Distributed systems software architecture modelling and research methods

Xiaochun Zhao*

Heilongjiang University of Science and Technology

Received 12 June 2014, www.cmnt.lv

Abstract

With the development of computer network technology, the open, heterogeneous and distributed systems have become the mainstream in current computer applications because of the sharing resources, high availability, parallel processing and so on. However, due to the problems of development, which are constant expansion of systems size, evolution and continuous improvement, maintenance that required, specific distribution, autonomy and heterogeneity, a lot of research and software development practice shows, the introduction of software architecture which guide distributed system to develop and assume component blueprint is a practical and effective way to solve the difficulties of the development of distributed systems and build distributed systems successfully. Therefore, how to improve the quality and efficiency of distributed systems development by using software architecture, and ensuring system maintenance and space evolution are the key to develop distributed systems, also the core of this study. Software architecture, formal description of distributed systems interaction style, refinement and mapping architecture, distributed architecture systems development methods under evolution and reconstruction driving were studied based on the current distributed systems development methods as well as the problem of inadequate means.

Keywords: Software architecture, distributed systems, interaction style, software architecture refinement

1 Introduction

The main question in distributed system development is: the complexity of design, construction, commissioning, configuration and maintenance distributed applicant system is high in a heterogeneous environment, and causing high costs and low efficiency in developing, large-scale distributed systems development seems a risky challenge. With the component development ideas become mainstream in software development, people gradually realize that the software architecture is an important mean to control software complexity, improve the quality of software systems, support software development and reuse.[1] Therefore, it has immediate practical significance to in-depth research on the software system architecture, explore effective large-scale distributed systems development method, system design, analysis, and tectonic environment of architecture system driven, which help support its entire life cycle.

In this paper, a distributed system software architecture modelling and developing methods have been studied mainly from the following aspects: (1) propose a system architecture abstract model DSAM distributed which is suited system architecture , and give its formal model, on this basis, design and implement an attribute grammar-based software architecture description language DISADL and description language Discid based on CCS distributed component interaction style; (2) propose architecture refinement guiding principles and design a set of mapping rules from a software architecture description DISADL to universal design model UML of software implementation; (3) propose a software architecture reconstruction based on fuzzy clustering analysis; (4) present distribution systems development method of the architecture driven, ADISC, establish its life cycle model, and put into system architecture supporting; (5) design and implement a visual supporting modelling environment for distributed software architecture structure, protocols, analysis, refinement, remodelling design, ADisDTool.

2 A Distributed software architecture description core model-DSAM

Model DSAM as a basis is designed a distributed software system architecture description language with component based by extending the traditional attribute grammars.

2.1 DSAM

Model DSAM includes: Event; Port; Interface; Connector Type; Component Type; Architecture Model. Figure 1 is a graphical representation of DSAM.

* ***Corresponding author*** e-mail: 986889936@qq.com

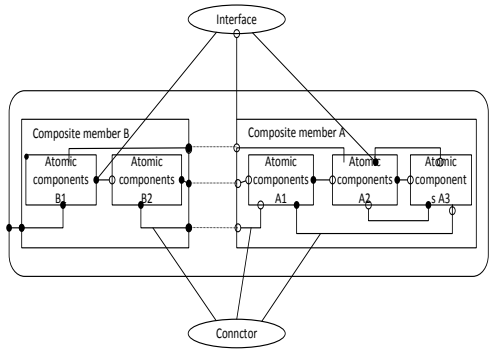

FIGURE 1 Architecture models DSAM

2.2 DESCRIPTION LANGUAGE DIS_ADL

2.2.1 Extending Attribute Grammar of Dis_ADL (EAG)

EAG is a seven-tuple: EAG = (T, N, P, Z, V, F, B), where T is the terminator collection; N is non-terminal symbol set; P is the production; Z is the starting character; V is the attribute range; F is calculation rules for the property; B is a finite set of conditions.

2.2.2 Semantic Description of Dis_ADL Based on EAG

Semantic description based on EAG includes: (a) logic timing and dependencies terminator; (2) parallel description terminator; (3) terminator @ (4) conditions production; (5) time constraints; (6) specific terminator

2.3 DESCRIPTION LANGUAGE - DISCID

Discid was meant to describe component interaction styles and verify formal nature. Discid begins with "Discid". Practical application shows, combination with Dis_ADL and Discid use various forms of mechanisms, describe the architecture from different viewpoints and form an organic whole, which greatly reduce the distributed system designer's cognitive difficulty, and also greatly improve the efficiency of the system design. Meanwhile, DIS_ADL and Discid are uncertainty and incompleteness in modelling, moreover, its adaptive capacity needs to be improved, Discid description needs further implementation language for mapping.

3 Models UML

UML is the most common object-oriented modelling language, modelling the software system by graph mode which from static structure and dynamic behaviour.

3.1 ARCHITECTURE MAPPING FOR

Distributed systems development with the guidance of software architecture build software systems architecture model by DIS_ADL, after the high-level completion of verification, seek to design fine from the top layer constantly, up to a certain size, take the UML as the middle part the DIS_ADL architecture description is mapped to the appropriate elements in UML, and then developed into the design and implementation phase. Therefore, DIS_ADL and UML are combined and complement, DIS_ADL focus on high-level grasp and semantics and depth of precise, UML is based on a viable practice.

3.2 MAPPING RULES FROM DISADL TO UML

Conversion rules from Dis_ADL architecture description elements to UML elements are shown in Table 1.

TABLE 1 Mapping table of Dis_ADL element to UML elements

Element--Dis_ADL	Element--UML1.x	Element--UML2.0
Component Name	Class Name,Package Name	Class Name,Package Name
Atomic Components	Class Diagrams,Packages	Component,Packages,Class Diagrams
Composite member	Aggregation,composition, package, subsystem layer	Composite structures、Subsystem
State variables	Class private property	Class private property
Total Interface	Interface	Component supply Interface
Demand Interface	Abstract class	Component Requirements Interface
Port	Class method names, parameters, return type	Port
Message return type	Class method return type	Class method return type
Message parameters	Class method parameter (variable + type)	Class method parameter (variable + type)
Connector	Class diagrams, association	Connectors, class diagram, association
Architecture configuration	Component diagram or class diagram	Component diagram, structured category
Binding	Object Linking (associated instances)	Object Linking (associated instances)
Comment	Document	

4 Analysis of distributed software architecture reconstruction

4.1 DEFINITION OF SOFTWARE ARCHITECTURE RECONSTRUCTION [3]

Software architecture reconstruction is the process that reverses extract the architecture from being achieved or already implemented systems, reflecting the "actual construction" software architecture. The core problem is extraction of the architecture and the assessment the architecture evolution.

4.2 SOFTWARE ARCHITECTURE RECONSTRUCTION MODEL

Software architecture reconstruction process can be divided into four parts: (l) confirm stakeholders, defining target point of view, and the view collections. (2) extract

the underlying architecture element information from a variety of data sources (such as source code, documentation, etc.). (3) Define and process the extracted information, coordinate and establish connections between elements, generate a cohesive view of the architecture. (4) Construct data abstraction to generate architecture representation.

5 Distributed system design method -- ADISc and distributed development tectonic environment – AdisDTool

5.1 DISTRIBUTED SYSTEM DESIGN METHOD OF ARCHITECTURE DRIVEN – ADISC [4]

The development process of distributed systems can be divided with ADiSC into: requirements analysis; software architecture design, modelling and refinement conversion; system detailed design; system implementation, component assembly deployment and reconfiguration; system evolution and reuse.

5.2 DISTRIBUTED DEVELOPMENT TECTONIC SETTING --ADISDTOOL

Modelling system of ADisDTool consists of interactive graphics interpreter, DISADL language converter, DISADL lexer / parser, verify the nature of the software architecture, system builder, Discid language compiler environment, refinement converter, UML mapping generator, architecture reconstruction and other components, the system model is shown in Figure 2.

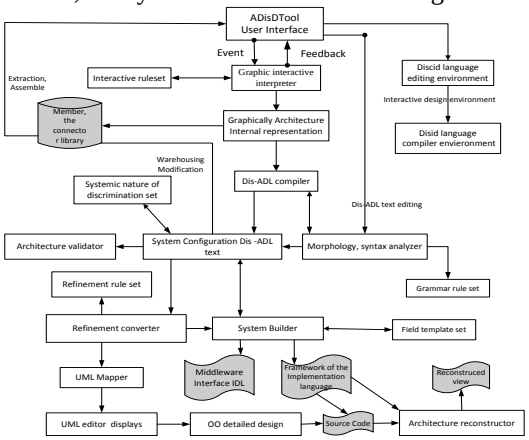

FIGURE 2 ADisDTool system model

5.3.1 Functional design of ADisDTool architecture modelling tool

The ADisDTool can be divided into eight modules: Project Management module, component and connector management module, visual modelling module architecture view modules, code generation module, system properties verification module, the system refinement mapping module, reconfigurable architecture.

5.3.2 ADisDTool reusable component library design [5]

ADisDTool established member (connector) library, which provided for the entire life-cycle management member. Provides the following functions: components, connectors, interfaces, warehousing; components, connectors, interfaces, query; components, connectors, export, evaluation, life cycle management, version control and so on.

6 Example authentication

Through such typical case of large-scale distributed systems development "digital content security platform based on DRM (National Innovation Fund project, project number 07c26226101995)", showing the core areas of applying ADISC methods to analyse, design and develop, providing convenient and effective design environment through ADisDTool, which does help convenient, fast and efficient analysis of large-scale distributed systems, nature verification, design, evolution and reuse, and prove effectiveness of DSAM models and ADISC method the thesis mentioned .

6.1 AS FOR "ORM-BASED DIGITAL CONTENT SECURITY PLATFORM."

The planform of Digital Content Protection Based on DRM is a system, which is analysed, designed, developed and achieved by distributed system, and the current one has been put to use for network environment of digital content security protection. PlatDRM ensured security of digital content in creating, distributing, using, sharing that throughout the life cycle. This paper briefly describes PlatDRM to validate the model DSAM and ADiSC method.

6.2 PLATDRM CORE FUNCTIONAL REQUIREMENTS

PlatDRM core functional requirements are: (1) to ensure digital content distribute secure, digital content is held being only in the specified environment and key user access, any duplicate is not available. (2) real-time authorization control, and can be changed the digital content access which have been distributed out at any time. (3) real-time monitoring and recording digital content action to take for the audit analysis, behaviour tracking provides detailed historical data. (4) support the parties authentication with domain authentication and other authentication systems integration, users only need to open an important document domain authentication.

6.3 TOP-LEVEL ARCHITECTURE DESIGN

According to ADiSC methods, requirement analysis selected core use cases; the resulting model is transformed to the software architecture to build a

language-independent architecture model. Meanwhile, it can be application-oriented modelling, describe domain business needs to facilitate greater reuse. PlatDRM is a typical distributed application, Figure 3 is its top-level conceptual architecture diagram, architecture consists of a client component, server component, WEB management component, of which the client is called CC member, the server is called SC members, managing client is called MC member.

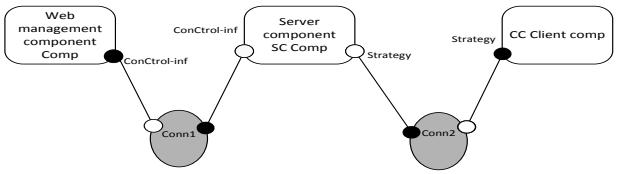

FIGURE 3 PlatDRM top Software Architecture Design

6.4 ARCHITECTURE REFINEMENT

When architecture is in a design stage, we use ADisDTool to analyse and refine PlatDRM architecture. First, for the interface refinement, the interface refinement between customer component and service component are set to be strategy interfaces, authentication interface, the operation log pick date, and digital content object interface. Then the client component CC internal structure refinement. After refining, Cc component architecture is shown in Figure 4.

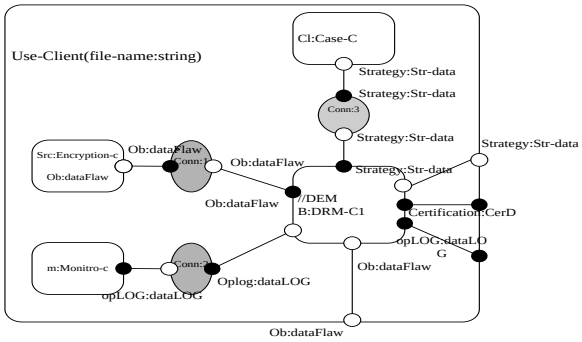

FIGURE 4 PlatFORM client component architecture after refinement

6.5 INTERACTION EVENTS CLIENT DEFINITION LOG_EVENT_CLIENT

After customers receiving log events, should be temporarily stored in the queue used to prepare the client component. LOG_EVENT_IO process defines the client endpoint behaviour, describes the state changing method after in response of API, also shows that when customers in the state of receiving start LOG_RECEIVED log events , the receiving message can only be placed in the event queue. To enable the server can send message directly at any time, avoiding the synchronization aborting between the server and the client, the definition of time endpoint contains two explicit intermediate state: STOP_LOG_EVENT_OUT and STOP_LOG_EVENT_ACK, instead of the direct state transition from LOG_RECEIVED to LOG_NEGLECT.

6.6 MAPPING SYSTEM

In the system design phase, we designed classes, interfaces, components which PlatDRM containing by using UML. In addition, object state diagrams, sequence diagrams, etc. to get the attributes and methods of classes. Figure 5 is PlatDRM overall package design. Figure 6 is a design package diagram after client finer.

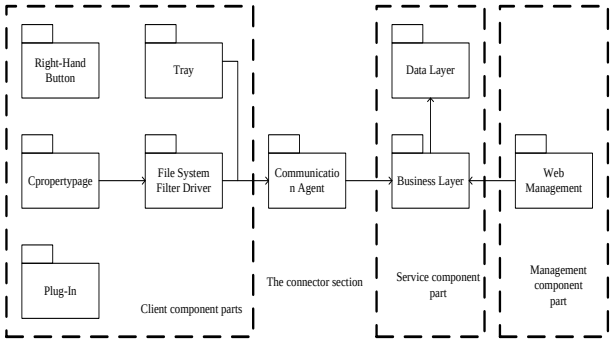

FIGURE 5 PlatDR, I overall design package diagram

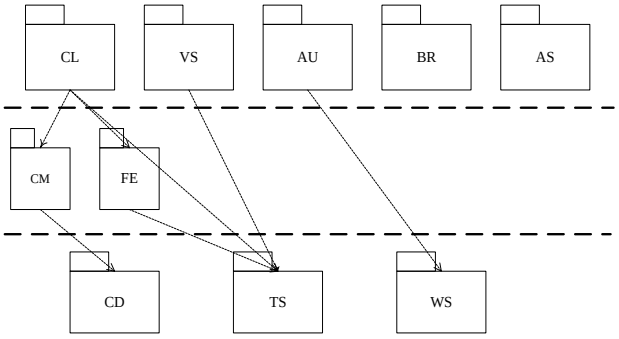

FIGURE 6 PlatDRM client design

6.7 PLATDRM FUZZY CLUSTERING ANALYSIS

Fuzzy clustering analysis, the results are shown in Figure 7.

		Client														
		AS	AU	BR	CC	CD	CM	DP	DM	FE	RM	SU	TS	TU	WS	VS
1	AS	Y														
2	AU		Y						Y							
3	BR			Y												
4	CC				Y			Y	Y		Y	Y		Y		
5	CD				Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y
6	CM				Y		Y	Y		Y						
7	DP				Y			Y								
8	DM				Y				Y							
9	FE				Y					Y						
10	RM								Y		Y					
11	SU							Y	Y		Y	Y				
12	TS				Y		Y	Y	Y	Y	Y	Y	Y	Y		Y
13	TU											Y		Y		
14	WS		Y												Y	
15	VS															Y

FIGURE 7 PlatDRM client component dependency matrix after refactoring

6.8 ANALYSIS AND RESULTS BY USING THE DSM CORRELATION ALGORITHM

Using the DSM-correlation algorithm, we can get matrix, shown in Figure 7, so we focused on {DP, DM, RM, SU, TU, CC} can be polymerized to obtain Figure 8. Relationship between the main clients' components can be clearly seen.

	BR	AU	WS	FE	DP	DM	RM	SU	TU	CC	CM	TS	VS	AS	CD
BR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
WS	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
FE	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
DP	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
DM	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
RM	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
R = SU	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
TU	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
CC	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
CM	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
TS	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
VS	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
AS	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
CD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

FIGURE 7 PlatDRM client component reachable matrix view after refactoring

Client		BR	AU	WS	FE	CL	CM	TS	VS	AS	CD
1	BR	Y									
2	AU		Y			Y					
3	WS		Y	Y							
4	FE				Y	Y					
5	CL					Y					
11	CM				Y	Y	Y				
12	TS				Y	Y	Y	Y	Y		
13	VS								Y		
14	AS									Y	
15	CD				Y	Y	Y	Y	Y		Y

FIGURE 8 Client DSM after the polymerization

Analyse the results of DSM after client code division. After refreshing, we discovered that according to results of the matrix analysis, Figure 9 shows a PlatDRMclient's system architecture and Figure 8 is the corresponding matrix. Compared the figure represented with the original architectural design (Figure 6), it is easy to see the dependencies between components changed, and dependency occurred between CL and AU, contrary the initial design expectations, therefore, it needs to be adjusted and avoided.

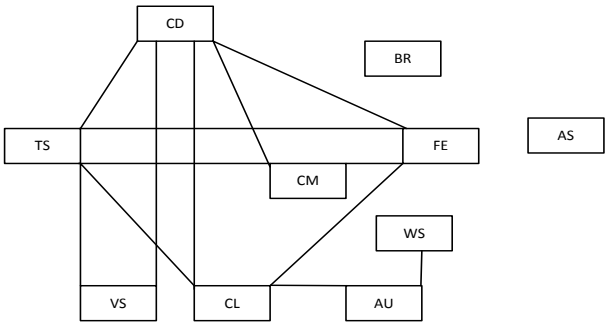

FIGURE 9 P1atDRM client view after polymerization system packing

"Digital content security platform based on DRM" Such a typical distributed system has been put into the core link in ADISC, Dis_ADL describes the software architecture and is given top-level component description of the system, then refined it, and give a formal and intuitive description for the language Discid we mentioned focus on the typical event interaction style and conduct of the verification. Finally, the code that system implementation has been found in the software architecture, fuzzy clustering analysis and analyse it by using the proposed design structure matrix theories. Currently, the system by using ADISC methods for designing and developing was put into use in Shan xi military giant, which has been got a good evaluation of the user.

7 Conclusion

In the analysis of the lacking and problems in current large-scale distributed systems development methods and means of supporting, this paper presents a method based on component structure distributed systems under the support of the software driver. Its core is a distributed software architecture abstraction model--DSAM, a description language supported such architecture modelling, validation, refinement, evolution and reverse extraction over the DSAM--DIS_ADL and a component interaction analysis verification language based on CCS --Discid. On this basis, this paper presents a visual support for distributed software architecture modelling, refinement and reconstruction of distributed system design environment--ADisDTool.

References

[1] Mei Hong, Jun-rong Shen 2006 Software Architecture Research *Journal of Software* **17**(06) 1257 - 75
[2] Shi-xian Zhang, Li-fu Wang, Fu-qing Yang 2000 Systems development based on COTS component *Computer Science* **27**(1) 6 - 8
[3] Lv-ai Sun, Mao-zhong Jin, etc 2002 Software Architecture Research *Journal of Software* **13**(07) 1225 - 37
[4] Ding-yi Fang 2001 Distributed software architecture research based on component interaction mechanism *Northwestern Polytechnical human science, Xi'an*
[5] Jian-jiang Zhu 2001 Software reuse research based software component *Nanjing University of Aeronautics and Astronautics*

Authors

Xiaochun Zhao, born on February 6, 1976, Jilin Province of China

Current position, grades: lecturer
University studies: Bachelor degree was earned in major of computer science and technology, Heilongjiang University of Science and Technology in 2007.
Scientific interest: computer application, web application, Web database technology, information processing