Spatio-Temporal Architecture-Based Framework for Testing Services in the Cloud

Huai Liu[a], Maria Spichkova[b], Heinz W. Schmidt[a], Timos Sellis[b], Matt Duckham[c]
[a]Australia-India Research Centre for Automation Software Engineering
[b]School of Computer Science and Information Technology
[c]School of Mathematical and Geospatial Sciences
RMIT University, Melbourne, Australia
{huai.liu, maria.spichkova, heinz.schmidt, timos.sellis, matt.duckham}@rmit.edu.au

ABSTRACT

Increasingly, various services are deployed and orchestrated in the cloud to form global, large-scale systems. The global distribution, high complexity, and physical separation pose new challenges into the quality assurance of such complex services. One major challenge is that they are intricately connected with the spatial and temporal characteristics of the domains they support. In this paper, we present our visions on the integration of spatial and temporal logic into the system design and quality maintenance of the complex services in the cloud. We suggest that new paradigms should be proposed for designing software architecture that will particularly embed the spatial and temporal properties of the cloud services, and new testing methodologies should be developed based on architecture including spatio-temporal aspects. We also discuss several potential directions in the relevant research.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; D.2.5 [**Software Engineering**]: Testing and Debugging

Keywords

Software architecture, software testing, spatio-temporal logic

1. INTRODUCTION

Nowadays, more and more systems are deployed as services in the highly distributed cloud computing environment, and they are composed to construct complex services, which coordinate, control, and/or support global large-scale systems. These services are normally connected with teams, offices, facilities, and computing infrastructures that are remote from each other. Some of these systems track mobile fleets, supply chains, movements of goods and services etc., and are thus intricately connected with the spatial characteristics of the domains they support. Monitoring adds real-time processing constraints and requirements, and offers temporal lenses on historic data to enable predictive and proactive testing, monitoring and control.

Quality is always the key factor for guaranteeing the success of any product or service. Testing, a mainstream approach to quality assurance, validation, and verification, has been extensively used in the context of software systems. However, testing cloud services and their compositions is fundamentally more difficult than conventional software, especially when these services are deployed across remote locations in globally distributed environments. Existing industry-strength methods in testing have evolved around integrated systems running locally on single servers or focus on performance characteristics in distributed systems from the network architecture upwards. Few of them have taken the inherent spatial and temporal properties into consideration.

Analysis of the spatio-temporal dependencies among services allows designers and developers to find better testing strategies and identify the source of failure faster. Thus, the spatio-temporal aspects are crucial for the complex services in cloud: to test a service more efficiently, we need not only the service's data itself, but also additional information in relation to the time and to the location, when and where the data are produced. This should be taken into account while developing the architecture, i.e. the architecture should be *ST-aware* (spatio-temporal aware).

In this vision paper, we propose the development of a testing framework that supports the quality improvement and maintenance of the complex services and their compositions in the highly distributed cloud computing environments, with the purposes of enhancing fault-detection effectiveness as well as reducing the cost. More specifically, we will discuss the following aspects:

- Design of architectures for globally-distributed complex services based on the spatial and temporal properties;
- Development of architecture-based testing methodologies for complex services and their compositions, including:
  - Architecture-based test case generation methods;
  - ST-aware techniques for verifying testing results;

ASWEC ' 15 Vol. II, September 28-October 01, 2015, Adelaide, SA, Australia
© 2015 ACM. ISBN 978-1-4503-3796-0/15/09…$15.00
DOI: http://dx.doi.org/10.1145/2811681.2811685

- Definition of resource allocation strategies for testing and monitoring in cloud.

The rest of the paper is structured as follows: Section 2 introduces some background information of spatial and temporal frameworks and their applications. In Section 3, we discuss how to embed spatial and temporal properties into software architecture design. Section 4 presents some initial ideas of testing methodologies based on spatio-temporal architecture. In Section 5, we discuss a new challenge in testing and present our solution. Section 6 concludes the paper.

2. BACKGROUND – SPATIO-TEMPORAL FRAMEWORKS

To analyse spatio-temporal phenomena in a particular domain, we need to define the corresponding spatial, temporal and event semantics. In addition, if the coordinates of particular systems or their components are not fixed (systems/components can change their location over time) as, e.g., in the case of road traffic, the mobility semantics should also be defined.

Spatio-temporal logic is the natural combination of spatial and temporal logic, which qualitatively represents the spatial and temporal aspects of systems with mobility and real-time properties. It has been widely applied into different domains. For example, the project CHOROCHRONOS [18], initiated by the European Commission, aimed to achieve systematic interaction and synergy between temporal and spatial databases. The focus of the project has been on the design, implementation, and application of spatio-temporal database management systems, also covering architectural aspects of them. Xiao and Eltabakh [22] presented a spatio-temporal engine for complex pattern queries, STEPQ system. STEPQs require correlation among events in time and space over multiple instances of the database. Alamri et al. [1] discussed an approach for indexing spatio-temporal objects in indoor environments, and developed a trajectories index structure for moving objects in indoor spaces, which is based on adjacency between cells. Other applications of spatio-temporal logic include sensor networks[10, 12], information privacy [6], and smart devices [3].

These approaches mostly focus on the modelling aspect of the development and on general analysis of the system. In contrast to them, we are aiming at *integration of testing methods* into spatio-temporal representation. For this purpose we can use the modelling framework FocusST[20], which focuses on the formal representation of spatio-temporal aspects and allows us to create concise but easily understandable specifications [19].

3. ST-AWARE ARCHITECTURE

Architecture should be designed particularly for globally-distributed complex services. Though a large number of methodologies have been proposed for designing software architecture, few of them have dealt with the spatial and temporal properties of the system under development, which have to be considered now in the global cloud computing environments. In particular, ST-aware models should be developed for describing the highly-distributed services and their compositions, and architectures should be designed to formally represent the spatial and temporal properties of complex systems.

A number of architecture description languages (ADLs) have been developed to specify compositional views of a system on an abstract level. For example, the TrustME ADL [17] combines software architecture specification approaches with ideas of design-by-contract. This approach allows capturing of complex behavioural interaction patterns, synchronous and asynchronous, between large-scale components of software and systems architectures, and gives a background for the Rich Architectural Description Language RADL [16] and its modelling of predictable component-based distributed control architectures by extending finite state automata and Petri nets [15], as well as a framework for a family of mutually-compatible parameterised contract models [14].

In the following, we discuss how to expand the RADL functionality to include the features of FocusST[20]. We define a special type of component that specifies real objects that can physically change their location in space over the time (*ST-objects*, refer to Figure 1 for the general architecture of the system). An ST-object is normally associated with three special variables to store current *location* (e.g., central point of the object), *speed*, and *direction* of movement. A variety of representations of these variables may be necessary, for example absolute position in some specified geodetic coordinate system, relative position in a local coordinate system, or even a linear reference in the case of movement through a transportation network (i.e., distance along specified edge).

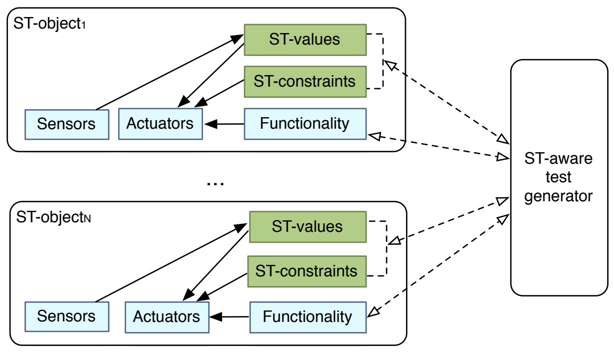

Figure 1: ST-aware Testing: General Architecture

By adding ST-constraints associated with every component, we can, for example, restrict the direction and speed of an ST-object; or verify whether the specified behaviour excludes the possibility that the object enters restricted areas during time intervals marked as dangerous. To calculate whether a collision of ST-objects is possible, we will assign to each ST-object a zone of influence that describes the maximal space the object can "cover" in the worst case. The maximal space may be represented as a simple radius of movement, or as a more complex polytope.

As shown on Figure 1, performance metrics of actuators are determined by the specified functionality of the object and its ST-constraints and ST-values. The sensors' data are used to update the values of the location, speed and direction of the object for a given timestamp (we call them *ST-values*). We can predict the values of these variables for the next timestamp, and also detect errors based on the inconsistencies between the predicted and the actual behaviour. For example, an unplanned stop can mean a collision of the objects or problem with the motion actuators; a sudden change of the location can mean sensors' failure. To generate "static" test cases based on ST-aware architecture, we can simplify the model by focusing on the location and time stamps only. However, the ST-constraints and the values of *direction* and *speed* of an ST-object build together a set of testing constraints that allows to decrease the minimal number of the required test cases.

4. NEW TESTING METHODS

In order to effectively test and monitor complex services in the globally-distributed and physically-remote environment, novel ST-aware techniques are needed. There exist many techniques for testing traditional software systems, but many of them may not be applicable to the state-of-the-art complex services and their compositions that are deployed in the cloud. Other approaches, e.g. [21], focus on testing in globally-distributed environments, but do not cover spatio-temporal aspects. We propose that spatial and temporal properties should be systematically introduced into testing.

Traditionally, there are two fundamental challenges in software testing, namely, reliable test set problem and oracle problem. The reliable test set problem refers to that it is practically infeasible to conduct testing with all possible program inputs, thus requiring the difficult selection of which inputs are to be used as test cases in order to reveal the largest set of program faults or to prove the program's correctness. The oracle problem means that in many practical situations, there does not exist a mechanism, namely an oracle, to help verify the correctness of test results given any possible input; or even if an oracle exists, it is infeasible to apply it. These two fundamental problems are still prominent and need to be addressed in the context of testing cloud services. On one hand, a series of methods should be proposed to design test cases making use of the ST-aware architecture. On the other hand, new techniques should be developed to verify the correctness of testing results in terms of spatial and temporal properties.

4.1 Test case generation based on spatio-temporal architecture

Lots of techniques have been developed to generate effective test cases to address the reliable test set problem. However, many of them cannot be directly applied into the testing of cloud services. White-box testing techniques, for example, construct test cases based on the source code of a component. However, the high abstraction in complex services makes it infeasible to apply such a low-level testing. Instead, testing should be conducted in the higher architecture level. What is more, the real time and high mobility of cloud services require the generated test case to sufficiently reflect spatial and temporal properties. In the following, we discuss some potential directions on how to make use of ST-aware architecture to enhance traditional testing techniques for their adoption in the context of cloud services.

4.1.1 *Architecture-based boundary-value analysis*

Boundary-value analysis [11] is a typical test case generation technique that is focused on the faults occurring at the boundaries of equivalence classes. Different from the equivalence partitioning method (which considers all inputs within the same class to be *homogeneous* in terms of fault detection [11]), boundary-value analysis selects test cases on and near the boundaries.

Under the ST-aware architecture, each component (or service in the context of cloud computing) is associated with certain location and timestamp (refer to Section 3). Suppose that a service executes the function f_1 when $x \leq x_0$, $y \leq y_0$, $z \leq z_0$, and $t \leq t_0$, while function f_2 will be executed otherwise (f_1 and f_2 are specified in the *Functionality*-block of the ST-object, shown in Figure 1). According to boundary-value analysis, two test cases can be generated as given in Table 1.

Table 1: Two test cases generated by boundary-value analysis

Test Case	Location	Timestamp	Note
TC#1	(x_0, y_0, z_0)	t_0	on boundary triggering f_1
TC#2	$(x_0+\epsilon, y_0 +\epsilon, z_0+\epsilon)$	t_0	near boundary triggering f_2

Note that a large number of test cases can be generated based on such a kind of ST-aware architecture-based boundary value analysis technique. Table 2 gives two other typical examples.

Table 2: Another two test cases based on boundary-value analysis

Test Case	Location	Timestamp	Note
TC#3	$(x_0-\epsilon, y_0, z_0)$	t_0	triggering f_1
TC#4	$(x_0+\epsilon, y_0, z_0)$	t_0	triggering f_2

4.1.2 *Architecture-based adaptive random testing*

As shown above, even for a single service that execute simple functions, a large number of test cases can be generated. Recall the example with the service triggering f_1 and f_2 under different conditions of location and timestamp. According to the basic idea of boundary-value analysis, given the three values for each parameter (for example, $x_0 - \epsilon$, x_0, and $x_0 + \epsilon$ for x), there can be a maximum of $3^4 = 81$ test cases. What is more, services are often composed to construct complex systems. The simple boundary-value analysis (and many other systematic testing methods) will generate a huge amount of test cases. Exhaustive testing using all these test cases is infeasible in many practical cases. Hence, a flexible technique should be used to decrease the number of generated test cases to fit the limits of testing resource and time.

It has been widely acknowledged that a positive correlation exists between the diversity among test cases and their fault-detection effectiveness [7]. Based on such an intuition, a number of testing techniques [5, 7] have been proposed to reduce the size of test suites while preserving the effectiveness of the original suites. Among these techniques, adaptive random testing [5] attempts to achieve a high diversity by evenly spreading random test cases across the whole input domain. Some recent studies [2, 4] have proposed techniques for applying adaptive random testing into programs with complex inputs. In these techniques, each program input is represented by a set of attributes (for example, categories and choices as defined in category-partition method [13]); distance between inputs are measured by how different these inputs are with respect to their associated attributes; and finally, diversity is achieved by maximizing the distance among all selected test cases.

We can make use of the spatial and temporal properties described in the ST-aware architecture as the attributes for adaptive random testing. In other words, enhanced testing techniques can be developed by applying some ST-aware architecture-based distance measure into adaptive random testing. One straightforward distance measure is the Euclidean distance between the services' locations. Another straightforward distance measure is the temporal distance, that is, the difference between the services' timestamps, for example, $|t_1 - t_2|$ given two services with timestamps t_1 and t_2. More sophisticated measures may be required where distance must be measured in a network, or between spatially and temporally extended objects (such as zones of influence or object trajectories). What is more, the functionalites associated with the services will also be considered for measuring the distance. Therefore, a weighted sum of multiple distance measures may in practice be required, where weights are defined through theoretical and empirical studies.

4.2 Test result verification based on ST-aware architecture

Unlike the large number of test case generation techniques to address the reliable test set problem, only a few techniques have been proposed for the oracle problem. One popular approach to test result verification is based on some innate properties that are normally extracted from software specifications. Metamorphic testing [8, 9] is a simple yet effective property-based technique for alleviating the oracle problem. In metamorphic testing, properties are represented in the form of relations (namely metamorphic relations) among different test inputs and outputs. A recent empirical study [8] has shown that a small number of diverse metamorphic relations can achieve a similar fault-detection effectiveness to a complete test oracle. However, in most of previous studies, metamorphic relations are normally identified in an ad hoc manner, that is, little research has been conducted on how to systematically identify metamorphic relations. With the introduction of ST-aware architecture, it is very likely that we can develop new guidelines for the identification of metamorphic relations, especially those with respect to the spatial and temporal properties. Such guidelines may be given in the form of question and answer: Some questions are provided to testers/users, who are required to give answers in the form of relations. The following lists some typical questions:

- Given the same service, if the location/timestamp/speed/direction has been changed, will the service's functionality also be changed? If yes, is there any relation between different functionalities?
- Given the same locations/timestamps/speeds/directions, will two homogeneous services have the same functionalities?
- Given two homogeneous services, if they have certain relations in the locations/timestamps/ speeds/directions, is there any relation between the functionalities of these services?
- Given two heterogeneous services that have some relations in their functionalities, if their locations/ timestamps/speeds/directions have been changed in a certain way, if there any certain change to the relations in functionalities?

It should be noted that though metamorphic relations are intuitively simple, they are quite expressive and useful in addressing the oracle problem. Refer to Tables 1 and 2 in the previous Section 4.1.1. Though we know which function will be triggered by each of the four test cases (TC#1, TC#2, TC#3, and TC#4), it may be very difficult in many practical situations to check whether or not the function is correctly executed, and thus infeasible to verify the result of one single test case (that is, the oracle problem exists). In such a scenario, we can at least verify part of spatial and temporal properties using metamorphic relations. For example, let us look at the following metamorphic relation:

MR#1 Given two different locations (x_1, y_1, z_1) and (x_2, y_2, z_2) as well as two different timestamps t_1 and t_2, the functionality of the service should not be changed as long as $x_1 < x_0$, $x_2 < x_0$, $y_1 < y_0$, $y_2 < y_0$, $z_1 < z_0$, $z_2 < z_0$, $t_1 < t_0$, and $t_2 < t_0$.

Though it is known that the function f_1 will be executed, we may not be able to verify whether f_1 is correctly executed. With MR#1, we can at least verify whether the same result will be given if we only change the locations/timestamps.

5. ST-AWARE RESOURCE ALLOCATION FOR TESTING AND MONITORING

As mentioned above, traditionally, there are two main fundamental problems in testing, namely reliable test set problem (addressed by test case generation methods) and oracle problem (addressed by test result verification techniques). Test execution, a necessary part of automated testing, was not considered as a major challenge when testing standalone programs: The programs were just run based on the generated test cases.

Nowadays, the highly distributed cloud computing environment poses another new challenge into testing, that is, how, when, and where to execute testing. We suggest the design of a set of new ST-aware models, which effectively allocate resources for executing the testing tasks all over the cloud. On this basis, we can perform testing on a cloud platform such as Chiminey [23].

We can make use of the spatial and temporal properties to design various testing scenarios, for example:

Same location & same time: manufacturing robots working together in a factory.

Different locations & same time: the virtual collaboration between services from remote fields.

Same location & different time: mining robots working in turn at the same place.

Different locations & different time: weather monitoring services relaying in raw data collection all over the world.

The ST-aware architecture will also be used to properly allocate resources for real-time testing and monitoring. Nowadays, numerous services are continuously executing all over the world. On one hand, testing and monitoring of these services are necessary to guarantee their quality. On the other hand, the testing and monitoring tasks should not affect the normal executions of services under test. Therefore, appropriate locations and timestamps will be selected such that services providing real-time testing and monitoring will not occupy the computing resources of the working services, but mostly use the spare resources. Conversely, the ST-aware architecture can also help design the models for load testing. Opposite to the real-time testing and monitoring, load testing requires the selection of the locations/timestamps where and when computing resources are heavily used by the working services.

6. CONCLUSION

The research we have discussed in this vision paper integrates knowledge from the three major disciplines of spatio-temporal logic, software architecture, and software testing. In the current cloud computing environment, everything is considered as a service. Multiple services can be composed to construct complex systems of systems. Physical services, such as robots, smart vehicles, etc, are distributed all over the world. Motivated by the features of real time and physical separation in these services, we propose the design of new ST-aware architecture for the complex physical services that are deployed in the highly distributed environment. Based on the new architecture, we can develop a new testing framework, which includes not only a family of novel testing techniques, but also new models for the execution of various testing tasks.

Our future work will focus on three research directions, namely new architecture design paradigm, ST-aware architecture-based testing techniques, and cloud-based test resource allocation strategies.

7. REFERENCES

[1] S. Alamri, D. Taniar, and M. Safar. Indexing of spatiotemporal objects in indoor environments. In *AINA2013*, pages 453–460, 2013.

[2] A. C. Barus, T. Y. Chen, F.-C. Kuo, H. Liu, R. Merkel, and G. Rothermel. A novel linear-order algorithm for adaptive random testing of programs with non-numeric inputs. Technical Report TR-UNL-CSE-2014-0004, Univ. of Nebraska – Lincoln, Dec 2014.

[3] A. Both, W. Kuhn, and M. Duckham. Spatiotemporal Braitenberg vehicles. In *ACM SIGSPATIAL GIS2013*, pages 74–83, 2013.

[4] T. Y. Chen, F.-C. Kuo, H. Liu, and W. E. Wong. Code coverage of adaptive random testing. *IEEE T. Reliab.*, 62(1):226–237, 2013.

[5] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse. Adaptive random testing: The ART of test case diversity. *J. Syst. Software*, 83(1):60–66, 2010.

[6] M. Duckham, L. Kulik, and A. Birtley. A spatiotemporal model of strategies and counter strategies for location privacy. In *GIScience2006*, pages 47–64, 2006.

[7] H. Hemmati, A. Arcuri, and L. Briand. Achieving scalable model-based testing through test case diversity. *ACM T. Software Eng. Meth.*, 22(1):6:1–6:42, 2013.

[8] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen. How effectively does metamorphic testing alleviate the oracle problem? *IEEE T. Software Eng.*, 40(1):4–22, 2014.

[9] H. Liu, I. I. Yusuf, H. W. Schmidt, and T. Y. Chen. Metamorphic fault tolerance: An automated and systematic methodology for fault tolerance in the absence of test oracle. In *ICSE2014*, pages 420–423, 2014.

[10] A. Mousavi, M. Duckham, R. Kotagiri, and A. Rajabifard. Spatio-temporal event detection using probabilistic graphical models (PGMs). In *CIDM2013*, pages 81–88, 2013.

[11] G. J. Myers. *The Art of Software Testing.* John Wiley and Sons, 2nd edition, 2004. Revised and updated by T. Badgett and T. M. Thomas with C. Sandler.

[12] A. G. Neiat, A. Bouguettaya, T. Sellis, and Z. Ye. Spatio-temporal composition of sensor cloud services. In *ICWS2014*, pages 241–248, 2014.

[13] T. J. Ostrand and M. J. Balcer. The category-partition method for specifying and generating functional tests. *Comm. ACM*, 31(6):676–686, 1988.

[14] I. Peake and H. W. Schmidt. Systematic simplicity-accuracy tradeoffs in parameterised contract models. In *QoSA-ISARCS2011*, pages 95–104, 2011.

[15] R. Reussner, H. W. Schmidt, and I. Poernomo. Reliability prediction for component-based software architectures. *J. Syst. Software*, 66(3):241–252, 2003.

[16] H. W. Schmidt. Trustworthy components – compositionality and prediction. *J. Syst. Software*, 65(3):215–225, 2003.

[17] H. W. Schmidt, I. Poernomo, and R. Reussner. Trust-by-contract: Modelling, analysing and predicting behaviour of software architectures. *J. Integ. Design Proc. Sci.*, 5(3):25–51, 2001.

[18] T. Sellis, M. Koubarakis, A. Frank, S. Grumbach, R. H. Guting, C. S. Jensen, N. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, B. Theodoulidis, N. Tryfona, H.-J. Schek, and M. Scholl. *Spatio-Temporal Databases: The CHOROCHRONOS Approach.* Springer, 2003.

[19] M. Spichkova. Design of formal languages and interfaces: "formal" does not mean "unreadable". In *Emerging Research and Trends in Interactivity and the Human-Computer Interface*, pages 301–314. 2013.

[20] M. Spichkova, J. Blech, P. Herrmann, and H. W. Schmidt. Modeling Spatial Aspects of Safety-Critical Systems with FocusST. In *MoDeVVa2014*, pages 49–58, 2014.

[21] M. Spichkova, H. W. Schmidt, and I. Peake. From abstract modelling to remote cyberphysical integration/interoperability testing. In *ISSEC2013*, 2013. arXiv:1403.1005.

[22] D. Xiao and M. Eltabakh. STEPQ: Spatio-temporal engine for complex pattern queries. In *Advances in Spatial and Temporal Databases*, pages 386–390. 2013.

[23] I. Yusuf, I. Thomas, M. Spichkova, S. Androulakis, G. Meyer, D. Drumm, G. Opletal, S. Russo, A. Buckle, and H. W. Schmidt. Chiminey: Reliable computing and data management platform in the cloud. In *ICSE2015*, pages 677–680, 2015.