Architecture Challenges for Internal Software Ecosystems: A Large-Scale Industry Case Study

Klaus-Benedikt Schultis†*, Christoph Elsner†, Daniel Lohmann*

†Corporate Technology, Siemens AG, Wladimirstr. 3, 91058 Erlangen
*Computer Science 4, Friedrich-Alexander University, Martensstr. 1, 91058 Erlangen
klaus-benedikt.schultis.ext@siemens.com, schultis@cs.fau.de
christoph.elsner@siemens.com
lohmann@cs.fau.de

Abstract: The idea of software ecosystems encourages organizations to open software projects for external businesses, governing the cross-organizational development by architectural and other measures. Even within a single organization, this paradigm can be of high value for large-scale decentralized software projects that involve various internal, yet self-contained organizational units. However, this intra-organizational decentralization causes architecture challenges that must be understood to reason about suitable architectural measures.

In our FSE paper, we present an in-depth case study on collaboration and architecture challenges in two of these large-scale software projects at Siemens. We performed a total of 46 hours of semi-structured interviews with 17 leading software architects from all involved organizational units. Our major findings are: (1) three collaboration models on a continuum that ranges from high to low coupling, (2) a classification of architecture challenges, together with (3) a qualitative and quantitative exposure of recurring issues along each collaboration model.

We identified compliant software development as the core challenge. It targets cross-cutting regulations and architecture governance to assist and check for compliance. In addition to our study results, we outline a framework along with tool support that allows to manage these regulations and their violations throughout the life cycle.

Internal Software Ecosystems

We have investigated two large-scale software projects (about 500 and 950 developers) within Siemens [SEL14]. These projects involve a set of internal organizational units that are self-contained profit centers with own business objectives, organizational independent with own product management, and have to a wide extent autonomous processes and software-engineering life cycles. Thus, the view on the organizational structure moves from strict hierarchies towards more decentralized topologies. We define those systems as internal software ecosystems (ISECOs) [SEL14]. ISECOs comprise a keystone organizational unit that provides a platform and multiple client organizational units that build applications upon it. The keystone acts in a creative role but does not have power to direct.

Our talk is laid out as follows: First, we outline our study results. Second, we present our current work, a framework that addresses the compliant software development challenge.
Case Study Results: We present the three identified collaboration models and a classification of architecture challenges that become particularly crucial due to collaboration.

Each collaboration model is defined by the type of client. The first model comprises the keystone and core clients. These units set the original scope, develop a set of products for a shared market and collaborate based on a strategic reuse approach. The second model involves consumer clients. They are outside of the initial scope and independently develop products for different markets. The units collaborate based on a platform reuse approach without strong strategic coupling. Within the third model, the platform scope was explicitly broadened to additionally involve extended core clients. They develop products for partially overlapping markets. As for core clients, they follow a strategic reuse approach but the broadened scope requires a more decoupled software development procedure.

Our classification of architecture challenges includes platform openness strategy, composition of decentralized developed software, preservation of the organizational-units independence, guarantee of software qualities across the ecosystem and compliance to architectural intentions and cross-cutting guidelines. The latter one turned out as the key challenge and is called the Compliant Software Development challenge.

Compliant Software Development: Compliant software development targets issues that need to be regulated across organizational units, for instance architecture dependencies or quality patterns. It necessitates the establishment and execution of architecture governance to assist and check for compliance. This requires to decide on topics that are most relevant for the ecosystem, to decide on processes and measures to execute governance and to define roles and responsibilities across the ecosystem that support the overall approach.

The decentralization with independent spheres of authority implies a range of hurdles that complicate the situation. For instance, ISECOs represent a consensus-based environment where the core organizational units are on a par regarding their influence. As guidelines often imply varying costs and benefits coming to mutual agreements can take significant amounts of time. As another example, guidelines often evolve over time. Thus, adopting a new guideline generally implicates a range of already existing violations that need to handled appropriately. In most cases, their resolution cannot be the generally viable way as this would require multiple organizational units to change working applications.

In addition to our case study results published at FSE [SEL14], we will outline our current work: An integrated framework along with tool support that addresses all identified hurdles and that allows to manage guidelines and their violations throughout the life cycle, from guideline definition to violation detection to violation management.

References