Towards Model-Driven Evolution of Performance Critical Business Information Systems to Cloud Computing Architectures

Steffen Becker
Model-Driven Software Engineering
Heinz Nixdorf Institute, Paderborn
steffen.becker@upb.de

Matthias Tichy
Software Engineering,
Computer Science and Engineering,
Chalmers University of Technology, Göteborg
mtt@tichy.de

Abstract Migrating legacy applications to todays emerging cloud infrastructures is still challenging. In this paper, we sketch an approach, that combines reverse engineering and performance analyses for applications evolving to the cloud.

1 Motivation

Long-living software systems undergo permanent evolution during their lifetime. These evolutions are triggered by a changing system context (system usage and technology stacks) and/or changing system requirements. Examples of such context or requirements changes include evolving functional or non-functional requirements, e.g., the capability of the system to deal with increasing system workload. The latter is often a consequence of opening the access to existing systems over the Internet, e.g., for the integration of the systems into novel service compositions.

Especially the latter scenario imposes major challenges for long-living software systems as their software architecture is often not prepared to deal with a high system load. Cloud computing offers a new technology which is supposed to address this issue by selling almost unlimited amounts of compute or storage resources. In order to utilize this new technological environment which is able to cope with the mentioned changing environment, usually long-living software systems have to be migrated to new environments. Often this implies major changes to the system structure for which no systematic engineering process is available today. This lack of understanding can lead to high risks or even project failures. For example, the migration of SAP R3 to SAP's ByDesign SOA solution almost failed as the legacy system architecture was unable to work properly in the new environment [1]. The consequence was that the performance was unacceptably low and the system went through a costly redesign process deferring product release by approximately 3 years.

The current state of the art for this kind of problems is twofold. On the one hand, classical architectural styles and patterns document best practices for systems built in the past. As an example, consider the three-tier layered architecture style often used successfully for business information systems. On the other hand, we can find new architectural styles that enable the efficient use of the almost unlimited computational resources in the cloud. Here, we find for example the multi-tenant architectural style (or SPOSAD style) that allows massive replication of the business logic which is enabled by a smart physical data distribution. Their aim is to support various load situations efficiently.

However, there is currently a lack of approaches that systematically bridge these two types of architectures, i.e., there is no systematic engineering support for migrating from one architecture type to the other to gain the benefits provided by the latter. A special focus of such systematic engineering support has to be the performance and scalability gained by migrating to the cloud architectural styles as these are the main drivers for the migration needs.

2 Solution Sketch

In this paper, we propose a systematic engineering approach to migrate long-living legacy software systems (which we assume will follow one of the classical architectures) to the conceptual and technological context provided by the emerging cloud architectures. We focus on supporting performance and scalability predictions.

In general, our proposed process is as follows (cf. Figure 1). First, existing systems have to be *reverse engineered* to obtain a performance prediction

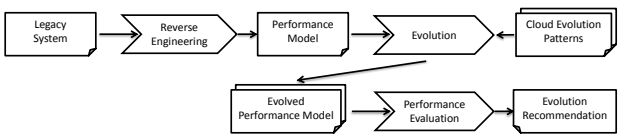

Figure 1: Solution Process Outline

[1] http://www.v3.co.uk/v3-uk/news/1970547/sap-update-business-bydesign-plans

model. These models contain static aspects, e.g., components and their connections, behavior, e.g., control flow and resource demands, and deployment aspects. Second, the software architect has to *select a set of potential target architecture styles or patterns* which have to be appropriately formalized. For example, the architect plans to evaluate the impact of migrating the classical system architecture to Map-Reduce or to the SPOSAD [Koz11] style and, thus, he semi-automatically adapts the reverse engineered performance prediction models by the selected architectural styles. Third, the performance of the target architectures is *evaluated* to get a final ranking and to come to a recommendation for the migration. Finally, based on the analyzed target architecture, the system's implementation has to be *adapted*. Note that some (most prominently the final step) of the illustrated steps cannot and should not be fully automated, i.e., human intervention has to be supported for all these steps.

3 State of the Art

The major foundations for the sketched process are already in place, i.a, software architectural patterns, software performance engineering, architecture evolution, and model transformations. We will sketch some selected state of the art in the following.

Software Architecture Patterns In the following, we discuss some examples of typical architectural styles or patterns used to implement today's Cloud-based applications. One example is the Map-Reduce style used by the Google search engine. It has been implemented for example in supporting frameworks like Hadoop [2]. Another example is a tuple oriented abstraction where data portions to be worked on are put into a tuple space which several nodes access and process accordingly. The SPOSAD style [Koz11] uses classical application tiers but ensures that the data access layer can be scaled by an appropriate physical distribution of the data.

Software Performance Engineering Software Performance Engineering aims at predicting the performance of software systems before their implementation to avoid costly redesigns. For Cloud systems the data flow often is the limiting factor as the data needs to be distributed among the manifold server available in the cloud. While all performance engineering approaches support the specification of control flow aspects, the handling of data flow varies significantly in its level of detail.

Hamlet [Ham09] proposed one of the first models where the data sent from one component to the next had an impact on the called component's performance. Drawing upon that, for example the Palladio component model [BKR09] for business information systems consider data flow impact on loop iteration counts, branch probabilities or parameters passed on to called required services. We plan to extend these models to reflect the cloud requirements in our research.

Architecture Evolution The CloudMIG approach [FH10] is closely related to our sketched process. Frey and Hasselbring develop an approach for the migration of software systems to cloud systems. The approach uses formal constraints to determine whether the target software satisfies or violates constraints imposed by the cloud environment. However, CloudMIG does not offer performance analyses.

4 Conclusion and Future Work

Many legacy applications are currently and will be even more in the future migrated to cloud architectures. To reap the benefits from cloud architectures and to avoid costly failures, this process should be appropriately supported. We sketched a software engineering process which supports this migration by employing formal patterns for cloud architectures and performing performance estimations.

While the foundations for the major parts of the proposed process are already laid by the software engineering community, there exists no related work which appropriately supports the outlined steps. Consequently, we will develop an implementation of the sketched process in the future.

References

[BKR09] Steffen Becker, Heiko Koziolek, and Ralf Reussner. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82:3–22, 2009.

[FH10] Sören Frey and Wilhelm Hasselbring. Model-based migration of legacy software systems to scalable and resource-efficient cloud-based applications: The cloudmig approach. In *Cloud Computing 2010: Proceedings of the 1st International Conference on Cloud Computing, GRIDs, and Virtualization*, 2010.

[Ham09] Dick Hamlet. Tools and experiments supporting a testing-based theory of component composition. *ACM Trans. Softw. Eng. Methodol.*, 18:12:1–12:41, June 2009.

[Koz11] Heiko Koziolek. The SPOSAD Architectural Style for Multi-tenant Software Applications. In *Proc. 9th Working IEEE/IFIP Conf. on Software Architecture (WICSA'11), Workshop on Architecting Cloud Computing Applications and Systems*, pages 320–327. IEEE, July 2011.

[2] `http://hadoop.apache.org/`