An Analysis Method for Error Propagation Reachability of Component-Based Software

Mingqi FAN¹, Min SONG²*, Zhengxian WEI³,⁴, Wanfeng MAO³
¹ Equipment Procurement Center of Navy Equipment Department, Beijing, China
² Information Technology Center, Beijing Foreign Studies University, Beijing, China
³ Systems Engineering Research Institute, Beijing, China
⁴ Joint Laboratory for Smart Ocean Technology, Pilot National Laboratory for Marine Science and Technology, Qingdao, China

Corresponding author: Min SONG, e-mail: songmin@bfsu.edu.cn

Abstract. Component-based software (CBS) is widely used in various industries and fields. One important characteristic of CBS is that when some error occurs in a component, the error will be transmitted to other components as the inter-component state propagates, the error propagation will form and the error propagation may cause the software system to malfunction or fail. From the perspective of software architecture (SA), an analysis process for error propagation accessibility of CBS and a generation algorithm for error propagation reachability graph event/state-based of CBS were established. In this algorithm, firstly, the elementary events are sorted by identifying the internal timing sequence of the components in the directed graph of SA, and the state groups that cannot be reached at the same time between the components are obtained; secondly, the Cartesian product of the reachable states between the components is established and all the reachable state groups of CBS are obtained; at last, the error propagation reachability graph of CBS is obtained according to the event connection table. The algorithm has laid a technical foundation for the analysis on CBS errors and reliability.

1. Introduction

Component-based software (CBS) is widely used in various industries and fields. At present, many industries such as the ship industry, the aviation industry and the ordnance industry have adopted this software mode in their software. Service oriented architecture (SOA), Web services and internetwork are typical representatives of this software mode. As a kind of new software form based on network platform, CBS is the software system in which various software entities in the network (e.g. components, services, agents, which are collectively called “components” in this paper) exist in all the nodes in an open and independent form and implement cross-network interconnection, intercommunication and collaboration in various collaborative methods in an open environment. At present, many industries such as the ship industry, the aviation industry and the ordnance industry have adopted this software mode in their software. With the development of technology, CBS will be more and more characterized by independent adaptability, collaboration, reactivity, evolution, diversity, etc., and provide customers with the services featuring 7*24, continuous evolution and update, and smartness. Meanwhile, CBS can perceive the dynamic changes of the environment and carry out online dynamic evolution in accordance with the function indexes, performance indexes, credibility indexes, etc. [2,3].
The reliability analysis of CBS is one of the hot research topics at present \cite{4,5,6}. The relation between a software error and the error propagation is an important basis for the software reliability analysis \cite{7,8,9}. One important characteristic of CBS is that when some error occurs in a component, the error will be transmitted to other components as the inter-component state propagates, the error propagation will form and the error propagation may cause the software system to malfunction or fail. From the perspective of software architecture (SA), the paper reified the software architecture into a directed graph, analyzed the transitivity of the elementary event (internal activity or state of the component which causes an error) interaction at the interfaces between components, then an analysis process for error propagation accessibility of CBS was established. More important, a generation algorithm for error propagation reachability graph event/state-based of CBS was proposed. In this algorithm, firstly, the elementary events are sorted by identifying the internal timing sequence of the components in the directed graph of the architecture, and the state groups that cannot be reached at the same time between the components are obtained; secondly, the Cartesian product of the reachable states between the components is established and all the reachable state groups of the software system are obtained; at last, the error propagation reachability graph of the software system is obtained according to the event connection table. The algorithm has laid a technical foundation for the analysis on CBS errors and reliability.

2. Analysis process for error propagation accessibility of CBS

One important characteristic of CBS is that when some error occurs in a component, the error will be transmitted to another component as the inter-component data state propagates, may cause some error to occur in the component and some error propagation to form. The scope and reachability of the error propagation needs to be analyzed. CBS is composed of components, and the functional (or service) interaction between components is implemented through the interfaces. Therefore, to analyze the errors of CBS, it is necessary to start from the perspective of software architecture (SA), and obtain the error behavior of component $C_i$ as well as the error behavior of the interaction $I_{ij}$ of the interfaces between components \cite{10}.

Currently, it is widely believed that the components, as well as the interactive relationship between components, work together to form the software architecture \cite{2}. Component $C$ consists of two parts, i.e., the interface specification and internal specification of the component. The interface specification is divided into two types, the first is called the service interface (provided/service/public interface), denoted as $I_p$, and it is used by component $C$ to provide functions (services); the other is called the request interface (required/entry interface), denoted as $I_e$, and it is a function (service) interface needed by the component $C$.

If a component is used to accomplish certain functions or provide corresponding services, these will be implemented through the activities and states of the internal functional domain of the component. The relationship between the activity and state of the internal functional domain of the component is shown in Fig 1. In Fig.1, the request interface $I_e$ can be used to input the states or activities, and in the meantime, the service interface $I_p$ can be obtained from the activities or states. In the CBS research process, when the interactive relationship of the interfaces between components is analyzed, the implementation of the internal functions of the component can be ignored and the component can be treated as a black box. If an analysis involves the internal function implementation of a component, it shall be necessary to pay attention to the relationship between the activities and states within the component. Due to the directivity of the interactive relationship of the interfaces between components, the architecture could be denoted as a directed graph $SA:=G(C,I)$, where $C$ refers to the component set and $I$ refers to the interface interaction set. If the implementation process within the component is involved in the analysis, the description method of the activities and states within the component as shown in Fig.1 shall be adopted. In addition, it is assumed that there are no loops in the directed graph of the architecture.
It is pointed out in [10], when the error propagation and reliability of the system is analyzed, three kinds of information shall be obtained: (1) architecture of the software system; (2) error behaviors of component $C_i$; (3) error behaviors of the interface interaction $I_{ij}$ between components. An activity or state of a component which causes an error is called an “elementary event”. The occurrence of an elementary event doesn’t necessarily cause a software system to malfunction or fail. Only when an elementary event occurs, it propagates through the interactive relationship of the interface between components and results in the loss of the user services or the system function performance, will it cause the software system to malfunction or fail (It is called a top-layer event). The occurrence of a top-layer event of the software system depends on the occurrence sequence and propagation route of the elementary event. In CBS, the elementary events are transmitted as the states propagate among components. In this paper, a state that can propagate carrying an error is called an error state. In the process of the elementary event (error) propagation, the interactive relationship of the interface determines whether the elementary event that had occurred has effects on the system. To handle the state space explosion is the key to the analysis on the error propagation and reliability of the software system[8]. To solve the state space explosion problem effectively, an error propagation reachability graph is established. Firstly, elementary events are sorted by identifying the internal timing sequence of the components in the directed graph of the architecture, and the state groups that cannot be reached at the same time between the components are obtained; secondly, the Cartesian product of the reachable states between the components is established and all the reachable state groups of the software system are obtained; at last, the error propagation reachability graph of CBS is obtained according to the event connection table.

In CBS, when event $t_i$ occurred within component $C_i$, if event $t_j$ occurred within component $C_j$ through interface $I_{ij}$, it is called “event $t_i$ triggers event $t_j$”. In fact, the triggering relationship between events reflects the propagation of the events with the propagation of the states. The triggering relationships between component events form through their interface interactions, and the interaction relationships of the interfaces between components are divided into direct interactive relationships and indirect interactive relationships. The event triggering relationship between components is also divided into two kinds: direct triggering relationships and indirect triggering relationships. As shown in Fig.2, component $C_2$ triggered the event of component $C_1$, $t_1$ through event $C_2$, $t_1$; component $C_1$ triggered the event of component $C_3$, $t_1$ through event $C_1$, $t_2$. Therefore, there was a triggering relationship between components $C_2$ and $C_3$. 

Based on the event triggering relationship and interface interaction relationship shown in Fig. 2, the analysis process for error propagation accessibility of CBS is shown in Fig. 3.
3. **Generation algorithm for error propagation reachability graph event/state-based of CBS**

In the process of the analysis on the CBS error propagation reachability, the proration of events and states are bound together. In order to analyze the CBS error propagation relationship, a generation algorithm for error propagation reachability graph event/state-based of CBS is proposed. The specific steps of the algorithm are shown as follows:

**Step1:** sort the events with some triggering relationship according to the internal activity timing sequence of the component.

1. Sort the internal events of the components according to the internal activity timing sequence of the component.
2. For the components with a direct triggering relationship, the occurrence time of the triggering event is equal to that of the triggered event.
3. For the components with an indirect triggering relationship, the timing sequence relationship of the events between nonadjacent components is established according to the activity timing sequence of the triggering relationship of the intermediate component.

Three rules are proposed based on the above three event timing sequence relationships:

Rule 1: For component \( C_1 \) containing two events \( t_1 \) and \( t_2 \), if the occurrence time \( t_1 \) of the event is earlier than \( t_2 \), then there is a timing sequence relationship: \( t_1 < t_2 \);

Rule 2: For the existing components \( C_1 \) and \( C_2 \), and there is a triggering relationship between event \( t_1 \) in component \( C_1 \) and event \( t_2 \) in component \( C_2 \), there is timing sequence relationship: \( t_1 = t_2 \);

Rule 3: For component \( C_1, C_2 \) and \( C_3 \), if there is a triggering relationship between event \( t_1 \) in component \( C_1 \) and event \( t_1 \) in component \( C_2 \), there is also a triggering relationship between event \( t_2 \) in component \( C_1 \) and event \( t_1 \) in component \( C_3 \), and \( t_1 < t_2 \), then there is timing sequence relationship \( C_2, t_1 < C_3, t_1 \), as shown in Fig. 4.

![Fig. 4 Rule3 the event timing sequence between components](image-url)
For component $C_1$ containing event $t_1$ and $t_2$, according to the timing sequence activities, if $C_1.t_1 < C_1.t_2$, $C_1.t_2 = C_2.t_1$ and $C_1.t_2 = C_3.t_1$, then $C_2.t_1 < C_3.t_1$. Then if $t_1$ is connected with the initial state, then the preceding state of the triggered event $t_1$ in $C_2$ and the subsequent state of the triggered event $t_1$ in $C_3$ form the unreachable state group.

**Step 2**: Calculate the preceding state and subsequent state of the event triggering relationship.

The unreachable state group refers to the combination of two states of different components. They cannot be activated simultaneously due to the timing sequence relationship. Define the reachability graph of a complete system as the state reachability graph of the system and the internal reachability graph of a component as the state reachability graph of the component.

1. Identify the preceding state set of the triggering event. Identify all the unreachable states of the triggering events according to the states of the component, use the Dijkstra algorithm beginning from the direct preceding state of the event. As for the unreachable state, the distance is infinitely great.

2. Identify the subsequent state set of the triggered event. Identify all the unreachable states beginning from the initial state according to the states of the component, implement the Dijkstra algorithm beginning from the initial state and identify all the unreachable states. The distance is infinitely great. Therefore, only the state which is directly reachable through the triggering relationship is used as the subsequent state of the triggered event.

**Step 3**: Identify unreachable state groups according to the timing sequence relationship.

1. Identify unreachable state groups with direct event triggering relationships between components according to the timing sequence relationship.

For the components with direct event triggering relationships, according to the timing sequence relationship between the triggering event and the triggered event, the preceding state of the triggering event and the subsequent state of the triggered event are unreachable. In this case, establish the Cartesian products of the preceding states and the subsequent states, and obtain the unreachable state groups. In the meantime, if a triggered event is uniquely connected with an initial state, then establish an unreachable state group according to the preceding state of the triggered event and the subsequent state of the triggering event. As shown in Fig. 5, component $C_1$ causes triggering event $t_1$, if component $C_2$ is in the state of $s_1$, then the event $t_1$ is triggered and it will reach the state of $s_2$.

![Fig.5 Component relationship diagram](image)

In Fig. 5, if there are successors $(C_2.t_1) = \{C_2.s_2\}$, predecessors $(C_1.t_1) = \{C_1.s_1\}$ and successors $(C_1.t_1) = \{C_1.s_2\}$, then the unreachable state group is: $\{C_1.s_1, C_2.s_1\}$, $\{C_2.s_1, C_1.s_2\}$.

2. Identify unreachable state groups with indirect triggering relationships between components according to the timing sequence relationship.

For components $C_1$ and $C_2$ with some indirect timing sequence relationship, there is $C_1.t_1 < C_1.t_2$ based on the activity timing sequence of the component, then $\{C_1.s_1, C_3.s_2\}$ is the unreachable state group with an indirect timing sequence relationship, as shown in Fig. 6.

**Step 4**: obtain all the reachable state groups of a system.

For the CBS $S = \{C_1, C_2, ..., C_k\}$, where $C_i = \{s_1, s_2, ..., s_m\}$ indicates that component $C_i$ contains $m$ states, and then establish Cartesian products of the reachable states between components.

Let the component set involved in the combinatorial computing of the Cartesian product be $D$, the reachable state group be $R$, and initially the set $D$ is empty, then $R$ is initially empty too.
Fig. 6 Component relationship diagram

1. At the initial time, take out \( C_i \), \( D = \{C_i\} \), \( S = S - D \), \( R=\{s_0, s_1, s_2, ..., s_m\} \) and \( i = 1, ..., k \) from the component set \( S \).

2. Select a component \( C_j \), \( D = \{C_i, C_j\} \) and \( (1 \leq i, j \leq k) \), from \( S - D \). For any state set \( R_i \) in \( R \), traverse the unreachable state set generated by the sub-states \( R_i \) and \( C_j \) in step (3), remove the states of the components with unreachable relationship in \( C_j \) and obtain \( \overline{C_j} \), then implement the combinatorial computing \( R = R \times \overline{C_j} \).

3. Repeat step (2), until all the components are included in \( D \), \( S \) is empty and \( R \) contains all the reachable state groups.

Step 5: Establish a connection table and obtain an error propagation reachability graph.

In CBS, each event corresponds to one activity or state, and each event or activity has a connection mode. Therefore, there is a connection mode corresponding to each event. The mode describes the transfer from the preceding state to the subsequent state. Therefore, an event state connection table is established according to the connection mode, and the first column of the table contains the only allowed switching between the two states. Remove all the unreachable states from the initial state in the graph, and then we obtain error propagation reachability graph of CBS.

4. Conclusion

CBS is widely used in various industries and fields. One important characteristic of CBS is that when some error occurs in a component, the error will be transmitted to other components as the inter-component state propagates, the error propagation will form and the error propagation may cause the software system to malfunction or fail. From the perspective of software architecture (SA), this paper reified the software architecture into a directed graph, analyzed the transitivity of the elementary event interaction at the interfaces between components, then an analysis process for error propagation accessibility of CBS was established and a generation algorithm for error propagation reachability graph event/state-based of CBS was proposed. The analysis process and algorithm have laid technical foundation for the analysis on CBS errors and reliability.

Acknowledgments

This work is sponsored by the National Natural Science Foundation of China under Grant No. 61502037 and 61772152, the Basic Research Project No. JCKY2016206B001 and JCKY2016604C010, and Innovation Special zone of National Defense Science and Technology Project.

References

[1] Tomar P, Mishra R, Sheoran K. Prediction of quality using ANN based on Teaching-Learning Optimization in component-based software systems[J]. Software Practice & Experience, 2018, 48(3).

[2] Fuqing Yang: Thinking on the Development of Software Engineering Technology. Journal
[3] Jian Lv, Xiaoxing Ma, Xianping Tao, Feng Xu, Hao Hu: Research and development of network software[J]. Science in China. 2006, 36(10): 1037-1080.

[4] Ling, Dongyi , B. Liu , and S. Wang. A component-based software reliability assessment method considering component effective behavior[C]. Second International Conference on Reliability Systems Engineering IEEE, 2017.

[5] Koziolek Heiko, Schlich Bastian, Becker Steffen. Performance and reliability prediction for evolving service-oriented software systems[J]. EMPIRICAL SOFTWARE ENGINEERING, 2013, 18(4):746-790.

[6] Joao M Franco, Raul Barbosa, Mario Zenha-Rela. Reliability Analysis of Software Architecture Evolution[C]. In: Proc. of the 6th Latin-American Symposium on Dependable Computing (LADC), 2013: 11-20.

[7] Hayat, Tayebe, and H. Seifzadeh. A New Approach to Reliability Prediction in Component-based Systems[C]. International Conference on Computer Modeling & Simulation 2017.

[8] Li, Kewen , et al. Reliability Evaluation Model of Component-Based Software Based on Complex Network Theory[C]. Quality and Reliability Engineering International (2016).

[9] Singh A P, Tomar P. A Proposed Methodology for Reliability Estimation of Component-Based Software[C]. International Conference on Optimization Modeling and Applications, 2012.

[10] Uokhale S S, Wong W E, Horgan J K, et al. An analytical approach to architecture based software performance and reliability prediction[J]. Performance Evaluation, 2004, 58(4): 391-412.

[11] YIN Guisheng, SONG Min, WEI Zhengxian, WANG Hongbin, ZHANG Wansong. Analysis of software architecture evolution orienting semantic relationship between components[J]. Journal of Harbin Engineering University, 2011, 32(10),1329-1335.

[12] Song M, Wei ZX, Yin GS. Evolution analysis of data flow oriented internetware service[J]. Journal of Software, 2013, 24(12):2797-2813.