On-demand Generation of Views to Support Online Evolution of Software Product Lines

Danny Weyns, Bartosz Michalik, Alexander Helleboogh, Nelis Boucke
DistriNet Labs, Katholieke Universiteit Leuven, Belgium

The description of the software architecture is considered as a key artifact in a system's design and evolution. Architectural views provide the means to express different perspectives on the system, capturing the stakeholder concerns. However, in practice, assuring conformance between architecture and code turns out to be very difficult to achieve. As the software evolves, the architectural models diverge from the actual system and the documentation gets obsolete. As pointed out by Shaw and Clements: "lack of conformance dooms an architecture to irrelevance as the code sets out on its own independent trajectory" [1]. In order to tackle this fundamental problem, we need to rethink the way we manage software architecture descriptions.

In a joint effort between Egemin NV [2], an industrial manufacturer of logistic systems, and DistriNet Labs, we faced the problem of lacking proper architecture descriptions in the context of the evolution of a software product line of logistic systems. A logistic system consists of a set of logistic subsystems that are built from a common set of core assets that can be customized to the client's needs. Typical logistic subsystems include a warehouse management system and control software for various transportation subsystems such as automated guided vehicles, cranes, and conveyers. Egemin's software product line comprises over 200 deployed logistic systems. During their lifespan (typically 10+ years), logistic systems obviously have to evolve. For example, the software of one of the subsystems is upgraded to improve performance, or a particular factory introduces a new conveyer belt and its control software needs to be integrated with the existing logistic system. Architects and maintainers at Egemin face various problems with such evolution scenarios.

Besides the intrinsic complexity of the logistic systems, the problems are caused by various concerns. (1) Each logistic system is unique in its composition and client-specific extensions. However, core assets cannot be composed arbitrary. The constraints between the assets introduce a complex management problem. (2) The logistic systems comprise a lot of legacy software components. Documentation of the deployed systems is often incomplete or outdated. The lack of documentation and detailed knowledge makes it difficult to trace the complex dependencies between the installed subsystems. (3) Logistic systems typically operate 24/7. Therefore, evolving the system with minimal interruption is crucial. Determining which processes have to be shut down and (re-)started and when is a complex problem. Restarting an incorrect configuration may compromise the consistency of the system. (4) The lack of detailed documentation about deployed systems leads to the ad-hoc update practices which are inefficient and error prone. Faulty updates increase maintenance costs, harm the company's reputation, or even worse, they may cause serious damage to industrial installations.

To tackle the problems, we developed an approach in which architecture models are dynamically generated from the deployed system sources, guaranteeing consistency between the models and the actual system. The approach comprises two complementary parts: an evolution viewpoint and a supporting tool to generate on-demand architecture models based on the viewpoint. The evolution viewpoint captures the various evolution concerns as explained above. The viewpoint defines four model kinds. The deployment models allow stakeholders to browse the currently deployed system (as-is) and the future version of the system (to-be) respectively. The update procedure model shows the tasks that have to be performed to evolve a system with minimal interruption of services (stop/start processes, add/remove/replace

resources). This model is derived from an analysis of the differences between the as-is and to-be system determining which resources have to be added and removed, and which processes have to be shut down and (re-)started. Finally, the update inconsistencies model deals with correctness, showing inconsistencies of the updated logistic system. This model is derived from an analysis of the dependencies between resources and version inconsistencies.

The evolution viewpoint is supported with a tool that allows generating on-demand views for deployed systems. Central in this tool is an architecture repository that reifies the necessary architectural knowledge of the system. The architecture repository can be populated with knowledge harvested from system sources, including deployed assemblies, versioning information, configuration files, and installation bundles. The repository is connected to a query engine that allows dynamic extraction of the architecture models supported by the evolution viewpoint. The architecture models guide system maintainers by listing the concrete tasks they need to perform when upgrading a system, and by showing inconsistencies when they fail to do so. We have used the viewpoint and the supporting tool to manage the evolution of logistic systems at Egemin. We evaluated a total of 68 updates of industrial logistic systems performed by 17 professionals, half of them with Egemin's traditional update approach, the other half with the tool. The results demonstrate that 44% of the updates with the traditional approach contain errors, while all the updates with the tool were performed correctly. Furthermore, the results show that with the traditional approach 58% of the process shutdowns were unnecessary, while there were only 7% redundant process shutdowns with the tool.

In the talk, we will pinpoint the problems with online evolution of software systems. We elaborate on the evolution viewpoint and explain how the viewpoint deals with the various stakeholder concerns. We zoom in on the tool for generating on-demand views. We explain how architecture knowledge is harvested and represented, and how this knowledge is used to generate the architecture models dynamically. We discuss our experiences with the approach and conclude with lessons learned and recommendations for future work.

[1] M. Shaw and P. Clements, The Golden Age of Software Architecture, IEEE Software, vol. 23, 2006
[2] Egemin NV, http://www.egemin.com/

Bio's

Danny Weyns is a post-doctoral researcher at DistriNet Labs of the Katholieke Universiteit Leuven, funded by the Research Foundation Flanders. His main research interests are in software architecture, self-adaptive systems, multiagent systems, and middleware for decentralized systems.

Bartosz Michalik is a doctoral student at DistriNet Labs of the Katholieke Universiteit Leuven. The goal of his research is to develop a systematic approach for evolving software product lines that are in use. Key concerns of the approach are correctness and minimal interruption of services.

Alexander Helleboogh is a postdoctoral researcher at DistriNet labs. His research interest is software architecture, more in particular: design, documentation and evaluation of software architectures, software product line architectures and architectures for autonomous, adaptive and decentralized systems. He investigates these topics in a real-world setting, in projects with industrial partners.

Nelis Boucké is a post-doctoral researcher at DistriNet labs. His research focuses on architectures of large distributed systems, including software quality, architecture evaluation, tools to support software architecture practice, architectural design, documentation and reconstruction, and software product lines. He works on practical solutions in collaboration with companies and large organizations.