Migrating Autonomic Self-Testing to the Cloud

Tariq M. King
Department of Computer Science
North Dakota State University
Fargo, ND 58108, USA
E-mail: tariq.king@ndsu.edu

Annaji Sharma Ganti
Department of Computer Science
North Dakota State University
Fargo, ND 58108, USA
E-mail: annaji.ganti@ndsu.edu

Abstract—The cloud computing model continues to gain much attention from software industry practitioners. As such, leading companies are investing in the development, packaging and delivery of cloud services over the Internet. However, although much work is being done to model and build cloud applications and services, there is significantly less research devoted to testing them. In this paper, we describe our research-in-progress towards migrating autonomic self-testing (AST) to the cloud. Our approach combines the development of an automated test harness for a cloud service, with the delivery of test support as-a-service (TSaaS). Both AST and TSaaS are supported by a virtual test environment, which utilizes the power of the cloud to enhance the self-testing process.

Keywords—software testing; cloud computing; autonomic computing; virtualization

I. INTRODUCTION

Leading companies such as IBM, Microsoft, Google, and Amazon have a vested interest in cloud computing [1]. As such several applications, platforms, and infrastructures are being developed to facilitate the delivery of computing resources as services over the Internet [2], [3], [4]. New cloud applications and services can be developed by tailoring existing services, while hiding the complexities of the underlying implementation.

Although much work is being done to model and build cloud applications and services, there is significantly less research devoted to testing them [5], [6]. Cloud services may be allowed to dynamically adapt and update, which requires that changes be validated at runtime. However, the high availability requirements of cloud services mean that they should employ a sufficiently transparent runtime testing process, i.e., any degradation of the timing, processing, and memory characteristics should be within the limits allowed by the quality of service requirements.

Developers and testers of cloud applications that use remote services generally do not have controllability or observability of these services, apart from what is exposed at the service interface. This can pose a significant challenge when validating applications that use stateful cloud services. A cloud service is said to be stateful if the result of its invocation does not only depend on the input parameters, but prior usage of the service [7]. One approach to testing applications that use remotely hosted stateful cloud services, is to develop a mock version of the service for the purpose of testing. However, this can be challenging since the implementation details of a cloud service are generally hidden from its consumers.

King et al. [8] apply concepts from autonomic computing to software testing of adaptive systems. The approach, referred to as autonomic self-testing (AST), deploys test managers throughout the software to validate dynamic adaptations and updates. AST was designed with flexible strategies for incorporating the approach into systems with different performance and availability requirements [8]. An empirical study of AST suggests that the replication with validation strategy can provide a highly transparent runtime testing process in distributed environments [9].

In this paper, we describe our research-in-progress towards migrating AST to the cloud, emphasizing the use of replication with validation. We also introduce the notion of test support as-a-service (TSaaS), which provides cloud partners with access to automated test operations for remotely hosted cloud services. Extending AST with TSaaS allows testing to cross administrative boundaries in the cloud. Our approach is supported by a virtual test environment that utilizes the power of the cloud to enhance the self-testing process.

The rest of this paper is organized as follows: Section 2 provides background material on cloud computing and autonomic self-testing. Section 3 presents our testing approach, and Section 4 describes the research-in-progress. Section 5 identifies some uses and potential benefits of the research direction. Section 6 discusses major research challenges and implications, and in Section 7 we conclude the paper.

II. BACKGROUND

This section provides background on cloud computing, and autonomic self-testing.

A. Cloud Computing

Cloud computing is the use of computer technology to provide services over the Internet [1], [6]. The term services refers to the delivery of computing resources, and includes: (1) Applications – programs designed to help users perform a task, e.g., Google Apps [3]; (2) Platforms – development,
In this section, we present our approach to testing cloud applications and services based on AST and TSSaaS. Due to the high availability requirements of cloud services, we emphasize the use of the replication with validation (RV) strategy. Since RV uses copies of managed resources for testing, validation can be performed on a separate computational node to avoid excessive degradation of the processing, memory, and timing characteristics of the cloud service. Lastly, our approach incorporates runtime virtualization techniques in an effort to enhance the self-testing process under RV.

### A. Developing the Self-Test Harness

High complexity in the underlying implementation, coupled with the distributed and asynchronous nature of cloud services, make developing a self-test harness for the cloud architecture extremely difficult. Pre-deployment testing of cloud services involves validating them for accurate functionality, interoperability, operational integrity, performance, security, fault tolerance, among others [6]. This involves writing test cases that check the software against the aforementioned quality attributes, and creating the scaffolding necessary to support test execution (e.g., test stubs).

Test engineering activities for a service produce a baseline test model, which specifies the: establishment of preconditions, application of test inputs, and test assertions to be made in terms of expected outputs. These specifications are then translated into scripts for automated testing tools. Under AST, the test model is made available to autonomic managers tailored for runtime testing activities, referred to as test managers.

Figure 1 shows the structure of a cloud provider under our approach. Each cloud provider hosts some X-as-a-Service, which is exposed to cloud consumers over the Internet (bottom-left). Partner providers can also act as consumers if they offer tailor-made services that invoke the hosted service at runtime. When the hosted service dynamically adapts or updates, a test manager (top-left) intercepts the change and validates it on a local test bed. The local test bed provides the hardware, software, and networking components necessary to validate the change using a copy of the hosted service. Once testing completes, the test manager compares the results against a predefined validation policy to determine whether or not the change should be accepted.

King et al. [8] provide a comprehensive description of test managers, including detailed workflows of their autonomic self-testing behavior. In addition, the development of test managers is supported by a reusable object-oriented design for building self-testable autonomic software [12]. We therefore provide these references for additional details on how test managers may be used to validate adaptive changes to resources such as X-as-a-Service [8], [12], [13], [14]
B. Exposing Test Support as-a-Service

After developing the self-test harness, test engineers model the sub-procedures of automated tests as operations to be realized on a separate test bed (see dotted box in Figure 1). These operations are then exposed as a service for partner providers, who can use them during the development, testing, and maintenance of tailor-made cloud applications and services. TSaaS provides cloud partners with access to automated test setup, input, assertion, and teardown operations for remotely hosted services. This enables them to execute on-line tests for their tailor-made applications, even when such tests require controllability and observability of remotely hosted services. However, in addition to using TSaaS on-line during development, we envision that enterprises would use TSaaS APIs to design, build, and deploy fully automated tests that can cross administrative domains.

C. Utilizing Runtime Virtualization

Managing an infrastructure for testing is an important, yet highly time-consuming aspect of the testing cycle. The replication with validation (RV) strategy to AST further complicates the problem by requiring that copies of managed resources be maintained for testing [8]. However, this generally leads to increased software production costs because of the need to purchase, install, setup, and manage additional computing resources to support testing. Therefore, in an effort to reduce some of the overhead costs associated with RV, we utilize runtime virtualization techniques when self-testing in the cloud.

As shown at the center of Figure 1, the local and partner test beds are realized in a virtual test environment. This environment hosts virtual machines and other virtual resources to support runtime testing. When the cloud infrastructure receives a command for validation services, a virtual machine containing a copy of the service is instantiated. Test-related operations are then performed on the virtualized copy of the service. Once the testing process completes, the virtual machine and its associated resources can be destroyed to relinquish valuable physical resources.

Runtime virtualization is achieved by linking the TSaaS API to a hypervisor, as shown at the bottom-right of Figure 1. A hypervisor, also known as a virtual machine monitor, is a platform that facilitates configuring and managing multiple virtual machines. Access to the hypervisor allows cloud partners to take snapshots of the system state at any point, and save these to a file server. Keeping snapshots of system state can be useful in reducing test setup/teardown times, and reproducing failures for online debugging.

IV. RESEARCH-IN-PROGRESS

The approach described in the previous section depicts a grand vision of automated testing and improved test support for cloud computing. Such a vision encompasses many research problems and practical issues related to testing adaptive, distributed, service-oriented architectures. The Software Testing Research Group (STRG) [15] at North Dakota State University, working under the guidance of the lead author, is currently pursuing a number of research directions that seek to address such issues. In this section, we present three of these research directions as work-in-progress towards migrating AST to the cloud.

A. System-Wide AST of Services

This research direction is aimed at formulating a practical approach to AST at the service-level. It defines a system-wide architectural perspective for AST, which validates adaptive changes in service-oriented applications.
Our architectural perspective views the adaptive portion of the application as a set of self-testable autonomic components (STACs), as shown at the center of Figure 2. At runtime, if an adaptation manager (top-left) makes a change to a service, self-testing is performed as part of its integration into the application. A test manager (top-right) coordinates the validation process by analyzing the dependencies between services, injecting stubs when necessary, and calling the appropriate test drivers for the service via a test interface. Using stub injection allows us to map AST to unit, integration, and system testing.

We are currently conducting a case study that applies system-wide AST of adaptive services to a Communication Virtual Machine (CVM) [16], [17]. CVM is a platform that enables rapid realization of user-centric communication models in the healthcare domain. Users can specify high-level models of their communication needs, and CVM synthesizes these models into a control script that automatically configures the underlying network devices and protocols [17]. CVM has been designed to configure itself with multiple open API communication services such as Smack [18] and Skype [19].

Although CVM does not use services that run over the Internet on a cloud platform, preliminary results suggest that system-wide AST of adaptive services is feasible. We are therefore planning to apply system-wide AST to a cloud application, and then conduct an empirical study to compare the self-testing cloud application with the self-testing CVM.

B. Propagating Changes to AST Models

This research direction focuses on tackling the problem of maintaining up-to-date test sets in the presence of dynamic adaptation and updating. It considers how three categories of adaptive changes affect the test model. These are: (1) additive changes – introduces a new component interface and/or implementation, (2) reductive changes – removes an existing component interface and/or implementation, and (3) mutative changes – transforms an existing component while retaining some aspects of the original interface and/or implementation.

We are currently investigating the use of change propagation [20], an emerging field of model-driven software development (MDSD), for automatically generating transformations for the test model of the adaptive system. A model of the runtime component configuration of the adaptive system is first analyzed for additive, reductive, and mutative changes. Traceability relationships between the two models are then combined with information on the change, to determine a general course of action for restoring consistency between the models.

Preliminary results have shown that propagating changes from the configuration model to the test model is difficult even for small examples. To prune the test model after a reductive change, the system must not only remove associated tests, but determine the course of action for all other test artifacts affected by the removal (e.g., stubs, data files). In addition, if the test case to be removed is part of a test dependency hierarchy (i.e., some other test cannot be run before this test is run and passes), then the logic in the test harness must be updated to reflect the test’s removal.

Additive and mutative changes typically require automatic test case generation, which is widely recognized as a highly challenging research problem. However, having TSaaS APIs in the cloud may lead to a ubiquity of test information on newly created, adapted, and updated services. We therefore plan to investigate how TSaaS can be used to support automatic test case generation for cloud services.

C. Developing a Prototype of TSaaS

This research direction represents the most recent addition to our work-in-progress towards migrating AST to the cloud. The immediate goal is to develop a proof-of-concept prototype of TSaaS, and the virtual test environment described in Subsection III-C. We are currently defining a detailed workflow of TSaaS, and acquiring the necessary hardware and software resources for building the prototype. So far the planned hardware configuration consists of two powerful servers to simulate a cloud computing environment.

An initial survey of platform development tools for cloud computing has led us to select the Microsoft Windows Server 2008 R2 Operating System (on which Microsoft Windows Azure is built) [4], [21]. This operating system provides hypervisor-based server virtualization technology, and supports runtime virtualization through Windows Management Instrumentation (WMI) and the Hyper-V API [21]. In addition, Microsoft Visual Studio already provides tools for developing cloud services for the selected platform. Once the prototype has been developed, we will evaluate the effectiveness and practicality of our approach through a series of controlled experiments.
V. USES AND BENEFITS

Since the trend of cloud computing is towards providing everything-as-a-service, the migration of AST to the cloud can be useful in several scenarios. In general AST facilitates testing the interactions between multiple entities at different levels of abstraction. At the sub-functional level (i.e., intra-service), this involves testing interactions between different functions of a single service after it has been modified. On the other hand, inter-service validation checks the resulting behavior of services that depend on the modified service.

Situations where intra-service validation of the cloud may be required include: adapting or updating hosted software applications and platforms, or after changes are made to the underlying infrastructure. For example, application-level services may dynamically add, remove, or replace components while balancing workloads. Platforms may automatically download and install security patches to protect themselves from attacks. Physical and virtual hardware/firmware may be upgraded to provide users with the latest device capabilities. AST would provide a means to determine whether or not such changes should be accepted, or if the system should roll back to previous versions.

At the higher level, AST may be used to detect faults arising from the interactions between services. This includes checking the interoperability of multiple application-level services, performing compatibility testing of applications with different platforms, or testing platforms with several hardware configurations. Furthermore, in addition to providing a test bed for developers, cloud providers may provide access to metrics from the latest test runs of their services. For example, this could include information relating to test set size and breakdown, item pass/fail results, and code coverage. These metrics could then be used by other providers as a measure of confidence in the hosted services.

VI. CHALLENGES AND IMPLICATIONS

The ideas presented in this paper represent significant research and technical challenges. From a research perspective, all the issues associated with runtime testing of adaptive and distributed systems should be considered. These include, but are not limited to: automatic test case generation and maintenance; dynamic test planning and optimization; and test synchronization and safety. From a technical perspective, a new wave of development and testing tools would need to be developed for the emerging cloud computing platforms.

The use of runtime virtualization for testing implies that developers and test engineers would need to be well-versed in the tools and techniques used to manage virtual resources. Testing in a virtualized environment also raises the question of whether or not one can rely on the test results. After all, one of the key advantages of testing is that it provides information about the behavior of a system in its actual environment. However, to guarantee that the results obtained from testing in the virtual environment are reliable, one would need to prove that the virtualization is consistent with the actual environment.

VII. CONCLUSION

This paper described our research-in-progress in the area of runtime testing of cloud applications and services. We proposed the migration of autonomic self-testing to the cloud, and introduced the notion of test support as-a-service. Our work is aimed at increasing the overall testability of cloud applications and services, and re-thinking the way software testing is performed on this emerging computational model.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Peter J. Clarke, Andrew A. Allen and the members of the Software Testing Research Group at NDSU for their contributions to this work. We extend a special thanks to Mohammed Akour and Akanksha Jáidev for their work on propagating changes to test models at runtime.

REFERENCES


