Summary of the 8th International Workshop on Models@run.time

Nelly Bencomo[1], Robert France[2], Sebastian Götz[3], and Bernhard Rumpe[4]

[1] Aston University, UK
[2] Colorado State University, USA
[3] Technische Universität Dresden, Germany
[4] RWTH Aachen, Germany
nelly@acm.org,sebastian.goetz@acm.org,
france@cs.colostate.edu,rumpe@se-rwth.de

Abstract. The 8th edition of the workshop Models@run.time was held at the 16th International Conference MODELS. The workshop took place in the city of Miami, USA, on the 29th of September 2013. The workshop was organised by Nelly Bencomo, Sebastian Götz, Robert France and Bernhard Rumpe. Here, we present a summary of the workshop and a synopsis of the papers discussed during the workshop.

Keywords: runtime adaptation, MDE, reflection, run-time abstractions

1 Introduction

The Models@run.time workshop series provides a forum for exchange of ideas on the use of run-time models. The main goal is to further promote cross-fertilization between researchers from different communities, including model-driven software engineering, software architectures, computational reflection, adaptive systems, autonomic and self-healing systems, and requirements engineering. This edition of the workshop successfully brought together at least twenty five researchers from different communities.

This edition had nine (9) papers presented. Improved versions of these papers are published in this post-workshop proceedings.

2 Workshop Format and Session Summaries

The workshop was held as a one-day workshop in MODELS. The activities were structured into presentations followed by an afternoon discussion session.

Nine (9) papers were presented. In total, seven (7) long presentations and two (2) short presentations were given. Authors of the papers with long presentations discussed their papers in a twenty-minute-time slot, and five minutes were allowed for questions and discussion. Shorter presentations, in turn, had fifteen (15) minutes.

In the second part of the afternoon, the workshop participants dedicated their time for discussions where a useful classification of presented and discussed approaches was defined. In the following we summarize the nine (9) paper presentations given in three sessions:

Session 1: Safety and Cyber Physical Systems

Enhancing Root Cause Analysis with Runtime Models and Interactive Visualizations by Michael Szvetits and Uwe Zdun. In this paper the authors propose the use of runtime models in combination with interactive visualizations to ease tracing between log file entries and corresponding software artefacts. The contribution of this paper is a repository-based approach to augment root cause analysis (of unwanted behaviour for example) with interactive tracing views while maximizing reusability of models created during the software development process.

Building heterogeneous models at runtime to detect faults in ambient-intelligent environments by Christophe Jacquet, Ahmed Mohamed, Frédéric Boulanger, Cecile Hardebolle and Yacine Bellik. This paper introduces an approach for fault detection in ambient-intelligent environments. It proposes to compute predictions for sensor values, to be compared with actual values. As ambient environments are highly dynamic, one cannot pre-determine a prediction method. Therefore, our approach relies on (a) the modeling of sensors, actuators and physical effects that link them, and (b) the automatic construction at run-time of a heterogeneous prediction model. The prediction model can then be executed on a heterogeneous modeling platform such as ModHel'X, which yields predicted sensor values.

A Model-driven Approach to Develop and Manage Cyber-Physical Systems by Adalberto Sampaio Jr., Fabio Costa and Peter Clarke. Cyber-Physical Systems (CPS) integrate computing, networking, and physical processes to digitally execute tasks on or using the physical elements of a system. Power microgrids are a particular kind of CPS that enables management and autonomic control of local smart grids, aiming at reliability, fault tolerance and energy efficiency, among other goals. Thw paper explores a new approach based on MDE that uses models at runtime techniques to manage and control microgrids. The approach employs a model execution engine that manages a causally connected runtime model of the microgrid and interprets user-defined models in order to generate controls for the microgrid elements. The authors demonstrate the approach by building the lower layer of the model execution engine. Furthermore, they explored a model-driven technique to build the execution engine and use the resulting experience to argue that the approach can be extended to other kinds of cyber-physical systems.

Session 2: Enterprise and Cloud

A Tool for Collaborative Evolution of Enterprise Architecture Models at Runtime by Sascha Roth, Matheus Hauder and Florian Matthes. In this paper, the authors propose a solution that empowers stakeholders to reveal their information demand collaboratively to facilitate Enterprise Architecture (EA) models that evolve with changing information demands at runtime. They present core concepts of our approach and insights of an implementation thereof as foundation to achieve our long-term goal of evolving EA models. In their implementation we extend a collaboration platform with capabilities to monitor the actual information demand and to maintain the EA model referring to this demand at runtime.

Towards Business Process Models at Runtime by Thomas Johanndeiter, Anat Goldstein and Ulrich Frank. Business Process Management (BPM) suffers from inadequate con-cepts and tools for monitoring and evaluation of process executions at runtime. Conversely, models at runtime promise to give insights into the state of a soft-ware system using the abstract and concrete appearance of design time process models. Therefore, the authors at first advocate to use models at runtime in business process (BP) modeling. Then, we outline the implementation of a prototypical modeling framework for BP runtime models based on metaprogramming. This framework supports the integration of BP type models - models that are en-hanced with statistics of runtime data - and instance models - visual representa-tions of executed BPs - resulting in versatile process monitoring dashboards. The approach is superior to object-oriented programming, as it provides a common representation for models and code at various levels of classification, and represents an attractive alternative to object-oriented languages for the im-plementation of runtime models in general.

Models@Runtime to Support the Iterative and Continuous Design of Autonomic Reasoners by Franck Chauvel, Nicolas Ferry and Brice Morin. Modern software systems evolve in a highly dynamic and open environment, where their supporting platforms and infrastructures can change on demand. Designing and operating holistic controllers able to leverage the adaptation capabilities of the complete software stack is a complex task, as it is no longer possible to foresee all possible environment states and system configurations that would properly compensate for them. The paper presents our experience in using models@runtime to foster the systematic design and evaluation of self-adaptive systems, by enabling the coevolution of the reasoning engine and its environment. The research was carried out in the context of the Diversify project, which explores how biodiversity can be used to enhanced the design of self-adaptive mechanisms.

Session 3: Model Extraction and Configuration

Exploring the use of metaheuristic search to infer models of dynamic system behaviour by James Williams, Simon Poulding, Richard Paige and Fiona Polack. As software systems become more pervasive and increase in both size and

complexity, the requirement for systems to be able to self- adapt to changing environments becomes more important. The paper describes a model-based approach for adapting to dynamic runtime environments using metaheuristic optimisation techniques. The metaheuristics exploit metamodels that capture the important components in the adaptation process. Firstly, a model of the environment's behaviour is extracted using a combination of inference and search. The model of the environment is then used in the discovery of a model of optimal system behaviour i.e. how the system should best behave in response to the environment. The system is then updated based on this model. The paper focuses on investigating the extraction of a behaviour model of the environment and describes how our previous work can be utilised for the adaptation stage. The authors contextualise the approach using an example and analyse different ways of applying the metaheuristic algorithms for discovering an optimal model of the case study's environment.

Simon Spinner, Samuel Kounev, Xiaoyun Zhu and Mustafa Uysal. Towards Online Performance Model Extraction in Virtualized Environments. Virtualization increases the complexity and dynamics of modern software architectures making it a major challenge to manage the end-to-end performance of applications. Architecture-level performance models can help here as they provide the modeling power and analysis exibility to predict the performance behavior of applications under varying workloads and configurations. However, the construction of such models is a complex and time-consuming task. In this position paper, the authors discuss how the existing concept of virtual appliances can be extended to automate the extraction of architecture-level performance models during system operation.

Yihan Wu, Ying Zhang, Yingfei Xiong, Xiaodong Zhang and Gang Huang. Towards RSA-based HA configuration in Cloud. High availability (HA) is a crucial concern in cloud systems. However, guaranteeing HA is challenging because of the complex runtime cloud environment, large number of HA mechanisms available, error-prone HA configuration, and ever-needed dynamic adjustment. In this study, the authors leverage runtime system architecture (RSA) for automatic configuration of HA in a cloud. With a causal connection between RSA and its corresponding cloud system, our approach has the following advantages. 1) The runtime changes of the system are abstracted, collected, and reflected on the RSA continuously, simplifying HA information collection. 2) With a model-based HA analyzer working on the collected HA-related information, an appropriate HA style (HAS) can be automatically selected for application to the current system. 3) Changing a HAS on RSA at runtime changes the system HA mechanism, which is suitable for the complex and ever-changing cloud environment. They implemented a prototype of our approach and evaluated it using our model-driven Yan-cloud platform.

The presentations and slides associated with discussions are in the web pages of the workshop.

3 Discussion and Classification of Presented and Discussed Approaches

At the end of the third session discussion topics were identified. All participants agreed on aiming for a classification of the approaches presented at the workshop using the following five dimensions:

1. Runtime Model: What is the runtime model in the approach ?
2. Purpose: What is its purpose?
3. User: Who is using it?
4. Properties: What are its properties?
5. Reflection: Is the runtime model used for reflection?

Eight (8) approaches were classified according to these dimensions. The approaches discussed were approaches developed by people who were in the discussions and were not necessarily related to the papers presented earlier during this workshop but certainly use runtime models.

In order to do the classification, the eight approaches were explained to the rest of the audience by at least one person. Then, the approaches were assigned to two (2) types based on this classification: (1) model-code synchronization and (2) self-adaptive systems / system configuration, where the second type can be further split by the applied domain into (2a) general purpose and (2b) cloud. Of course, further domains exist, e.g., mobile devices and cyber-physical systems, but approaches of these domains were not been discussed at the workshop. The results of the discussions can be summarized as follows:

(1) Approaches on Model-Code Synchronization

Michael Szvetits (University of Applied Sciences Wiener Neustadt) – *Interactive Visualization*

Runtime Model:	Class and sequence diagrams (design time).
Purpose:	Extraction of information from the running system.
User:	Developers use models to understand the design and the running system.
Properties:	Descriptive; there are trace links between model elements and runtime elements.
Reflection:	Is used to determine relevant information.

Mohammed Al-Refai (Colorado State University, USA) – *Fine-grained Adaptation Framework (FiGA)* [1] (not discussed as a paper in this workshop)

Runtime Model:	Class, sequence and activity diagrams.
Purpose:	Enables developers to change the system at runtime by applying changes to the runtime models.
User:	Developers.
Properties:	Visual (representation of source code), abstract, not executable, descriptive.
Reflection:	Yes. Causal connection between models and code.

(2a) General Purpose Self-adaptive Systems

Simon Spinner (KIT, Germany) – *Descartes Meta-Model*

Runtime Model: Architectural performance models, behavior models describing how the system utilizes resources and adaptation models describing the adaptation process and what can be adapted.
Purpose: Optimization of ressource allocation to meet performance objectives.
User: Autonomous systems, i.e., the system is the user.
Properties: Self-reflective, self-predictive, course-grained (i.e., architectural), interpretative.
Reflection: Yes. Causal connection between model and system.

Sebastian Götz (Technische Universität Dresden) – *Multi-Quality Auto-Tuning* [2] (not discussed as a paper in this workshop)

Runtime Model: Architecture of soft- and hardware, behavioral models to predict non- functional properties, Quality-of-Service contracts covering non- functional behavior of implementations and a model reflecting the current state of the overall system.
Purpose: Optimal configuration meeting multiple objectives concurrently.
User: End-user or system are the user.
Properties: descriptive, self-reflective, self-predictive, interpretative, heterogeneous.
Reflection: Yes. Causal connection between model and running system.

Nelly Bencomo (Aston University, UK) – *Runtime Goal-based models for requirements-aware systems* [3] (not discussed as a paper in this workshop)

Runtime Model: Requirement and goal models.
Purpose: Self-adaptive system, thus, configuration.
User: End-user or system itseldcan be user.
Properties: Descriptive
Reflection: Top-down, but not bottom-up.

(2b) Self-adaptive Systems in Cloud Environments

Nicolas Ferry (SINTEF, Norway) – *CloudML*

Runtime Model: Model describing the distribution of components in a cloud environment.
Purpose: Service Provisioning, Optimal Deployment meeting objectives.
User: End-user or system can be the user (also simultaneously).
Properties: Descriptive, can be simulated, i.e., interpretable.
Reflection: Yes. Causal connection between model and system.

Xiadong Zhang (Peking University) – *SM@RT*

Runtime Model:	Architecture model of a cloud environment.
Purpose:	Optimal configuration meeting cost minimization objective.
User:	Enterprises (i.e., management staff) and administrators.
Properties:	heterogeneous, descriptive.
Reflection:	Yes. Causal connection between model and system.

Interestingly, all participants could immediately classify their approach by answering the five questions. Thus, for future work, we plan to do a broader discussion to collect more approaches based on models at runtime and, in consequence, identify more types of approaches.

The classification shown above for the presented approaches can certainly be applied to other approaches. The classification shows that "models@runtime" has evolved as a paradigm covering new types of dynamic systems (like the CPSs described above) and enables a structured investigation of domains like safety (by separation of concerns due to models). Note that during the edition of the workshop last year, models@runtime has been discussed much more fundamentally (e.g., executable models vs. runtime models). This year we saw the application of the paradigm to various domains. A more extensive application of this classification can certainly be useful as a way to evaluate and compare the approaches developed based on the use of runtime models so far.

4 Final Remarks

A general wrap-up discussion was held at the very end of the afternoon. The workshop was closed with a friendly "thank you" from the organizers to all participants for a fruitful workshop. After the workshop, the organizers used the feedback from attendees and PC members to conclude that there is still need for the discussions in this workshop for next year–even after eight years. However, we are planning a new approach for next year while doing the call for papers. It was concluded that the research topic of the workshop has got a certain level of maturity. Several approaches exist and have been applied along all these years and certainly the workshop should take those facts into account.

Acknowledgements

We would also like to thank the members of the program committee who acted as anonymous reviewers and provided valuable feedback to the authors: Christoph Bockisch, Walter Cazzola, Lars Grunske, Martin Gogolla, Bradley Schmerl, Hui Song, Matthias Tichy, Franck Chauvel, Peter J. Clarke, Fabio Costa, Holger Giese, Gang Huang, Jean-Marc Jezequel, Rui Silva Moreira, Brice Morin, Hausi Müller, Arnor Solberg, Mario Trapp and Thaís Vasconcelos Batista.

References

1. Cazzola, W., Rossini, N.A., Al-Refai, M., France, R.: Fine-grained software evolution using uml activity and class models. In: To be published in Proceedings of the 16th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Springer (2013)
2. Götz, S., Wilke, C., Cech, S., Aßmann, U.: Architecture and Mechanisms for Energy Auto Tuning. In: Sustainable ICTs and Management Systems for Green Computing. IGI Global (June 2012) 45–73
3. Bencomo, N., Whittle, J., Sawyer, P., Finkelstein, A., Letier, E.: Requirements reflection: Requirements as runtime entities. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, Cape Town, South Africa, ACM (May 2010) 199–202