Evolving Software

Evolutionary Architecting of Software-Intensive Systems

by Jakob Axelsson

Most industrial software-intensive systems have a very long life span, and undergo many changes after their initial conception. To ensure that they maintain the desired system-level properties, it is essential that their architecture is also updated to reflect the added functionality. To this end, we have investigated how Evolutionary Architecting should be carried out in order to provide a strong basis for a system, not only at conception but throughout its life. The results are concrete and based on empirical findings, and include both a new state-of-the-art process description and a light-weight maturity evaluation method.

Embedded software is a critical component to be considered in the development of many new products such as cars, airplanes, industrial robots, and telecommunications systems. There is great scope for innovation within software, and innovative software can keep a product competitive. Often, a product line approach is used, where many variants are spawned off the same platform over decades. However, with separate teams working on different products and modifying the platform to suit their needs, coordination between them becomes essential to ensure that the platform keeps its integrity and stays useful for future needs. It is a central role of the architects to maintain this integrity and provide technical prerequisites for the different design teams.

Plenty of research has been done on software and systems architecture, but much of it focuses on the development of new systems. However, our empirical research shows that evolution of existing products is much more common than new development. Therefore, through interviews and surveys with around 100 active architects at a dozen companies, we have tried to identify needs and challenges within architecting to support the evolutionary development of embedded software. Many of the organizations suffer from unclear processes and responsibility for architecture, and lack methods to evaluate the business value when making architectural decisions. In short, the companies rely on individuals instead of on processes and methods.

These issues are typical signs of immaturity in architecting organizations. To identify where improvements can be made, it is useful to have a way of assessing an organization’s maturity. Several well-known maturity models exist, such as Capability Maturity Model Integration (CMMI), which captures the best practices for development in general. However, it has a few drawbacks when applied to architecting. Firstly, it is heavy to use for a small team, and secondly, it emphasizes formal appraisals by external assessors, whereas architects are more in need of self-assessment. As an alternative, we have developed an Evolutionary Architecting Maturity Model (EAMM) which is an instantiation and simplification of CMMI. The appraisals in EAMM are extremely simple, with just 53 questions taking about one hour to complete, but it still pinpoints where the organization should focus on making improvements.

We have applied EAMM at a handful of industrial companies, and a typical company ranks between level two and three on the five-level maturity scale used by EAMM and CMMI. Specific areas that are often weak include management of architectural requirements, quality assurance, verification, measurement and analysis, and risk management. It is
Automated Synthesis of CONNECTors to support Software Evolution

by Amel Bennaceur, Paola Inverardi, Valérie Issarny and Romina Spalazzese

Software evolution is a hot topic in today’s distributed software systems, which are characterized by an increasing level of heterogeneity and dynamism. For instance, updated versions of software are continually made available, new applications are created and dynamically join the networked environment, old or broken applications are unavailable or removed, and new needs emerge.

Evolution management in this environment raises the need for novel computing paradigms able to ensure continuous run-time composition and interoperability of heterogeneous and dynamically available applications in an automated manner.

Although in principle they could interact since they have compatible (ie complementary) functionalities, heterogeneous applications populating this landscape can be characterized by discrepancies that may undermine their ability to seamlessly interoperate (ie communicate and coordinate). Discrepancies include incompatible interaction protocols and/or different interfaces meaning different actions and/or data models.

Examples of heterogeneous applications are shown in Figure 1(a) where the light blue entities show independently developed clients and servers for the organization of trips. The server exhibits separate searches for flights and hotels but the client would expect to have a single search with all the information.

To enable perpetual and automated composition and interoperability, the ICT FET IP European project CONNECT pursues the automated synthesis of CONNECTors that overcome interoperability barriers. This responds to the evolution of functionalities provided to and required from the networked environment of today’s software systems. Automatically synthesized CONNECTors are concrete emergent entities that mediate the application discrepancies, ie translate and coordinate mismatching interaction protocols, actions, and/or data models, letting applications interact effectively. Examples of CONNECTors are illustrated in Figure 1(b) -green entities.

Automated Synthesis of CONNECTors

We have developed a theory for the automated synthesis of application-layer CONNECTors in order to formally ground and automatically reason on the interoperability problem among heterogeneous applications. The theory has also served as a principled basis for the development of a prototype tool that automatically generates CONNECTors.

A Theory of CONNECTors

Our theory for the automated synthesis of CONNECTors assumes the descriptions of two Networked Systems (NSs) focusing on their application-layer pro-

striking that many companies lack a written description of their architecting process, and this is a fundamental flaw if one wants to work systematically with process improvement. To address this, we have devised an Evolutionary Architecting Process (EAP) which can be used as a starting point for a company to improve its architecting practices.

The top level EAP process is shown in Figure 1. The process starts with a change request from the product planners. These requests are planned and prioritized and placed in a backlog. The architects then start a task by analyzing these requirements to find out which of these are significant for the architecture. This involves close interaction with various stakeholders, and the final requirements are stored in a database for use in future iterations. Based on the requirements and on quality attributes of the architecture, architectural solutions are developed and evaluated. From the selected option, prerequisites are generated that can be used by system developers for subsequent detailed design and implementation.

Quality attributes play an important role, and include factors such as flexibility, scalability, safety, and understandability. Maintaining these attributes, while dealing with the specific requirements of a particular function, is at the core of architecting. Sometimes it is also necessary to induce specific activities called refactorings that do not alter the functionality of the system but change its internal structure to improve the quality attributes.

With the development of the EAMM and EAP, we provide a new foundation for the kind of architecting that is most important for the software-intensive systems industry. In the future, we will further investigate and refine these methods in close co-operation with partner companies.

Link: http://www.mrtc.mdh.se/index.php?choice=staff&id=0148

Please contact:
Jakob Axelsson
SICS and Mälardalen University, Sweden
Tel: +46 72 734 29 52
E-mail: jakob.axelsson@sics.se