The challenge for dealing with runtime variability is that it should support a wide range of product customizations under various circumstances that might be unknown until execution as new product variants can be identified only at runtime [29]. Contemporary variability models face the challenge of representing runtime variability to therefore allow the modification of variation points during the execution of the system, and underpin the automation of the system reconfiguration [41]. The runtime representation of feature models (i.e., their runtime models) is required to automate the decision-making.

Towards runtime models to support feedback loops Even though the feedback loop is a core concept behind self-adaptation and it has been shown to be beneficial to separate it from the system itself [190], there are only few approaches investigating the usage of an explicit model of this feedback loop for self-adaptation (6 of 275 papers; cmp. Fig. 9). We expect that further research is done with respect to this aspect. An example of its use is to enable analyses and reasoning about feedback loops, e.g., to support resource management as shown in [118]. Multiple runtime models would be required to provide views for various stages of feedback loops, such as monitoring, analysis, decision-making or adaptation [180].

6.3 Challenges based on the purposes for using models@run.time

Regarding the purpose of models@run.time, it can be observed that self-adaptation is dominant in the surveyed approaches. Self-adaptation approaches mostly exploit structural (i.e., architecture) and behavioural models (cf. Fig. 9). Only few approaches using quality and variability models exist for self-adaptation (10 and 8 of 275 papers; cmp. Figure 9). Below we discuss challenges on the purposes for using models@run.time that can be related to self-adaptation but also can transcend it.

Model-driven techniques have been poorly investigated for the purposes of prediction, fault tolerance and interoperability (6, 7 and 9 of 275 papers; see Fig. 11). However, all three purposes are highly required to build future software systems. For example, for the Internet of Things (IoT), vast amounts of connected devices from different vendors need to work together by interoperability [69]. Some of these devices will be safety critical (e.g., autonomous cars) and, hence, need to be fault-tolerant. Finally, IoT devices have to operate energy efficiently as they are often battery-powered, i.e., only have a limited capacity of energy until they need to be recharged. Prediction is one central element for realizing energy-efficient software in general [93]. The following are the challenges we have identified with respect to the purposes runtime models may serve.

Models@run.time for assurances Assurance is another purpose besides self-adaptation that has attracted reasonable research interest (cf. Sect. 3.2.3). However, other than at the architecture level and using structural models, there is a lack of approaches with respect to this purpose (cmp. Figs. 7 and 9). Still further research on assurances for models@run.time-based systems is required, as already stated four years ago in [30].

Among others, a compelling application for assurances in the context of models at runtime is autonomous vehicles. Increasingly, cars offer intelligent driver assistance [50,122]. Such driving software is safety critical and, thus, poses the need for methods to assure required safety qualities. For this, such systems require quality models to capture the respective non-functional properties, environmental context models to enable the adaptation to contextual changes and goal models representing the connection between required quality assurances and contextual situations. Currently, such models are typically implicit and coded manually into the running system. In order to provide assurance of properties, these models need to be leveraged explicitly during the full life cycle, including runtime. An interesting first step in this direction has been described by Schneider et al. in [156–159], who introduced the concept of conditional safety certificates, which allow safety checks at system runtime.

Assurance is required for functional (i.e., those which specify the functions of the systems) and non-functional properties of the system (i.e., those which specify how these functions need to work, e.g., efficiency, performance, availability, robustness, and stability). The ability to guarantee these properties at runtime poses challenges due to the variations of the systems, their environment and their inherent uncertainty [70,145]. However, addressing these challenges offers as well new opportunities for runtime verification and validation, enabling assurances for critical system properties at runtime. Further, since runtime models form the foundation of many assurance tasks [50], their quality depends upon the quality of the runtime models. The definition of performance and accuracy for the assessment of runtime models is a crucial research challenge [174]. Efforts in the research area of models@run.time are fundamental to the development of runtime assurance techniques [50]. The central issue in this context is the modelling of uncertainty, as understanding and leveraging uncertainty is central to deliver assurance guarantees. This topic is discussed further in Sect. 6.5 as it can be considered as fundamental and not applied research in models@run.time.

Models@run.time for development Development has, too, attracted some research interest (cf. Sect. 3.2.3). As for assurances, there are no approaches despite than at the architecture level and using structural models (see the Figs. 7, 9).

Software engineers have come to the conclusion that there is no clear separation between development time and runtime [16]. Here, there are opportunities for models@run.time as it can act as a vehicle to understand and address the issues that inevitably arise. However, as in development time while using MDE, different interrelated models need to be used to systematically build up a software system. Different interrelated runtime models as representation of different parts of the executing systems are employed simultaneously, and their relationships need to be maintained at runtime. The result is a complex development process as the models and their relationships need to be managed [182] that in the case of runtime is aggravated by the fact that the runtime models are treated and even conceived during the execution of the system. Just few initiatives exist that address the issues of managing runtime models and their relationships [182]. However, the issues are rather neglected by applying ad hoc solutions if at all. Further, some authors [181] advocate for the unification of development and runtime models to systematically realize their integration and management. It may not be the case that a comprehensive unification is possible or even advised; however, it may be a way to support an incremental strategy of adoption of runtime models from manual maintenance to automated management.

Models@run.time for self-awareness Models@run.time is at the core of self-aware systems [40], and their role in self-awareness is studied in [115,116]. Runtime models provide the vehicle for the self-representation needed by a self-aware system. Runtime models correspond to the models that contain knowledge about the environment and the system itself. They support learning of that knowledge. Runtime models can be traversed and consulted to provide up-to-date information for analysis, prediction and planning needed for self-awareness [115]. Different types of runtime models may be needed to capture different facets of which a system needs to be self-aware. What aspects a system needs to be aware of and what kinds of runtime models are needed is subject to future studies.

6.4 Challenges based on applied model-driven techniques

The major model-driven technique that has been transferred from design to runtime is the model-transformation technique. We indeed found work on runtime model transformations at all identified abstraction levels (cmp. Fig. 8). The main focus again, as for all other model-driven techniques, was on architectural runtime models. However, other techniques are relevant for the models at runtime vision. Below we highlight a set of relevant techniques to be exploited to advance further the state of the art of runtime models and the challenges we have identified in doing so.

Machine learning Machine learning can be used to build context-aware systems, systems-of-systems and self-aware systems. As discussed before, we consider work on models@run.time for systems aware of their context and their interactions with other systems as a promising research challenge. The use of machine learning is a promising direction to address this challenge as for example shown in [75] for the case of self-organizing systems, in [185] for the case of resource management in data centres and in [48] for the case of non-functional properties in the context of cloud-based systems. In general, we only found 15 approaches investigating the use of machine learning in the context of models@run.time. Further, this challenge is intrinsically related to the challenge of providing runtime model inference during execution explained in Sect. 6.5, as it is with the use of techniques such as machine learning techniques that runtime models would be able to be inferred or learned at runtime. The need to apply machine learning to address the challenges of context-aware systems, systems-of-systems and self-aware systems among others has just recently been highlighted again in the context of self-aware computing systems [116]. Accordingly, the rise of machine learning should be tackled by the community of models@run.time.

Towards runtime model comparison A counter-intuitive observation we found was that very little work has been conducted on applying model comparison techniques at runtime (7 of 275 papers, see Sect. 3.2.4). The motivation of the approaches with non-trivial runtime model comparisons we found in our survey is either originating from the problem domain (e.g., error detection [104]) or from novel types of models (e.g., aspect-oriented models [130]). This is in contrast to the finding that most initiatives focus on approaches for self-adaptive software, for which comparisons between the current and desired system state are essential. Presumably, current approaches only use simple, custom-made comparison approaches for this purpose. The application of comparison techniques at runtime poses additional requirements compared to design time. Namely, especially for self-adaptive systems, the runtime model required to perform comparisons is getting more important, e.g., to meet real-time deadlines. We, thus, assume that the application or transfer of existing comparison techniques to runtime is a promising candidate for future research.

Towards runtime models for workflows Another poorly investigated technique is workflow modelling and execution (6 of 275 papers; see Sect. 3.2.4). Although this topic is close to the research area of model execution (which we explicitly excluded from this survey), the need to couple workflows with the system for which they specify the order of actions to take is inevitably required. Surprisingly, most of the papers we found in our survey have been published at least 4 years in the past (e.g., [131] on self-tuning BPEL processes in 2009 or [110] with a vision paper on business processes at run-

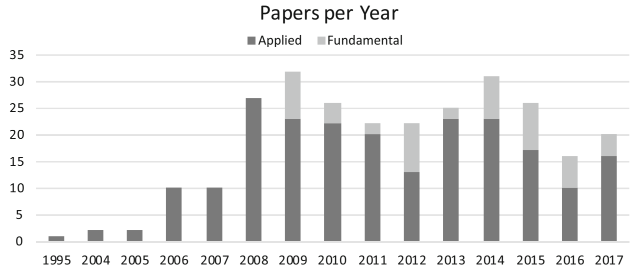


Fig. 12 Research on Models@run.time per year

time in 2013). In consequence, approaches focusing on the causal connection between workflows and the systems they are bound to are a promising field for future research, where initial investigations have been made, but the research efforts almost stagnated.

6.5 Challenges for fundamental research on models@run.time

As already discussed in Sect. 2.4, 20% of the papers we included in this survey address fundamental research on models@run.time. A more detailed look is provided in Fig. 12, which shows the number of applied and fundamental papers published per year as stacked bar chart. Notably, a small but constant research effort focusing on fundamental research for models@run.time can be observed since 3 years after the term models@run.time was coined (i.e., since 2009). This section summarizes the challenges we have concluded for fundamental research on models@run.time.

Towards managed uncertainty at runtime Further techniques to deal with uncertainty and incompleteness of events and information from systems and their environment are required [51]. One aspect of this challenge is monitoring and sensing, which is widely used by modern intelligent and adaptive software. Monitoring can be imprecise and can provide just partial information. Using the correct runtime abstractions to enable the measurement of uncertainty is a core challenge. Runtime models can be used to represent uncertainty, while more evidence is collected by the running system. In [22], the authors use Bayesian inference to model the level of confidence related to the monitoring infrastructure.

Further, in order to design software systems that are able to tackle uncertainty, inferring the knowledge necessary to reason about the system behaviour seems to be an essential task. Such models can be used to dynamically build runtime models during the execution of the system. The acquired knowledge could support solving uncertainty, but, on the other hand, it could incorporate more. Suitable mathematical and formal abstractions should be used to represent and reason upon uncertainty. Probability theory, fuzzy set theory with the use of machine learning should be used to further investigate this issue. Probability theory, based on historical data, can be used to identify which non-functional properties are less likely to be satisfied [27]. Bayesian inference can be used to manipulated values of probabilities or parameters of utilities that change over time and therefore, enable the quantification of the impact of these values during the decision-making [139]. Likewise, fuzzy set theory can be used to produce an initial model of flexible design that can be progressively completed as more information about the environment and the system itself becomes available [31].

Towards runtime model inference during execution This challenge is somehow a consequence of the previous one posed by the uncertainty modern software systems face. In the area of self-adaptive systems, conventional software adaptation techniques and more contemporary models@runtime approaches usually require an a priori model to specify the system's behaviour and structure. In contrast, runtime models can be learned and reified at runtime. However, just few research efforts have been done towards these research lines. For instance, the authors in [71] proposed mining software component interactions based on the execution traces of the underlying running system, in order to build a probabilistic model that will be used to analyze, plan, and execute adaptations. The domain in [71] was self-adaptation. In [28], the runtime models are automatically inferred during execution and refined by exploiting learning techniques and ontologies. The final goal was the dynamic synthesis of code to generate mediators to support the interoperability of systems that were built without previous knowledge of the interaction needed. Several issues exist that are worth further consideration. For example, the replacement or insertions of a component may introduce new unexpected behaviours by changing the functional behaviour of the system, or their non-functional properties (e.g., availability, reliability, etc.). The recent progress in machine learning and Bayesian inference, among other techniques, is key for the extraction of information at runtime to dynamically build the models [23,75,140].

Towards runtime code synthesis Runtime models can support the extension of the success of model-driven engineering to synthesize code at runtime [28,154]. This issue still requires much more research and, obviously depends on the previous issues on runtime model inference and uncertainty. Ontology-based solutions seem to be a promising direction in this respect. Ontologies have been exploited to enrich the runtime models with information that was not know before runtime [23], but is required to dynamically synthesize code.

Towards distributed runtime models Current and previous methods to support the discovery of runtime architecture take centralized approaches, meaning that the process of discovery is carried out from a single location [84,176]. These methods are inadequate for large distributed systems because they either present a single point of failure or do not scale up well [148,177]. A key characteristic of future software systems is that they will operate in collaboration with other systems as covered by the terms systems-of-systems [124], collective adaptive systems [8] and collective self-aware computing systems [64]. In our survey, we only found 8 papers investigating distributed models@run.time, whereof two [32,142] only describe the necessity for such approaches. The most prominent approach in this area originates from the EUREMA project (cf. Sect. 3.6) by Holger Giese and Thomas Vogel, who investigated how the integration and/or synchronization of multiple runtime models of different systems can be systematically described and automated [179,182,183,186]. For this, they transfer the term megamodel [34], i.e., a model comprised of other models, to runtime. In their approach, the processing of models at runtime can be described as a workflow. But, still several research questions for this topic remain open. For example, how to handle partial distributed runtime models [91], i.e., models of different systems representing overlapping knowledge. Thus, further research on distributed runtime models is required.

Towards transaction-safe causal connections The causal connection between the system and the runtime model should support the concept of transactions to, for example, offer roll-back capabilities for consistency. Various research questions in this regard are still unanswered: When are system and model allowed to be out of sync? What happens when a decision is made based on outdated information in the model? What happens when an effect is realized on the system based on an outdated model? What about race conditions when multiple models reflect upon and control the same system? To the best of our knowledge, these questions related to transactional concepts and the frequency of the synchronization have been addressed, until now, only once by the research community [62], even if the topic has been highlighted in the models@run.time workshop's call for papers [95] since 2015.

Towards self-modelling, self-designing systems Increasingly, more and more approaches are proposed for the engineering of systems with *emergent* properties [26]. Such approaches go beyond the state of the art in self-adaptive or -organizing systems, as they aim at self-modelling, self-designing systems [21]. However, we argue that the research community of self-adaptive and self-organizing systems [56] can offer useful techniques as self-modelling and self-designing systems can be casted as self-adapting systems. Thus, making a complex system build itself can be seen as both letting it autonomously change/adapt the organization of its components and, by enabling these latter parts change/adapt their behaviour in an autonomous way. Autonomous, self-adaptive and self-organizing systems, hence, would act as a non-human modellers, treating models according to high-level goals rather than a predefined script [83]. Traditional approaches to self-adaptive and self-organizing software require human experts for the specification of models, policies and processes by which software can adapt according to its goals and environment. For future software systems, these tasks need to be automated, i.e., the systems need to be enabled to perform modelling and design themselves. In other words, future systems required self-modelling capabilities. Some initial approaches have already been proposed [28,71,75], where the authors present approaches, which do not require defining the system's behaviour beforehand, but instead involve techniques to infer the interactions from system execution using, e.g., probabilistic usage models and machine learning. The challenges of self-modelling systems still require much more research efforts. Formal methods and models@run.time-based solutions, with the aid of model/constraint checking, seem to be a promising direction to follow [28]. For example, models@run.time can offer support for reasoning by the system as the runtime model can assemble the learned knowledge based on observation in order to allow the systems to redesign itself to better suit the environment around it.

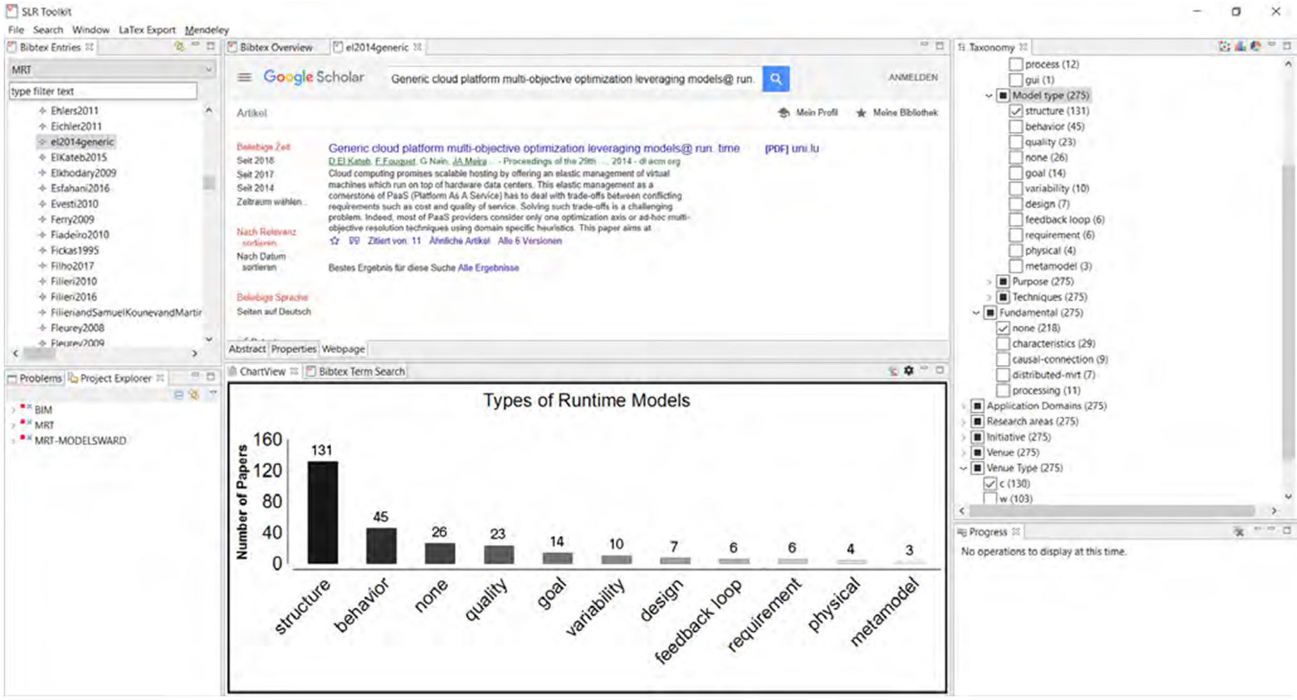
7 Conclusions

The main motivation for this work was to investigate the state of the art in models@run.time, by identifying three main research questions and systematically mapping the literature creating and using a taxonomy to answer those questions. Supported by the taxonomy, we have determined the issues to study, as well as their meaning and relevance to the research topic of models@run.time, to therefore provide aid and guidance to researchers who are planning future research in the area.

The list of all papers included in our survey as well as our classification and taxonomy is available online.[7] The SLR Toolkit, a software tool for systematic literature reviews, was developed by one of the authors of this paper with the support of his students. The SLR Toolkit was developed and used while we performed this survey [94]. Figure 13 depicts a screenshot of the tool, showing the list of papers on the left, a detail view of a selected paper in the centre top, the taxonomy on the right and, at the centre bottom, a bar chart enumerating all papers by the of runtime models used.

Throughout our literature study, we followed three objectives stated Sect. 1, which were formulated as three research questions in Sect. 2.1. In the following, we provide the answers to the three research questions, and which represent the main contribution of this article.

[7] https://github.com/sebastiangoetz/slr-toolkit/tree/master/examples/mrt.



Generic cloud platform multi-objective optimization leveraging models@ run.

Generic cloud platform multi-objective optimization leveraging models@ run. time [PDF] uni.lu
D El Kateb, F Fouquet, G Nain, JA Meira ... - Proceedings of the 29th ..., 2014 - dl.acm.org
Cloud computing promises scalable hosting by offering an elastic management of virtual machines which run on top of hardware data centers. This elastic management as a cornerstone of PaaS (Platform As A Service) has to deal with trade-offs between conflicting requirements such as cost and quality of service. Solving such trade-offs is a challenging problem. Indeed, most of PaaS providers consider only one optimization axis or ad-hoc multi-objective resolution techniques using domain specific heuristics. This paper aims at ...
☆ 99 Zitiert von: 11 Ähnliche Artikel Alle 6 Versionen

Bestes Ergebnis für diese Suche Alle Ergebnisse

☐ process (12)
☐ gui (1)
■ Model type (275)
☑ structure (131)
☐ behavior (45)
☐ quality (23)
☐ none (26)
☐ goal (14)
☐ variability (10)
☐ design (7)
☐ feedback loop (6)
☐ requirement (6)
☐ physical (4)
☐ metamodel (3)
■ Purpose (275)
■ Techniques (275)
■ Fundamental (275)
☑ none (218)
☐ characteristics (29)
☐ causal-connection (9)
☐ distributed-mrt (7)
☐ processing (11)
■ Application Domains (275)
■ Research areas (275)
■ Initiative (275)
■ Venue (275)
■ Venue Type (275)
☑ c (130)
☐ w (103)

No operations to display at this time.

Fig. 13 Screenshot of SLR Toolkit

RQ1: How can existing research on models@run.time be classified? In Sect. 3, we presented a novel taxonomy to classify research on models@run.time. We classified 275 papers on models@run.time using this taxonomy, which evaluates its suitability. Our taxonomy, concisely depicted in Fig. 5, is comprised of four main dimensions: the type of research, the domain of application (e.g., healthcare), the intersecting research areas and related research initiatives. The type of research was further refined into applied and fundamental research, where applied research denotes approaches that use models@run.time to address particular research questions and fundamental research denotes approaches that answer research questions about the models@run.time paradigm itself (e.g., how to realize a transaction-safe causal connection). Finally, the dimension of applied research is further refined into the modelled artefact (e.g., the architecture or context), the type of runtime model used (e.g., behavioural or structural models), the purpose for which the runtime model was used (e.g., to enable the interoperation of systems) and the model-driven techniques, which have been applied in the approach (e.g., model comparison).

RQ2: What is the state of the art of models@run.time research with respect to the classification? To answer this research question, we first analyzed the frequency of the values we found for each dimension of the taxonomy (see Sect. 3.2 until 3.6). Later, as shown in Sect. 5, we performed a cross-dimensional analysis using bubble matrix charts for the classification covering 275 papers. Our main findings are that most research on models@run.time has a strong focus on particular topics with respect to our taxonomy. Namely, by far the most applied research on models@run.time has (a) focused on the architecture, (b) used structural runtime models, (c) used the runtime models for self-adaptive software and (d) used model transformations. Also, an analysis of the application domains (25) and intersecting research areas (25) shows that models@run.time is a highly interdisciplinary research topic. Finally, a conclusion to be drawn from analyzing related research initiatives is that even if there is a plethora of initiatives that use models@run.time (45), virtually no initiatives exist and that focus on fundamental research topics.

RQ3: What can be inferred from the results associated with RQ2 that will lead to timely, relevant research directions for further investigation? In Sect. 6, we derived and motivated a set of research challenges, which have been rarely investigated according to our study. We categorized our identified challenges using our taxonomy into fundamental research challenges and challenges with respect to the four dimensions of applied research on models@run.time. In total, we described 18 challenges as shown in Fig. 14. The challenges have been strategically connected to related discussions, references and concepts.

In conclusion, the work on models@run.time over the last 13 years has been very proliferous and has been applied onto different domain areas. However, we have seen that it has also been largely focused on the use of structural models at the

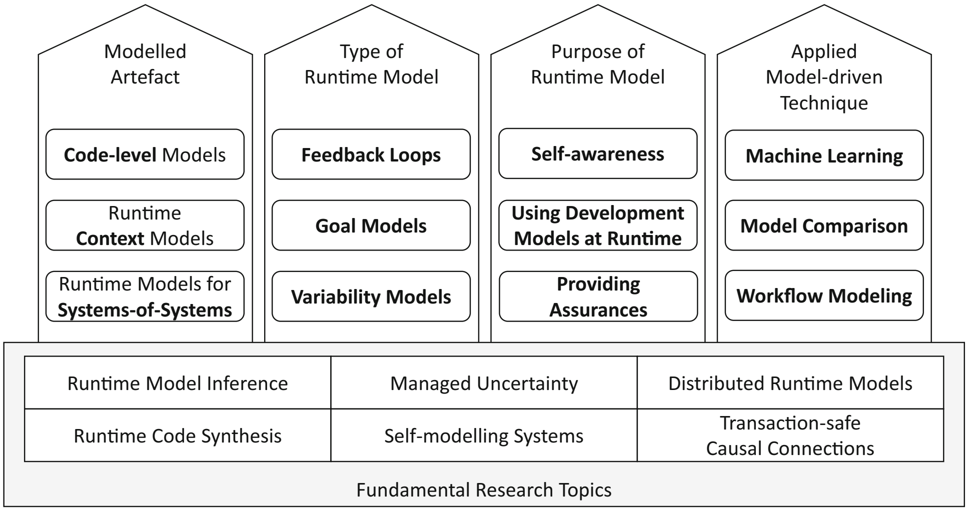


Fig. 14 Challenges for research on Models@run.time

architectural abstraction level to build self-adaptive systems using model-transformations. The researchers of the area of models@run.time still have many research opportunities to develop future software that inevitably will increasingly need to work under uncertainty, will be distributed, will need to take advantage of new techniques (such as those based on machine learning or nature-inspired algorithms) and exploit the power provided by new technologies such as IoT and Cloud. Models at runtime will certainly serve as a vehicle to underpin the building of such systems.

Acknowledgements This work has been partially funded by the German Research Foundation (DFG) under Project Agreement SFB912/2 and GRK1907 and the Systems Analytics Research Institute (SARI) in Aston University.

Appendices

A List of application domains

The following list summarizes all domains to which models@run.time has been applied so far according to our body of literature.

- **Enterprise Software (23)**, e.g., enterprise resource planning (ERP) or customer relationship management (CRM) software (e.g., [130]).
- **Cloud-based (17)** systems, especially Software as a Service (SaaS) (e.g., [48]).
- **Energy-efficient Software (11)** of software systems like, e.g., optimization approaches trading performance and energy consumption (e.g., [99]).
- **Home Automation Systems (10)**, e.g., approaches for the Smart Home (e.g., [54]).
- **Communication Technology (8)**, i.e., telecommunication networks (e.g., [132]).
- **Cyber-Physical Systems (8)**, i.e., networked embedded systems (e.g., [101]).
- **Monitoring Systems (7)**, i.e., approaches to intelligently observe the state of a running physical or virtual system (e.g., [28]).
- **eCommerce Systems (7)**, e.g., sales platforms and webshops (e.g., [87]).
- **Embedded Systems (6)**, i.e., single devices, which are embedded into a physical environment and react to changes in it (e.g., [171]).
- **Healthcare (6)**, e.g., approaches to monitor patient data (e.g., [2]).
- **Robotics (6)**, e.g., approaches to reason about the collaboration of multiple robots (e.g., [85]).
- **Traffic Advising (5)**, i.e., routing/navigation software (e.g., [12]).
- **Ambient Assisted Living (AAL) (5)**, i.e., systems designed with the aim to help elderly people or people with special needs in their everyday life (e.g., [159]).
- **Games (4)**, e.g., approaches to improve the reasoning about strategies of non-player characters (e.g., [61]).
- **Crisis Management (4)**, e.g., flood warning systems (e.g., [15]).
- **Travel Advising (4)**, i.e., software suggesting holiday packages, including flights, hotel, rental car and activities (e.g., [162]).
- **IT Management Systems (4)**, i.e., systems used to manage all electronic devices in a building (e.g., [118]).
- **Internet of Things (3)**, i.e., approaches to capture the network of connected devices, typically with the aim to integrate previously unknown system with each other (e.g., [49]).

- **Database Management Systems (3)**, i.e., approaches to reason about how (data format) and where to store data (e.g., [65]).
- **Mobile Software (2)**, i.e., software applications running on mobile devices, which need to react to changes in their environment (e.g., [82]).
- **Office Management Systems (1)**, i.e., systems used to manage all software applications of a company (e.g., [42]).
- **eGovernment (1)**, i.e., software systems enabling citizens to interact with governmental administration over the Internet [106].
- **Java Virtual Machine (1)**, i.e., approaches to improve garbage collection [108].
- **Scientific Computing (1)**, e.g., simulations of climate models [9].
- **Social Networks (1)**, i.e., approaches to analyze trends and to identify hot topics based on what people share in social networks [175].
- **None (127)**, i.e., no case study has been conducted.

B List of supporting research initiatives

In the following, we list all research projects we found, grouped by their origin of funding. For each funding organization, we provide the number of identified research projects in braces.

- **European Union (19)**
  - **NeCS** European Network for Cyber-security. EU H2020 (EU.1.3.1).
  - **ALIVE** Coordination, Organisation and Model Driven Approaches for Dynamic,Flexible, Robust Software and Services Engineering. EU FP7-ICT.
  - **CHOReOS** Large Scale Choreographies for the Future Internet. EU FP7-ICT.
  - **CONNECT** Emergent Connectors for Eternal Software Intensive Systems. EU FP7-ICT.
  - **DiVA** Dynamic Variability in Complex, Adaptive Systems. EU FP7-ICT.
  - **DIVERSIFY** Ecology-inspired software diversity for distributed adaptation in CAS. EU FP7-ICT.
  - **EINS** Network of Excellence in Internet Science. EU FP7-ICT.
  - **MASSIF** MAnagement of Security information and events in Service InFrastructures. EU FP7-ICT.
  - **MODAClouds** MOdel-Driven Approach for design and execution of applications on multiple Clouds. EU FP7-ICT.
  - **Lucretius** Foundations for Software Evolution. ERC Advanced Investigator Grant.
  - **PaaSage** Model-Based Cloud Platform Upperware. EU FP7-ICT.
  - **PERSIST** PERsonal Self-Improving SmarT spaces. EU FP7-ICT.
  - **RECOGNITION** Relevance and cognition for self-awareness in a content-centric Internet. EU FP7-ICT.
  - **REMICS** REuse and Migration of legacy applications to Interoperable Cloud Services. EU FP7-ICT.
  - **S-Cube** Software Services and Systems Network. EU FP7-ICT.
  - **SeSaMo** Security and Safety Modelling. EU FP7-JTI.
  - **SMSCom** Self-Managing Situated Computing. EU FP7-IDEAS-ERC.
  - **MODELPLEX** Modelling solution for complex software systems. EU FP6-IST.
  - **MUSIC** Self-adapting applications for mobile users in ubiquitous Computing Environments. EU FP6-IST.
- **German Research Foundation (DFG) (4)**
  - **CRC 912—HAEC** Highly Adaptive Energy Efficient Computing. DFG collaborative research centre (CRC).
  - **RTG 1907—RoSI** Role-based Software Infrastructures for continuous-context-sensitive Systems. DFG research training group (RTG).
  - **SPP 1593** Design For Future-Managed Software Evolution. DFG priority programme (SPP).
  - **RAMSES** Reflective and Adaptive Middleware for Software Evolution of Non-stopping Information Systems.
- **German Federal Ministry of Education and Research (BMBF) (4)**
  - **CoolSoftware** BMBF cluster of excellence.
  - **SysPlace** EcoSystem of Displays.
  - **OptimAAL** Kompetenzplattform für die Einführung und Entwicklung von AAL-Lösungen.
  - **SPES2020** Software Plattform Embedded Systems.
- **France National Research Agency (ANR) (2)**
  - **FAROS** Composition Environment for Building Reliable Service-oriented Architectures.
  - **SALTY** Self-Adaptive very Large disTributed sYstems.
- **French Institute for Research in Computer Science and Automation (Inria) (1)**
  - **Project M@TURE** Models @ run Time for self-adaptive pervasive systems: enabling User-in-the-loop, REquirement-awareness, and interoperability

  in ad hoc settings. Inria/Brazil International Scientific Cooperation Program (year 2014).
  - **Project M@TURE 2** Inria/Brazil International Scientific Cooperation Program (year 2015).

- **Netherlands Organisation for Applied Scientific Research (TNO) funded projects (2)**
  - **AMSN** Adaptive Multi-Sensor Networks research program.
  - **Trader** Reliability by design.

- **iMinds Funded Projects (2)**
  - **D-BASE** Decentralized support for Business Processes in Application Services.
  - **DMS2** Decentralized Data Management and Migration of SaaS.

- **UK Engineering and Physical Sciences Research Council (EPSRC) Funded Projects (2)**
  - **DAASE** Dynamic Adaptive Automated Software Engineering.
  - **LSC-ITS** Large Scale Complex IT System.

- **Projects Funded by other Grants (9)**
  - **ARM** Adaptive Resource Management Project. Funded by University of Milano-Bicocca.
  - **CAPUCINE** Context-aware Service-oriented Product Lines. Funded by Fonds Unique Interministeriel (France).
  - **CARAMELOS** Collaborative Action Research on Agile Methodologies for Enterprises in the Little, adhering to the Open Source principle. Funded by the Vlaamse Interuniversitaire Raad (Belgium).
  - **GenData 2020** Data-Driven Genomic Computing. Funded by the Ministry of Education, University and Research (Italy).
  - **GIOCOSO** GIOchi pediatrici per la COmunicazione e la SOcializzazione (Regione Lombardia).
  - **MAIS** Multichannel Adaptive Information Systems. Funded by Politecnico di Milano (Italy).
  - **MEDICAL** Embedded middleware for sensor and application integration for in-home services. Finded by Minalogic.
  - **MORISIA** Models at Runtime for Self-Adaptive Software. Funded by HPI.
  - **Value@Cloud** Model-Driven Incremental Development of Cloud Services Oriented to the Customers' Value. Funded by CICYT.

References

1. Abeywickrama, D.B., Serbedzija, N., Loreti, M.: Monitoring and visualizing adaptation of autonomic systems at runtime. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, pp. 1857–1860. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2695664.2695983
2. Albassam, E., Porter, J., Gomaa, H., Menasci, D.A.: Dare: a distributed adaptation and failure recovery framework for software systems. In: 2017 IEEE International Conference on Autonomic Computing (ICAC), pp. 203–208 (2017). https://doi.org/10.1109/ICAC.2017.12
3. Alfarez, G., Pelechano, V., Mazo, R., Salinesi, C., Diaz, D.: Dynamic adaptation of service compositions with variability models. J. Syst. Softw. **91**, 24–47 (2014). https://doi.org/10.1016/j.jss.2013.06.034
4. Almorsy, M., Grundy, J., Ibrahim, A.S.: Adaptable, model-driven security engineering for SaaS cloud-based applications. Autom. Softw. Eng. **21**(2), 187–224 (2014)
5. Al-Refai, M., Cazzola, W., France, R.: Using models to dynamically refactor runtime code. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC '14, pp. 1108–1113. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2554850.2554954
6. Amoui, M., Derakhshanmanesh, M., Ebert, J., Tahvildari, L.: Achieving dynamic adaptation via management and interpretation of runtime models. J. Syst. Softw. **85**(12), 2720–2737 (2012). https://doi.org/10.1016/j.jss.2012.05.033
7. Anaya, I.D.P., Simko, V., Bourcier, J., Plouzeau, N., Jézéquel, J.M.: A prediction-driven adaptation approach for self-adaptive sensor networks. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, pp. 145–154. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2593929.2593941
8. Anderson, S., Bredeche, N., Eiben, A., Kampis, G., van Steen, M.: Adaptive Collective Systems: Herding Black Sheep. Bookprints, Minneapolis (2013)
9. Andersson, J., Ericsson, M., Löwe, W.: Automatic rule derivation for adaptive architectures. In: 7th Working IEEE/IFIP Conference on Software Architecture, pp. 323–326. IEEE (2008)
10. Andersson, J., Lemos, R., Malek, S., Weyns, D. (2009) Modeling dimensions of self-adaptive software systems. In: Cheng B.H., Lemos R., Giese H., Inverardi P., Magee J. (eds.) Software Engineering for Self-Adaptive Systems, Chap. Modeling Dimensions of Self-Adaptive Software Systems, pp. 27–47. Springer, Berlin. https://doi.org/10.1007/978-3-642-02161-9_2
11. Anthony, R., Pelc, M., Ward, P., Hawthorne, J., Pulnah, K.: A run-time configurable software architecture for self-managing systems. In: International Conference on Autonomic Computing, 2008. ICAC '08, pp. 207–208 (2008). https://doi.org/10.1109/ICAC.2008.23
12. Arcaini, P., Riccobene, E., Scandurra, P.: Modeling and analyzing MAPE-K feedback loops for self-adaptation. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '15, pp. 13–23. IEEE Press, Piscataway, NJ, USA (2015). http://dl.acm.org/citation.cfm?id=2821357.2821362
13. Arcega, L., Font, J., Haugen, Ø., Cetina, C.: An infrastructure for generating run-time model traces for maintenance tasks. In: Proceedings of the 11th International Workshop on Models@run.time co-located with 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), Saint Malo, France, 4 October 2016, pp. 35–42 (2016). http://ceur-ws.org/Vol-1742/MRT16_paper_7.pdf

14. Arias, T.B.C., America, P., Avgeriou, P.: Defining execution viewpoints for a large and complex software-intensive system. In: Joint Working IEEE/IFIP Conference on Software Architecture, 2009 and European Conference on Software Architecture. WICSA/ECSA 2009, pp. 1–10. IEEE (2009). **(They never use the term "models@runtime", nor cite our paper, but it is essentially the same idea)**
15. Barbier, F., Cariou, E., Le Goaer, O., Pierre, S.: Software adaptation: classification and a case study with state chart xml. IEEE Softw. **32**(5), 68–76 (2015)
16. Baresi, L., Ghezzi, C.: The disappearing boundary between development-time and run-time. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, FoSER '10, pp. 17–22. ACM, New York, NY, USA (2010). https://doi.org/10.1145/1882362.1882367
17. Baresi, L., Pasquale, L., Spoletini, P.: Fuzzy goals for requirements-driven adaptation. In: RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, 27 September–1 October 2010, pp. 125–134 (2010). http://dx.doi.org/10.1109/RE.2010.25
18. Baresi, L., Pasquale, L.: Live goals for adaptive service compositions. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '10, pp. 114–123. ACM, New York, NY, USA (2010). https://doi.org/10.1145/1808984.1808997
19. Baresi, L.: Self-adaptive systems, services, and product lines. In: Proceedings of the 18th International Software Product Line Conference—Volume 1, SPLC '14, pp. 2–4. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2648511.2648512
20. Baxter, I.: Keynote: supporting forward and reverse engineering with multiple types of models. In: Proceedings of the 20th International Conference on Model-driven Engineering, Systems and Languages. IEEE (2017)
21. Bellman, K.L., Landauer, C., Nelson, P., Bencomo, N., Götz, S., Lewis, P., Esterle, L.: Self-Modeling and Self-Awareness, pp. 279–304. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-47474-8_9
22. Bencomo, N., Belaggoun, A., Issarny, V.: Dynamic decision networks for decision-making in self-adaptive systems: a case study. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, pp. 113–122. IEEE Press, Piscataway, NJ, USA (2013). http://dl.acm.org/citation.cfm?id=2487336.2487355
23. Bencomo, N., Belaggoun, A., Issarny, V.: Dynamic decision networks for decision-making in self-adaptive systems: a case study. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2013, San Francisco, CA, USA, 20–21 May 2013, pp. 113–122 (2013). https://doi.org/10.1109/SEAMS.2013.6595498
24. Bencomo, N., Grace, P., Flores-Cortés, C.A., Hughes, D., Blair, G.S.: Genie: supporting the model driven development of reflective, component-based adaptive systems. In: 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, 10–18 May 2008, pp. 811–814 (2008). https://doi.org/10.1145/1368088.1368207
25. Bencomo, N., Whittle, J., Sawyer, P., Finkelstein, A., Letier, E.: Requirements reflection: requirements as runtime entities. In: 2010 ACM/IEEE 32nd International Conference on Software Engineering, vol. 2, pp. 199–202 (2010). https://doi.org/10.1145/1810295.1810329
26. Bencomo, N.: The role of models@run.time in autonomic systems: keynote. In: 2017 IEEE International Conference on Autonomic Computing, ICAC 2017, Columbus, OH, USA, 17–21 July 2017, pp. 293–294 (2017). https://doi.org/10.1109/ICAC.2017.55
27. Bencomo, N., Belaggoun, A.: Supporting Decision-Making for Self-Adaptive Systems: From Goal Models to Dynamic Decision Networks, pp. 221–236. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-37422-7_16
28. Bencomo, N., Bennaceur, A., Grace, P., Blair, G., Issarny, V.: The role of models@run.time in supporting on-the-fly interoperability. Computing **95**(3), 167–190 (2012)
29. Bencomo, N., Hallsteinsen, S., De Almeida, E.S.: A view of the dynamic software product line landscape. Computer **45**(10), 36–41 (2012). https://doi.org/10.1109/MC.2012.292
30. Bencomo, N., France, R., Cheng, B.H.C., Aßmann, U.: Models@run.time. Foundations, Applications, and Roadmaps, vol. 8378. Springer, Cham (2014)
31. Bencomo, N., Torres, R., Salas, R., Astudillo, H.: An architecture based on computing with words to support runtime reconfiguration decisions of service-based systems. Int. J. Comput. Intell. Syst. **11**(1), 272–281 (2018). (Copyright 2018, the Authors. Published by Atlantis Press. This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/). Funding: UNAB Grant DI-1303-16/RG, grant FONDEF IDeA ID16I10322, FONDECYT Grant 1140408)
32. Bennaceur, A., France, R.B., Tamburrelli, G., Vogel, T., Mosterman, P.J., Cazzola, W., Costa, F.M., Pierantonio, A., Tichy, M., Aksit, M., Emmanuelson, P., Huang, G., Georgantas, N., Redlich, D.: Mechanisms for leveraging models at runtime in self-adaptive software. In: Models@run.time—Foundations, Applications, and Roadmaps (Dagstuhl Seminar 11481, 27 November–2 December 2011), pp. 19–46 (2014). https://doi.org/10.1007/978-3-319-08915-7_2
33. Bennaceur, A., Issarny, V.: Automated synthesis of mediators to support component interoperability. IEEE Trans. Softw. Eng. **41**, 221–240 (2015)
34. Bézivin, J., Jouault, F., Valduriez, P.: On the need for megamodels. In: Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development Workshop, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, Vancouver, Canada (2004). https://hal.archives-ouvertes.fr/hal-01222947
35. Blair, G., Bencomo, N., France, R.: Models@run.time. Computer **42**(10), 22–27 (2009). https://doi.org/10.1109/MC.2009.326
36. Bosch, J.: Delivering customer value in the age of autonomous, continuously evolving systems. In: 2016 IEEE 24th International Requirements Engineering Conference (RE), pp. 1–1 (2016). https://doi.org/10.1109/RE.2016.16
37. Calinescu, R., France, R., Ghezzi, C.: Models@run.time. Computer **95**(3), 165–166 (2013)
38. Calinescu, R., France, R.B., Ghezzi, C.: Editorial. Computing **95**(3), 165–166 (2013). https://doi.org/10.1007/s00607-012-0238-4
39. Cámara, J., Correia, P., De Lemos, R., Garlan, D., Gomes, P., Schmerl, B., Ventura, R.: Evolving an adaptive industrial software system to use architecture-based self-adaptation. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, pp. 13–22. IEEE Press, Piscataway, NJ, USA (2013)
40. Cámara, J., Bellman, K.L., Kephart, J.O., Autili, M., Bencomo, N., Diaconescu, A., Giese, H., Götz, S., Inverardi, P., Kounev, S., Tivoli, M.: Self-Aware Computing Systems: Related Concepts and Research Areas, pp. 17–49. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-47474-8_2
41. Capilla, R., Bosch, J.: The promise and challenge of runtime variability. Computer **44**(12), 93–95 (2011). https://doi.org/10.1109/MC.2011.382
42. Castañeda, L., Villegas, N.M., Müller, H.A.: Self-adaptive applications: on the development of personalized web-tasking systems. In: Proceedings of the 9th International Symposium on Software

Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, pp. 49–54. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2593929.2593942

43. Cazzola, W., Rossini, N.A., Bennett, P., Mandalaparty, S.P., France, R.B.: Fine-grained semi-automated runtime evolution. In: Models@run.time—Foundations, Applications, and Roadmaps (Dagstuhl Seminar 11481, 27 November–2 December 2011), pp. 237–258 (2014). https://doi.org/10.1007/978-3-319-08915-7_9
44. Cazzola, W., Rossini, N.A., Al-Refai, M., France, R.B.: Fine-Grained Software Evolution Using UML Activity and Class Models, pp. 271–286. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-41533-3_17
45. Cetina, C., Giner, P., Fons, J., Pelechano, V.: A model-driven approach for developing self-adaptive pervasive systems. In: Proceedings of the 3rd International Models@ Runtime Workshop, pp. 97–106 (2008)
46. Cetina, C., Giner, P., Fons, J., Pelechano, V.: Autonomic computing through reuse of variability models at runtime: the case of smart homes. Computer **42**(10), 37–43 (2009)
47. Chen, B., Peng, X., Yu, Y., Nuseibeh, B., Zhao, W.: Self-adaptation through incremental generative model transformations at runtime. In: 36th International Conference on Software Engineering, ICSE '14, Hyderabad, India—31 May–07 June 2014, pp. 676–687 (2014). https://doi.org/10.1145/2568225.2568310
48. Chen, T., Bahsoon, R.: Self-adaptive and online qos modeling for cloud-based software services. IEEE Trans. Softw. Eng. **43**(5), 453–475 (2017). https://doi.org/10.1109/TSE.2016.2608826
49. Chen, X., Li, A., Zeng, X., Guo, W., Huang, G.: Runtime model based approach to iot application development. Front. Comput. Sci. **9**(4), 540–553 (2015)
50. Cheng, B.H.C., Eder, K.I., Gogolla, M., Grunske, L., Litoiu, M., Müller, H.A., Pelliccione, P., Perini, A., Qureshi, N.A., Rumpe, B., Schneider, D., Trollmann, F., Villegas, N.M.: Using models at runtime to address assurance for self-adaptive systems. In: Models@run.time—Foundations, Applications, and Roadmaps (Dagstuhl Seminar 11481, 27 November–2 December 2011), pp. 101–136 (2011). https://doi.org/10.1007/978-3-319-08915-7_4
51. Cheng, B.H.C., Eder, K.I., Gogolla, M., Grunske, L., Litoiu, M., Müller, H.A., Pelliccione, P., Perini, A., Qureshi, N.A., Rumpe, B., Schneider, D., Trollmann, F., Villegas, N.M.: Using models at runtime to address assurance for self-adaptive systems. In: Models@run.time—Foundations, Applications, and Roadmaps (Dagstuhl Seminar 11481, 27 November–2 December 2011), pp. 101–136 (2014). https://doi.org/10.1007/978-3-319-08915-7_4
52. Combemale, B., Broto, L., Crégut, X., Daydé, M., Hagimont, D.: Autonomic management policy specification: from uml to dsml. In: Model Driven Engineering Languages and Systems, pp. 584–599. Springer (2008)
53. Criado, J., Vicente-Chicote, C., Padilla, N., Iribarne, L.: A model-driven approach to graphical user interface runtime adaptation. In: Proceedings of the 5th Workshop on Models@run.time, pp. 49–59 (2010)
54. Dávid, I., Ráth, I., Varró, D.: Foundations for streaming model transformations by complex event processing. Softw. Syst. Model. (2016). https://doi.org/10.1007/s10270-016-0533-1
55. de Grandis, P., Valetto, G.: Elicitation and utilization of application-level utility functions. In: Proceedings of the 6th International Conference on Autonomic Computing, pp. 107–116. ACM (2009). https://doi.org/10.1145/1555228.1555259
56. de Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., Weyns, D., Baresi, L., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Desmarais, R., Dustdar, S., Engels, G., Geihs, K., Göschka, K.M., Gorla, A., Grassi, V., Inverardi, P., Karsai, G., Kramer, J., Lopes, A., Magee, J., Malek, S., Mankovskii, S., Mirandola, R., Mylopoulos, J., Nierstrasz, O., Pezzè, M., Prehofer, C., Schäfer, W., Schlichting, R., Smith, D.B., Sousa, J.P., Tahvildari, L., Wong, K., Wuttke, J.: Software Engineering for Self-Adaptive Systems: A Second Research Roadmap, pp. 1–32. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-35813-5_1
57. De Oliveira Filho, J., Papp, Z., Djapic, R., Oosteveen, J.: Model-based design of self-adapting networked signal processing systems. In: IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2013, pp. 41–50 (2013). https://doi.org/10.1109/SASO.2013.16
58. Debbabi, B., Diaconescu, A., Lalanda, P.: Controlling self-organising software applications with archetypes. In: IEEE 6th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2012, pp. 69–78 (2012). https://doi.org/10.1109/SASO.2012.21
59. DeLoach, S.A., Ou, X., Zhuang, R., Zhang, S.: Model-driven, moving-target defense for enterprise network security. In: Models@run.time—Foundations, Applications, and Roadmaps (Dagstuhl Seminar 11481, 27 November–2 December 2011), pp. 137–161 (2014). https://doi.org/10.1007/978-3-319-08915-7_5
60. Denker, M., Ressia, J., Greevy, O., Nierstrasz, O.: Modeling features at runtime. In: Model-Driven Engineering Languages and Systems, pp. 138–152. Springer (2010)
61. Derakhshanmanesh, M., Amoui, M., O'Grady, G., Ebert, J., Tahvildari, L.: Graf: graph-based runtime adaptation framework. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '11, pp. 128–137. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1988008.1988026
62. Derakhshanmanesh, M., Grieger, M., Ebert, J.: On the need for extended transactional models@run.time. In: Götz, S., Bencomo, N., Blair, G., Song, H. (eds.) Proceedings of the 10th International Workshop on Models@run.time, pp. 21–30. CEUR-WS.org (2015)
63. Devries, B., Cheng, B.: Using models at run time to detect incomplete and inconsistent requirements. In: Proceedings of the 12th International Workshop on Models@run.time Co-located with 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), 19 September 2017, Austin, TX, USA (2017)
64. Diaconescu, A., Bellman, K.L., Esterle, L., Giese, H., Götz, S., Lewis, P., Zisman, A.: Architectures for Collective Self-Aware Computing Systems, pp. 191–235. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-47474-8_7
65. Didona, D., Romano, P., Peluso, S., Quaglia, F.: Transactional auto scaler: elastic scaling of in-memory transactional data grids. In: Proceedings of the 9th International Conference on Autonomic Computing, pp. 125–134. ACM (2012). https://doi.org/10.1145/2371536.2371559
66. Ding, Y., Namatame, N., Riedel, T., Miyaki, T., Budde, M.: Smartteco: context-based ambient sensing and monitoring for optimizing energy consumption. In: Proceedings of the 8th ACM International Conference on Autonomic Computing, pp. 169–170. ACM (2011). https://doi.org/10.1145/1998582.1998612
67. Ebraert, P., Tourwe, T.: A reflective approach to dynamic software evolution. In: Cazzola, W., Chiba, S., Saake, G. (eds.) Research Report C-196, pp. 37–43. Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo (2004)
68. El Kateb, D., Zannone, N., Moawad, A., Caire, P., Nain, G., Mouelhi, T., Le Traon, Y.: Conviviality-driven access control policy. Requir. Eng. **20**(4), 363–382 (2015). https://doi.org/10.1007/s00766-014-0204-0
69. Elkhodr, M., Shahrestani, S.A., Cheung, H.: The Internet of Things: new interoperability, management and security challenges. CoRR arXiv:1604.04824 (2016)

70. Esfahani, N., Malek, S.: Uncertainty in Self-Adaptive Software Systems, pp. 214–238. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-35813-5_9
71. Esfahani, N., Yuan, E., Canavera, K.R., Malek, S.: Inferring software component interaction dependencies for adaptation support. ACM Trans. Auton. Adapt. Syst. **10**, 26:1–26:32 (2016)
72. Evesti, A., Ovaska, E.: Ontology-based security adaptation at runtime. In: 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2010, pp. 204–212 (2010). https://doi.org/10.1109/SASO.2010.11
73. Ferry, N., Hourdin, V., Lavirotte, S., Rey, G., Tigli, J.Y., Riveill, M.: Models at runtime: service for device composition and adaptation. In: Proceedings of the 4th Workshop on Models@run.time, pp. 51–60 (2009)
74. Fiadeiro, J.L., Lopes, A.: A model for dynamic reconfiguration in service-oriented architectures. In: Proceedings of 4th European Conference on Software Architecture, ECSA 2010, Copenhagen, Denmark, 23–26 August 2010, pp. 70–85 (2010). https://doi.org/10.1007/978-3-642-15114-9_8
75. Filho, R.R., Porter, B.: Defining emergent software using continuous self-assembly, perception, and learning. ACM Trans. Auton. Adapt. Syst. **12**(3), 16:1–16:25 (2017). https://doi.org/10.1145/3092691
76. Filieri, A., Ghezzi, C., Grassi, V., Mirandola, R.: Reliability analysis of component-based systems with multiple failure modes. In: Proceedings of 13th International Symposium on Component-Based Software Engineering, CBSE 2010, Prague, Czech Republic, 23–25 June 2010, pp. 1–20 (2010). https://doi.org/10.1007/978-3-642-13238-4_1
77. Filieri, A., Tamburrelli, G., Ghezzi, C.: Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. IEEE Trans. Softw. Eng. **42**(1), 75–99 (2016). https://doi.org/10.1109/TSE.2015.2421318
78. Fleurey, F., Dehlen, V., Bencomo, N., Morin, B., Jezequel, J.M.: Modeling and validating dynamic adaptation. In: Proceedings of the 3rd International Models@ Runtime Workshop, pp. 36–46 (2008)
79. Fouquet, F., Morin, B., Fleurey, F., Barais, O., Plouzeau, N., Jézéquel, J.: A dynamic component model for cyber physical systems. In: Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering, CBSE 2012, Part of Comparch '12 Federated Events on Component-Based Software Engineering and Software Architecture, Bertinoro, Italy, 25–28 June 2012, pp. 135–144 (2012). https://doi.org/10.1145/2304736.2304759
80. Fouquet, F., Nain, G., Morin, B., Daubert, E., Barais, O., Plouzeau, N., Jézéquel, J.M.: An eclipse modelling framework alternative to meet the models@ runtime requirements. In: Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems, pp. 87–101. Springer (2012)
81. France, R., Rumpe, B.: Model-driven development of complex software: a research roadmap. In: Briand, L., Wolf, A. (eds.) Future of Software Engineering. IEEE-CS Press, Piscataway (2007)
82. Gamez, N., Fuentes, L., Troya, J.: Creating self-adapting mobile systems with dynamic software product lines. IEEE Softw. **32**(2), 105–112 (2015)
83. Garcia, A., Bencomo, N.: Non-human modelers: Can they work? In: Proceedings of Workshops, STAF 2017, Software Technologies: Applications and Foundations (2017)
84. Garlan, D., Schmerl, B.: Using Architectural Models at Runtime: Research Challenges. Springer, Berlin (2004)
85. Georgas, J.C., van der Hoek, A., Taylor, R.N.: Using architectural models to manage and visualize runtime adaptation. Computer **42**(10), 0052–60 (2009)
86. Gerbert-Gaillard, E., Lalanda, P.: Self-aware model-driven pervasive systems. In: 2016 IEEE International Conference on Autonomic Computing (ICAC), pp. 221–222 (2016). https://doi.org/10.1109/ICAC.2016.26
87. Ghahremani, S., Giese, H., Vogel, T.: Efficient utility-driven self-healing employing adaptation rules for large dynamic architectures. In: 2017 IEEE International Conference on Autonomic Computing (ICAC), pp. 59–68 (2017). https://doi.org/10.1109/ICAC.2017.35
88. Ghezzi, C., Mocci, A., Sangiorgio, M.: Runtime monitoring of component changes with spy@runtime. In: Proceedings of the 34th International Conference on Software Engineering, ICSE '12, pp. 1403–1406. IEEE Press, Piscataway, NJ, USA (2012). http://dl.acm.org/citation.cfm?id=2337223.2337430
89. Gjerlufsen, T., Ingstrup, M., Olsen, J.W.: Mirrors of meaning: supporting inspectable runtime models. Computer **42**(10), 61–68 (2009). (This paper is focused on the reflection of programs' runtime status)
90. Gonzalez-Herrera, I., Bourcier, J., Daubert, E., Rudametkin, W., Barais, O., Fouquet, F., Jézéquel, J.M.: Scapegoat: an adaptive monitoring framework for component-based systems. In: IEEE/IFIP Conference on Software Architecture (WICSA), 2014, pp. 67–76. IEEE (2014)
91. Götz, S., Gerostathopoulos, I., Krikava, F., Shahzada, A., Spalazzese, R.: Adaptive exchange of distributed partial models@run.time for highly dynamic systems. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). IEEE (2015)
92. Götz, S., Kühn, T.: Models@run.time for object-relational mapping supporting schema evolution. In: Götz, S., Bencomo, N., Blair, G., Song, H. (eds.) Proceedings of the 10th International Workshop on Models@run.time, pp. 41–50. CEUR-WS.org (2015)
93. Götz, S., Schöne, R., Wilke, C., Mendez, J., Assmann, U.: Towards predictive self-optimization by situation recognition. In: Proceedings of 2nd Workshop "Energy Aware Software—Engineering and Development" (EASED) (2013)
94. Götz, S.: Supporting systematic literature reviews in computer science: the systematic literature review toolkit. In: MoDELS Companion, pp. 22–26. ACM (2018)
95. Götz, S., Bencomo, N., France, R.B.: Devising the future of the models@run.time workshop. ACM SIGSOFT Softw. Eng. Notes **40**(1), 26–29 (2015). https://doi.org/10.1145/2693208.2693249
96. Grohmann, J., Herbst, N., Spinner, S., Kounev, S.: Self-tuning resource demand estimation. In: 2017 IEEE International Conference on Autonomic Computing (ICAC), pp. 21–26 (2017). https://doi.org/10.1109/ICAC.2017.19
97. Guo, T., Shenoy, P.: Model-driven geo-elasticity in database clouds. In: 2015 IEEE International Conference on Autonomic Computing (ICAC), pp. 61–70 (2015). https://doi.org/10.1109/ICAC.2015.46
98. Hallsteinsen, S., Hinchey, M., Park, S., Schmid, K.: Dynamic software product lines. Computer **41**(4), 93–95 (2008). https://doi.org/10.1109/MC.2008.123
99. Hartmann, T., Moawad, A., Fouquet, F., Le Traon, Y.: The next evolution of MDE: a seamless integration of machine learning into domain modeling. Softw. Syst. Model. (2017). https://doi.org/10.1007/s10270-017-0600-2
100. Hartmann, T., Moawad, A., Fouquet, F., Nain, G., Klein, J., Traon, Y.L.: Stream my models: Reactive peer-to-peer distributed models@run.time. In: ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2015, pp. 80–89 (2015). https://doi.org/10.1109/MODELS.2015.7338238
101. Heinzemann, C., Becker, S., Volk, A.: Transactional execution of hierarchical reconfigurations in cyber-physical systems. Softw.

Syst. Model. (2017). https://doi.org/10.1007/s10270-017-0583-z

102. Hinchey, M., Park, S., Schmid, K.: Building dynamic software product lines. Computer **45**, 22–26 (2012). https://doi.org/10.1109/MC.2012.332

103. Hong, Jy, Suh, Eh, Kim, S.J.: Context-aware systems. Expert Syst. Appl. **36**(4), 8509–8522 (2009). https://doi.org/10.1016/j.eswa.2008.10.071

104. Hooman, J., Hendriks, T.: Model-based run-time error detection. In: Giese, H. (ed.) Models in Software Engineering, Lecture Notes in Computer Science, vol. 5002, pp. 225–236. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-69073-3_24

105. Hussein, M., Han, J., Yu, J., Colman, A.: Enabling runtime evolution of context-aware adaptive services. In: 2013 IEEE International Conference on Services Computing, pp. 248–255 (2013). https://doi.org/10.1109/SCC.2013.77

106. Iordanov, B., Alexandrova, A., Abbas, S., Hilpold, T., Upadrasta, P.: The semantic web as a software modeling tool: an application to citizen relationship management. In: Model-Driven Engineering Languages and Systems, pp. 589–603. Springer (2013)

107. Jacques-Silva, G., Challenger, J., Degenaro, L., Giles, J., Wagle, R.: Towards autonomic fault recovery in system-s. In: 4th International Conference on Autonomic Computing, 2007. ICAC '07, pp. 31–31 (2007). https://doi.org/10.1109/ICAC.2007.40

108. Janik, A., Zielinski, K.: Transparent resource management and self-adaptability using multitasking virtual machine RM API. In: Proceedings of the 2006 International Workshop on Self-Adaptation and Self-Managing Systems, SEAMS '06, pp. 51–57. ACM, New York, NY, USA (2006). https://doi.org/10.1145/1137677.1137688

109. Javed, F., Arshad, N.: Adopt: an adaptive optimization framework for large-scale power distribution systems. In: 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2009. SASO '09, pp. 254–264 (2009). https://doi.org/10.1109/SASO.2009.26

110. Johanndeiter, T., Goldstein, A., Frank, U.: Towards business process models at runtime. In: Proceedings of the 8th Workshop on Models@run.time, pp. 13–25. CEUR-WS.org (2013)

111. Junior, A.S., Costa, F., Clarke, P.: A model-driven approach to develop and manage cyber-physical systems. In: Proceedings of the 8th Workshop on Models@run.time, pp. 62–73. CEUR-WS.org (2013)

112. Karol, S., Bürger, C., Aßmann, U.: Towards well-formed fragment composition with reference attribute grammars. In: Grassi, V., Mirandola, R., Medvidovic, N., Larsson, M. (eds.) Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering, CBSE 2012, Part of Comparch 12 Federated Events on Component-Based Software Engineering and Software Architecture, pp. 109–114. ACM (2012)

113. Kitchenham, B.: Procedures for Performing Systematic Reviews (2004)

114. Kounev, S., Brosig, F., Huber, N.: Self-aware QoS management in virtualized infrastructures. In: Proceedings of the 8th ACM International Conference on Autonomic Computing, pp. 175–176. ACM (2011). https://doi.org/10.1145/1998582.1998615

115. Kounev, S., Kephart, J.O., Milenkoski, A., Zhu, X. (eds.): Self-Aware Computing Systems. Springer, Cham (2017)

116. Kounev, S., Lewis, P.R., Bellman, K.L., Bencomo, N., Cámara, J., Diaconescu, A., Esterle, L., Geihs, K., Giese, H., Götz, S., Inverardi, P., Kephart, J.O., Zisman, A.: The notion of self-aware computing. In: Self-Aware Computing Systems, pp. 3–16 (2017). https://doi.org/10.1007/978-3-319-47474-8_1

117. Křikava, F., Collet, P., France, R.B.: Actress: domain-specific modeling of self-adaptive software architectures. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC '14, pp. 391–398. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2554850.2555020

118. Krikava, F., Rouvoy, R., Seinturier, L.: Infrastructure as runtime models: towards model-driven resource management. In: ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2015, pp. 100–105 (2015). https://doi.org/10.1109/MODELS.2015.7338240

119. Kuhn, A., Verwaest, T.: FAME—a polyglot library for metamodeling at runtime. In: Proceedings of the 3rd International Models@Runtime Workshop, pp. 57–66 (2008)

120. Kusic, D., Kandasamy, N., Jiang, G.: Approximation modeling for the online performance management of distributed computing systems. In: 4th International Conference on Autonomic Computing, 2007. ICAC '07, pp. 23–23 (2007). https://doi.org/10.1109/ICAC.2007.8

121. Lee, J., Muthig, D., Naab, M.: An approach for developing service oriented product lines. In: Proceedings of the 12th International on Software Product Line Confer SPLC 2008, pp. 275–284 (2008). https://doi.org/10.1109/SPLC.2008.34

122. Loulou, H., Saudrais, S., Soubra, H., Larouci, C.: Adapting security policy at runtime for connected autonomous vehicles. In: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 26–31 (2016). https://doi.org/10.1109/WETICE.2016.16

123. Maes, P.: Concepts and experiments in computational reflection. In: Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications, OOPSLA '87, pp. 147–155. ACM, New York, NY, USA (1987). https://doi.org/10.1145/38765.38821

124. Maier, M.W.: Architecting principles for systems-of-systems. Syst. Eng. **1**(4), 267–284 (1998). https://doi.org/10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D

125. Maoz, S.: Using model-based traces as runtime models. Computer **42**(10), 0028–36 (2009)

126. Mocci, A., Sangiorgio, M.: Detecting component changes at run time with behavior models. Computing **95**(3), 191–221 (2013). https://doi.org/10.1007/s00607-012-0214-z

127. Mongiello, M., Pelliccione, P., Sciancalepore, M.: Ac-contract: run-time verification of context-aware applications. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '15, pp. 24–34. IEEE Press, Piscataway, NJ, USA (2015). http://dl.acm.org/citation.cfm?id=2821357.2821363

128. Morin, B., Fleurey, F., Bencomo, N., Jézéquel, J.M., Solberg, A., Dehlen, V., Blair, G.: An aspect-oriented and model-driven approach for managing dynamic variability. In: Model Driven Engineering Languages and Systems, pp. 782–796. Springer (2008)

129. Morin, B., Nain, G., Barais, O., Jezequel, J.M.: Leveraging models from design-time to runtime. A live demo. In: Proceedings of the 4th Workshop on Models@run.time, pp. 21–30 (2009)

130. Morin, B., Barais, O., Jezequel, J., Fleurey, F., Solberg, A.: Models@ run. time to support dynamic adaptation. Computer **42**(10), 44–51 (2009)

131. Mosincat, A.D., Binder, W.: Self-tuning BPEL processes. In: Proceedings of the 6th International Conference on Autonomic Computing, pp. 47–48. ACM (2009). https://doi.org/10.1145/1555228.1555239

132. Moyano, F., Fernandez-Gago, C., Lopez, J.: A model-driven approach for engineering trust and reputation into software services. J. Netw. Comput. Appl. **69**, 134–151 (2016). https://doi.org/10.1016/j.jnca.2016.04.018. http://www.sciencedirect.com/science/article/pii/S1084804516300698

133. Mullins, R.: The EternalS Roadmap—Defining a Research Agenda for Eternal Systems, pp. 135–147. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-45260-4_10

134. Nascimento, A., Rubira, C., Castor, F.: Using CVL to support self-adaptation of fault-tolerant service compositions. In: IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2013, pp. 261–262 (2013). https://doi.org/10.1109/SASO.2013.34

135. Neamtiu, I.G.: Practical Dynamic Software Updating. Ph.D. Thesis (2008)

136. Park, S., Hinchey, M., In, H.P., Schmid, K.: 8th International workshop on dynamic software product lines (dspl 2014). In: Proceedings of the 18th International Software Product Line Conference—Volume 1, SPLC '14, pp. 355–355. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2648511.2648554

137. Parra, C., Blanc, X., Cleve, A., Duchien, L.: Unifying design and runtime software adaptation using aspect models. Sci. Comput. Program. **76**(12), 1247–1260 (2011). https://doi.org/10.1016/j.scico.2010.12.005

138. Pasquale, L., Baresi, L., Nuseibeh, B.: Towards adaptive systems through requirements@runtime. In: Proceedings of the 6th Workshop on Models@run.time, pp. 13–24 (2011)

139. Paucar, L.H.G., Bencomo, N., Yuen, K.K.F.: Juggling preferences in a world of uncertainty. In: 25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, 4–8 September 2017, pp. 430–435 (2017). https://doi.org/10.1109/RE.2017.12

140. Paucar, L.H.G., Bencomo, N.: Runtime models based on dynamic decision networks: enhancing the decision-making in the domain of ambient assisted living applications. In: Proceedings of the 11th International Workshop on Models@run.time Co-located with 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), Saint Malo, France, 4 October 2016, pp. 9–17 (2016). http://ceur-ws.org/Vol-1742/MRT16_paper_12.pdf

141. Pickering, B., Robert, S., Menoret, S., Mengusoglu, E.: Model-driven management of complex systems. In: Proceedings of the 3rd International Models@ Runtime Workshop, pp. 117–126 (2008)

142. Piechnick, C., Piechnick, M., Götz, S., Püschel, G., Aßmann, U.: Managing distributed context models requires adaptivity too. In: Götz, S., Bencomo, N., Blair, G., Song, H. (eds.) Proceedings of the 10th International Workshop on Models@run.time, pp. 61–70. CEUR-WS.org (2015)

143. Porter, J., Menascé, D.A., Gomaa, H.: Desarm: a decentralized mechanism for discovering software architecture models at runtime in distributed systems. In: Proceedings of the 11th International Workshop on Models@run.time Co-located with 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), Saint Malo, France, 4 October 2016, pp. 43–51 (2016). http://ceur-ws.org/Vol-1742/MRT16_paper_3.pdf

144. Ramirez, A.J., Cheng, B.H., Bencomo, N., Sawyer, P.: Relaxing claims: coping with uncertainty while evaluating assumptions at run time. In: Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems, pp. 53–69. Springer (2012)

145. Ramirez, A.J., Jensen, A.C., Cheng, B.H.C.: A taxonomy of uncertainty for dynamically adaptive systems. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '12, pp. 99–108. IEEE Press, Piscataway, NJ, USA (2012). http://dl.acm.org/citation.cfm?id=2666795.2666812

146. Redlich, D., Blair, G.S., Rashid, A., Molka, T., Gilani, W.: Research challenges for business process models at run-time. In: Models@run.time—Foundations, Applications, and Roadmaps (Dagstuhl Seminar 11481, 27 November–2 December 2011), pp. 208–236 (2014). https://doi.org/10.1007/978-3-319-08915-7_8

147. Ressia, J., Renggli, L., Girba, T., Nierstrasz, O.: Run-time evolution through explicit meta-objects. In: Proceedings of the 5th Workshop on Models@run.time, pp. 37–48 (2010)

148. Riva, C., Rodriguez, J.V.: Combining static and dynamic views for architecture reconstruction. In: Proceedings of the 6th European Conference on Software Maintenance and Reengineering, pp. 47–55 (2002). https://doi.org/10.1109/CSMR.2002.995789

149. Rothenberg, J., Widman, L.E., Loparo, K.A., Nielsen, N.R.: The nature of modeling. In: Artificial Intelligence, Simulation and Modeling, pp. 75–92. Wiley (1989)

150. Sabatucci, L., Cossentino, M.: From means-end analysis to proactive means-end reasoning. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '15, pp. 2–12. IEEE Press, Piscataway, NJ, USA (2015). http://dl.acm.org/citation.cfm?id=2821357.2821361

151. Salehie, M., Tahvildari, L.: Self-adaptive software: landscape and research challenges. ACM Trans. Auton. Adapt. Syst. **4**(2), 141–1442 (2009). https://doi.org/10.1145/1516533.1516538

152. Samuel, A.L.: Some studies in machine learning using the game of checkers. IBM J. Res. Dev. **44**(1.2), 206–226 (2000). https://doi.org/10.1147/rd.441.0206

153. Sanchez, M., Barrero, I., Villalobos, J., Deridder, D.: An execution platform for extensible runtime models. In: Proceedings of the 3rd International Models@ Runtime Workshop, pp. 107–116 (2008)

154. Saudrais, S., Staikopoulos, A., Clarke, S.: Using specification models for runtime adaptations. In: Proceedings of the 4th Workshop on Models@run.time, pp. 109–117 (2009)

155. Sawyer, P., Bencomo, N., Whittle, J., Letier, E., Finkelstein, A.: Requirements-aware systems: a research agenda for re for self-adaptive systems. In: 2010 18th IEEE International Requirements Engineering Conference, pp. 95–103 (2010). https://doi.org/10.1109/RE.2010.21

156. Schneider, D., Becker, M., Trapp, M.: Approaching runtime trust assurance in open adaptive systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '11, pp. 196–201. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1988008.1988036

157. Schneider, D., Becker, M.: Runtime models for self-adaptation in the ambient assisted living domain. In: Proceedings of the 3rd International Models@ Runtime Workshop, pp. 47–56 (2008)

158. Schneider, D., Trapp, M.: A safety engineering framework for open adaptive systems. In: 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2011, pp. 89–98 (2011). https://doi.org/10.1109/SASO.2011.20

159. Schneider, D., Trapp, M.: Conditional safety certification of open adaptive systems. ACM Trans. Auton. Adapt. Syst. **8**(2), 8:1–8:20 (2013). https://doi.org/10.1145/2491465.2491467

160. Schöne, R., Götz, S., Aßmann, U., Bürger, C.: Incremental runtime-generation of optimisation problems using rag-controlled rewriting. In: Proceedings of the 11th International Workshop on Models@run.time Co-located with 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), Saint Malo, France, 4 October 2016, pp. 26–34 (2016). http://ceur-ws.org/Vol-1742/MRT16_paper_5.pdf

161. Sheikh, M.B., Minhas, U.F., Khan, O.Z., Aboulnaga, A., Poupart, P., Taylor, D.J.: A Bayesian approach to online performance modeling for database appliances using Gaussian models. In: Proceedings of the 8th ACM International Conference on Autonomic Computing, pp. 121–130. ACM (2011). https://doi.org/10.1145/1998582.1998603

162. Simmonds, J., Ben-David, S., Chechik, M.: Monitoring and recovery for web service applications. Computing **95**(3), 223–267 (2013). https://doi.org/10.1007/s00607-012-0215-y

163. Song, H., Huang, G., Chauvel, F., Sun, Y.: Applying MDE tools at runtime: experiments upon runtime models. In: Proceedings of the 5th Workshop on Models@run.time, pp. 25–36 (2010). **(Tool demo paper)**

164. Song, H., Huang, G., Xiong, Y.F., Chauvel, F., Sun, Y., Mei, H., et al.: Inferring meta-models for runtime system data from the clients of management APIs. In: Proceedings of the 13th International Conference on Model-Driven Engineering Languages and Systems (MODELS 2010), vol. 6395 (2010)

165. Song, H., Xiong, Y., Chauvel, F., Huang, G., Hu, Z., Mei, H.: Generating synchronization engines between running systems and their model-based views. In: Proceedings of the 4th Workshop on Models@run.time, pp. 11–20 (2009)

166. Song, H., Zhang, X., Ferry, N., Chauvel, F., Solberg, A., Huang, G.: Modelling adaptation policies as domain-specific constraints. In: Model-Driven Engineering Languages and Systems, pp. 269–285. Springer (2014)

167. Spinner, S., Kounev, S., Zhu, X., Lu, L., Uysal, M., Holler, A., Griffith, R.: Runtime vertical scaling of virtualized applications via online model estimation. In: IEEE 8th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2014, pp. 157–166 (2014). https://doi.org/10.1109/SASO.2014.29

168. Staikopoulos, A., Saudrais, S., Clarke, S., Padget, J., Cliffe, O., De Vos, M.: Mutual dynamic adaptation of models and service enactment in alive. In: Proceedings of the 3rd International Models@ Runtime Workshop, pp. 26–35 (2008)

169. Stehle, E., Lynch, K., Shevertalov, M., Rorres, C., Mancoridis, S.: On the use of computational geometry to detect software faults at runtime. In: Proceedings of the 7th International Conference on Autonomic Computing, pp. 109–118. ACM (2010). https://doi.org/10.1145/1809049.1809069

170. Szvetits, M., Zdun, U.: Enhancing root cause analysis with runtime models and interactive visualizations. In: Proceedings of the 8th Workshop on Models@run.time, pp. 38–49. CEUR-WS.org (2013)

171. Szvetits, M., Zdun, U.: Reusable event types for models at runtime to support the examination of runtime phenomena. In: ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2015, pp. 4–13 (2015). https://doi.org/10.1109/MODELS.2015.7338230

172. Szvetits, M., Zdun, U.: Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. Softw. Syst. Model. **15**(1), 31–69 (2016)

173. Tallabaci, G., Souza, V.E.S.: Engineering adaptation with Zanshin: an experience report. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, pp. 93–102. IEEE Press, Piscataway, NJ, USA (2013)

174. Tamura, G., Villegas, N.M., Müller, H.A., Duchien, L., Seinturier, L.: Improving context-awareness in self-adaptation using the dynamico reference model. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, pp. 153–162. IEEE Press, Piscataway, NJ, USA (2013)

175. Tanvir Al Amin, M., Li, S., Rahman, M., Seetharamu, P., Wang, S., Abdelzaher, T., Gupta, I., Srivatsa, M., Ganti, R., Ahmed, R., Le, H.: Social trove: a self-summarizing storage service for social sensing. In: IEEE International Conference on Autonomic Computing (ICAC), 2015, pp. 41–50 (2015). https://doi.org/10.1109/ICAC.2015.47

176. Taylor, R.N., Medvidovic, N., Oreizy, P.: Architectural styles for runtime software adaptation. In: Joint Working IEEE/IFIP Conference on Software Architecture, 2009 and European Conference on Software Architecture. WICSA/ECSA 2009, pp. 171–180. IEEE (2009). **(Need to define for fundamental)**

177. Vasconcelos, A., Werner, C.: Software architecture recovery based on dynamic analysis. In: XVIII Brazilian Symposium on Software Engineering, Workshop on Modern Software Maintenance (2004)

178. Vialon, A., Tei, K., Aknine, S.: Soft-goal approximation context awareness of goal-driven self-adaptive systems. In: 2017 IEEE International Conference on Autonomic Computing (ICAC), pp. 233–238 (2017). https://doi.org/10.1109/ICAC.2017.25

179. Vogel, T., Giese, H.: A language for feedback loops in self-adaptive systems: executable runtime megamodels. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 129–138 (2012). https://doi.org/10.1109/SEAMS.2012.6224399

180. Vogel, T., Giese, H.: Language and framework requirements for adaptation models. In: Proceedings of the 6th Workshop on Models@run.time, pp. 1–12 (2011)

181. Vogel, T., Giese, H.: On unifying development models and runtime models. In: Götz, S., Bencomo, N., France R. (eds.) Proceedings of the 9th International Workshop on Models@run.time, pp. 5–10. CEUR-WS.org (2014)

182. Vogel, T., Seibel, A., Giese, H.: Toward megamodels at runtime. In: Proceedings of the 5th Workshop on Models@run.time, pp. 13–24 (2010)

183. Vogel, T., Giese, H.: Model-driven engineering of self-adaptive software with eurema. ACM Trans. Auton. Adapt. Syst. **8**(4), 18:1–18:33 (2014). https://doi.org/10.1145/2555612

184. Vrbaski, M., Mussbacher, G., Petriu, D., Amyot, D.: Goal models as run-time entities in context-aware systems. In: Proceedings of the 7th Workshop on Models@Run.Time, MRT '12, pp. 3–8. ACM, New York, NY, USA (2012). https://doi.org/10.1145/2422518.2422520

185. Walter, J., Marco, A.D., Spinner, S., Inverardi, P., Kounev, S.: Online learning of run-time models for performance and resource management in data centers. In: Self-Aware Computing Systems, pp. 507–528. IEEE Press, Los Alamitos, CA, USA (2017). https://doi.org/10.1007/978-3-319-47474-8_17

186. Wätzold, S., Giese, H.: Classifying distributed self-* systems based on runtime models and their coupling. In: Götz, S., Bencomo, N., France, R. (eds.) Proceedings of the 9th International Workshop on Models@run.time, pp. 11–20. CEUR-WS.org (2014)

187. Weissbach, M., Chrszon, P., Springer, T., Schill, A.: Decentralized coordination of adaptations in distributed self-adaptive software systems. In: 2017 IEEE 11th International Conference on Self-Adaptive and Self-Organizing Systems (SASO) (2017)

188. Welsh, K., Bencomo, N., Sawyer, P., Whittle, J.: Self-explanation in adaptive systems based on runtime goal-based models, pp. 122–145 (2014). https://doi.org/10.1007/978-3-662-44871-7_5

189. Welsh, K., Sawyer, P., Bencomo, N.: Run-time resolution of uncertainty. In: RE 2011, 19th IEEE International Requirements Engineering Conference, Trento, Italy, 29 August 2011–2 September 2011, pp. 355–356 (2011). https://doi.org/10.1109/RE.2011.6051673

190. Weyns, D., Iftikhar, M.U., Söderlund, J.: Do external feedback loops improve the design of self-adaptive systems? A controlled experiment. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, pp. 3–12. IEEE Press, Piscataway, NJ, USA (2013). http://dl.acm.org/citation.cfm?id=2487336.2487341

191. Wolfe, C., Graham, T.N., Phillips, W.G.: An incremental algorithm for high-performance runtime model consistency. In: Model Driven Engineering Languages and Systems, pp. 357–371. Springer (2009)

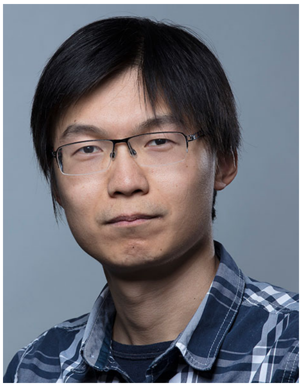
192. Zhang, X., Chen, X., Zhang, Y., Wu, Y., Yao, W., Huang, G., Lin, Q.: Runtime model based management of diverse cloud resources. In: Model-Driven Engineering Languages and Systems, pp. 572–588. Springer (2013)

193. Zhong, C., DeLoach, S.A.: Runtime models for automatic reorganization of multi-robot systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '11, pp. 20–29. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1988008.1988012

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Nelly Bencomo is a lecturer in Computer Science in Aston University, UK. Before, she was a Marie Curie Fellow at INRIA Paris (2011–2013) and a Senior Research Associate in Lancaster University, UK, where she also got her Ph.D. She has built a portfolio of successful collaborative and innovative research in areas such as decision-making under uncertainty and distributed, self-adaptive and autonomous systems. She was the Program Chair of the Symposium Software Engineering for Adaptive and Self-Managing Systems (SEAMS) in India, in 2014, and a Co-program Chair of the 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Italy, in 2018.

Sebastian Götz is a senior researcher at Technische Universität Dresden, Germany. His research interests are model-driven self-optimizing systems, energy-efficient software systems and software engineering for robotics.



Hui Song is research scientist in SINTEF Digital, Norway. He got his Ph.D. from Peking University, China, and before joining SINTEF, he worked as Post-Doc in Trinity College Dublin, Ireland. His research interest is Software Engineering techniques, especially model-driven engineering, and the application of them on cloud computing and Internet of Things.