A Bird’s-eye View on Modelling Malleable Multi-Cloud Applications

Mohammad Hamdaqa
Supervised by: Prof. Ladan Tahvildari
Department of Electrical and Computer Engineering
University of Waterloo, Canada
mhamdaqa@uwaterloo.ca

Abstract—Cloud platforms advances have changed the application development landscape. Cloud platforms abstract the complexity of application delivery to enable rapid development and easy management. This changes the way development teams need to think about and deal with the underlying resources while building and managing their applications. This research describes a new methodology supported by a modeling framework to enable organizations that build cloud applications (e.g., SaaS providers) to unbiasedly exploit the cloud platform building blocks to leverage the flexibility, reliability and scalability that these platforms provide to the application layer.

I. INTRODUCTION

According to Gartner the software as a service (SaaS) market will grow to $45.6 billion in 2017. To reap the benefits of cloud computing, SaaS providers should design and configure their applications to leverage cloud platform capabilities. Moreover, to take advantage of cost savings, applications should be platform independent to avoid vendor lock-in. This is challenging. First, it requires a full understanding of the service offerings from different providers and the metadata artifacts required by each provider to deploy, run and manage an application. Second, it involves complex decisions that are specified by different stakeholders. Examples include financial decisions (e.g., selecting a platform to reduce costs), architectural decisions (e.g., partitioning the application to maximize availability), and operational decisions (e.g., distributing modules to insure availability). Finally, while each stakeholder may conduct a certain type of change to address a specific concern, the impact of a change may span multiple models and influence the decisions of several stakeholders.

While several issues related to developing cloud applications have been individually addressed in the literature, there is no complete framework that supports them all at all service levels (i.e. infrastructure, platform and application) for the whole cloud application life cycle. The goal of my research is to build a multi-cloud framework that enables consumers of cloud platforms (SaaS providers) to exploit cloud platforms capabilities in order to architect malleable cloud applications that can guarantee a desired service level and minimize operational costs. Therefore, I am proposing: (i) a new architectural view model that focuses on service operation, and reflects cloud stakeholder perspectives along the application lifecycle, and (ii) a novel framework that provides holistic and partial architectural views, and enables the generation of the required platform artifacts by fragmenting the model into artifacts that can be easily modified separately. Moreover to keep the framework up-to-date as new cloud platforms emerge or evolve, this research also introduces (iii) a new schema matching approach that is tailored to the cloud domain.

The work in this research follows a model-driven approach that is heavily based on domain analysis to construct the framework. The domain analysis process will use information gathered from three of the most popular cloud platforms (Window Azure, Amazon Web Services, and Google App Engine). The usefulness and practical applicability of the proposed framework will be demonstrated through a representative domain example. The next section discusses this approach, as well as current progress. Section III highlights the research plan. Section IV presents related work, and Section V concludes this abstract.

II. APPROACH AND RESEARCH QUESTIONS

This research investigates the process of building, deploying and managing cloud applications by harnessing the underlying cloud platform and infrastructure capabilities in a cost effective manner. To tackle this problem, a Model Driven Engineering (MDE) approach has been adopted. An overview of the research approach is depicted in Figure 1. The approach aims to provide answers for four research questions. For each of these questions, we explain its objective, the methodology followed to address it, and the current research status.

A. RQ1. How is architecting cloud applications different from architecting other applications?

Objective: The objective of the first question was to devise a software architecture framework for cloud applications that captures the common practices of the different cloud application stakeholders for creating, interpreting, analyzing and using architecture descriptions to specify cloud applications deployment and management aspects.

Methodology: To understand the architecture of cloud applications, we started with a study of some of the major cloud computing reference architectures [1]. Consequently, a list of cloud services and stakeholder roles have been identified. We considered roles that are directly related to the cloud application development and operation process, and studied the different financial, design and operational decisions...
required to build a cloud application, deploy it, and manage its operations while running it in the cloud. Initial observations were that: (i) cloud applications need to dynamically evolve at runtime to meet performance, availability, and scalability targets, and (ii) the quality of a cloud application depends on its configuration and the architecture of its service model. These are fundamentally different from the concerns addressed in existing software architecture frameworks, where the focus is on an application’s implementation. In cloud computing, architecture evolves during deployment; therefore, runtime operation needs as much architectural modeling as functional design does. This shift in emphasis from architecting for implementation to architecting for service operation motivated the need for a view model that is constructed around the application service and deployment model.

**Status:** In [2] we proposed an architectural framework for cloud applications: the (5+1) architectural view model. The (5+1) view model consists of five model views, namely: availability, adaptation, performance, workflow, and provider models. These models specify a core model that represents the application service model that addresses five different but interleaved cloud concerns related to application deployment and operation. In [2] we explained what each model represents, why these models have been selected, and how the (5+1) architectural view model captures all the essential information for architecting malleable applications that can change their structure and behavior at runtime through configuration.

**B. RQ2. How can applications be architected for deployment into multiple clouds?**

**Objective:** Deploying an application on a cloud platform requires specifying how the application service model will use the platform resources of that particular provider. Our objective is to facilitate application migration between different cloud providers, by: (i) providing an abstraction layer (reference model) for cloud applications to enable modeling the service structure and configuration independently from the target cloud platform specifications; (ii) then mapping the platform-independent models onto target platform specifications as needed.

**Methodology:** To create a reference model for cloud applications, we started with a domain analysis for the different artifacts required to successfully deploy and manage an application on three cloud platform providers (i.e., Amazon AWS, Windows Azure and Google App Engine). Our analysis focused on the schemas provided by these platforms. These schemas specify the syntax and structure of the information that is required by the platform. Our methodology includes both manual and semi-automated schema matching processes to identify similar domain concepts and then create meta-concepts that describe them independently from the target platform.

**Status:** The initial cloud application reference model was presented in [3]. This reference model has been extended to address various views in the (5+1) view model [2].

**C. RQ3. How can the cloud modeling space and the configuration space be bridged?**

**Objective:** The objective of this phase was to keep architectural models in sync with configuration space as applications evolve. Consequently, this phase focused on developing a modeling framework that supports transforming the high-level architectural models into platform specific artifacts.

**Methodology:** To build such a framework, we created a meta-model for each of the (5+1) view models. Then integrated the meta-models into one meta-model, by extending the top element of each meta-model from a common meta-meta component, and defining associations between the elements of the different meta-models. The outcome meta-model is then realized as a layered Domain Specific Modeling Language (DSML), where we utilized the hierarchical structure of the meta-model to divide the model into layers to facilitate incremental and collaborative modeling. To facilitate model transformation, a model generator was implemented. We used template-based transformation, which provides the necessary flexibility and portability, since the only thing that needs to be changed to support a new platform is the template.

**Status:** So far, “StratusML”, our cloud application modeling framework and DSML, has been partially covered in IC2E 2015 [4]. There, we demonstrate some of the framework’s capabilities to model cloud service structure and behavior, validate the model, and generate target platform artifacts based on the transformation templates. Other capabilities of StratusML (e.g., cloud application performance modeling) have been left for future work.

**D. RQ4. How can the framework be kept up-to-date?**

**Objective:** We needed to explore how to support framework evolution as new cloud platforms evolve. In MDE, the process of domain model creation, evolution and transformation depends on the ability to find mappings between the concepts of source and target models. Consequently, this phase focused on devising a semi-automated approach to facilitate matching
domain concepts of different providers’ schemas to facilitate creating new concept abstractions and transformation rules.

**Methodology:** To automate the generation of alignments between providers’ schemas, we used schema matching to identify possible matches between the schemas of two major cloud providers (Window Azure, and Google App Engine). The shortcomings of traditional schema matching techniques is that they rely on linguistic and structural similarities to identify the possible matches. In the cloud, schema terms diverge so much that such matching is impossible. To address this challenge we incorporated domain knowledge in the schema matching process. We implemented a new web similarity matcher that uses web search results to achieve this goal.

**Status:** In [5], I presented “PrisonBreak” a schema matching approach that uses web search results to compute the similarity between any two schema-elements. The paper shows that Prison Break outperforms the traditional schema matching for finding correspondences between cloud schemas.

### III. Research Plan

My main focus is on framework evaluation. Moreover, I am working on extending both the StratusML framework and the PrisonBreak approach. I am extending StratusML to capture and generate analytical performance models, and Prison Break to solve a more generic problem related to models migration on the fly.

### IV. Related Work

Using MDE for managing and configuring software systems at runtime to satisfy desired quality attributes is not new [6]. Example of approaches that address this problem are surveyed in [7]–[9]. The shortcomings of these approaches are: (i) most of them are limited to performance, (ii) they focus on domains other than cloud computing, and (iii) they normally capture the dynamics of the software components only without considering the dynamics of the underlying resource model. StratusML addresses these issues. While recently there have been several proposals for exploiting MDE to address cloud modeling concerns, most of the current frameworks are not comprehensive enough. They either focus on portability [10]–[12], or security [13]. The most comprehensive cloud modeling frameworks that we are aware of, in terms of the number of cloud concerns they address, are MODAClouds [14], CloudML [15] and the cloud DSM by Caglar et al. [16]. Those are the closest relatives to StratusML. These modeling frameworks can be differentiated based on the features they provide. What distinguishes StratusML from both MODACloud and CloudML is (i) its ability to provide partial and holistic views of different cloud application concerns using the concept of layers, (ii) its ability to visually model adaptation rules and actions, (iii) its comprehensive validation constraints that covers a wide range of the cloud application requirements, (iv) its template-based transformation that can automatically generate ready-to-deploy platform-specific cloud application artifacts, and most importantly (v) its support for semi-automatic schema matching, which facilitates language evolution and template generation as new platforms emerge.

### V. Conclusions

To support the process of developing, deploying and managing cloud applications, I propose a new cloud specific architecture view model that takes service models as first class entities, and builds views that address cloud application evolution at runtime. So far, we have successfully realized the architectural view model as a cloud DSM that uses layers to toggle between partial and holistic views, model dynamic behaviors using adaptation rules and actions, and “weave” stakeholder concerns together to generate useful artifacts for supported target platforms. While the framework currently supports generating artifacts for three cloud platforms, support for framework evolvability has been considered, using template-based transformation and advanced schema matching.

### REFERENCES