Advanced Engineering Forum *Online: 2012-09-26*
ISSN: 2234-991X, Vols. 6-7, pp 865-870
doi:10.4028/www.scientific.net/AEF.6-7.865
© 2012 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.

A Lightweight Framework for Trust Cloud Software Development

Xiaoli Liu[1, a], Yujuan Quan[1,b] and Weizhen Jiang [1,c]

[1] College of Information Science and Technology, Jinan University,Guangzhou, Guangdong, China

[a] txlliu@jnu.edu.cn, [b]quanyj@126.com, [c] Vivianjiang00@Sina.com

Keywords: Trust; cloud software; formal method.

Abstract. Trust is particularly important for cloud service. In this paper, we introduce a lightweight framework for trust software development in cloud computing by comparing the differences of web service and cloud service. Formal method is used to ensure the correctness of the whole process. Furthermore, the theory implementation is simply discussed, including the related research results which support our framework.

Introduction

Cloud computing is receiving enormous attention in recent years. Cloud computing represents a new tipping point for the value of network computing. It is designed to cut costs for companies by moving such functions as data storage, security, and enterprise applications onto the Internet. Cloud application delivers software as a service over the Internet, eliminating the need to install and run the application on the customer's own computers and simplifying maintenance and support. In cloud computing, service provides common business applications on-line through a web browser, while the software and data are stored on the servers. It is widely recognized that trust is critical for cloud computing, especially for cloud services.

Armbrust, M. et al. summarized top 10 obstacles and opportunities for cloud computing and analyzed the difference between cloud service and previous web service or traditional software in [1]. Too many researchers devote to reputation based service discovery in cloud computing. Trust and Reputation Management is detailed investigated in [2]. However, the trust analysis about service development is ignored. Bad-quality services are allowed in cloud platform and the performance of will be affected. Although the privacy is taken considered into consideration while service designing, the correctness is not ensured. A formal framework for developing adaptable service-based applications is proposed in [3].

Cloud Service Engineering is of critical importance to Cloud Computing. Up to now, there are no generic software development methods for trust cloud application. Recent research results provide a significant foundation toward building trust cloud applications.

In this paper, we introduce a lightweight framework for trust software development in cloud computing. In this framework, the correctness of each step in the software development workflow is ensured by formal verification. Related formal methods and supported tools are also analyzed for process implementation.

a) Web Service and Cloud Service

The main differences of web service and cloud service are listed as below:

(1) Web service is a software system designed to support interoperable machine-to-machine interaction over a network. Cloud services are services related to storing or processing data in cloud remotely. For example, data storage, data processing. Usually, these services are provided on-demand, thus allowing scalability of the application.

(2) Web services are used as interface to access these resources / information through the web. Cloud computer is sharing resources / information as a service using the cloud. Too many issues should be studied in detail, include security, massive data storage, parallel computing and so on.

(3) Web service is running on web server, it might become overloaded and fail. Cloud service is running on some kind of cloud, the systems managing the cloud will bring up additional resources automatically to address the increased demand.

Cloud service is similar to the improved version of web service. However, the design of cloud service/software is different from web service on account of the specific characteristics of cloud platform.

b) Cloud Software

Cloud applications have special characteristics that differentiate it from traditional software because of centralization of data and millions of diverse concurrent users in cloud platform. The simple implementation of cloud services is figured in Fig.1.

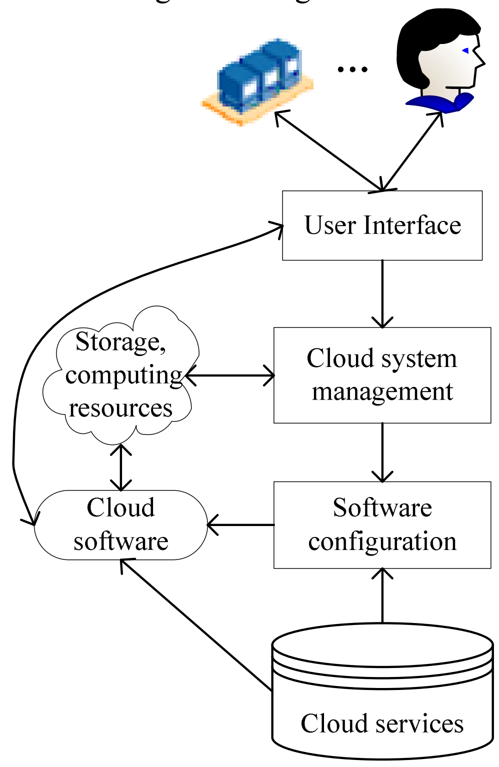

Fig.1 Implementation of cloud services

Complex cloud application is usually composed of many independent and autonomous services. Cloud service should be agile, reliable, scalable, further secure, sustainable, simple to maintain. To get trusted cloud software, in software development phase, following characteristics of cloud service must be considered:

(1) Trustworthiness

Software in cloud computing must be trusted enough to ensure the privacy about data processing. One of the difficult challenges in Cloud Computing is removing errors in these very large scale distributed systems [1]. The software should be correct and its behaviors must be expected. Bugs or redundant codes are not allowed. The software engineers need to pay close attention to the trustworthiness and development cost of cloud services.

(2) Rapid development

Cloud platform varies at any time and diverse software application is usually requisite urgently. The lifecycle of cloud software development should be short enough to meet the customers' needs.

Reliable development methods and supported tools should be introduced to shorten the development cycle.

(3) Reusing

There are lots of services in cloud and it is complicated work to manage these services. Redundant services or repeated function units should be eliminated from cloud platform. Improving the reusability of services is a way to realize green computing.

Therefore the granularity of cloud service should fit for the characteristic of cloud application.

(4) Easy change

Cloud application should provide with the flexibility and scalability to adapt as requirement change. The cloud platform is open and constantly changing. The software development method should deal with rapidly changing requirements of today's applications, data and clients. Cloud services need to both evolve quickly and provide high availability.

The Framework

Future software will likely have a piece that runs on clients and a piece that runs in the Cloud. The cloud piece needs to both scale down rapidly as well as scale up, which is a new requirement for software systems [1]. The application which is formed by fine-grained services can scale up or scale down quickly. And also, the fine-grained services have high reusability.

Service granularity generally refers to the size of a service. Service granularity types are detailed analyzed in [4]. Many researchers are trying to find an optimal granularity for service oriented application. In this section, we introduce a cloud software development framework which is based on fine-grained services. The framework workflow is showed in Fig. 2.

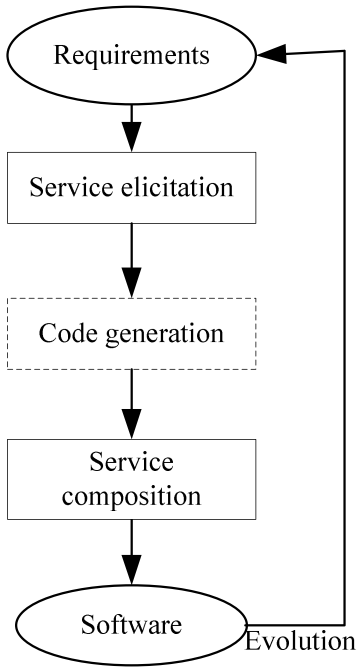

Fig. 2 The workflow of trusted cloud software development

My PhD thesis has completed the work in dashed rectangle in Figure 1. The framework contains following steps:

Step 1 Services Elicitation.

Functional or non-functional units of cloud service are elicited from requirement documents. Formal verification will be implemented to ensure the consistency of each service.

Step 2 Automatic Code Generation

For each service, the code is generated separately. The code generator can convert the formal specification into executable code directly according to the proved translation rules.

Step 3 Service Composition

Service composition is the process of creating a new cloud software/application by assembling existed cloud services. Formal models will be employed for service composition.

Step 4 Software Evolution

Software evolution is critical for cloud platform. The changed requirements must be analyzed rapidly and designed.

Process Implementation In Theory

In our framework, formal method is adopted to facilitate the whole development process to ensure the correctness of the software. Researchers have been devoted to some of these research areas. In this section, we explain possible implementation method about our framework.

A. Service Elicitation

A generic method for service elicitation is needed in service oriented engineering, especially for cloud software development. Since service granularity affects the quality of the software, the elicitation must satisfy the following characteristics: reusability, high cohesion and low coupling.

We want to build a formal business model for requirement analysis and inference rules for service elicitation which is presented in Fig. 3. The requirements will be represented by formal expression, and then verify the correctness and consistency. With inference rules separated services are elicited.

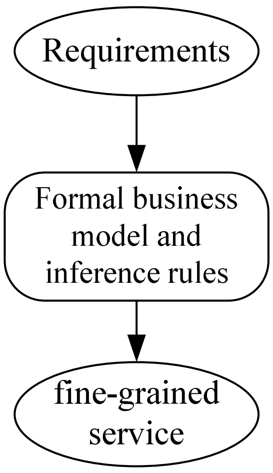

Fig. 3 Fine grained service elicitation

There are too many formal business models for requirement analysis [5, 6]. Therefore, for our framework the critical work is to construct inference rules. Service granularity analysis and logics are the basis of inference rules.

B. Code Generation

Automatic code generation form formal specification can promote the speed of software development. Code generation from formal specification is the very important research direction in formal method. The conversion is implemented according to the semantic translation rules. Each translation rule is proved with the semantics of source formal specification and the target language. There are many theoretical results. Model to model transformation about code generation is detailed discussed in [7]. A semantically configurable code generator is presented [8].

For model transformation, the semantics about source and target language should be investigated firstly. Then translation rules are constructed based on the semantic equivalence. Code generation from B formal specification to Java code is studied in my PhD thesis [9]. There are also commercial code generators for formal specification for different target language.

Automatic code generation can be difficult to implement, especially for commercial use. The case studies in experiment are usually in small scale. The commercial-grade generators should be fast and high-throughput.

C. Service Composition

The diversity and complexity of cloud services call for detailed composition and aggregation models. For enterprise oriented application, dynamic composition of capabilities into just-in-time service plans is requisite [10]. We plan to compose the services with formal method to ensure the correctness of the software. The service composition has been explored by many researchers for web services. Too many formal models are adopted for service composition, such as automata, semantic web, Petri-net, process algebra, and so on [11]. These research results can be used for our framework.

Some researchers also point out that in future the cloud users should be customize the application according to their requirements by themselves. It is great challenge for software engineers.

The construction of services and its ability to adapt to dynamic environments are important to cloud software.

D. Software Evolution

Software evolution aims to improve predictions about functional and structural transitions as scale increases. For object oriented application, the software evolution can be realized by defining a formal foundation of reuse contracts to capture similarities. Then, the principles behind software evolution can be illustrated [12]. For cloud application, we can also use similar method. What's more important is to get the users' feedback and to trace the requirements. How to trace the requirements in formal manner is the very important work [13]. In [14], dynamic requirement management at runtime is proposed at length. What we should do is to detail the process.

With the above analysis, we can see that existed formal methods and related research results lay a solid foundation for the proposed framework. Formal methods and supported tools are mature enough to be the foundation for process implementation.

Future Work

To implement our framework in practice, following work should be studied firstly.

A. Formal Foundation

Formal method is the basis for correct cloud software development in our proposed framework. In each phase, appropriate formal methods are adopted to do the modeling or verification. There are too many kinds of formal method using in service oriented software, such as B method, Petri-Net, and so on. How to balance the relative merits of these formal methods is the foremost work.

The first and foremost step is to analyze and improved existed formal methods on the basis of our framework and the characteristics of cloud.

B. Supported Tools

Rapid development is one of our aims in the proposed framework. Supported is needed to assist the development process. Especially for code generation, manual programming is absolutely inappropriate for correct design since bugs are often introduced inadvertently. Minor faults in code may bring great troubles for cloud platform.

Conclusion

This paper presents one approach to addressing the difficulties in the trust software development of cloud computing. A lightweight framework for trust software development is proposed. We also investigate the implementation of this framework. The method has the benefit that it can be combined to formal methodology and supported tools. With the assistance of formal methods and supported tools, the degree of software automation will be improved. The lifecycle of cloud software will be shortened.

This paper provides a lightweight framework to explore the field of Cloud Service Engineering. Its principal limitation at this time is the lack of concrete methods. Future work is also presented in this paper. The detailed implementation and proof obligation for our framework is the focus of current work.

Overall, we believe that our study has formed an important first step towards creating trust cloud software/services for cloud platform by using formal methods.

Acknowledgment

This research is supported by "the Fundamental Research Funds for the Central Universities" (No.21612337), National Natural Science Foundation of China (Grant No. 61003014/F020101) and the Natural Science Foundation of Fujian Province (Grant No.2010J05142).

References

[1] Armbrust, M., A. Fox, et al. "A view of cloud computing." Communications of the ACM, 53(4):2010, 50-58.

[2] K. Hwang, S. Kulkareni, and Y. Hu, "Cloud Security with Virtualized Defense and Reputation-Based Trust Management", in IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009, pp.717-722.

[3] L. Lambers, L. Mariani, H. Ehrig, and M. Pezz¨", "A formal framework for developing adaptable service-based applications", 2008, pp. 392-406.

[4] R. Haesen, M. Snoeck, W. Lemahieu, and S. Poelmans, "On the definition of service granularity and its architectural impact", 2008, pp. 375-389.

[5] J. Xiang, L. Liu, W. Qiao, and J. Yang, "SREM: A Service Requirements Elicitation Mechanism based on Ontology", 2007.

[6] M. Kerrigan, A. Mocan, E. Simperl, and D. Fensel, "Modeling semantic web services with the web service modeling toolkit", Journal of Network and Systems Management, vol. 17, pp. 326-342, 2009.

[7] Z. Hemel, L. C. L. Kats, D. M. Groenewegen, and E. Visser, "Code generation by model transformation: a case study in transformation modularity", Software and Systems Modeling, vol. 9, pp. 375-402, 2010.

[8] A. Prout, J. Atlee, N. Day, and P. Shaker, "Semantically configurable code generation", Model Driven Engineering Languages and Systems, pp. 705-720, 2010.

[9] X. Liu, "Research on code generation based on B formal method," PhD thesis, Wuhan University, 2008.

[10] D. Oppenheim, K. Ratakonda, and Y. M. Chee, "Enterprise oriented services", 2009, pp. 82-95.

[11] T. V. Pham, H. Jamjoom, K. Jordan, and Z. Y. Shae, "A service composition framework for market-oriented high performance computing cloud", 2010, pp. 284-287.

[12] T. Mens, "A formal foundation for object-oriented software evolution", 2001, pp. 549-552.

[13] M. Jastram, S. Hallerstede, M. Leuschel, and A. Russo, "An approach of requirements tracing in formal refinement", Verified Software: Theories, Tools, Experiment, pp. 97-111, 2010.