Poster: Cloud API Testing

Junyi Wang, Xiaoying Bai, Haoran Ma , Linyi Li , Zhicheng Ji
Department of Computer Science and Technology
Tsinghua University
Beijing, China
baixy@tsinghua.edu.cn, {wangjuny15, ly-li14, mhr15, jzc15} @mails.tsinghua.edu.cn

***Abstract*—Following the Service-Oriented Architecture, Cloud services are exposed as Web APIs (Application Program Interface), which serve as the contracts between the service providers and service consumers. With increasing massive and broad applications of Cloud-based development, a large number of diversified APIs are emerging. Due to their wide impacts, any flaw in the cloud APIs may lead to serious consequences. API testing is thus necessary to ensure the availability, reliability, and stability of cloud services. The research proposed an approach to automating API testing following the model-driven architecture, so that services can be continuously fetched, analyzed and validated. A prototype system ATcloud was constructed to illustrate the process of API understanding, test scenario modeling using directed graph annotated with transfer probabilities between operations, cloud-based test resources management, distributed workload simulation, and performance monitoring.**

Keywords—API testing; test automation; cloud computing

I. INTRODUCTION

APIs (Application Program Interface), especially Web APIs for Internet software, provide a natural way to wrap and deliver software functions as self-contained services that can be accessed through standard protocols. By serving as the contracts between service providers and service users, APIs can effectively shield heterogeneity as well as enforce decoupling. However, the inherent open, collaborative, and dynamic characteristics of Web APIs raise new threats to the quality of systems developed by composing services. API testing is thus gaining more and more attentions .

Cloud platforms, such as Amazon, Azure and Ali, provide APIs for various services including infrastructure services, storage services, data services, and more and more rich application services such as cognitive services and machine learning services. According to programmable Web report [1], there have been over 15,000 APIs available nowadays, a considerable increase from around 200 APIs in 2005. As Cloud APIs are widely used and continuously evolve online, there is a pressing need of an automatic testing approach to constantly detect the changes and potential defects in the services [2, 3].

The paper reported ATcloud, a model-based automatic testing framework and a prototype system to support Cloud API testing. Fig. 1 gives an overview of the proposed approach.

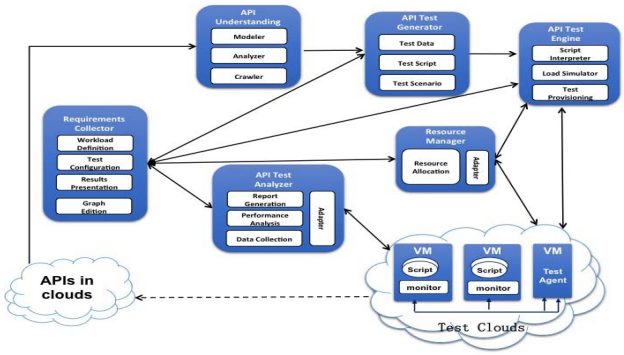

Fig. 1. Approach Overview

1) API Understanding. It automatically gatheres the API specifications from Cloud websites, interpretats the syntax and semantics of service data and operations, then transforms it into internal semi-formal representations.

2) Test Generation. With built-in strategeis, test cases are generated at different levels using different algorithms.

3) Test Engine. Test cases are encoded in executable scripts to be deployed at host environment, trigger on schedule. In this research, a test Cloud was built to dynamically allocate testing resources on demand across platforms.

4) Test Analyzer. Test results are collected, verified and validated against requirements. A monitor was built to gather peformance indicators, visulize and report system status during testing.

II. API UNDERSTANDING

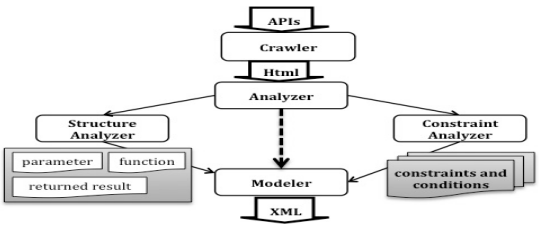

Fig. 2. API Understanding

Fig. 2 shows the process of API Understanding, which crawls Cloud services, interprets API specifications, and transforms them into XML-encoded description for automatic test generation.

978-1-5090-6676-6/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSTW.2017.71

A. Domain Knowledge Capture

In addition to capturing data and operation definition, a key issue in API understanding is the modeling of domain knowledge in terms of constraints and conditions. For example, a constraint of parameter *InstanceName* says that: *"the name of the instance is required to contain [2,128] English or Chinese characters, and it must be in uppercase letters or "-". InstanceId, which default is instance, cannot start with 'http: //' or 'https: //'"*. This type of constraint is critical to system robustness and reliability testing, but usually insufficiently defined in interface specification. To enhance the modeling capability with interface semantics, it has a great potential to improve test effectiveness and efficiency.

B. API Scenario Modeling

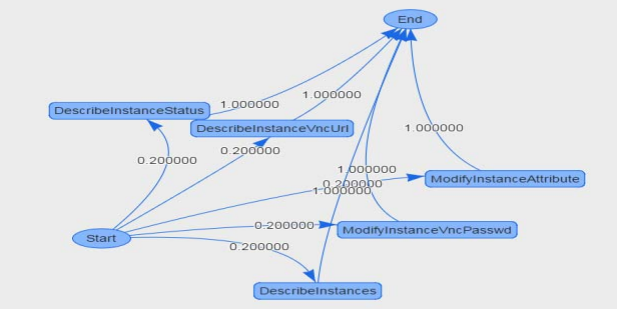

Fig. 3. An Example API Scenario Model.

A complex scenario by composing multiple APIs executed in sequence usually has a higher potency to reveal defects than single API testing. To capture API composition scenarios, a directed diagraph is defined to model the control flow and data flow among API invocations, as shown in Fig. 3. A node in the graph represents an API operation, and a directed link between two nodes represents a valid execution sequence between two operations. The link can be a conditional transfer, which is annotated with a Boolean expression of the conditions, or an iteration, which is annotated with the number of repeated times. In addition, the link is annotated with transfer probability to identify the differences in invocation frequencies among different operations. A formal definition of the scenario model is defined as follows:

Scenario: =<ID, Name, Description, Duration, Size, ThinkTime, Services, Matrix, Start, End >

Where *Size* represents the number of simulated users. *ThinkTime* represents the time interval for user application. *Matrix* is the transfer probability matrix, which is generated after traversing the directed transfer probability diagraph. *Start* and *End* defines the entrance and exit of the matrix.

III. API Test Generation

Test cases are generated at following levels:

- Test data generation based on API data specification and domain knowledge
- Test operation generation based on API operation specification.
- Operation sequence generation by traversing the paths in the directed graph of the scenario model.
- Workload generation by generating the operation sequences following the statistical distributions of the transferring probabilities in the scenario model.

The algorithm is described as Fig. 4:

```
// Test data generation based on API data specification and domain knowledge
load and parse the XML description of the current API;
for every combinations of parameters
    for each parameter parsed by the current API
        Obtain the constraint of the parameter;
        generate a value according to the parameter constraint and the current combination;
the API is called according to the parameters generated;
get the result;
log API response time;
// Test operation generation based on API operation specification.
if the parameter combinations are normal
    if results are normal
        for every Effect parsed out of the Effects section in the current ApiXML
            operate the local status data;
        for each data in the returned result
            compare the result with the state data stored in this computer;
            if there are differences
                report call error for the current API;
                exit;
    if an error warning was returned
        report call error for the current API;
        exit;
if there is an abnormality in the parameter combination
    if no error warning was returned
        report call error for the current API;
        exit;
report that the current API call is correct;
```

Fig. 4. API test generation algorithm

IV. Massive Scalable Workload Simulation

To simulate large-scale API invocations from geographically distributed clients, ATcloud maintains a pool of Cloud resources from various vendors and data centers to provision test resources on demand. As shown in Fig. 5, test tasks are wrapped, scheduled and remotely deployed to available VM (Virtual Machine) instances.

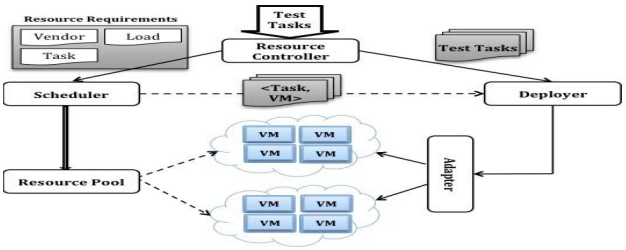

Fig. 5. On-demand resource allocation.

V. Summary and Conclusion

A large number of Cloud APIs have been published, which constantly evolve online and widely influence system constructions. The paper presents the testing framework and the ATcloud system to support Cloud API testing. It made an early attempt to address several key issues of API testing in Internet-scale. ATcloud aims to promote automatic API understanding and testing to facilitate continuous quality control of Cloud-based software systems.

References

[1] Programmable Web, "programmable Web report," https://www.programmableweb.com, 2005, accessed 26 January 2017.

[2] J. Gao, X. Bai, W.T. Tsai, "Cloud testing-issues, challenges, needs and practice," Software Engineering: An International Journal, 2011, Vol. 1, no.1, pp.9-23.

[3] J. Gao, X. Bai, W. T. Tsai and T. Uehara, "Testing as a Service (TaaS) on Clouds," 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, Redwood City, 2013, pp. 212-223.