Model-Based Migration of Legacy Software Systems to Scalable and Resource-Efficient Cloud-Based Applications: The CloudMIG Approach

Sören Frey and Wilhelm Hasselbring
Software Engineering Group
University of Kiel
24118 Kiel, Germany
{*sfr, wha*}*@informatik.uni-kiel.de*

***Abstract*—The paper describes the model-based approach CloudMIG. Cloud computing supplies software, platforms, and infrastructures as a service (SaaS, PaaS, and IaaS, respectively) over a network connection. Cloud providers frequently offer the services according to the utility computing paradigm. Therefore, cloud computing provides means for reducing over- and under-provisioning through enabling a highly flexible resource allocation. Running an existing software system on a cloud computing basis usually involves extensive reengineering activities during the migration. Current migration approaches suffer from several shortcomings. For example, they are often limited to specific cloud environments or do not provide automated support for the alignment with a cloud environment. We present our model-based approach CloudMIG which addresses these shortcomings. It aims at supporting SaaS providers to semi-automatically migrate existing enterprise software systems to scalable and resource-efficient PaaS and IaaS-based applications.**

***Keywords*-Approach CloudMIG; Cloud Computing; Model-based software migration to cloud-based applications; Resource-efficient cloud-based applications.**

I. Introduction

Most enterprise applications' workload underlies substantial variations over time. For example, user behavior tends to be daytime-dependent or media coverage can lead to rapidly increasing popularity of provided services. These variations often result in over- or under-provisioning of data center resources (e.g., #CPUs or storage capacity). Cloud computing provides means for reducing over- and under-provisioning through supplying elastic services. Thereby, the conformance with contractually agreed service level agreements (SLAs) has to be ensured [1]. Considering legacy software systems, is there a way established enterprise applications can benefit from present cloud computing technologies? For reasoning about this issue, it is useful to clarify the main participants in providing and consuming cloud computing services. According to [2], three different roles can be distinguished. SaaS providers (cloud users) offer software services which are being utilized by SaaS users. For this purpose, the SaaS providers may build upon services offered by cloud providers (cloud vendors). In the following, we will employ the terms SaaS user, SaaS provider, and cloud provider.

Newly developed enterprise software may easily be designed for utilizing cloud computing technologies in a greenfield project. Though, SaaS providers may also consider to grant responsibility of operation and maintenance tasks to a cloud provider for an already existing software system. Running established enterprise software on a cloud computing basis usually involves extensive reengineering activities during the migration. Nevertheless, instead of recreating the functionalities of an established software system from scratch for being compatible with a selected cloud provider's environment, a migration enables the SaaS provider to reuse substantial parts of a system. The number of system parts which might be migrated is dependent on the weighting of several parameters in a specific migration project. For example, implications concerning the performance or structural quality metrics regarding the resulting software architecture can be taken into account. Furthermore, aligning a software system to a cloud environment's special properties during the migration process has the potential to increase the software system's efficiency. For example, a reengineer could decide to prefer utilization of certain resources according to their pricing. Considering such kinds of favorable resource utilization and a cloud environment's specific scalability mechanisms can improve overall resource efficiency (e.g., according to the aforementioned prioritization) and scalability. However, there are several major obstacles which can impede such migration projects. Current approaches are often limited to specific cloud environments or do not provide automated support for the alignment with a cloud environment, for instance. In this work, we propose our model-based approach CloudMIG which addresses these shortcomings and focuses on the SaaS provider perspective. The semi-automated approach aims at assisting reengineers in migrating existing enterprise software systems to scalable and resource-efficient PaaS and IaaS-based applications (e.g., see [3]).

The remainder of the paper is structured as follows: Section II describes the shortcomings of existing approaches. Our approach CloudMIG is presented in Section III, before Section IV draws the conclusions and outlines the future work.

II. Current Shortcomings

Migrating typical enterprise software to a cloud-based application usually implies an architectural restructuring step. However, knowledge about the internal structure of the existing software system is often insufficient and therefore an architectural model has to be reconstructed first. The architectural model serves as a starting point for restructuring activities towards a cloud-compatible target architecture, which most often has to be created manually. This often is not an easy task, as construction of the advanced architecture usually presumes profound comprehension of the existing one. Furthermore, the target architecture must comply with the specific cloud environment's offered resources and imposed constraints, for example application frameworks and limitations of programming interfaces, respectively. A mapping model that describes the relationships between system parts of the status quo and the target architecture is needed as well. Future workload in combination with the target architecture arrangement will determine resource utilization of the cloud environment during operation. As most cloud providers follow the paradigm of utility computing and therefore charge resource utilization on a pay-as-you-use basis, the arrangement of the target architecture has a direct impact on the operational costs. Moreover, running an application in a cloud environment does not solve scalability issues per se. For example, an IaaS-based application often needs to have built-in self-adaptive capabilities for leveraging a cloud environment's elasticity.

Shortcomings of today's migration projects from typical enterprise software to cloud-based applications can therefore be summarized as follows:

- ***S1*** **Applicability:** Solutions for migrating enterprise software to cloud-based applications are limited to particular cloud providers.
- ***S2*** **Level of automation:** The target architecture and the mapping model often have to be built entirely manual. Additionally, the target architecture's violations against the cloud environment's constraints are not identified automatically at design time.
- ***S3*** **Resource efficiency:** Various migrated software systems are not designed to be resource-efficient and do not leverage the cloud environments' elasticity, because even transfering an established application to a new cloud environment is a challenging task itself. Furthermore, means for evaluating a target architecture's dynamic resource utilization at design time are most often inadequate.
- ***S4*** **Scalability**: Automated support for evaluating a target architecture's scalability at design time is rare in the cloud computing context.

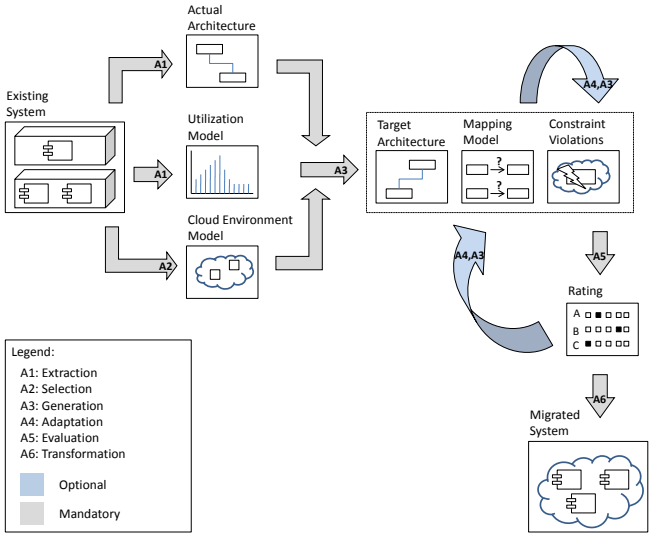

Figure 1. CloudMIG Overview.

III. The Approach CloudMIG

CloudMIG is composed of six activities for migrating an enterprise system to a cloud environment. It provides model-driven generation of considerable parts of the system's target architecture. Feedback loops allow for further alignment with the specific cloud environment's properties and foster resource efficiency and scalability on an architectural level. Fig. 1 outlines the approach. Its activities (A1-A6) are briefly described in the following including the involved models.

A. Activity A1 - Extraction

CloudMIG aims at the migration of established enterprise applications. Usually, the architecture of software systems tends to erode over time. Therefore, initially envisioned architectures frequently diverge from actual implementations. The knowledge about the internal structure is often incomplete, erroneous, or even missing. As CloudMIG utilizes a model transformation during generation of its target architecture (cf. A3), a representation of the software system's actual architecture has to be available first. Concerning this issue, an appropriate model is extracted by means of a software architecture reconstruction methodology. We propose OMG's Knowledge Discovery Meta-Model[1] (KDM) for building a suitable meta-model.

For leveraging the commonly applied utility computing paradigm, the target architecture has to be laid out resource-efficient and elastic. Therefore, CloudMIG includes the extraction of an established software system's utilization model acting as a starting point. The utilization model (resp. its meta-model) includes statistical properties concerning user behavior like service invocation rates over time or average submitted datagram sizes per request. Relevant

[1]http://www.omg.org/spec/KDM/ (Accessed August 16, 2010)

information can be retrieved from various sources. For example, considering log files or instrumenting the given system with our tool Kieker[2] for setting up a monitoring step constitute possible techniques. Furthermore, the utilization model contains application-inherent information related to proportional resource consumption. Metrics of interest could be a method's cyclomatic complexity or memory footprint. We propose OMG's Software Metrics Meta-Model[3] (SMM) as a foundation for building the related meta-model.

B. Activity A2 - Selection

Common properties of different cloud environments are described in a cloud environment meta-model. Selecting a cloud provider specific environment as a target platform for the migration activities therefore implies the selection of a specific instance of this meta-model. For example, this meta-model comprises entities like VM instances or worker threads for IaaS and PaaS-based cloud environments, respectively. As a result, for every cloud environment which shall be targeted with CloudMIG a corresponding meta-model instance has to be created once beforehand. Transformation rules define possible relationships to the architecture meta-model.

We plan to attach further information related to scalability issues to the included entities, which can be configured by the reengineer in activity A4. For example, VM instances could provide hooks for controlling their lifetime dependent on dynamic resource utilization during runtime. Furthermore, the meta-model includes constraints imposed by cloud environments restricting the reengineering activities. For example, the opening of sockets, the manual spawning of threads, or the access to the file system are often constrained.

C. Activity A3 - Generation

The generation activity produces three artefacts, namely a target architecture, a mapping model, and a model characterizing the target architecture's violations of the cloud environment constraints. The latter lists the features of the target architecture which are non-conform with the cloud environment's specification. These constraint violations explicitly highlight the target architecture's parts which have to be redesigned manually by the reengineer (cf. A6). The mapping model assigns elements from the actual architecture to those included in the target architecture. Finally, the target architecture constitutes a primary artefact. It is realized as an instance of the cloud environment meta-model. We propose three phases P1-P3 for the generation of the target architecture.

P1 - Model transformation: The phase P1 produces an initial assignment from features of the existing architecture to cloud-specific features available in the cloud environment model. The initial assignment is created applying a model-to-model transformation according to the transformation rules included in the cloud environment model (cf. activity A2).

P2 - Configuration: The phase P2 serves as a configuration of the algorithm used for obtaining a resource-efficient feature allocation in phase P3. During P2, a reengineer may adjust rules and assertions for heuristic computation (cf. P3). A rule could be formulated like the following examples: "Distribute the five most frequently used services to own virtual machines" or "The server methods responsible for at least 10% of overall consumption of the CPU time shall be moved to client side components if they do not need access to the database". An exemplary assertion could be: "An existing component must not be divided in more than 3 resulting components". It is intended to provide a set of default rules and assertions. In addition to that, the reengineer will be given the possibility to modify them either via altering the regarding numerical values or applying a corresponding DSL. In both cases, the rules and assertions have to be prioritized after their selection. Hereby, the reengineer determines their significance during execution of P3. This means that architectural features which are related to higher-weighted rules will be considered priorly for assignment and therefore have a stronger impact on the further composition of the target architecture. Furthermore, a reengineer may pin architectural features. This prevents the reallocation of previously assigned architectural features to other target architecture components in phase P3.

P3 - Resource-efficient feature allocation: The phase P3 improves the initial assignment of architectural features generated in phase P1 referring to resource-efficiency. Therefore, the formulated rules are utilized and the compliance of the resulting architecture with the defined assertions is considered. There exists an enormous number of possible combinations for assigning architectural features. Efficiency improvements for one resource can lead to degradation for other resources or impair some design quality attributes. For example, splitting a component's parts towards different virtual machines can improve relative CPU utilization, but may lead to increased network traffic for intra-component communication and a decreased cohesion. Additionally, those effects do not necessarily have to move on linearly and moreover, the interrelations are often ambiguous as well. Therefore, we propose application of a heuristic rule-based approach to achieve an overall improvement. A potential algorithm is sketched in Fig. 2 and it works as follows.

The rules are considered successively according to their priority. Thus, rules with higher priorities are weighted higher and have a stronger impact on the generated target architecture. The selection criterion of a rule is defined to deliver a set of scalar architectural features. All possible subsets of the set are rated respective to the quality of

[2]http://kieker.sourceforge.net/ (Accessed August 16, 2010)

[3]http://www.omg.org/spec/SMM/ (Accessed August 16, 2010)

```
1: F_Pinned ← Pinned architectural features
2: R ← All rules
3: A ← All assertions
4: R_Sort ← Sort R descending by priority
5: F_AllAffected ← F_Pinned
6: for all r in R_Sort do
7:    F_r ← All architectural features delivered by r's
      selection criterion
8:    P_r^F ← Power set of F_r
9:    Score ← New associative array
10:   for all p_r^F in P_r^F do
11:      Score[p_r^F] ← Rate p_r^F
12:   end for
13:   Score_Sort ← Sort Score descending by score
14:   Score_Sort^Keys ← Keys of Score_Sort
15:   for all p_r^F in Score_Sort^Keys do
16:      F_FormerlyAffected ← p_r^F ∩ F_AllAffected
17:      F_NeedReallocation ← Elements of
         F_FormerlyAffected that need reallocation conc. r
18:      if F_NeedReallocation == ∅ then
19:         A_HigherPrio ← All a ∈ A with higher priority
            than r
20:         if ∄a ∈ A_HigherPrio with r violates a then
21:            Apply rule r to all features in p_r^F
22:            F_AllAffected = F_AllAffected ∪ p_r^F
23:         end if
24:      end if
25:   end for
26: end for
```

Figure 2. Rule-based heuristics for creating a resource-efficient feature allocation.

the target architecture that would result, if the features in the subset would be assigned correspondingly. This aims at considering interdependencies at the level of a single rule. For regarding interdependencies on an inter-rule level, the formulated assertions are taken into account. A rule is only applied if the reengineer did not formulate an assertion with a higher priority that would be violated after the rule's execution. Furthermore, the rule is applied to all mentioned subsets in order of their score. However, the rule is only utilized if no rearrangement of features is necessary whose subset was rated higher. The same applies to assignments that would lead to rearrangement of features that were placed by rules of higher priority or formerly pinned features.

D. Activity A4 - Adaptation

The activity A4 allows the reengineer to adjust the target architecture manually towards case-specific requirements that could not be fulfilled during generation activity A3. For example, the generation process might not have yielded an expected assignment of a critical component. Furthermore, for leveraging the elasticity of a cloud environment, the reengineer might configure a capacity management strategy by means of utilizing the hooks provided by entities contained in the cloud environment meta-model (cf. A2).

E. Activity A5 - Evaluation

For being able to judge about the produced target architecture and the configured capacity management strategy, A5 evaluates the outcomes of the activities A3 and A4. The evaluation involves static and dynamic analyses of the target architecture. For example, the metrics LCOM or WMC can be utilized for static analyses. Considering the target architecture's expected runtime behavior, we propose to apply a simulation on the basis of CloudSim [4]. Thus, we intend to contribute a transformation from CloudMIG's cloud environment meta-model to CloudSim's simulation model.

F. Activity A6 - Transformation

This activity comprises the actual transformation of the enterprise system from the generated and improved target architecture to the aimed cloud environment. No further support for actually accomplishing the implementation is planned at this time.

IV. Conclusion and Future Work

We presented an early work concerning our model-based approach CloudMIG for migrating legacy software systems to scalable and resource-efficient cloud-based applications. It concentrates on the SaaS provider perspective and facilitates the migration of enterprise software systems towards generic IaaS and PaaS-based cloud environments. CloudMIG is intended to generate considerable parts of a resource-efficient target architecture utilizing a rule-based heuristics. The future work focuses on the realization, improvement, and evaluation of CloudMIG's target architecture generation and evaluation activities (A3 and A5, respectively).

References

[1] W. Iqbal, M. Dailey, and D. Carrera, "SLA-Driven Adaptive Resource Management for Web Applications on a Heterogeneous Compute Cloud," in *CloudCom*, ser. Lecture Notes in Computer Science, M. G. Jaatun, G. Zhao, and C. Rong, Eds., vol. 5931. Springer, 2009, pp. 243–253.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[3] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2009.

[4] R. N. Calheiros, R. Ranjan, C. A. F. D. Rose, and R. Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services," *CoRR*, vol. abs/0903.2525, 2009.