Scientific Research Publishing

Communications and Network, 2017, 9, 155-163
http://www.scirp.org/journal/cn
ISSN Online: 1947-3826
ISSN Print: 1949-2421

Uncertainty Analysis for Software Service Evolution in the Heterogeneous Cloud Environment

Hang Qin[1], Li Zhu[2*]

[1]Computer School, Yangtze University, Jingzhou, China
[2]Oujiang College, Wenzhou University, Chashan University, Wenzhou, China
Email: *68681700@qq.com, *yeah_1397118@hotmail.com

How to cite this paper: Qin, H. and Zhu, L. (2017) Uncertainty Analysis for Software Service Evolution in the Heterogeneous Cloud Environment. *Communications and Network*, **9**, 155-163.
https://doi.org/10.4236/cn.2017.93010

Received: April 5, 2017
Accepted: May 31, 2017
Published: June 3, 2017

Abstract

To solve the problem of resource heterogeneity and the dynamic structure, loose coupling of integrated applications has brought a lot of benefits in clouds environment. Thus, the development of highly robust service-oriented applications has many challenges, especially for the autonomy of service resources over the system components to the end-user portal. In this paper, a proposed method for the business users can satisfy the service availability changes in the early warning and application for service relationship adjustment. Then, the designed mechanism can deal with exception not available for service in a real-time development application for a business user. Based on the heterogeneous model of service-oriented applications, an availability process with lifecycle analysis is proposed to ensure that service resources are available to integrate components at different levels.

Keywords

Heterogeneous Cloud, Service-Oriented Computing, Anomalies, Exception Handling, Service Availability

1. Introduction

Homogeneous cloud is about the entire software stack, from the remote cloud provider, through various intermediate management layers, all the way to the end-user portal provided by one vendor [1] [2]. It integrates components by many different vendors, either at different levels or even at the same level. Specifically, in the release of subscription technology based on the change availability in service availability changes caused by abnormal prediction method, high

system on the issue of rapid detection capabilities. In addition, on the basis of the abnormal forecast, the prerequisite of the equivalent service brings about rules and algorithms [3] [4].

The existing exception handling methods are mostly based on the definition of the specific system, from the perspective of the specific processing strategy [5] [6]. According to the feature extraction technology of the service set, we have the general function service abstract and use the role of annotation, the definition of the application and the service to describe the relationship between the algebraic systems [7] [8]. Consequently, the user can configure the adaptive adjustment mechanism and algorithm, the business user's business layer configuration and system software layer for the combination, through a given algorithm to eliminate the impact of service on the application needs [9] [10].

The above model contains an early warning model and support exception, which can reduce the rate of abnormalities and the adaptive adjustment model of the availability matching to improve the adaptability [11] [12]. By using the event as a carrier, this paper develops a new model, which can cope with the problem of behavior control of distributed components in service-oriented environment. The model takes advantage of the service resources in the service-oriented environment, by improving the relationship adaptability between the application and the service to complete the exception handling.

In this paper, given the guidance on the definition of exception handling methods involving business users, some types of exception handling modes, which are characterized by business user participation, are proposed with possible human-computer interaction. It embodies how to maximize the introduction of business users to the implementation of the ability to deal with the original then.

2. Heterogeneous Service Exception Method

2.1. Basic Ideas

In business-oriented, service-oriented application development, the goal of the approach is to support for business user participation, early warning of service availability changes, and relationship adjustment to achieve the purpose of improving service usability. From the point of view of business-side programming, business users do not have to understand some of the specific concepts of the software layer, but also difficult predict and understand the dynamic behavior of software system components. To this end, the primary task is to guide the software layer system components, with regard to the basic idea of the architecture over dynamic behavior shown in **Figure 1**.

Firstly, from the service availability change model and the business needs of the basic constraints of business-level availability constraints, the system can automatically detect the availability of changes available to heterogeneous environment. Second, if the service reservation is unsuccessful, the exception is triggered by adjusting the application and service source set of the availability with the relationship to complete the exception handling.

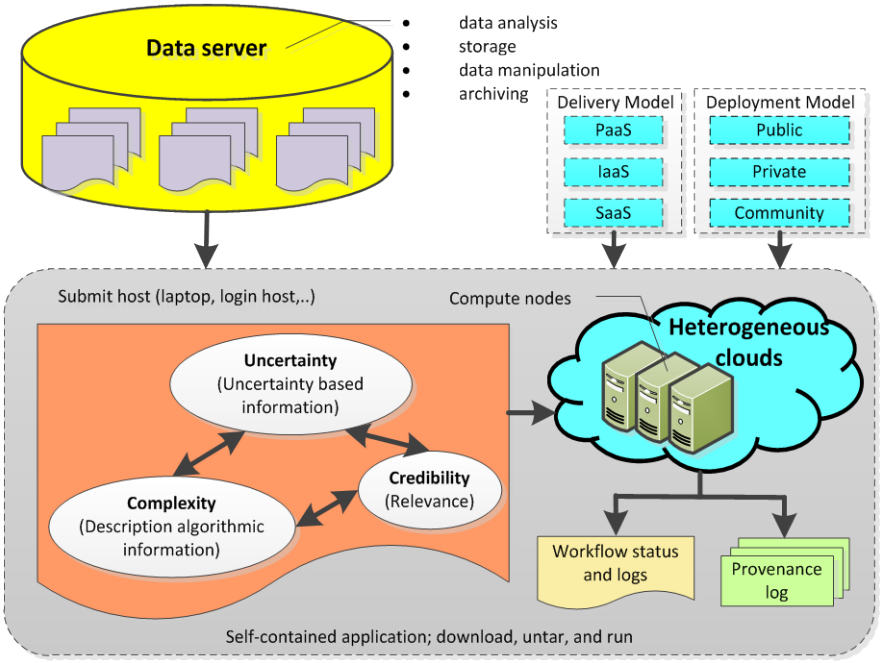

Figure 1. Workflow system architecture over dynamic behavior with heterogeneous clouds.

2.2. The Core Content

In order to achieve the basic idea, our method involves a lot of content, such as semantic description of the infrastructure establishment, service description management. Corresponding to the two research questions pointed out in the introduction, this paper focuses on the following two core content: pre-detection and maintenance of business layer availability constraints and software layer changes.

2.2.1. Maintenance of Business Layer Availability and Software Layer Changes

Service-level defined services unavailable boundary conditions need to be converted to software-level processing logic to complete the service not available for tracking. The current service availability judgment is primarily based on the application of static constraint information and service static description information matching to complete most of the current service selection process are the core process. It is difficult to adapt to the changes in dynamic changes.

For the above reasons, the objective is to achieve the service layer based on the availability of constraints in the software layer to complete the service availability. The primary problem to be addressed is that the business layer service is not constrained to the software layer service unavailable source mapping problem. This is on the basic elements of business needs and the basic elements of service description. To this end, we introduce events as a carrier of behavioral change basics to complete the mutual notification of dynamic behavior between system components. Finally, after pre-detection of potential anomalies, we can improve

the availability of service resources to the application requires a single connection between the application and the service.

2.2.2. Recovery after Unavailable Events

In the general study of the introduction, we recognize that in the application of the construction process for all abnormal case description and definition of the corresponding processing method. At the same time, abnormal in the process of eliminating the impact on running services on the application is a very important part of the link.

Further, in order to be able to make better use of exception handling rules, it is possible to classify them with feasible exception handling methods and business users based on the ability to analyze, to define the business users to participate directly in the exception handling mode. The process is concerned with the content of the internal implementation of the system, taking the last participation of the business user as a means.

The mode of the method is that during the process of handling anomalies through adaptive adjustment, the user participates in the angle definition of the processing mode that facilitates the definition of the new processing method. System implementation is the final implementation and verification of the method and its content to provide tools on the support holding, will also help it in the practical application of the field.

3. System Model

3.1. Objective

The purpose of the model is to provide a clear and clear view of the core content of the method and then lay the foundation for the corresponding implementation and implementation of this method. For the above purpose, the model consists of the following two parts:

From the point of view of business users, considering business users to complete the description of the availability constraints, it is required to provide business-level constraints for business users. In order to be able to use business users, requiring such constraints to reflect the content of the business area can be used to meet the needs of business users, and can be understood by business users.

From the software layer, it can describe the information point of view, in order to ensure that from the dynamic and static description of the two levels can be the description of the change in nature requires the ability to effectively establish the association between the two types of data. Based on the above analysis, the business layer usability constraints and software layer changes in the predetection of the following conclusions:

- It is necessary to provide a business concept that reflects the usability constraints of the business area, allowing the business user to directly define the business level constraints and make it possible with business-side programming;

- The need to provide a description of the usability of the service concept, so that these concepts can be described in detail through the service dynamic behavior related availability;
- The business users to shield the underlying system components of the availability of changes, making business users only need to consider the industry the definition of the layer.

For the above objectives, we use the ontology as the infrastructure to achieve business layer availability constraints semantics and software layers available the mapping between changes. The main goal of the software layer usability predetection is to achieve the follow-up to the source of the change, as well as the effective control of the random change process during operation. In the maintenance of usability, we are in the existence of a large number of services in a service-oriented environment, through the rational use of service resources to protect the corresponding use of usability.

3.2. Reducing the Availability Change of Exception Rate

3.2.1. Association with Software Layer Availability Changes

One of the issues to be considered when building business applications and service resources is to adapt to service-oriented computing the dynamic and open nature of the environment. The goal of this approach is to influence the impact of dynamic changes in service resources by service users. From the perspective of business needs description, such methods can be more effective to help business users to develop applications. From the application of operating angle, it is difficult to use this method to provide timely feedback on the dynamic changes of service resources. In the context of the discussion of the application, the basic idea of designing the usability constraint clustering is to meet the above design principles:

1) Business users autonomously define business rules that constrain service availability business-side programming requires business users to directly describe the basic needs in the business area.

2) Professionals predefine the behavior that leads to the change in availability the discussion in the introduction shows that the basic causes of usability changes are varied. While it is not possible to see when these changes occur, the professionals can define these changes in advance.

3) Implement the association of business constraints to availability changes according to a given association from the implementation point of view, while business constraints are associated with a certain type of availability change, to achieve the synchronization between the two.

In order to achieve the above objectives, we first analyze the different levels of the system description of the content. Events that are not available for services are called exception triggering events. The so-called exception triggering event refers to the behavior that occurs at a particular point in time during the run of the application, which may cause the application to continue service. On the whole, the usability constraints of the business layer and the usability of the

software layer change occur at two levels of independence. To change the details of the software layer, we need to define a reasonable mapping between the two, so that the software layer changes can be clustered to the business layer of the constraint concept.

As can be seen from the composition of the root causes, changes in software layer availability may eventually lead to business-level constraints not satisfied, resulting in an exception. The business layer constraint rules can be divided into functional constraint rules and nonfunctional constraint rules a large category. Changes to the software layer availability also affect the service function attributes and non-functional attributes.

3.2.2. Early Warning and Handling of Availability Changes

In order to achieve pre-detection of availability changes, the primary problem to be addressed is that the service layer service is not constrained to software services are not available for source mapping problems. The association relationship mapping defined in the previous has been built between the two the interrelated bridges, which can lay a good foundation for the pre-detection of anomalies that are not available. To this end, we have events as a carrier of behavioral changes, in terms of a basis for the completion of system components mutual notification of dynamic behavior.

3.2.3. Characteristics

Software layer service availability changes are random and dynamic, through the availability of change warning processing model can be business layer as a subscription to the subject and content. In the application when the service of the initial call fails, it is judged by the check of the operation result of the reservation service whether a service has been smooth running finished.

Automatic service is not available for pre-detection. The software system constrains the event subject semantics and business layer availability semantic mapping to complete the association between dynamic and static constraints. The definition of the software layer availability change event can be done independently, so that it can customize the new exception trigger event subject according to the new application environment.

Based on the idea of publishing subscriptions, service availability changes are based on events in the automatic transmission, which can reduce the impact of other external factors on the event capture. After the availability of the original binding service has changed, the system automatically completes the appointment of other available services to establish multiple implementations of individual requirements. At the same time, we can see that the appointment of the service is to increase the load on the premise of the system.

3.3. Adaptability Adjustment of Exception Recovery

In the first phase of the processing of usability-related issues, we define the availability change warning processing model to low service availability changes

lead to unusual risks. But if the process still cannot avoid the occurrence of abnormal, we need to further consider how to deal with the occurrence of abnormal problems. As the number of services in the service-oriented environment grows, processing exceptions due to service unusable can be done as much as possible make full use of other available service resources in the system environment, and take into account the rationalization of application needs to meet. The analysis the goal of exception handling is to ensure that the application is running between the application and the service between the effective dynamic connections, because the dynamic disruption of the connection relationship is the cause of the service being unusable and causing an exception.

4. Lifecycle for Autonomic Cloud Computing

Autonomic cloud computing system is different with the general software system. Also, the autonomous unit is different with the general components, which have a unique the life cycle. The life cycle should be the provision of self-management support, with an autonomous unit begins with the design and implementation, tested, validated. Then, it can install, configure, then start the deployment run; at run time. The autonomous unit in the transition from clients, service catalogs, resource provisioning, virtualization, management, to data centers is shown in Figure 2.

In the design and implementation of autonomous units, taking into account the interoperability between autonomous units, we have different states from multiple cloud providers, such as initializing, active, suspending, migration, and waiting. Because the autonomous unit can provide both services, it is important to describe and match the service. Users in the design of independent units, with the need for powerful tools to the preparation of autonomous units, can ask for the customization of the strategy.

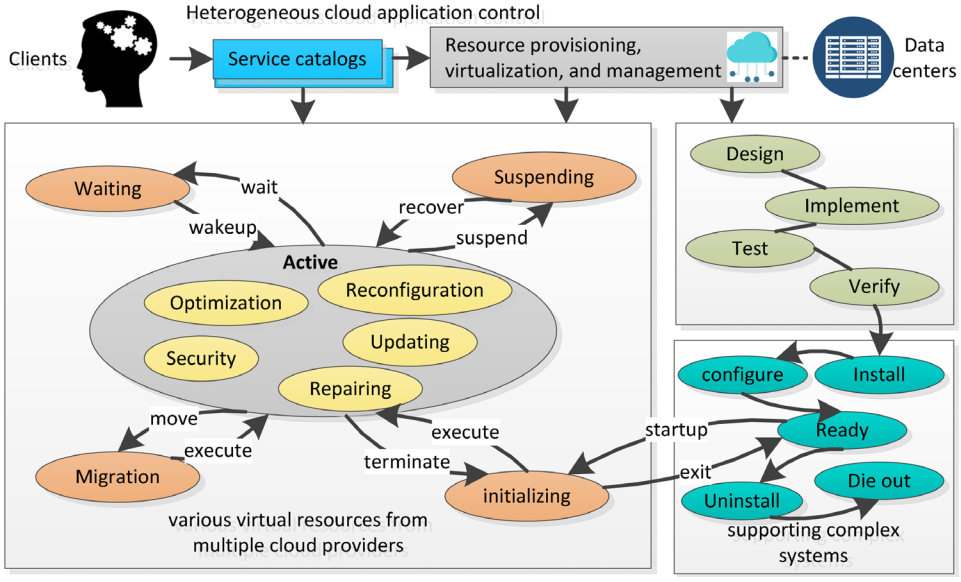

Figure 2. The lifetime for autonomous unit over data centers.

When installing and configuring autonomous units, we need to register installation and configuration with a special directory server in the autonomic cloud computing system information for use in subsequent self-management. After the installation configuration is complete, the autonomous unit enters the ready state at any time standby run. The autonomous unit is initialized by the ready state after initialization. If the autonomous unit terminates exit, a new ready state is entered. In this state, if it is unloaded, the autonomous unit can enter the demise of the state, thus completing its complete life cycle.

5. Conclusion

This paper deals with the heterogeneous service exception related to usability issues for the cloud service applications. The workflow system architecture is developed over dynamic behavior with heterogeneous perspective of business users involved in the processing. Moreover, for the maintenance consideration, the proposed method can not only reduce the availability change of exception rate, but also have early warning over handling process. Finally, from clients to data center, the autonomous units have lifetime of exceptions related to usability in service-oriented applications, in terms of the basic content of the method and the processing ability of the problem.

References

[1] Wu, L., Garg, S.K., Versteeg, S., *et al.* (2014) SLA-Based Resource Provisioning for Hosted Software-as-a-Service Applications in Cloud Computing Environments. *IEEE Transactions on Services Computing*, **7**, 465-485. https://doi.org/10.1109/TSC.2013.49

[2] Wang, S., Liu, Z., Sun, Q., *et al.* (2014) Towards an Accurate Evaluation of Quality of Cloud Service in Service-Oriented Cloud Computing. *Journal of Intelligent Manufacturing*, **25**, 283-291. https://doi.org/10.1007/s10845-012-0661-6

[3] Lee, I. and Lee, K. (2015) The Internet of Things (IoT): Applications, Investments, and Challenges for Enterprises. *Business Horizons*, **58**, 431-440. https://doi.org/10.1016/j.bushor.2015.03.008

[4] Jamshidi, P., Pahl, C. and Mendonça, N.C. (2016) Managing Uncertainty in Autonomic Cloud Elasticity Controllers. *IEEE Cloud Computing*, **3**, 50-60. https://doi.org/10.1109/MCC.2016.66

[5] Fitzgerald, B. and Stol, K.J. (2017) Continuous Software Engineering: A Roadmap and Agenda. *Journal of Systems and Software*, **123**, 176-189. https://doi.org/10.1016/j.jss.2015.06.063

[6] Bhattacherjee, A. and Park, S.C. (2014) Why End-Users Move to the Cloud: A Migration-Theoretic Analysis. *European Journal of Information Systems*, **23**, 357-372. https://doi.org/10.1057/ejis.2013.1

[7] Letier, E., Stefan, D. and Barr, E.T. (2014) Uncertainty, Risk, and Information Value in Software Requirements and Architecture. *Proceedings of the* 36*th International Conference on Software Engineering*, Hyderabad, May 31-June 7 2014, 883-894.

[8] Cusumano, M.A., Kahl, S.J. and Suarez, F.F. (2015) Services, Industry Evolution, and the Competitive Strategies of Product Firms. *Strategic Management Journal*, **36**, 559-575. https://doi.org/10.1002/smj.2235

[9] Seethamraju, R. (2015) Adoption of Software as a Service (SaaS) Enterprise Resource Planning (ERP) Systems in Small and Medium Sized Enterprises (SMEs). *Information Systems Frontiers*, **17**, 475-492. https://doi.org/10.1007/s10796-014-9506-5

[10] Truong, H.L. and Berardinelli, L. (2017) Testing Uncertainty of Cyber-Physical Systems in IoT Cloud Infrastructures: Combining Model-Driven Engineering and Elastic Execution. *Proceedings of the* 1*st ACM SIGSOFT International Workshop on Testing Embedded and Cyber-Physical Systems*, Santa Barbara, CA, 10-14 July 2017, 5-8.

[11] Von Der Emde, M., Hoffmann, T., Nowotny, D., *et al.* (2014) Providing Payment Software Application as Enterprise Services. U.S. Patent 8671032.

[12] Suarez, F.F., Cusumano, M.A. and Kahl, S.J. (2013) Services and the Business Models of Product Firms: An Empirical Analysis of the Software Industry. *Management Science*, **59**, 420-435. https://doi.org/10.1287/mnsc.1120.1634