Software Adaptive Mechanism Based on Software Architecture in Software Running Time

Haiyun XIANG[1, a], Xiao FU[2], Xu LI[3]

[1]Modern Educational Technology Center, Southwest Petroleum University, Si Chuan, ChengDu, China

[2]College of Computer Science , Southwest Petroleum University, Si Chuan, ChengDu, China

[3]Modern Educational Technology Center, Southwest Petroleum University, Si Chuan, ChengDu, China

[a]email: l xianghaiyun_xa@163.com

Keywords: Bigraph; Response System; Adaptive Software; Software Architecture; Formal Methods

Abstract. In the Internet environment, software gradually moves from closed, static and controllable status towards open, dynamic and unpredictable state. How to propose suitable software theory for such adaptive software has become the challenging issue facing the computer science and technology. Applicable formal theoretical basis is one of signs that software technology achieves maturity, while support for protocol, analysis and verification of adaptive software architecture is inadequate in existing mobile and concurrent theories. Although the software architecture technology has now entered into the golden era of development, there are still many issues to be resolved, one of which is the need for effective mechanism to describe, analyze and verify software architecture. Bigraph puts emphasis on two factors of calculated position and connection on the basis of the existing theories, and a relatively complete and extensible theoretical framework is established. Nowadays, bigraph theory has now begun to be applied, and studies are gradually carried out on extension and shift of bigraph theory basis, description of concurrency theory, bigraph logic, modeling of pervasive computing system, and BPL programming language of bigraph. Hence, bigraph theory can provide a solid foundation for the formal methods of adaptive software architecture.

Introduction

In recent years, with the osmotic expansion of Internet to every corner of society, and the rapid development of the computing models typically represented by pervasive computing, grid computing and network configuration software and so on, the computing environment facing software system becomes open, pluralistic and variable. To adapt to changes in the network, equipment, resources and in users' needs in run-time computing environment, people have a growing demand for adaptive software. Adaptive software usually includes the software environment itself and the external environment, able to adapt to changes in needs and the environment in run time by adjusting its structure or behavior. At the same time, the complexity of the adaptive software also brings increasing challenges: how to understand, describe and model such applications? What kind of theory suitable for analysis and verification of the nature of such evolutionary systems? Which elements need to be extended for existing theories to support the development of such complex systems? Etc.

Bigraph Model of Adaptive Software Architecture

Combined with our research work, this section uses bigraph theory and existing research results to explore the formalization of adaptive software architecture. Formalization of dynamic architecture should include structure, behavior and change. Thus, formal methods of adaptive software architecture should cover change in environment, structure, behavior and the relationship

among them, thus bigraph needs to be used to describe external environment and the architecture itself, of which the architecture also includes the structure and behavior.

Structure

Next, description is made by the case of three-tier Client-Server architecture style. A network information service system is considered and the user can send service requests to the distributed server via Core. In a dynamic network environment, the number of client and server is constantly changing. For example, users can join or leave, and the server can be disconnected from the Core in the absence of the requested services. Thus, such architecture can be adjusted adaptively according to the requesting number of the users. Figure 1 shows a specific operation of such architecture style.

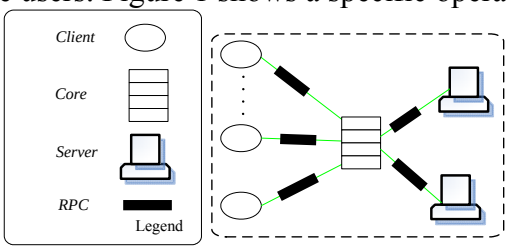


Fig.1. Bigraph description of living example of three-tier Client-Server style

Components and connectors of the example include: *Client*{port *request*}, *Server*{port *reply*}, *RPC*{role *caller*, *callee*}, *Core*{port $\overrightarrow{in}, \overrightarrow{out}$}.

We regard specific architecture as a bigraph, and changing operations of the architecture include adding, deleting and replacing members or linkers, and configuring the ports and the connectivity of roles and so on. We can modify bigraph to achieve operations of these architectures. Among them, A represents architecture; U and V are the items in A; P represents members or linkers; p represents the port and r is role.

定 Definition 1 (substitution) assuming U, V and F are elements of bigraph, and $S_V^U F$ means any V arisen in F replaced by U.

Add(*P*, *U*(*V*), *A*)=$S_{U(V)}^{U(V\|P)}A$	// Add P to U
Remove(*P*, *A*)=$S_P^1 A$	//Remove P in A
Replace(*P′*, *P*, *A*) = $S_P^{P'} A$	//Replace P with P in A
Connect(*p*, *r*, *A*) =$/_{x.x}/_{pr}.A$	/ *Connect p and r*
Disconnect(*p*, *r*, *A*) = $S^e_{/x.x/pr}A$	//Disconnect p and r
Rely(*t*, *t'*, *A*) =$/_{x.x}/_{tt'}.A$	/*Rely t and t '*
Disrely(*t*, *t'*, *A*) = $S^e_{/x.x/tt'}A$	//Disrely *t and t '*

For example, a Client is added in this case and we can describe through operations in the following sequence:

Add a Client {

Add(*Client*$_{request}$, *S*)

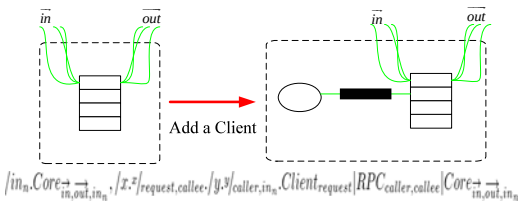
Add(*RPC*$_{caller,callee}$, *S*)

Connect(*request*, *caller*, *S*)

Connect(*in*, *callee*, *S*)

}

In addition, we can use reaction rules in BRS to represent structural changes, so as to visually represent the effect of architecture operations. For example, we use reaction rules of Figure 2 to show the addition of a Client.



$/_{in_n}.Core_{\overrightarrow{in},\overrightarrow{out},in_n}.\,/_{x.z}/_{request,callee}.\,/_{y.y}/_{caller,in_n}.Client_{request}|RPC_{caller,callee}|Core_{\overrightarrow{in},\overrightarrow{out},in_n}$

Fig.2. Reaction rule after addition of a Client

Behavior

In terms of the architecture system, most of the existing formal methods employ π calculation,

graph theory and other theories, and as described above, one of the bigraph goals is to provide unified framework for mobile and concurrent theory. Thus, these formal methods are described by bigraph, and that is, these acts protocols are directly converted to bigraph models.

For changes in components or linkers behavior, we can conduct representation through join, quit, activate and deactivate operations in dynamically bind mechanisms. Thus, the components and connectors can dynamically join or exit certain behavior specification, and the behavior specification can be placed in the active or inactive state to exhibit different behaviors in a changing environment. Status and operations of behavior specification in dynamic binding mechanism are shown in Figure 3.

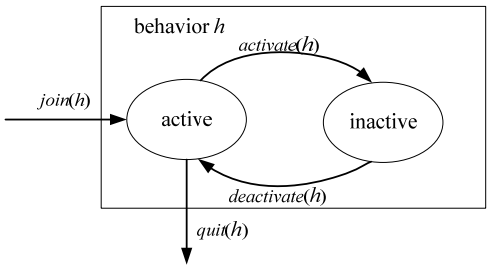


Fig.3. Dynamic binding mechanism

For example, if we have added Authentication component in the Core, then the initial behavior specification of Core is Ordinarybehavior, and certified behavior specification is Safetybehavior, specifically described as follows:

Ordinarybehavior = via in receive n; inaction; via out send n;

Safetybehavior = via in receive n; if authentication (n) then {via out send n} else {via in send FAILURE};

The changes in behavior can be explained by dynamic binding operation, namely *quit*(Ordinarybehavior); *join*(Safetybehavior);

$H' = \mathbf{join}(behaviour) = H \mid behaviour$

$H' = \mathbf{quit}(behaviour) = S^{1}_{behaviour} H$

$H' = \mathbf{activate}(behaviour) = S^{behaviour}_{INACTIVE(behaviour)} H$

$H' = \mathbf{deactivate}(behaviour) = S^{INACTIVE(behaviour)}_{behaviour} H$

Environment

As introduced in section 2.2, bigraph has universal expressiveness. Therefore, users can define environment in their own way. For example, nodes in the bigraph may represent positions, entities, events, etc., and edges can represent connections, relationships, trigger conditions and so on. We may also introduce environmental information for the example: Container is employed to indicate that the current system may also accept the number of Client (corresponding to the number of circles inside). Added response rule of a Client is shown in Figure 4.

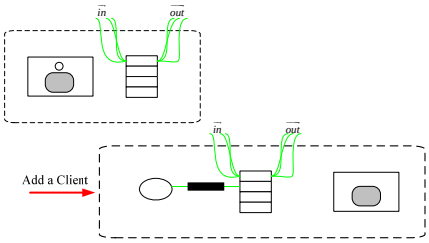


Fig.4. Reaction rule with environmental information

In terms of adaptive software, the environmental changes will have an impact on the architecture, which leads to changes in the structure and behavior. Changes in the relationship between environment, structure and behavior can be expressed as:

```
// Environmental changes cause structural and behavioral changes
EnvChange→(StructuralChange | BehavioralChange)+
// In the structural changes, op is the architecture operation
StructuralChange::=op+ | reaction rule
// Behavioral change is operated with dynamic binding
BehaviralChange::=(join(h) | quit(h) | activate(h) | deactivate(h) )+
```

Property Verification

When these changes are implemented in run time, it needs to ensure that these changes do not undermine the consistency and integrity of the system structure and behavior, which is a necessary condition for the implementation of adaptive evolution.

Consistency

Definition 2. For behavior expressions of BE1 and BE2, it is called that BE2 simulates behavior BE1 (abbreviated as $BE_1 \preccurlyeq BE_2$), if one of the following conditions are met:

$BE_1 = BE_2$

For reaction rule (r, r'), if $BE_1 \rightarrow BE_1'$, then there exists BE_2', thus $BE_2 \rightarrow BE_2'$ and $BE_1' \preccurlyeq BE_2'$.

From Definition 3, we can believe that $BE_1 \preccurlyeq BE_2$ means that disposing capacity of BE2 is the same with BE1 or stronger than BE1, which can also be regarded that BE1 is the refinement of BE2.

Definition 3 For behavioral expression BE1 and BE2, if there must be a reaction sequence tr formed by response, making $BE_1|BE_2 \rightarrow inaction$, thus it is called that BE1 and BE2 are compatible (abbreviated as $BE_1 \cong BE_2$).

Definition 4. For port p and role r, their behavioral expressions are BEp and BEr. If $BE_p \cong p/r.BE_r$ is met, it is called that p and r are compatible (abbreviated as $p \cong r$).

This definition indicates that their behavior must be compatible, and namely, interaction must be able to be completed, and the situations similar to deadlock cannot occur.

Integrity

Integrity means that the evolution of the system can not destroy the constraint conditions in specification of architecture. Integrity also means that the state of the system cannot be lost before and after evolution, or the system will become "unsafe", even unable to run correctly. Because evolution is decided by the operation system according to the adaptive rules, thus the integrity needs to be verified.

For example, for the style of three-tier Client-Server, the number of clients and servers is unlimited, but there must be a Core. Thus, this constraint condition in three Client-Server style can be expressed as A = ◇Core with BiLog. As another example, Authentication component is added in order to provide security mechanism, but it needs to be ensured that Authentication component is embedded into the Core. This constraint can be expressed as *A=Authentication–∘Core* with BiLog.

Summary and Further Work

Currently, there are still some issues to be resolved: supervising method of contextual information: the existing system has increasingly requirements for continuous operation, and often requires making respond to changing resources and internal state, thus contextual information needs to be increased to describe conditions causing this change. Hence, to study how to specify contextual information and what methods to guide such a specification is required; adaptive system architecture description language: the existing ADL lacks description of the environment, and its theoretical foundation is also insufficient to verify evolutionary properties of adaptive software. We want to design adaptive software-oriented ADL using bigraph theory; supporting of auxiliary means: Like other formal methods, design, analysis and verification of architecture require large amounts of auxiliary means to support such as: design tools, inspection tools and code generation tools and so on.

Reference

[1] Time-of-Arrival for Wireless Wearable Sensors in Multipath Environment, IEEE Sensor Journal, 14(11), 3996-4006, Nov. 2014

[2] Lv, Zhihan, Liangbing Feng, Haibo Li, and Shengzhong Feng. "Hand-free motion interaction on Google Glass." In SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications, p. 21. ACM, 2014.

[3] Zhong, Chen, Stefan Müller Arisona, Xianfeng Huang, Michael Batty, and Gerhard Schmitt. "Detecting the dynamics of urban structure through spatial network analysis." International Journal of Geographical Information Science 28, no. 11 (2014): 2178-2199.

[4] Li, Wubin, Johan Tordsson, and Erik Elmroth. "An aspect-oriented approach to consistency-preserving caching and compression of web service response messages." In Web Services (ICWS), 2010 IEEE International Conference on, pp. 526-533. IEEE, 2010.

[5] Y. Geng, J. He, K. Pahlavan, Modeling the Effect of Human Body on TOA Based Indoor Human Tracking[J], International Journal of Wireless Information Networks 20(4), 306-317