Runtime Exceptions Handling for Collaborative SOA Applications

Bin Wen(✉), Ziqiang Luo, and Song Lin

School of Information Science and Technology, Hainan Normal University, Haikou 571158, China
binwen@hainnu.edu.cn

Abstract. For all kinds of computing infrastructure, services have become the important channel to enlarge theirs resource utilization performance. Services computing achieves resource sharing value. Because of the lack of runtime exception handling consideration, collaborative service-based software system often encounters unexplained interrupt and collapse. This paper focuses on the collaborative ability of service-based system, especially in the adjustment mechanism for runtime exception handling. Main contributions: (1) Self-adaptive exception handling architecture for services resource has been built. (2) Runtime collaborative adjustment mechanism has been designed to deal with requirements and context changes. Experiment and empirical analysis for Hainan agricultural E-business platform have been acquired so as to support collaborative SOA with above-mentioned approaches. Combination of theoretical research and empirical validation, the paper tries to provide a technical operational and cost-effective solution with collaborative mechanism and promoting SOA runtime adaptive ability for runtime requirements evolution and exception handling. The solution will provide systematical support to build collaborative SOA.

Keywords: Collaborative SOA · Exception handling · Runtime

1 Introduction

Software and resource usage have entered into the cloud infrastructure for consumer use in the form of services. Service has been become the basic way to access and enlarge the capability of all kinds of computing infrastructure. Services computing achieves resource sharing value. In the study of Bigdata, Analytics as a service (AaaS), and Software as a service (SaaS), Platform as a service (PaaS), Infrastructure as a service for Cloud computing, the core is services provisioning. Services computing related researches in recent years also more involved in resource sharing and application integration based on infrastructure. "Software as a service", "Big service: resource as a service". Making full use of the available online software resource to meet the requirements of diversified and distributed stakeholders, we are going to the service-oriented software engineering (SOSE) era [1–3].

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 252–261, 2017.
DOI: 10.1007/978-3-319-59288-6_23

IOSOC2013 is the famous conference for services computing held on Nov. 3, 2013. Carlo Ghezzi, Fellow of ACM and IEEE, published keynote titled "Surviving in a world of change: forward evolvable and self-adaptive service-oriented systems" [4] and also aimed to focus on the adaptability and collaborative capability of service-oriented system. He pointed out that runtime adaptability is a major challenge to promote the development of software technology. The preface on the proceeding of IOSOC2014 [5] also emphasized that "In addition to traditional topics such as service-oriented architecture, service design, service description, and service composition, service change management is a key topic that reflects the need for services to adapt to dynamic environments."

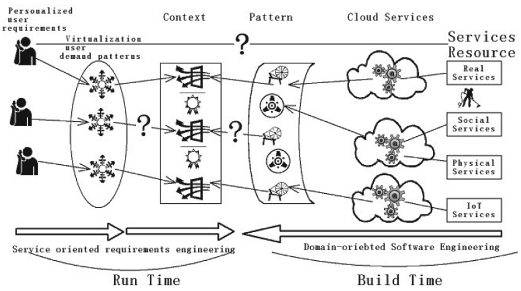

Fig. 1. Software Production Methods and Problems Based on the Services

General process for service-based software (SBS) production is as follow: producing services resource→publishing services→selecting services→services aggregation (services binding and combination). It is similar to the waterfall model or mixed alignment mode between top-down and bottom-top approach (Fig. 1). But the process lacks of runtime exception handling without starting from the requirements to overcome the exceptions, services resource insufficiency and context changes (Fig. 1: "?").

Therefore, establishing the direct feedback channel from user requirements to services aggregation, and implementing SOA runtime adjustable exception handling to build collaborative SOA are the motivations of the studying and significance.

Services resource provisioning has become the consensus of SBS development for industrialization. Now knowledge body of services computing mainly includes lifecycle planning, resource production, publishing, billing and management of SBS. Industry and academic community generally focus on the services discovery and services composition, the starting point of research is to assume that services resource is rich enough. Actually, services resource for meeting the users' needs often does not exist or cannot get in runtime at present stage. As mentioned in the literature [6], "If no services are available for some parts, the application developer can register them in the service broker's directory and wait until the needed services are available." Thus these problems lead to the difficulty, high complexity and low availability of services composition.

At the same time, due to the dynamic nature of web services and error-prone supply environment of services resource, all kind of provisioning exceptions may occur during the process of composition [7,8]. Exception refers to the services failure (fault), network error or abnormal events caused by resource or requirements changes. Lack of exception handling mechanism, it will lead to these problems such as poor performance, resource waste, poor optimized services and even failure. So services resource provisioning must be able to actively produce. They should have adaptive runtime exception capability.

The main SBS development platform in industry, such as IBM RSA[1], ActiveBPEL[2], Websphere Integration Developer[3], are also lack of considerations for active provisioning and runtime exception handling.

The main problem of the existing SOA mainly lies in:

1. Traditional SOA does not have runtime adaptive regulating mechanism and ability to meet the runtime context changes.
2. Lack of effective runtime exception handling solutions for SBS.

The current services resource is produced in advance by services provider. The process is lack of runtime exception handling consideration. Therefore, adjusting the runtime architecture to deal with context changes and resolving interrupt and collapse with runtime exception handling have become the urgently solved problems.

How to build self-adaptive exception handling architecture for services resource? How to design the runtime collaborative adjustment mechanism to deal with requirements and context changes? This is what our motivations focus on the runtime exceptions handling for collaborative SOA applications.

This paper aims at study the runtime exception handling for Collaborative SOA. We will mainly investigate the applicable SOA runtime exception handling mechanism. The rest of the article is organized as follows. Section 2 introduces the runtime exception handling. Section 3 presents runtime adjustment mechanism to deal with context changes. Section 4 shows the experiment and empirical analysis with the approach. Conclusions with main contributions of proposed approach and further work plans are also touched upon in Sect. 5.

2 Runtime Exception Handling

Due to lack of considering time changes and integrating method between requirements and architecture, we propose a adaptive approach with predictive control to put the requirements of problem space adaptively mapped to the runtime architecture of solution space. The approach can learn the model based on wavelet transform to predict the performance of services component, and induce requirements evolution or model transformation of architecture to achieve

[1] http://www.ibm.com/developerworks/rational/products/rsa/.
[2] http://www.activebpel.org/samples/samples-4/samples.php.
[3] http://www-01.ibm.com/software/integration/wid/.

runtime adaptive ability. And dynamic self-adaptive production based on customized services resource and legacy software servicilization will be choose for runtime adaptive exception handling according to users' requirements.

Service virtualization is designed to shield the heterogeneous properties of IT resource. Software abstraction expression will be decoupled with concrete IT resource to realize semantic equivalence mapping between web service of IT level and business functional requirements [9].

With the aid of service virtualization, active customization of personalized services resource is explored to complete instant and on-demand production for unmatched services resource in runtime. These efforts should compensate the lack of on-demand adaptive services customization for research community.

Services requester-centric collaborative SOA is different from triangle equal traditional SOA. Service requesters (consumers) lie in active status. Services consumers not only launch services requirements to select match and combine among the provided services, but also they can dominate the customized process. Predictive control can monitor requirements changes so as to map requirements and requirements evolution into the elements of architecture at run time.

As the project experimental carrier, we choose a software system based on Internet focusing on Hainan agricultural E-business services. Because the service-oriented Hainan agricultural platform is faced with rich stakeholders, more services resource, diversified individuation needs and SOA style to lead to the variety of customized requirements. Also, legacy software components for agricultural products processing are numerous. These situations conform to the demand for empirical carrier.

3 Runtime Adjustment Mechanism to Deal with Context Changes

Here in this section, runtime exception handling mechanism for collaborative SOA will be discussed based on context and requirements changes.

Services actively satisfy the consumers' needs to produce. It can change passive services selection shortcoming that is unable to satisfy users' needs. Unmatched and unqualified services in runtime will be appended into the queue of customization that will be processed to notify providers for on-demand production using Atom subscribe/inform style by customization manager. For customized and matched services, they will rerun to the aggregation flow to continue to complete the business process.

3.1 Self-adaptive Custom Service Resources Optimization

The main core algorithms includes the services customer preferred selection, services aggregation with customization, self-adaptive optimization, services evaluation algorithm, customization management process, customization information feedback and services re-aggregation with joined custom services and application effectiveness analysis for services resource customized production.

Self-adaptive optimization algorithm partly adopts the early results of ours effort, namely SSOA (Space Search Optimization Algorithm) (see Algorithm 1). In custom resource provisioning, selection and search efficiency will be improved. Space search algorithm uses search operation to realize optimization object. Algorithm starts from the known solution, and also produces new subspace and searches this subspace.

```
INPUT:solution set (population)
OUTPUT:optimization space
Begin
  Initialization:
    1) Initialize a solution set (population) at random.
    2) Opposition-based space search.
  While (the termination conditions are not met)
    IF (rand(0, 1) < C_r) // C_r is a fixed given number
      Local space search:
        1) Generate a new space: Generate a new space based on three given solutions.
        2) Search the new space: Reflection, Expansion, and Contraction.
      Global space search: Cauchy search (Cauchy mutation).
    Else
      Opposition-based space search.
  End While
End
```

Algorithm 1. SSOA Algorithm pseudo code.

The algorithm consists of three types of space search operations.

1. Local space search:
   The operation has been improved based on the Simplex algorithm (plus the search with constraint conditions)
2. Global space search:
   Essentially, the operation is Cauchy mutation.
3. Reverse operation:
   To accelerate the convergence speed of the algorithm, the operation referred the "reverse number". The result has proved better than pure random search.
4. Algorithm characteristics:
   SSOA has stronger local search ability. For example, compared with most of the DE algorithm, SSOA algorithm has relatively stronger global search ability, this is because it includes Cauchy mutation. SSOA algorithm has faster convergence speed.
5. Algorithm advantages:
   The experimental results have showed that SSOA has faster convergence speed, and has more possibility to obtain the approximate solution or more precise value compared some famous DE algorithm. Especially in the high dimensional optimization problem, these advantages perform more outstanding.

3.2 Custom Service Resources Management Monitoring Mechanism

On-demand customization of services resource needs runtime abnormal monitoring to trigger the customization process. So services quality evaluation metrics should be explored. Monitoring mechanism needs to define the boundary conditions of abnormal action, and it is also a real-time system that must meet the demand of real-time triggering, releasing and feedback.

When an exception occurs, the process of services aggregation will be interrupted. Services aggregation should be restarted and worked again with the recovery of unavailable services. Implementation method is drawn lessons from the scientific workflow's transaction in that the process either completed or terminated. At the same time, related process operation data should be retained. For the terminated process aggregation again, resumed process does not affect the services resource providers. So related data tables of system database are required to design. These data records will support the aggregation restart again.

The main operation points of services resource custom management method are as follows.

1. We propose a personalized active custom approach driven by requirements fragments for services requester-centric SOA with adaptive mechanism. It will reinforce current status for customized services resource without runtime on-demand customization.
2. We have designed a full set of architecture and implementation including self-adaptive custom, services aggregation restart and exception handling monitor for abnormal case of user services. Also, feasibility and simplicity will be focused.
3. Through some mathematical methods such as custom optimization algorithm of services resource, exception handling ability with adaptability will be built to optimize and quantity services resource production in runtime.

3.3 Runtime Adaptive Adjustment Mechanism with the Changes

By real-time adjusting the runtime software architecture to adapt to the requirements and context changes, software system can guarantee its performance under dynamic load. Due to lack of general method, how to adaptive map the requirements of problem space into architecture evolution of solution space, it will become a key issue. This section presents an adaptive control method based on predictive control, combining with the requirements model and software architecture to drive the adaptive SOA. This approach adopts learning model based on wavelet transform in order to accurate/flexible predict the evolution of services resource. Through real-time predictive control induced requirements model changes to realize the evolution of software architecture in runtime, the runtime adaptive adjustment of SOA should be finished.

Now runtime adaptive adjustment is still a difficult problem for Internet-intensive software system. It needs to solve how to map requirements changes into architecture units at runtime. Driven by predictive control for SaaS components to induce requirements evolution, runtime architecture change will be

realized, and the validity of predictive control can be proved in the aspect of requirements/architecture evolution [10]. But this method is not extended to SOA level with requirements change-driven architecture evolution.

As shown in Fig. 2, we propose the runtime adaptive adjustment scheme combined with effective predictive control method [10] and MAPE-K control circuit model [11]. The proposed adjustment system divided into real-time monitor, analysis engine, software architecture adjustment manager, requirements evolution manager and Aspect execution engine. First of all, the real-time monitor acquires QoS values of services resource and saves into the log records. Then the QoS value will be transferred to the analysis engine that uses analysis model based wavelet transformation to predict next QoS value according to the logs. Software architecture adjustment manager seeks the best runtime design decision under the requirements constraints according to the predicted QoS value and the current QoS values. If the feasible runtime model can be found, the Aspect script will be automatically generated. Execution engine runs the script to complete the model transformation at runtime. Otherwise, design decision manager should identify the specific improvement points of requirements evolution to induce these changes.

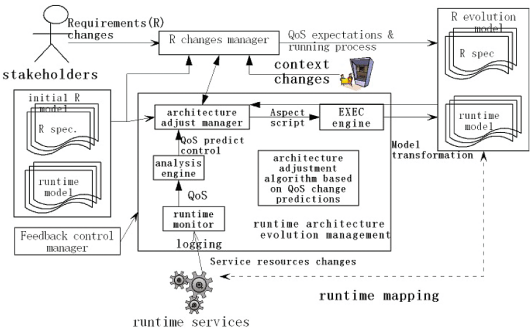

Fig. 2. Runtime adaptability to deal with the changes of requirements and scenario

The core points of the scheme are as follows.

(1) How to select the predictive QoS value for runtime services resource? Wavelet transformation is the best choice.
(2) How to build software architecture adjustment mechanism based on QoS changes to predict in advance? The runtime adaptive adjustment algorithm is designed (Algorithm 2) for solving the above concerned points.

1 INPUT: QoS value of service resources for SBS in time t and t+1; QoS value of the expected output.
2 OUTPUT:control operation vector in t+1 moment

```
Begin
  Initialization: training classification prediction model; the points of tags to improve
training requirements model;
  IF Classification prediction (QoS value of services resource in run time
t and t+1, expected output of QoS value) = requirements
    THEN
      control operation vector in t+1 moment=tag improvement point
(QoS value of services resource in t and t+1 moment at runtime, the expected output
of QoS value)
    ELSE
      control operation vector in t+1 moment = architecture evolution (AOP style)
(QoS value service resources in t and t+1 moment at runtime, the expected
output of QoS value)
  END IF
 RETURN control operation vector in t + 1 moment
End
```

Algorithm 2. Runtime adaptive adjustment algorithm.

4 Experiment and Empirical Analysis

Hainan agricultural e-commerce platform-NongBo Mall[4] should be chosen as experimental carrier which integrated the Internet of things, cloud and big data technology to build a service-oriented Internet based application software system. NongBo Mall is a service-oriented platform faced with richness stakeholders, diversity personalized requirements, thus leading to a variety of personalized customization requirements. Meanwhile, the platform adopts SOA development style and a large number of service resources (including Microsoft Asmx or Java Axis) have developed. The above characteristics conform to exploring empirical collaborative SOA carrier requirements. The author's team is the technical support of online/offline design for NongBo Mall. Close cooperation can help research achievements timely and effective application for Hainan agricultural electric business platform. Through continuous iteration, we can get a comprehensive CASE tool needs to support collaborative SOA.

A large number of services have been developed for the platform. For example, only for agricultural products traceability information service, the WCF interface service address is as follows:

http://218.77.186.198:8000/TracesDataService.svc

Web service interface address is:

http://218.77.186.198:8000/TracesDataWebService.asmx

Its identification code is 2C516EF7-CBD8-4C1C-9EE0-00EB34AFBCB5.

[4] http://www.963110.net.

The test data:

PID(Product ID): A003121910013001
PRID(Planting ID): XH03020130401

For example, production history batch No. is
ProductionHistoryGetProductionHistoryByPID(string PID, string IDs)
Parameters:

PID: 16 digits batch no.
IDs: identification codereturns a ProductionHistory object.

The main data structures, such as ProductionHistory (production record) class structure is as follows:

$\langle summary\rangle$ Batch no. $\langle /summary\rangle$
$publicstringPID\{get; set;\}$
$\langle summary\rangle$ Planting number $\langle /summary\rangle$
$publicstringPRID\{get; set;\}$
$\langle summary\rangle$ Planting time $\langle /summary\rangle$
$publicDateTimePlantTM\{get; set;\}$
$\langle summary\rangle$ Picking time $\langle /summary\rangle$
$publicDateTimePickTM\{get; set;\}$
$\langle summary\rangle$ Pesticide recordset $\langle /summary\rangle$
$publicList\langle PesticideHistory\rangle PesticideHistoryList\{get; set;\}$
$\langle summary\rangle$ Fertilization recordset $\langle /summary\rangle$
$publicList\langle FertilizerHistory\rangle FertilizerHistoryList\{get; set;\}$

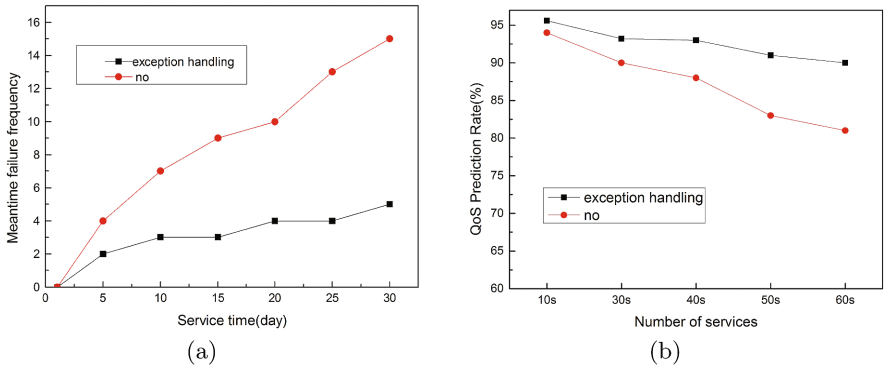

Fig. 3. Effect comparison for E-business platform using exception handling mechanism

In addition to platform-developed services, system also has called a large number of external services, such as map services, weather services and other physical services etc., which are typical SBS applications. In the early stage, platform operation is extremely unstable. No corresponding runtime exception

handling mechanism is the main reason. Figure 3(a) is the comparison result for the two sets of same system platform running at the same time. One of platforms is running under the part of the designed exception handling mechanism in this paper. Comparison results (Fig. 3) show that the embedded runtime exception handling mechanism obviously improved the platform's ability (including the rate of QoS prediction) to deal with all kinds of uncertainty.

5 Conclusions

The main contributions of this paper are summarized as follows: (1) Self-adaptive exception handling architecture for active services resource provisioning has been built. (2) Runtime adaptive adjustment mechanism has been designed to deal with requirements and context changes. This paper gives a general solution framework, but the details are still to be further in-depth study.

The paper aims at explore the collaborative SOA with runtime exception handling to enhance the operation reliability. We mainly focus on collaborative SOA runtime exception handling mechanism. The results and progresses of the study will provide systematical solutions to perfect the collaborative SOA.

Acknowledgements. This research has been supported by the Natural Science Foundation of China (No. 61562024, No. 61463012) and Natural Science Foundation of Hainan Province (No. 20156236).

References

1. Liu, L.: Editorial: services computing in 2016. IEEE Trans. Serv. Comput. **9**(1), 1 (2016)
2. Zhang, L.J.: Big services era: global trends of cloud computing and big data. IEEE Trans. Serv. Comput. **5**(4), 467–468 (2012)
3. Bin, W.: On-demand Service Software Engineering for Cloud Computing. National Defence Industry Press, Beijing (2014)
4. Ghezzi, C.: Surviving in a world of change: towards evolvable and self-adaptive service-oriented systems. In: Keynote Speech at Proceedings of the 11th International Conference on Service Oriented Computing, ICSOC 2013. Springer, Heidelberg (2013)
5. Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.): ICSOC 2011. LNCS, vol. 7084. Springer, Heidelberg (2011)
6. Yau, S.S., An, H.G.: Software engineering meets services and cloud computing. Computer **44**(10), 46–52 (2011)
7. Allier, S., et al.: Multitier diversification in web-based software applications. IEEE Softw. **32**(1), 83–90 (2015)
8. Lemos, A.L., Daniel, F., Benatallah, B.: Web service composition: a survey of techniques and tools. ACM Comput. Surv. **48**(3), 1–41 (2015)
9. Chouiref, Z., Belkhir, A., Benouaret, K., Hadjali, A.: A fuzzy framework for efficient user-centric web service selection. Appl. Soft Comput. **41**, 51–65 (2016). Elsevier
10. Xiong, W., et al.: A self-adaptation approach based on predictive control for SaaS. Chin. J. Comput. **39**(2), 364–376 (2016)
11. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–45 (2003)