Moving Towards Industrial Software Ecosystems: Are Our Software Architectures Fit for the Future?

Klaus-Benedikt Schultis
Siemens Corporate Technology
Erlangen, Germany
klaus-benedikt.schultis.ext@siemens.com

Christoph Elsner
Siemens Corporate Technology
Erlangen, Germany
christoph.elsner@siemens.com

Daniel Lohmann
Friedrich-Alexander University
Erlangen-Nuremberg, Germany
lohmann@cs.fau.de

***Abstract*—The development of large-scale software product-lines within large enterprises commonly involves various internal business units. Although within the same enterprise, each business unit has individual motivations and participation interests. For coordinating development, the emerging discipline of software ecosystems intents to explicitly discover and analyze the different players' interests, and manage them, often by means of a suitable software architecture. Already within a single enterprise, this discipline can be of high value. Instead of detailed managerial orders to coordinate internal interactions, an analysis of the players, their interests, and a suitable software architecture may slacken organizational structures and simplify processes.**

We have started to analyze the ecosystems of several Siemens internal product-lines in order to determine the different players and their interests, and to derive suitable software architectural requirements from this setting. This will enable us to compare these requirements to the actual architecture, for identifying reusable pain points and best practices of the existing system. However, there is no systematic (A) approach to model and analyze the collaboration among the participants from a technical perspective, as well as (B) to derive reusable architectural design-patterns and anti-patterns from such software ecosystems. By illustrating these problems using an existing software product-line that moves towards a software ecosystem, we are looking for answers to the two questions above to evaluate whether our product-lines are fit for a future as internal software ecosystems.

I. Introduction

In the last years, more and more software product-line businesses leverage the advantages of opening up their technological base outside their organizational boundary to involve external business units. For this effect, the term software ecosystem (SECO) [1] has been coined. There are many reasons to adapt SECO-like characteristics: reduction of costs involved in software development and innovation by larger-scale reuse, improvement of R&D efficiency, as well as the chance to address more customer needs for reaching a broader customer base [1]. Adapted from the definition of Jansen [2], a SECO consists of a set of co-acting business units together with the relationships among them, a shared market for software and services, a common technological base comprising a reference architecture, core assets or standards and operates through the exchange of information, resources and artifacts.

Up to now, SECOs are mainly considered as a set of co-acting companies where a company itself is regarded as some kind of black box [1], [2], [3], [4], [5]. Nevertheless, we have observed that large-scale development of software product-lines within large enterprises also involves a set of co-acting business units with dedicated interests, consequently adopting SECO-like characteristics. To account for this, we introduce the term industrial software ecosystem (ISECO). We define an industrial software ecosystem as a software ecosystem where the focus is on one key company maintaining large software projects, involving mainly internal business units with partially different motivations and interests. Thus, the view on the organizational structure moves from strict hierarchies towards more decentralized topologies.

Compared to classical software product-line architectures, the software architecture of an ISECO also explicitly shall achieve the coordination among the involved business units [4]. Explicitly considering this effect, as a consequence, introduces a set of additional software architectural challenges. Although they have received attention in literature [4], [2], [5], [6], [7], the identified challenges are derived from informal experiences or abstract considerations. They are not put into a context and backed up by more explicit empirical evidence. As a consequence, they are too generic to identify concrete architectural pain points and best practices for the different kinds of relationships existing between the players of our ISECOs. To improve on this situation, we started to analyze several of our ISECOs to derive suitable architectural requirements for their particular ecosystem settings. This will enable us to compare these requirements to the actual architectures to identify the actual pain points and best practices within the individual ISECOs. To achieve traceable results, we want to set our extraction method on a solid base, using a more reproducible, systematic and methodical procedure. In this paper we motivate the problem of a currently missing method to extract reusable architectural pain points and best practices from existing ISECOs. In our opinion such a method comprise

- an approach to model and analyze the collaboration among the participants from a technical perspective and
- an approach to derive reusable architectural design-patterns and anti-patterns for ISECOs from the analysis results.

We look for answers to the questions that arise from the two points in order to make our software product-lines fit for a future in which software architecture plays a more crucial role to coordinate internal software development.

978-1-4673-6449-2/13 © 2013 IEEE

PLEASE 2013, San Francisco, CA, USA

The remainder of this paper is structured as follows: We first motivate the problem by describing the ISECO of an existing software product-line within our company. We then point out the two challenges of extracting reusable architectural pain points and best practices from ISECOs in detail, providing in each case a formulation of the problem and open questions. We discuss related work, conclude the paper with an outlook on future work, and indicate when the problem can be considered as resolved.

II. Industrial Software Ecosystems

The ISECO we have started to analyze[1] offers a set of software products allowing to assemble and configure hardware devices and their topologies, in various domains, such as building, energy and industry. On the Siemens site, the ISECO comprises a keystone business unit that provides a software product-line platform as technological base, as well as business units from three divisions that develop the engineering software for the hardware devices of their respective domains upon that platform. Several other Siemens-internal business units provide various further plug-ins to represent the hardware devices to be available within the software platform. Moreover, the ISECO contains few external business units providing software components and infrastructure for the product-line platform, as well as external hardware vendors that also provide plug-ins for their hardware devices.

The relationships between most business units differ strongly, what results in different interaction patterns. For example, some business units work on the same source-code management system as the keystone business unit and have the same release schedule – whereas other business units are completely decoupled. Next, some business units develop .NET-based engineering software upon the platform – whereas other business units develop software representations of their hardware devices using an ISECO-specific scripting language. There are many further differences, like level of platform access, kind of contribution, integration or composition mechanisms, binding time, and so on. In the following we provide a simplified excerpt that describes the collaboration among two business units in more detail.

Scenario 1: The keystone business unit delivers a compiled version of its software product-line platform as well as the rights to exchange components using XML descriptor files to the business unit *B*. Then, *B* develops its products upon their own platform instance in a relative decoupled fashion. When the keystone player releases a new platform version, *B* has principally the free choice whether to retain the old version or adapt their product to the new version to leverage the new functionality.

Compared to SECOs, like the Apple iOS, ISECOs often comprise only a relative small number of business units. Besides, we have noticed a relative equable distribution of influence, power and control between most internal business units, so we have not such a strong and dominant player like Apple in the iOS SECO. This implies the need for consensus rather than normative approaches to achieve the coordination among the participants. Furthermore, most ISECOs were initially not designed as ISECO; rather they have gradually adapted ISECO-specific characteristics. As a result, the software architecture also explicitly shall manage the different players. Since the interaction patterns among most business units of our ISECOs differ strongly, so do the arising architectural pain points and best practices. Being not able to identify, understand and prioritize those pain points and best practices in detail hampers the possibility to counter or apply them in an adequate manner. This highlights the importance of a method for extracting reusable architectural pain points and best practices from existing ISECOs.

Our first observations base on several interviews with involved, well experienced software architects from Siemens Corporate Technology.

[1]To maintain confidentiality, our ISECO example needed to be anonymized. Still, the illustrated settings all stem from a real, existing ISECO within Siemens that we are analyzing at the moment.

III. Extract Pain Points and Best Practices

A method to extract reusable architectural pain points and best practices should comprise (A) an approach to model and analyze the collaboration among the participants from a technical perspective as well as (B) an approach to derive reusable architectural design-patterns and anti-patterns for ISECOs from the analysis results. In the following, we discuss the two points in detail, providing in each case a formulation of the problem, first results, and open questions. In order to make our software product-line architectures fit for the future, we look for answers to the highlighted questions that arise from the two approaches.

A. Model and Analyze the Collaboration Among Participants

To extract architectural pain points and best practices it is necessary to analyze their main source, which is the collaboration with and the collaboration among the involved business units. Thus, in our opinion a sound method for extracting pain points and best practices from existing ISECOs should base on a detailed modeling and analysis of the collaboration among the participants. Since the relationships between most business units differ strongly, resulting in different collaboration models, the emerging architectural pain points and best practices depend on the involved business units and their concrete interaction patterns. For this reason, the modeling and analysis approach should consider the properties of each single relationship.

ISECO participants gain intrinsic value when they contribute in the form of new products, providing or leveraging the technological base [3]. So the collaboration among the participants is strongly driven by their organizational motivations, in fact by the offer resulting from the joint ecosystem initiative. For this reason, it is important to understand the value proposition of the ISECO as a whole as well as the respective business goals of the participants [3]. By the discovery and analysis of the players' relationships that are relevant to fulfill

their objectives, the modeling and analysis approach should deliver a technical view on the participating business units, the relationships among them and their contributions. Thereby, based on the definition of ISECOs, the collaboration among the participants should be modeled through the exchange of information, resources and artifacts [8].

By the leverage of those data, we want to analyze each intermediate step of the collaboration among the participants from a technical perspective. In fact, we want to derive suitable architectural requirements for the respective relationship (as in the example below, the requirement to reflect critical usage restrictions to architectural guidelines), and in addition, we want to derive architectural pain points and best practices through the comparison of these requirements with the actual existing architecture. Based on our current experiences during analysis of the ISECO, a suitable approach should be able to model the following three characteristics in order to guide the analysis process: pure technical characteristics (like level of platform access, integration or composition mechanism, development environment, etc.), collaborative characteristics (like coupling of release schedules and development processes, communication, etc.), and business-driven characteristics, such as possible conflicts of aims. Analyzing each intermediate step of the collaboration under consideration of those characteristics might be an appealing approach for identifying architectural pain points and best practices for individual ISECOs, tackling the problems at its source and considering the individual problems and pitfalls of each single relationship.

For example, we analyzed *Scenario 1* with respect to the collaborative characteristic *communication*. Due to the low coupling of the business units and the resulting insufficiencies in direct communication, the architecture should compensate for this. Thus, a suitable architectural requirement should demand to reflect the architects intentions and critical usage restrictions to architectural guidelines. Furthermore, those guidelines should be spread in an easy accessible manner to the developers. Comparing this requirement with the actual situation, there is insufficient use of architectural mechanisms to compensate for the low coupling among the two players. As a consequence, the developers used the functionality beyond the planned and tested scope, which resulted in considerable performance problems. Regarding this individual relationship, we have identified as architectural pain point the lack of communication of critical usage restrictions through the use of architectural mechanisms.

In a nutshell, the envisaged modeling and analysis approach should comprise: (1) a detailed model of the collaboration, (2) technical, collaborative and business-driven characteristics to guide the analysis process, and (3) the derivation of architectural pain points and best practices through the comparison of suitable architectural requirements with the actual architecture, based on the modeled collaboration and guided by the characteristics. To enhance the traceability and reproducibility of this approach we look for answers to the following three open questions.

A1) Besides UML and SysML, what might be adequate notations to model the collaboration, the characteristics and the extracted architectural pain points and best practices, bearing in mind required tool support?

A2) How to link the modeled collaboration, the characteristics and the architectural pain points and best practices?

A3) What are further technical, collaborative and business-driven characteristics to guide the analysis process, or are there further types of characteristics valuable to explore?

B. Derive Reusable Architectural Design-Patterns for ISECOs

Our objective is the creation of a solid knowledge base as enabler for managing the architecture of our ISECOs. Thereby, the planed usage of the knowledge base is twofold. First, our existing software product-line architectures should be adjusted according to the paint points and best practices that stem from the individual transitions to ISECOs. Second, the derived pain points and best practices should be reused for our future ISECOs in order to design high-quality software architectures that explicitly manage the coordination among the involved business units. For achieving the second purpose, we must derive reusable architectural design-patterns and anti-patterns from the extracted best practices and architectural pain points.

As described in the section before, architectural pain points and best practices are extracted from single relationships of individual ISECOs; based on the modeled collaboration and guided by technical, collaborative and business-driven characteristics. Through the analysis of multiple ISECOs, we will get a pool of (relationship, pain point) and (relationship, best practice) tuples. By leveraging this pool, we want to derive reusable architectural design-patterns and anti-patterns for our ISECOs. In our opinion there are two possibilities to identify patterns. First, there might be recurring best practices and pain points that stem from several relationships, maybe from different ISECOs. Those best practices and pain points might be perfect candidates to carve them out as relationship-crosscutting architectural design-patterns and anti-patterns. Second, relationships that reappear in similar ways could be an indication to carve out the respective best practices and pain points as architectural design-pattern and anti-pattern for those kinds of relationships. Nevertheless, this is a challenging task since the relationships differ strongly, in particular with regard to the technical, collaborative and business-driven characteristics. Since we have only little experiences in this field of research, we are looking for answers to the following two questions.

B1) How to carve out reusable patterns and anti-patterns from specific pain points and best practices?

B2) How to document architectural design-pattern and anti-patterns, bearing in mind the strong influence relationships have on the adequacy of applying the patterns?

IV. Related Work

There are several promising approaches to model SECOs. Nevertheless, none of them were initially designed for the

special notion of ISECOs, and most of them do not adopt a technical perspective. Instead, they model SECOs form a business-driven perspective for purposes such as strategy assessment [9], industry taxonomy engineering [10], service systems for collaborative innovation [11] or management [12]. McGregor developed a method for analyzing software product-line ecosystems using scenarios [13]. However, also this method is rather business-driven and does not pursue the intention to extract architectural pain points and best practices. A promising approach to model the participating business units, the relationships among them and their contributions might be the use of software supply network (SSN) and product deployment context (PDC) diagrams [8], [2]. Nevertheless, the SSN and PDC diagrams we have become aware of neither deliver a pure technical view on ISECOs nor do they model the collaboration among the participants in a sufficient manner for deriving architectural pain points and best practices. Another approach with good prospects might be the visualization of SECOs based on the analysis of software repositories [14]. However, the information derived from software repositories also can only be a first step towards modeling the collaboration of the players in further detail.

Software architectural challenges that arise through the transition to SECOs already get attention in the SECO community. Although the yet discussed challenges are too generic to apply them directly for our purposes, they deliver an inspiration to identify the characteristics that guide our extraction process. As example, Bosch et al. [15] show the need to consider the *release schedules* of the participants as well as *integration* and *composition mechanisms*, whereas the research about interface translucence as enabler for scalable collaboration [6] motivates the importance of *communication* issues. Furthermore, the *level of platform access* is discussed by the evaluation of the architectural openness of mobile software platforms [7].

V. Conclusion and Future Work

We have started to analyze several Siemens-internal software product-lines that move towards ISECOs in order to make them fit for the future, where software architecture plays a more crucial role to manage the different players. Therefore, we have envisaged a method to extract reusable architectural pain points and best practices from our existing ISECOs. To get more reproducible, traceable and reusable results we have highlighted arising questions that still need to be answered.

In a long term, we consider the challenge resolved when we have extracted a set of reusable architectural design-patterns and anti-patterns that can be applied to design high-quality software architectures for our prospective ISECOs. In a short term, our objective is to identify architectural pain points and best practices from our current software product-lines that move towards ISECOs in order to adapt their architectures to the special needs that arise through the transition. By the use of interviews, we plan to evaluate the success through the comparison of the extracted results with those pain points and best practices that are already known by the respective architects. We want to show that, compared to the available information, the extracted results are far better documented, more traceable, and more complete to support architectural decision making. Finally, we plan to monitor our ISECOs in order to analyze if their adjusted architectures achieve the coordination among the players in a more sufficient manner.

Acknowledgment

The authors would like to thank the software architects from Siemens Corporate Technology for their contributions and their efforts during the interviews. Their expertise has positively influenced the quality of this research.

References

[1] J. Bosch, "From software product lines to software ecosystems," in *Proc. of the 13th Int. Software Product Line Conf.*, ser. SPLC '09. Pittsburgh, PA, USA: Carnegie Mellon University, 2009, pp. 111–119.

[2] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st Int. Conf. on*, may 2009, pp. 187 –190.

[3] P. R. J. Campbell and F. Ahmed, "A three-dimensional view of software ecosystems," in *Proc. of the Fourth European Conf. on Software Architecture: Companion Volume*, ser. ECSA '10. New York, NY, USA: ACM, 2010, pp. 81–84.

[4] J. Bosch, "Architecture challenges for software ecosystems," in *Proc. of the Fourth European Conf. on Software Architecture: Companion Volume*, ser. ECSA '10. New York, NY, USA: ACM, 2010, pp. 93–95.

[5] R. Pereira, C. Maria, and L. Werner, "A Proposal for Software Ecosystems Engineering," in *Proc. of the Workshop on Software Ecosystems 2011*, 2011, pp. 40–51.

[6] M. Cataldo and J. D. Herbsleb, "Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration," in *Proc. of the Fourth European Conf. on Software Architecture: Companion Volume*, ser. ECSA '10. New York, NY, USA: ACM, 2010, pp. 65–72.

[7] M. Anvaari and S. Jansen, "Evaluating architectural openness in mobile software platforms," in *Proc. of the Fourth European Conf. on Software Architecture: Companion Volume*, ser. ECSA '10. New York, NY, USA: ACM, 2010, pp. 85–92.

[8] V. Boucharas, S. Jansen, and S. Brinkkemper, "Formalizing software ecosystem modeling," in *Proc. of the 1st int. workshop on Open component ecosystems*, ser. IWOCE '09. New York, NY, USA: ACM, 2009, pp. 41–50.

[9] I. van den Berk, S. Jansen, and L. Luinenburg, "Software ecosystems: a software ecosystem strategy assessment model," in *Proc. of the Fourth European Conf. on Software Architecture: Companion Volume*, ser. ECSA '10. New York, NY, USA: ACM, 2010, pp. 127–134.

[10] I. Hunink, R. van Erk, S. Jansen, and S. Brinkkemper, "Industry taxonomy engineering: the case of the european software ecosystem," in *Proc. of the Fourth European Conf. on Software Architecture: Companion Volume*, ser. ECSA '10. New York, NY, USA: ACM, 2010, pp. 111–118.

[11] T. Janner, C. Schroth, and B. Schmid, "Modelling service systems for collaborative innovation in the enterprise software industry - the st. gallen media reference model applied," in *Services Computing, 2008. SCC '08. IEEE Int. Conf. on*, vol. 2, july 2008, pp. 145 –152.

[12] B. Iyer, C. Lee, and N. Venkatraman, "Managing in a small world ecosystem: Some lessons from the software sector," *California Management Review*, vol. 48, no. 3, pp. 28–47, 2006.

[13] J. D. McGregor, "A method for analyzing software product line ecosystems," in *Proc. of the Fourth European Conf. on Software Architecture: Companion Volume*, ser. ECSA '10. New York, NY, USA: ACM, 2010, pp. 73–80.

[14] M. Lungu, M. Lanza, T. Gîrba, and R. Robbes, "The small project observatory: Visualizing software ecosystems," *Science of Computer Programming*, vol. 75, no. 4, pp. 264 – 275, 2010, experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008).

[15] J. Bosch and P. Bosch-Sijtsema, "From integration to composition: On the impact of software product lines, global development and ecosystems," *J. Syst. Softw.*, vol. 83, no. 1, pp. 67–76, Jan. 2010.