Encouraging Business Flexibility by Improved Context Descriptions

Johan Silvander, Magnus Wilson and Krzysztof Wnuk
Department of Software Engineering, Blekinge Institute of Technology, Karlskrona, Sweden
{*Johan.Silvander, Magnus.Wilson, Krzysztof.Wnuk*}*@bth.se*

Keywords: Context Description, Business Flexibility, Business Support System, Requirements Engineering, Business Model.

Abstract: Business-driven software architectures are emerging and gaining importance for many industries. As software-intensive solutions continue to be more complex and operate in rapidly changing environments, there is a pressure for increased business flexibility realized by more efficient software architecture mechanisms to keep up with the necessary speed of change. We investigate how improved context descriptions could be implemented in software components, and support important software development practices like business modeling and requirement engineering. This paper proposes context descriptions as an architectural support for improving the connection between business flexibility and software components. We provide initial results regarding software architectural mechanisms which can support context descriptions as well as the context description's support for business-driven software architecture, and the business flexibility demanded by the business ecosystems.

1 INTRODUCTION

Business requirements are shaped by collaboration and continuous knowledge creation (Nonaka, 1994) between stakeholders, who are driven by intents while acting in business ecosystems. The business requirements and the speed of implementing them become the dominant concern for Software Intensive Product Development (SIPD) (Bharadwaj et al., 2013) companies and forces these companies to reach new levels of agility and orchestration of digital resources.

For SIPD, this challenge translates into creating efficient software architectures which support business flexibility in order to adapt existing business models or support new business models as a response to changes in the business ecosystems. Software components are often expensive to re-use and maintain in new or multiple business models due to a lot of business logic connecting various components while functions are hard-wired to certain business environments. Components cannot be re-coded every time a business model changes. Therefore, new software architectures need to support the complete lifecycle of connecting business models to software components with an efficient support for changeability.

This paper focuses on supporting business flexibility by using specific context descriptions. The aim is to transform these context descriptions into executable containers which could be used to support the needed business flexibility. The remaining part of the paper is structured as follows: Section 2 presents background and related work, Section 3 and 4 provide information about how the ideas have been used by practitioners, and Section 5 concludes the paper.

2 BACKGROUND AND RELATED WORK

Business architecture flexibility focuses on business trade-offs that need to be quickly resolved and how they impact both function layers and realization layers. Depending on the estimated future value for relevant stakeholders (Khurum et al., 2013), the business architecture flexibility allows for agile changes to the realization layer. Availability and flexibility are recognized as important aspects in high uncertain business environments (Richter et al., 2010).

The transition to service driven economy has given the birth to Industrial Product-Service Systems (Meier et al., 2010) with the focus on lifecycle-integration of products and services. New possibilities for capturing value as well as for "on-demand lock/un-lock" of business value options are possible with the digital delivery of software and value. This requires the software components to support new lev-

els of flexibility for option-locking support, including governance.

Several significant contributions have been made in decomposing value for software products (Kang et al., 1990) or describing industrial context in software engineering (Petersen and Wohlin, 2009). Castro et al. focused on bridging the gap between the software systems and their operational environment using i* modeling framework (Castro et al., 2002) leveraging on goal based modeling. Goal based modeling of requirements and agent-based software engineering are common approaches to capture the requirements on software components, e.g. KAOS, MAS, and TAO (Silva et al., 2003). However, in practice these frameworks still lacks usability (Dalpiaz et al., 2016) in which industry can effectively and efficiently industrialize these practices and develop efficient software architecture.

Creating software components that can be orchestrated and bring value to the relevant stakeholders in business ecosystems and timely respond to frequent changes remains the main challenge. This is partly addressed by Software product lines (Bosch, 2009) and industrial Product-Service systems (Meier et al., 2010) which focused on changeability (Richter et al., 2010), as ways to create flexible, adaptable and efficient component-based software architectures.

Supporting business flexibility requires support for agile business policies and business rules (Business Rules Group, 2003) which are used to govern how an enterprise does its business (OMG, 2014). It is desired to have a common governance structure and a standardized way of handling the business rules. Rosca et al. have contributed valuable knowledge in the area of common governance of business rules (Rosca et al., 2002; Rosca et al., 1997). However, we use to the more declarative nature of the business rules (Business Rules Group, 2003).

To fully support business flexibility, we need to better understand and define the business context. Modeling context is also critical for developing context-aware software systems (Brun et al., 2009). Baldauf et al. (Baldauf et al., 2007) summarized context-aware systems including methods to achieve context-awareness, e.g. resource discovery, sensing, context model, context processing, hierarchical context data. Despite several similarities, context-aware software systems focus on dynamically discoverable services rather than dynamically changing business opportunities. This paper builds upon the definition provided by Baldauf et al. (Baldauf et al., 2007) and introduces context description and context frame as concepts for achieving context-aware business architectures.

By composition, context descriptions can be used to create efficient re-usable descriptions that can be used not only in business requirements but also in business rules. The context description is what gives a context frame a scope (boundary) and defines a meaning (semantics). In this paper, we propose the context frame as a fundamental building block in new software business architectures to create self-adaptive software components that can understand, negotiate and adapt to a business context.

3 CASE CONTEXT

In today's implementations of business support systems, business rules are configured in different places of the system, and in different formats. This makes it hard to have a common view of what is defined, and to execute the same logic in different parts of systems, without re-implementing the rules.

Since humans are defining the business rules, these rules are usually ambiguous. Visual and logical support to verify the correctness of the defined business rules are desired. Parts of the business rules could be made executable in order to make the operation of the business more efficient and effective. The process of translating business rules to executable business rules is error-prone due to human interpretation. Sharma et al. (Sharma et al., 2014) have proposed a method to find business rules in requirements documents. This is a good start but most business rules are not about the information system it selves but rather about the business the information system shall support.

It is desired to use a software algorithm to translate business rules into executable business rules. A way to execute business rules is to use a common rule engine for all the components in a business support system. This approach might not be desired or possible. Instead, the possibility to express executable business rules in a different software language, which could be distributed to the different components in a business support system, might be an option. Contrary to many proposed solutions, we believe that a business rule could be triggered by several different events. This makes the use of a simple event-condition-action architecture not suitable for the problem at hand.

Together with Ericsson we have performed a proof-of-concept to investigate if it is possible to support visual and logical verification of business rules, and to generate executable business rules. We have chosen to investigate a limited part of an enterprise's business rules. The business rules we have chosen to study are targeting support for value propositions,

based on different business models. The business rules are based on following five parts of the Osterwalder canvas (Osterwalder and Pigneur, 2010): customer type, customer relationship, channels, revenue streams, and a specific area of the value propositions.

4 PROOF OF CONCEPT

Since a business model is supported by a set of legal contracts ,we started to derive the business rules from these type of contracts. Some of the information in a legal contract is not meaningful to translate into a business rule which should be executed in software, e.g. which country laws should be used to solve a dispute. Many times the nature of the language used in legal contracts requires human interpretation. However, the majority of the terms and conditions in a legal contract can be translated into meaningful business rules which could be implemented in software.

We have implemented machine learning pipelines which make it possible to conduct visual and logical verification of business rules, and generate executable business rules. This process can be regarded as the creation of a context frame. We have added different types of functionality which is regarded as needed when handling business rules. Missing data is handled as a wild card, i.e. all values are true. Continuous values have a defined boundary and there are no value gaps in the data. Since a human is defining the business rules, entering all possible combinations by hand is not an option. A meta-data file is used to describe the nature of the features.

In order for the solution to exist in an event-driven environment, the extracted business rule was extended with the events it is intended for. We have added the possibility to use two classification columns. These classification columns respectively represents eligible objects and the allowed actions on the business rules. The idea is borrowed from the gaming industry where a specific context gives the character the possibilities to, for example find specific treasures and stipulates how these treasures can be handled. The combination of event, eligibility and action makes it possible to mimic a business process.

There is a strong demand on the possibility to separate the design from the execution and the need for governance of the business rules throughout their lifecycle. This demand is in-line with TMForum's eTOM (TMForum, 2015). The design is supported by the possibility to, visually and logically, validate the correctness of the business rules before they are put in execution. The execution is supported by the possibility to logically validate the correctness of the business rules before they are put in operation, and to deploy and operate the executable business rules as a context frame. The governance views are supported by the fact that the executable business rules can be handle as immutable artifacts.

The pipe-lines are considered as a proof of concept, and as such is regarded as successful by the four practitioners involved in its evaluation. Three of them are system architects and one is a business support system expert. The pipe-lines make it possible to, visually and logically, validate the correctness of the business rules before they are put in production. Generating executable code representing the model of the business rules, makes it possible to execute the same model in different components without the need of re-implementation. This might improve the coherence of the business rules in a business support system. It was concluded that this way of supporting business processes can support the business models of the business support system it selves, as well as the business models of the enterprises which are running their business with the help of the business support system.

There are several improvements to the pipe-lines which should be considered. The precision of feature value has to be configurable feature by feature, and with different values for the maximum limit and minimum limit. The executable code representing the model of the business rules should support additional languages, for example Java™. The ability to extract information from the legal contracts must be improved. We will investigate how we can leverage on the research made in the area of common governance of business rules (Rosca et al., 2002; Rosca et al., 1997). During the proof-of-concept we elaborated with the logical visualization of the deployed business rules. We plan to investigate if a graph database can support the needed deployment capabilities regarding visualization and semantics.

There are no real-time requirements on the transformation from business rules to executable business rules. Since the data set will vary in the number of used feature columns and since each feature has its own characteristic, the use of a typed-language is not ideal. Based on this, Python fulfills the requirements as a suitable implementation language for the problem at hand.

5 CONCLUSIONS AND FURTHER WORK

This vision paper illustrates the potential benefits with introducing context descriptions and context frames to support business flexibility. The implementation

makes it possible to, visually and logically, validate the correctness of the business rules before they are put in production. The possibility to generate executable code representing the model of the business rules, makes it possible to execute the same model in different components without the need of re-implementation. However, the PoC does not include support for business requirements and the software architecture mechanisms supporting the context frame are very basic.

Together with Ericsson we plan to improve the solution in order to make it useful in a business support system offering. This includes the improvements discussed in this section and in Section 4.

ACKNOWLEDGEMENTS

This work was partially supported by Ericsson AB. We thank the members in our former units for their support and the time they spent with us during this work.

REFERENCES

Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263.

Bharadwaj, A., El Sawey, O. A., Pavlou, P. A., Venkatraman, N., El Sawy, O. a., Pavlou, P. A., and Venkatraman, N. (2013). Digital Business Strategy: Toward a Next Generation of Insights. *MIS Quarterly*, 37(2):471–482.

Bosch, J. (2009). From Software Product Lines to Software Ecosystems. *13th International Software Product Line Conference Proceefings*, (Splc):111–119.

Brun, Y., Di Marzo Serugendo, G., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. (2009). *Engineering Self-Adaptive Systems through Feedback Loops*, pages 48–70. Springer Berlin Heidelberg, Berlin, Heidelberg.

Business Rules Group (2003). The Business Rules Manifesto. http://businessrulesgroup.org/brmanifesto/BRManifesto.pdf Last checked 2017-02-01.

Castro, J., Kolp, M., and Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: the Tropos project. *Information systems*, 27(6):365–389.

Dalpiaz, F., Franch, X., and Horkoff, J. (2016). iStar 2.0 Language Guide. *CoRR*, abs/1605.0.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (FODA) feasibility study. Technical report, DTIC Document.

Khurum, M., Gorschek, T., and Wilson, M. (2013). The software value mapan exhaustive collection of value aspects for the development of software intensive products. *Journal of Software: Evolution and Process*, 25(7):711–741.

Meier, H., Roy, R., and Seliger, G. (2010). Industrial Product-Service systems-IPS2. *CIRP Annals - Manufacturing Technology*, 59(2):607–627.

Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation. *Organization Science*, 5(1):14–37.

OMG (2014). Business Motivation Model. http://www.omg.org/spec/BMM/1.2/ Last checked 2015-12-01.

Osterwalder, A. and Pigneur, Y. (2010). *Business Model Generation*. Wiley, 1st edition.

Petersen, K. and Wohlin, C. (2009). Context in Industrial Software Engineering Research. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 401–404, Washington, DC, USA. IEEE Computer Society.

Richter, A., Sadek, T., and Steven, M. (2010). Flexibility in industrial product-service systems and use-oriented business models. *CIRP Journal of Manufacturing Science and Technology*, 3(2):128–134.

Rosca, D., Greenspan, S., Feblowitz, M., and Wild, C. (1997). A decision making methodology in support of the business rules lifecycle. *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, (October):236–246.

Rosca, D., Greenspan, S., and Wild, C. (2002). Enterprise modeling and decision-support for automating the business rules lifecycle. *Automated Software Engineering*, 9(4):361–404.

Sharma, R., Bhatia, J., and Biswas, K. K. (2014). Automated identification of business rules in requirements documents. *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, pages 1442–1447.

Silva, V., Garcia, A., Brandão, A., Chavez, C., Lucena, C., and Alencar, P. (2003). Software Engineering for Large-scale Multi-agent Systems. chapter Taming Age, pages 1–26. Springer-Verlag, Berlin, Heidelberg.

TMForum (2015). eTOM. https://www.tmforum.org/business-process-framework/ Last checked 2015-12-01.