A Model-Driven Framework for Domain Specific Process Design and Governance

Adrian Mos[1], Mario Cortes-Cornax[1], José Miguel Pérez-Álvarez[1,2], María Teresa Gómez-López[2]

[1] Xerox Research Center, 6 Chemin de Maupertuis, Meylan, France
{adrian.mos, mario.cortes, jose.perez,}@xrce.xerox.com
[2] Universidad de Sevilla, C/ S. Fernando, 4, C.P. 41004-Sevilla, Spain
{josemi, maytegomez}@us.es

Abstract. Current BPM approaches and standards have not sufficiently reduced the Business-IT gap. Indeed, today's solutions are mostly domain-independent and platform-dependent, which limits the ability of business matter experts to express business intent and enact process change. In contrast, the tool presented in this paper supports an approach that focuses on BPM and SOA environments in a domain-dependent and platform-independent way. We propose to add a domain specific-layer on top of current solutions so that business stakeholders can design and understand their processes in a more intuitive way. This significantly improves the agility and governance of processes. The demo shows the appropriateness and the feasibility of the approach.

Keywords: Model-driven engineering, Process Design, DSL, Governance

1 Introduction

Today, the Business Process Model and Notation [6] (BPMN 2.0) has become the de-facto standard for business process modelling. With the aim at filling the Business-IT gap, significant effort has been put into bringing BPMN closer to Service Oriented Architectures (SOA). A BPM Suite (BPMS) manages the process execution directing SOA calls to the appropriate services and generally provides some monitoring infrastructure. While these components help alleviate agility problems, we observed that most of the existing solutions are domain-independent and platform-dependent, which significantly limits the involvement of business matter experts, especially when targeting execution. Business analysts require dedicated means (i.e., specific type of task with implicit domain knowledge) to effectively model processes in their business domains such as logistics, healthcare, transportation, etc. [7]. Domain Specific Languages (DSLs) are an effective means to deal with application domains providing improvements in expressiveness and ease of use. More specifically, it has been shown that domain-specific process modelling languages (DSPML) [1] can have significant advantages over BPMN. However, domain specific language development is hard,

requiring both domain knowledge and language development expertise [2]. In addition, such languages typically do not benefit from the kind of advanced tooling support and platform integration that BPMN has. In previous work [5] we describe the problems that tool providers face and propose a solution to generate appropriate graphical studios for DSPMLs. Overall, regardless of the specific approach, *a framework that helps building the technical infrastructure to use DSPMLs is critical*.

The presented tool combines technical solutions [3, 4] in a domain-dependent and platform-independent way. This paper focuses on the added novelty of the support and automatic artefact generation for data and forms in the domain-specific layer. The addition of these new features enables the actual BPMN generation and deployment (BonitaBPM[3] in our case) through a one-click action. As mentioned, the framework relies on a model-driven approach, which facilitates the development of studios in order to take advantage of the DSPMLs.

2 Tool Significance for BPM

The environment presented here is essentially an additional layer over the typical BPM stack. The idea is to design using a DSPML, which will be converted to an enriched BPMN that executes in a BPMS. The approach has two main steps: 1) the *Domain Specification*, where the main concepts of a domain are well defined, and 2) the *Process Definition*, where a non technical analyst will compose the predefined building blocks. This separation has the potential to bring *better governance and re-usability*, *faster process evolution*, *multi-target deployment*, and *non-ambiguous monitoring*. Figure 1 gives an overview of the framework. Each number corresponds to one key component described below.

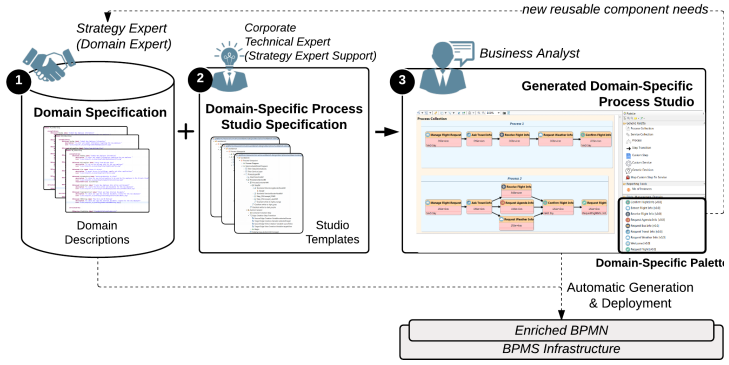

Fig. 1. Framework Overview

The **Domain Specification** relies on the definition of the domain concepts (Data, Services, Activity Types and Forms), which correspond to the organizational know-how [3]. A *Strategy (or Domain) Expert* assisted by technical staff will specify, incrementally, all the common information that any domain

[3] http://www.bonitasoft.com/

process should have access to. To support this, a generic domain meta-model (DomainMM) is used, which can be extended for specific organization needs. The individual domain descriptions (one per domain) correspond to instances of the DomainMM.

The **Domain-Specific Process Studio Specification** corresponds to the definition of several configurable templates that manage the studio functionalities. At this stage, a *Technical Expert* supported by the *Strategy Expert* defines the functionalities that the studios should have (i.e., transformation capabilities, monitoring capabilities, etc). The framework provides the possibility to enable them easily. Here is where the graphical constructs for the domain concepts are defined. We propose some default ones, but they can easily be changed.

The **Generated Domain-Specific Process Studio** brings a design and governance layer on top of existing BPM technology infrastructure for a business domain. The process design is supported by the Mangrove Meta-model[4] that has been extended to support domain connections. *Business Analysts* create their domain specific processes (DSPs) which are eventually transformed to a common BPMN denominator, which can leverage the BPMS investments while preserving the specificity and power of DSPMLs. Our hypothesis is that a reduced amount of symbols (semantically enriched) is enough to define high-level domain-specific process models. These models could potentially be refined in an analysis level [8], using BPMN 2.0. However if all the details are considered correct and no additions are needed, the generated BPMN models can be automatically deployed in one-click. The framework provides the connection to a (generated) platform-independent layer for managing persistence and forms. If at any time, new elements are identified by the business analysts while designing their process models, they could request an update of the domain which in turn will eventually trigger an injection of the new element in the studio palette. This ensures that needs that are identified in the context of a process could benefit the entire domain if deemed important across the domain scope.

3 Prototype Implementation

This prototype (see video[5]) relies on a the set of mature and open-source Eclipse technologies. We believe they are highly relevant for any BPMS, many of which are actually built using Eclipse. Figure 2 shows the architecture of the solution.

The domain-specific studio relies on a plugin-based infrastructure based on Eclipse Rich Client Platform (RCP)[6] and Equinox (OSGI implementation)[7]. The Eclipse Modelling Framework (EMF)[8] is used for the definition of the meta-models such as the Domain MM, the Mangrove MM (i.e. the process MM), the abstract binding repository (ABR MM) and the BPMN MM. The Mangrove

[4] https://www.eclipse.org/mangrove/
[5] http://xerox.bz/2oc6X5W
[6] https://eclipse.org/ide/
[7] http://www.eclipse.org/equinox/
[8] http://www.eclipse.org/modeling/emf/

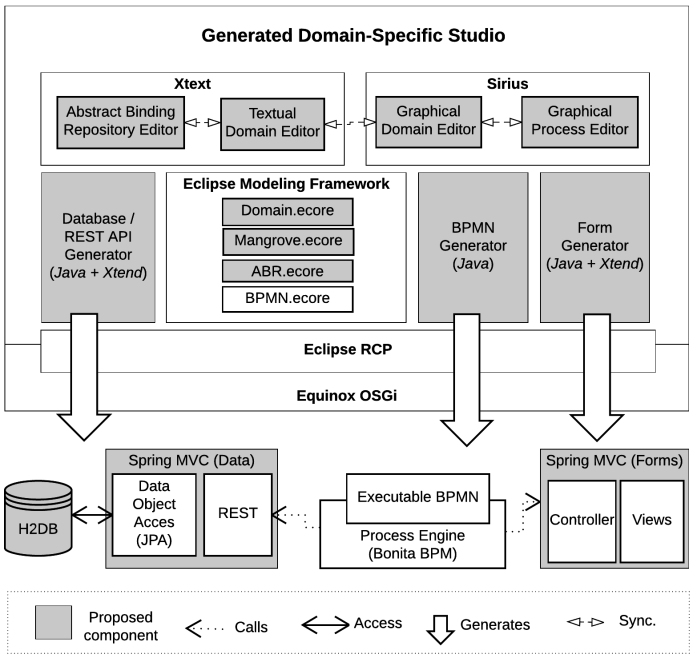

Fig. 2. Technology-Oriented Architecture of the Implemented Prototype

MM[9] is used as the pivot meta-model that enables the link between the DomainMM and the BPMN 2.0 MM for generation purposes. The ABR defines the bindings between the domain services and the actual technical services, indicating the method, the URI, the port, etc. On top of EMF, the Eclipse Xtext framework[10] generates a fully featured textual editor for domain descriptions relying on the DomainMM. The EMF ecore meta-models are the inputs for the Sirius domain-specific editor[11], which allows an easy creation of configurable graphical modeling studios. The BPMN 2.0 transformations are managed by a generator coded in Java.

The data-centric aspects as well as the form generation are managed using the Spring framework[12]. Concerning data, the framework generates Data Access Objects (DAO) in order to reach an H2 Database through a REST API. BonitaBPM is used as target platform. Using the public APIs, the technical information (e.g., the called service) is injected[13] in the generated activity tasks and gateways. In a similar way, forms are generated relying on the domain data objects and then deployed in a Model View Contoller (MVC) infrastructure easily accessible from any process engine. The generators use Xtend[14] to create the necessary files and place them into a base project (containing a static skeleton

[9] https://www.eclipse.org/mangrove/
[10] https://eclipse.org/Xtext/
[11] https://eclipse.org/sirius/
[12] https://projects.spring.io/spring-framework/
[13] https://github.com/jozemi/bonita.explore
[14] http://www.eclipse.org/xtend/

of the project). Once the forms are deployed, the framework permits the regeneration and update of the files at runtime, with no need to modify the process model nor redeploy the process.

4 Conclusion and Future Development

The presented framework provides the business stakeholders with means to design and govern their processes from a high-level, with impact to the entire collection of business processes in a domain, if required. A domain concept repository enables easier re-usability of data, services, activity types and forms independently from the target platform. Relying on the generative approach, changes are spread through the different layers. The framework is supported by tools that automate generation and synchronization. We used a mature set of open-source tools to implement a prototype and used a running example that illustrates the interest and applicability of our approach. Our next explorations include the integration of complex decisions at the domain level as well as state tracking in order to pro-actively help the process designers in the design and analysis of their processes.

Acknowledgment

The authors would like to acknowledge the support of the European Regional Development Fund (ERDF/FEDER) and the Spanish Ministry of Science and Technology (TIN2015-63502-C3-2-R) that are partially funding the work of the IDEA research group in U. of Seville.

References

1. Jablonski, S., Volz, B., Dornstauder, S.: Evolution of business process models and languages. In: 2nd International Conference on Business Process and Services Computing (BPSC). pp. 46–59. Citeseer (2009)
2. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM computing surveys (CSUR) 37(4), 316–344 (2005)
3. Mos, A.: Domain specific monitoring of business processes using concept probes. In: Service-Oriented Computing (ICSOC Workshops). pp. 213–224. Springer (2015)
4. Mos, A., Cortes-Cornax, M.: Business matter experts do matter: A model-driven approach for domain specific process design and monitoring. In: Business Process Management (BPM –Forum). pp. 210–226. Springer (2016)
5. Mos, A., Cortes-Cornax, M.: Generating domain-specific process studios. In: Enterprise Distributed Object Computing Conference (EDOC). pp. 1–10. IEEE (2016)
6. OMG: Business process model and notation (BPMN) version 2.0 (2011). Available on: http://www. omg. org/spec/BPMN/2.0 (2011)
7. Pinggera, J., Zugal, S., Weber, B., Fahland, D., Weidlich, M., Mendling, J., Reijers, H.A.: How the structuring of domain knowledge helps casual process modelers. In: Conceptual Modeling (ER), 2010, pp. 445–451. Springer (2010)
8. Silver, B.: BPMN Method and Style: A levels-based methodology for BPM process modeling and improvement using BPMN 2.0. Cody-Cassidy Press, US (2009)