Revisiting Model-driven Engineering for Run-time Verification of Business Processes

Wei Dou, Domenico Bianculli, and Lionel Briand

SnT Centre - University of Luxembourg, Luxembourg, Luxembourg
{wei.dou,domenico.bianculli,lionel.briand}@uni.lu

Abstract. Run-time verification has been widely advocated in the last decade as a key technique to check whether the execution of a business process and its interactions with partner services comply with the application requirements. Despite the substantial research performed in this area, there are very few approaches that leverage model-driven engineering (MDE) methodologies and integrate them in the development process of applications based on business process descriptions.
In this position paper we describe our vision and present the research roadmap for adopting MDE techniques in the context of run-time verification of business processes, based on our early experience with a public service partner in the domain of eGovernment. We maintain that within this context the adoption of MDE would contribute in three ways: 1) expressing, at a logical level, complex properties to be checked at run time using a domain-specific language; 2) transforming such properties in a format that can leverage state-of-the-art, industrial-strength tools in order to check these properties; 3) integrating such property checker in run-time verification engines, specific to a target run-time platform, without user's intervention.

1 Introduction

Enterprise information systems are usually realized leveraging the principles of service-oriented architecture [18] and business process modeling. These paradigms foster the design of systems that rely on composition mechanisms, like service orchestrations defined in BPEL [23] or BPMN [24], where added-value applications are obtained by integrating different components, possibly from different divisions within the same organization or even from third-party organizations. This emerging scenario is highly dynamic, open, and decentralized. The global system is not under control and coordination of a single authority. In principle, and according to an extreme viewpoint, multiple autonomous stakeholders contribute to the wealth of available resources [5].

Run-time verification has been widely advocated as a key technique to check whether the execution of a business process and its interactions with partner services comply with the application requirements [1]. Run-time verification becomes very important in the dynamic scenario described above, since it complements traditional design-time verification, which cannot deal with the unex-

pected changes of the system and its environment, typical of open-world software [3].

In the last decade, substantial research has been performed in the areas of design- and run-time verification (see, for example, the surveys in [2, 9, 25]) of business-process-driven, service-based applications. However, we notice there are very few approaches that leverage model-driven engineering (MDE) methodologies and integrate them in the development process of applications based on business process descriptions. We contend that MDE techniques should be revisited in the context of run-time verification of business-process-driven, service-based applications. More specifically, we argue that in this context the adoption of MDE would contribute in three ways: 1) expressing, at a logical level, complex properties to be checked at run time using a domain-specific language; 2) transforming such properties in a format that can leverage state-of-the-art, industrial-strength tools in order to check these properties; 3) integrating such property checker in run-time verification engines, specific to a target run-time platform, without user's intervention.

In this paper we outline a research road map for performing run-time verification of business processes using MDE techniques. This research road map is based on the early experience gained in the context of a project in collaboration with a public service partner in the domain of eGovernment.

The rest of this paper is structured as follows. Section 2 introduces our vision of model-driven run-time verification of business processes, and Sect. 3 describes the challenges we face and our research roadmap to tackle them. Section 4 discusses related work; Sect. 5 concludes the paper.

2 Our Vision

In this section, we describe our long-term vision of a model-driven developement methodology for run-time verification of business processes. As depicted in Fig. 1, the methodology encompasses both the design-time and run-time phases for business processes; in addition, there is an additional layer, called *meta*, which virtually sits in between the design-time and the run-time ones. These three layers are described below.

Design-time Layer

At this layer, the analyst designs the business process, based on requirements specifications. The analyst defines different models, such as use cases, business process models, and data models. Use cases and process models should be annotated with *properties* to be checked at run time. We envision these properties to be expressed in *Restricted Natural Language (RNL)*, using some predefined templates based on property specification patterns (such as the systems defined in [13], [20], [6], [16]).

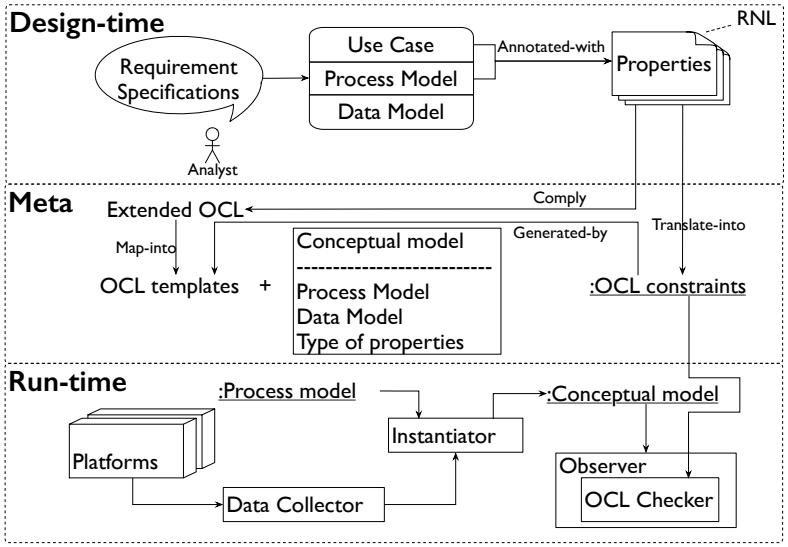

Fig. 1. Our Vision of Model-driven, Run-time Verification of Business Processes

Meta Layer

This layer captures the modeling information that is needed to develop a model-driven approach for run-time verification. Internally to the run-time verification "machinery", we plan to represent the properties to verify at run time using OCL (Object Constraint Language), since it is the standard language in the MDE community and is supported by industrial-strength tools. We then need to translate properties expressed in RNL and defined at the upper design-time layer into plain *OCL constraints*. This translation can be defined by introducing an intermediate language, in the form of an extended version of OCL, which maps RNL templates into corresponding OCL templates.

This layer also defines a *conceptual model* that captures the information that is needed for performing run-time checking: this includes the process model, the data model, the kind of properties to check (e.g., temporal or quality-of-service properties), the information to be collected at run time at the infrastructure level.

Run-time Layer

This layer defines the actual environment in which run-time checks happen. We assume there are different run-time *platforms* (e.g., business process execution engines, JavaEE application servers) on which a *process model* is deployed (possibly, after several model-transformations) and operated. Each platform contains a platform-specific *data collector* that probes and gathers various kinds of run-time information. The process model instance and the information available from

the data collector are given as input to an *Instantiator*, which builds a run-time instance of the conceptual model defined at the *meta* level. This instance is kept alive (and updated) at run time, based on the information coming from the process execution and the platform. The instance is fed into an *Observer*, which receives from the *meta* level also the *OCL constraints* to check. The Observer includes an *OCL checker*, which performs a check of the constraints against the model instance, possibly in an *incremental way*, responding to changes in the model instance. The output of the Observer, in case of violation of a property, can then be used to perform activities such as *root cause analysis*, *debugging*, and *adaptation* (in the form of corrective actions).

3 Research Roadmap

In this section we present our research roadmap for the development of the methodology presented above, and briefly discuss the challenges faced through it and how MDE could contribute to tackle them.

3.1 Requirements Specification Language

Specifications play a significant role in the realm of business-process-driven applications implemented as service compositions. In practice, services are developed by independent parties and are exposed as black boxes that can only be invoked by clients. Their specifications are the only information available to clients, while their implementations are normally inaccessible. A well-designed specification language is thus required to capture and constrain the requirements that a composite business process and its partner services should guarantee. More importantly in the context of run-time verification, these requirements specifications represent the properties to be checked at run time, to assess the correct behavior and quality-of-service (QoS) provided both by the composite business process and by its partner services.

Our initial experience with our public service partner shows that such a requirements specification language should support the specification of functional and non-functional requirements that include the characterization of quantitative aspects of the system, possibly involving temporal constraints. Examples of these requirements are QoS attributes like response time, throughput, which can be expressed as bounds on the sequence and/or number of occurrences of system events, conjuncted with constraints on the temporal distance of events. More in general, the specification language should support the well-known property specification patterns, including temporal [13], real-time [20], and service provisioning [6] patterns.

One of the challenges in the design of such a specification language is to find the right balance between expressiveness and usability, while guaranteeing efficiency for the verification of the properties expressed in this language.

Following this direction, we have developed *OCLR* (OCL for Runtime Verification) [12], a novel temporal extension of OCL based on common property

specification patterns, and extended with support for referring to a specific occurrence of an event in scope boundaries, and for specifying the distance between events and/or from boundaries of the scope of a pattern. *OCLR* extends OCL in a minimal fashion, complementing it to express temporal properties based on Dwyer et al.'s property specification patterns [13]. Moreover, the syntax is very close to English to foster its adoption among practitioners.

3.2 Property Checking

The second step related to run-time verification is how to efficiently verify the properties that can be expressed in the language detailed in the previous subsection.

The efficiency of the verification depends on the expressiveness of the specification language, on the formal model underlying the language, and on the tool support for the verification of the corresponding formal models.

In our case, since *OCLR* is based on OCL, our formal model is actually OCL itself and the tools that can be used for checking *OCLR* properties are represented by existing OCL checkers, such as Eclipse OCL.

Our idea is to recast the problem of the verification of *OCLR* properties at run time in terms of the checking of OCL constraints on instances of a model (kept alive at run time, as described in Sect. 2) corresponding to the actual execution of a business process. This approach leverages existing MDE techniques and technologies and we believe it is a safer and more efficient choice, with respect to developing a dedicated checker for *OCLR* from scratch.

In this regards, we have started assessing the feasibility of checking *OCLR* constraints over execution traces by proposing a mapping of *OCLR* constraints into OCL, based on a conceptual model for traces. The trace checking problem has been re-casted in terms of a check of OCL invariants [11]. The results of our preliminary evaluation using a proof-of-concept tool are encouraging, since the verification of traces with up to 10 million events takes only few minutes, depending on the complexity of the properties.

The trace checking approach reported in [11] focuses on *offline checking*. The next research steps will focus on tuning up the checking procedure to provide adequate performance when used for *run-time* trace checking of *OCLR* properties. We will also consider the use of incremental checking techniques of OCL constraints (see, for example [8] [7] [26] [14] [21], possibly following a syntactic-semantic approach [4]) as well as techniques for efficiently managing the history of monitored events [10].

3.3 Integration with Run-time Platforms

The last step of our research roadmap is the integration of the property checking procedure outlined in the previous section within the actual run-time execution platforms of business process applications. We plan to support at least two main execution platforms, which correspond to the ones adopted by our public service

partner: a) JavaEE for business processes delivered as Web applications; b) executable BPMN 2.0 process description executed on a process execution engine. In both cases, the idea is to embed the property checker (based on an OCL checker) within the run-time platforms. While the checker is the same across the platforms, the data collection architecture used to feed the checker will be different and platform-specific. Based on the expected inputs (and outputs) of the checker, a data collector should be put in place, for example using message interceptors in the business process engine or a dedicated middleware component (for example, implemented with EJB) for the case of JavaEE applications.

4 Related Work

The research on design- and run-time verification of business-process driven, service-based applications spans for more than a decade (see, for example, the surveys in [2, 9, 25]). However, to the best of our knowledge, the solutions proposing a complete model-driven approach to run-time verification for this application domain are very few; in the rest of this section we review them and comment on their limitations.

The model-driven approach presented in [27] relies on a graph-based model that includes *Key Performance Indicators (KPIs)* (e.g., process execution time, server availability); *correlation rules* that specify event patterns to be matched; *action policies* defining the actions to be taken when a certain event occurs and when the KPIs have certain values. Based on the correlation rules, this graph is then decomposed and transformed into several BPEL processes, which are extended with a logic to monitor KPIs and execute action policies. This approach focuses on generating business processes with (KPIs) monitoring capabilities, but it does not provide any mechanisms to link back to high-level requirements and embeds the monitoring code directly within the process structure.

A fully integrated approach for design and implementation of monitored web service compositions is presented in [22]. The approach proposes a set of metamodels for defining performance indicators and their calculation rules, as well as a set of model transformations that are used to generate an executable implementation on top of IBM WebSphere Business Monitor. Although the approach promotes the adoption of reusable calculation templates for specifying custom indicators, the ones that can be expressed are still limited by the basic properties of the process activities (e.g., start time) that can be referenced within the templates.

Reference [19] proposes the ProGoalML language as an extension of BPMN with additional modeling elements for metrics, KPIs, and goals. Based on these elements, monitoring CEP (Complex Event Processing) rules are generated to collect the proper information, which is then used to assess the fulfillment of the goals. However, the approach allows for only simple metrics and does not support a temporal dimension for goal fulfillment.

A model-driven approach for transformation from regulatory policies to event correlation rules is presented in [15]. Policies are expressed using real-time tem-

poral logic and then transformed into IBM ACT rules using some parameterized temporal patterns. The definition of the policies is disconnected from the models of the business processes; moreover, the type of policies is limited by the restricted set of temporal patterns supported during the transformation phase.

The model-aware monitoring approach presented in [17] is also related to policy compliance checking. The approach correlates low-level monitoring events with high-level business events by means of traceability information inserted into business process models. This information is then used at run time by a business intelligence component to perform the actual check on the process model instance to which the events refer to. However, the paper does not indicate which kind of policies can be checked using this approach.

5 Conclusion

In this paper we have presented our vision and the research road map to follow for run-time verification of business processes. This vision is currently being developed in collaboration with our public service partner CTIE (Centre des technologies de l'information de l'Etat, the Luxembourg national center for information technology), which has developed in-house a model-driven methodology for designing eGovernment business processes. Our goal is to complement this methodology with the model-driven run-time verification techniques discussed in this paper. At the time of writing this paper, CTIE has already started using *OCLR* [12] for specifying the requirements of business processes. Our next steps will focus on the integration of our model-driven trace checking technique [11] for *OCLR* within their business process execution platforms.

Acknowledgments. This work has been supported by the National Research Fund, Luxembourg (FNR/P10/03).

References

1. Baresi, L., Bianculli, D., Ghezzi, C., Guinea, S., Spoletini, P.: Validation of web service compositions. IET Softw. 1(6), 219–232 (December 2007)
2. Baresi, L., Di Nitto, E. (eds.): Test and Analysis of Web Services. Springer Heidelberg (2007)
3. Baresi, L., Di Nitto, E., Ghezzi, C.: Toward open-world software: Issue and challenges. IEEE Computer 39(10), 36–43 (2006)
4. Bianculli, D., Filieri, A., Ghezzi, C., Mandrioli, D.: Syntactic-semantic incrementality for agile verification. Sci. Comput. Program. (2013), DOI:10.1016/j.scico.2013.11.026
5. Bianculli, D., Ghezzi, C.: Towards a methodology for lifelong validation of service compositions. In: SDSOA 2008. pp. 7–12. ACM (May 2008)
6. Bianculli, D., Ghezzi, C., Pautasso, C., Senti, P.: Specification patterns from research to industry: a case study in service-based applications. In: ICSE 2012. pp. 968–976. IEEE (2012)

7. Cabot, J., Teniente, E.: Incremental evaluation of OCL constraints. In: Dubois, E., Pohl, K. (eds.) Advanced Information Systems Engineering. LNCS, vol. 4001, pp. 81–95. Springer Heidelberg (2006)
8. Cabot, J., Teniente, E.: Incremental integrity checking of UML/OCL conceptual schemas. J. Syst. Softw. 82(9), 1459–1478 (2009)
9. Canfora, G., Di Penta, M.: Service oriented architectures testing: a survey. In: De Lucia, A., Ferrucci, F. (eds.) ISSSE 2006–2008, LNCS, vol. 5413, pp. 78–105. Springer Heidelberg (2009)
10. Chomicki, J.: Efficient checking of temporal integrity constraints using bounded history encoding. ACM Trans. Database Syst. 20, 149–186 (June 1995)
11. Dou, W., Bianculli, D., Briand, L.: A model-based approach to trace checking of temporal properties with OCL. Tech. Rep. TR-SnT-2014-5, SnT Centre - University of Luxembourg (March 2014)
12. Dou, W., Bianculli, D., Briand, L.: OCLR: a more expressive, pattern-based temporal extension of OCL. In: Cabot, J., Rubin, J. (eds.) ECMFA 2014. LNCS, vol. 8569. Springer Heidelberg (July 2014), to appear
13. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: ICSE 1999. pp. 411–420. IEEE (1999)
14. Garcia, M., Möller, R.: Incremental evaluation of OCL invariants in the essential MOF object model. In: Modellierung 2008. vol. 127, pp. 11–26 (2008)
15. Giblin, C., Müller, S., Pfitzmann, B.: From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Tech. Rep. Research Report RZ-3662, IBM Research GmbH (2006)
16. Gruhn, V., Laue, R.: Patterns for timed property specifications. Electron. Notes Theor. Comput. Sci. 153(2), 117–133 (2006)
17. Holmes, T., Mulo, E., Zdun, U., Dustdar, S.: Model-aware monitoring of SOAs for compliance service engineering. In: Service Engineering, pp. 117–136. Springer Vienna (2011)
18. Josuttis, N.: SOA in Practice: The Art of Distributed System Design. O'Reilly Media, Inc. (2007)
19. Koetter, F., Kochanowski, M.: Goal-oriented model-driven business process monitoring using ProGoalML. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012, LNBIP, vol. 117, pp. 72–83. Springer Heidelberg (2012)
20. Konrad, S., Cheng, B.H.C.: Real-time specification patterns. In: ICSE 2005. pp. 372–381. ACM (2005)
21. Menet, L., Lamolle, M., Le Dc, C.: Incremental validation of models in a MDE approach applied to the modeling of complex data structures. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM Workshops. LNCS, vol. 6428, pp. 120–129. Springer Heidelberg (2010)
22. Momm, C., Gebhart, M., Abeck, S.: A model-driven approach for monitoring business performance in web service compositions. In: ICIW 2009. pp. 343–350. IEEE (2009)
23. OASIS: Web Services Business Process Execution Language Version 2.0 (2007)
24. OMG: BPMN 2.0 specification. `http://www.bpmn.org` (January 2011)
25. Salaün, G.: Analysis and verification of service interaction protocols - a brief survey. In: TAV-WEB 2010. EPTCS, vol. 35, pp. 75–86 (2010)
26. Vajk, T., Mezei, G., Levendovszky, T.: An incremental OCL compiler for modeling environments. ECEASST 15 (2008)
27. Yu, T., Jeng, J.: Model driven development of business process monitoring and control systems. In: Chen, C.S., Filipe, J., Seruca, I., Cordeiro, J. (eds.) ICEIS 2005. pp. 161–166 (2005)