Intelligent Collaboration Technology of Software Developers Based on Data Driven

Ming Wan[*]
Aviation Engineering Institute, Xuchang Vocational Technical College, XuChang, Henan, China
**Corresponding Author.*

Abstract

Software development highly depends on the group contribution of developers, so improving the cooperation efficiency of developers is an important issue to improve the efficiency and quality of software development. In recent years, it has become a hot topic in the field of software engineering to improve the intelligent level of software development by mining and using the knowledge contained in software big data. However, the research on software developers and their group collaboration methods has not yet formed systematic research results. Therefore, this paper takes the developer group as the research object, through in-depth analysis of the behavior history data of developers, studies the key technologies of intelligent cooperation, and develops the corresponding support environment based on this. This paper first analyzes and summarizes the related research work. Then, the capability feature model and collaboration relationship model of software developers are given, and the knowledge map of developers is constructed. Furthermore, based on the knowledge map of developers, the collaborative development method based on intelligent recommendation is introduced by taking two developer recommendation methods as examples. Based on the above key technologies, the corresponding supporting tools are developed, and the prototype system of intelligent collaborative development environment is constructed.

Keywords*: Software development, group contribution, collaboration efficiency, data driven*.

I. Introduction

The simple object-oriented, component-oriented and service-oriented software development technology can not meet the needs of software development under the Internet environment, and there are some limitations in the basic software model, software methods and technology, basic support mechanism and so on [1-2]. Software development in the Internet environment usually can not assume that all parts of the whole system comply with the unified design and management, and can not completely and accurately determine the structure of the system and the behavior of each component. At the same time, due to the dynamic environment and decentralized management, different collaborative behaviors may need to be implemented at different times. Therefore, on the one hand, the software model under the Internet environment should have an effective collaboration mechanism to support, manage and control the interaction between entities on the Internet, on the other hand, it must provide enough flexibility to adapt to the different needs of the environment and applications. In order to adapt to such an open environment, large-scale resource sharing and integration and a variety of new computing modes, such as grid computing, mobile computing, pervasive computing and so on, marked by WWW, computing grid, network embedded system, have emerged [3-5]. However, none of these technologies can solve the problems of autonomy and collaboration of Internet based software systems in dynamic open environment.

The software system based on Internet can be regarded as a software alliance which is formed dynamically by a series of distributed autonomic computing resources to complete specific tasks. Accordingly, the software system began to present a flexible, multi-objective, continuous reactive new system form. From a technical point of view, software entities supported by software components and other technologies exist on each node of the Internet in an open and autonomous way [6]. Any software entity can interact, interact, collaborate and alliance with other software entities in various collaborative ways in an open environment, and can perceive the dynamic changes of the external network environment and change with the development of the Internet According to the function index, performance index and credibility index, these changes are adjusted statically and dynamically to make the system have as high user satisfaction as possible. References [7-10] call such a new software form network software. Specifically, network architecture software includes a group of software entities distributed in each node of the Internet environment with the characteristics of subjectivity, and a group of connectors used to support these software entities to cooperate with each other in various ways. These entities can sense the changes of the external environment, through the method of architecture evolution (mainly including the increase and decrease of software entities and connectors) Network architecture software, which is quite different from traditional software, is a behavior mode of cooperation between entities on demand in the micro level, and an application mode of spontaneous formation of entities in the macro level Correspondingly, the development of network architecture software presents the basic system of "order" by combining the original "out of order" basic software resources. With the passage of time, the changes of these systems and resources in function, quality and quantity lead to the state of "out of order" again. This process from "out of order" to "in order" goes back and forth Ring is basically a spiral way from bottom to top and from inside to outside.

II. Problem statement

Because there are many differences between the conceptual framework and logical connotation of Netsoftware and the classical software system, it inevitably challenges the traditional classical software method and technology system in theory, model and technology. These challenges are reflected in the software collaboration mechanism:

(1) Separation of Software Collaboration: how to separate the collaboration mechanism from software entities and provide flexible and diverse collaboration methods to support the collaboration of software structure model.

(2) Adaptability of operation mechanism: how to make the software system adaptive to the changes of external environment in an open, dynamic and difficult to control environment.

(3) Formalization of core theory: how to find a more appropriate and consistent formal system for the above technologies to meet the needs of open environment, which is the theoretical basis of the new generation of software collaborative methodology. Therefore, the research of software collaboration framework and its theory should focus on the above key issues.

On the basis of the above analysis, we propose to use Petri net technology and mobile agent which have been widely used in the field of computer and automation to study the cooperation mechanism of network software. The idea can be simply expressed as follows: modeling the composition of software entity services with the help of Petri net theory to complete the tailoring and integration of software entity functions in the open environment. Using data-driven as the implementer of collaborative mechanism, an XML format assembly model with data-driven path information and function body separation is designed to support the dynamic assembly between software entities. The ripple effect of entity evolution on the system is measured by influencing factors. According to the dynamic operation rules, the dynamic adjustment strategy is determined when the system evolves.

Network cooperation software is an abstraction of the basic form of software system in the open, dynamic and changeable environment of Internet. It is not only a natural extension of the traditional software structure, but also has unique basic characteristics different from the traditional software form developed in the centralized and closed environment. Network collaboration software autonomy refers to the relative independence, initiative and adaptability of software entities. From a technical point of view, network collaboration software entities are generally developed and managed independently, and they may run independently on different network nodes. Their goals and services are determined by their owners, and their behaviors are driven by their own goals, rather than just passively used for assembly or deployment. In the process of running, they may collect all kinds of change information of the environment in real time, and automatically adjust their own behavior to adapt to the change of the environment according to the preset strategy.

(1) Collaboration refers to the interconnection, intercommunication, collaboration and alliance between software entities in a network collaboration software system in an open network environment. From a technical point of

view, traditional software systems tend to adopt a single static connection mode in a closed and centralized environment, while network collaboration software supports the adaptation of the connection mode Adaptability adjustment, such as switching of different interoperability protocols, rising and falling of connection security level, transition of synchronization and asynchronism, adjustment of message passing reliability, etc.

(2) Reactivity refers to the ability of network collaboration software to perceive the external operation and use environment and provide useful information for system evolution. From a technical point of view, the external environment of network collaboration software is composed of other network collaboration software and the underlying support platform. Therefore, reactivity requires network collaboration software to be able to expose its own state and behavior information in a certain way, as well as provide information for system evolution Network collaboration software support platform is required to open the bottom implementation details and running state.

(3) Evolvability refers to the dynamic evolution of network collaboration software structure according to the application requirements and network environment changes. It is mainly manifested in the variability of the number of entity elements, the adjustability of structural relationship and the dynamic configurability of structural form. From the technical point of view, evolvability requires the dynamic adjustment ability of software architecture.

(4) Polymorphism means that the effect of network collaboration software system reflects the compatible multi-objective. It can meet a variety of compatible target forms under the dynamic network environment according to some basic collaboration principles. From the technical point of view, polymorphism requires not only the support of multi-objective modeling in the process of system development, but also the dynamic modeling based on the environment changes during the system operation Based on the above analysis, the main technical commonness of many characteristics of network collaboration software can be attributed to the adaptability, which is embodied in the adaptability of software entity and software structure, that is, the network collaboration software can accurately capture changes and make reasonable adaptive adjustment at the right time and in the right situation, To meet the requirements of function and quality.

III. Software collaboration framework for network architecture software

From the perspective of engineering design, a domain world can be regarded as a distributed system in which multiple autonomous nodes cooperate to complete specific tasks. The system has two basic elements: autonomous entity and activity. Autonomous entity refers to the experts, organizations or tools that participate in the design activities in the system, and the activity is the process of product design by autonomous entity. The activities of a collaborative design system can be described by three levels which is shown in Figure 1.

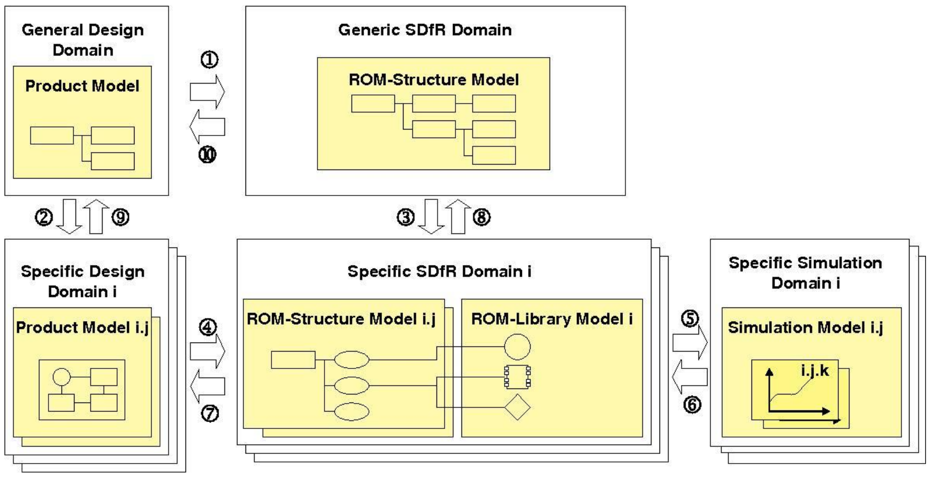

Fig 1: Software collaborative design system

The first level is the analysis and decision-making level, whose main activities are browsing, retrieval, data analysis, master planning and decision-making. The second layer is the information sharing and exchange layer, whose main activities are to provide support for information exchange, resource sharing, collaborative work and analysis and decision-making of the lower layer. The third layer is the collaborative design layer, which carries out specific design activities.

Therefore, a collaborative design system should include the core and control entities for analysis and decision-making, the functional entities to support collaborative work, information exchange and resource sharing, and the design transaction entities to participate in specific design activities. The collaborative process model for network architecture software is shown in Figure 2. The collaborative process model can be divided into five parts: analysis part, design part, assembly part, deployment part and maintenance and evolution part.

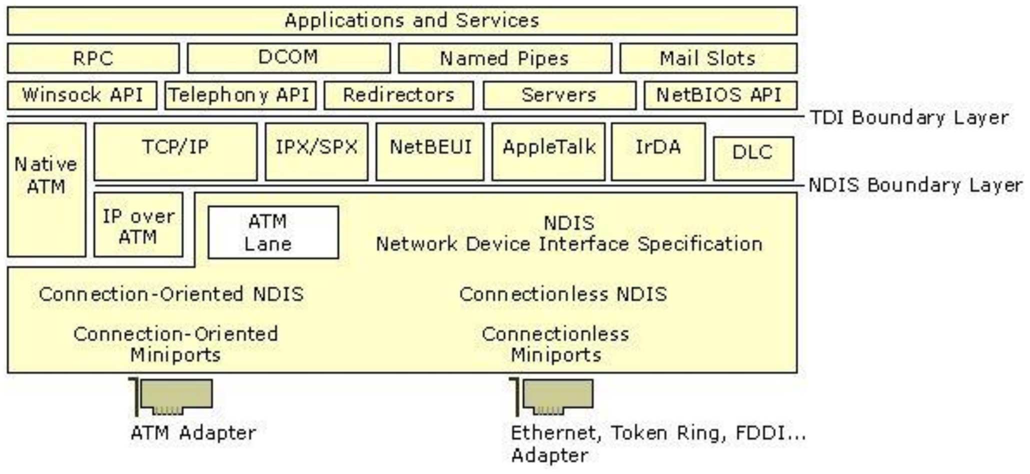

Fig 2: Collaborative model of network architecture software

(1) The main purpose of the analysis part is to smooth the gap between requirement and design. In the requirement analysis stage, the problem space and user requirements are organized in a structured way. At this stage, we have no requirements for the specific form of the product. We only need to be able to convert it into software architecture relatively conveniently, naturally and directly.

(2) In the design part, the requirements specification of the software system is studied, the corresponding global design decisions are made, the components and connectors in the conceptual software architecture are further refined, the necessary software entities and mobile agent connectors are created, the static and dynamic software architecture model (including type diagram, instance diagram and process diagram) is established, and the requirements specification and software architecture are maintained The mapping between architectures.

(3) The assembly part is equivalent to the implementation phase of traditional software development. Different from the traditional software development method of programming according to the design to achieve the target system, the basic functional unit of network software is the existing and running network software entity. Therefore, the focus of network software implementation is not programming but assembly, that is, selecting the entity that conforms to the software architecture, and making all entities interact according to the provisions of the software architecture. If the entity or interaction does not meet the requirements of software architecture, it needs to be adapted. If the adaptation fails. New entities need to be developed to achieve the target system.

(4) The deployment part is a component set for the normal operation of the network architecture software. The information needed for deployment is complex, which often needs to be filled in manually. In fact, most of the deployment information already exists in the system design and implementation stage, and can be reused after transformation or fusion. On the other hand, new entities or new collaborations may need to change the organizational relationship between existing entities, and the implementation of this organizational relationship is also one of the main tasks of deployment.

(5) Maintenance and evolution stage: in a sense, the development method of network architecture software can be regarded as the continuous and iterative refinement, mapping and transformation of the software architecture of the target system in different views. After each refinement and transformation, the syntax and semantic information of the software architecture becomes more accurate and complete.

Architecture is the basic and main form of the whole system. It should show the characteristics of the software. Architecture includes a series of abstract patterns to guide the design of large software system. For large-scale complex software systems, the overall architecture design and specification is much more important than the algorithm design and data structure selection. A good architecture is conducive to improving software productivity and solving software maintenance problems. The framework system described in this paper refers to the idea of software reuse based on component composition, and adopts the loose coupling structure of separating the software entity part from the collaboration part. The software entity part and the collaboration part are independently developed by the service provider and the service integrator, as shown in Figure 3. This structure abandons the traditional software structure characterized by tight coupling structure, centralized development and reprogramming. In the open network environment, facing the individual needs of integrators, taking into account the individual characteristics of software entities, various software services are integrated through the collaborative part, and in the process of adapting to the external dynamic environment change or user needs change, the static adjustment or dynamic evolution is constantly carried out, so as to improve the user satisfaction as much as possible.

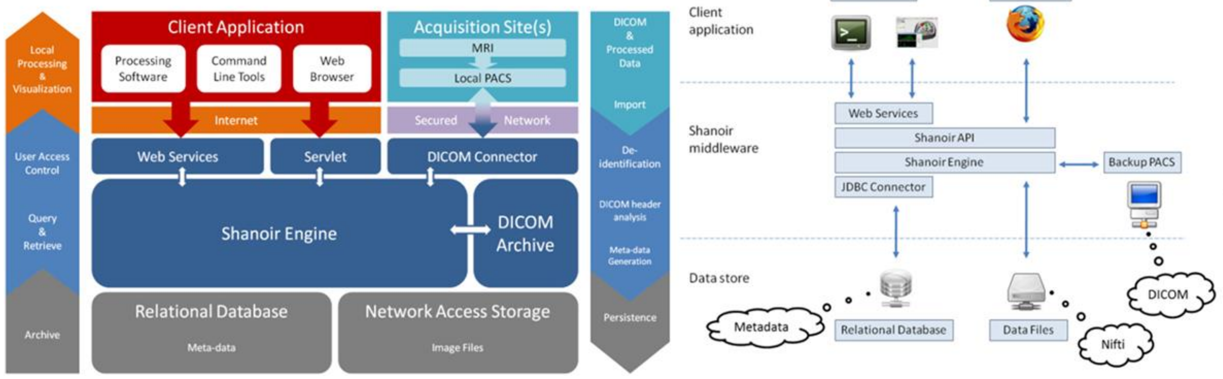

Fig 3: Data driven collaborative process model

(1) Application layer: the application layer mainly includes two parts: user requirements description and application software function description. It adopts a way similar to function decomposition to organize problem space and user requirements, semi automatically generates conceptual software architecture, smoothes the gap between requirements and design, and provides a basis for the organization and co evolution of software architecture in control layer. Data driven software collaboration framework is shown in Figure 4.

(2) Control layer: control layer is the core of software collaboration system, which is also the main content of this paper. It is mainly composed of collaborative engine module and monitoring part, which is the intelligent part of the whole architecture. The main work of the collaborative engine is to assemble the isolated software entities in the entity layer into a logical application software system through the corresponding connectors according to a certain logical relationship. The monitoring part can monitor the changes of user requirements and environment in real time, and feed the structure back to the collaborative engine. The design of this layer draws on the knowledge of web service composition, software architecture dynamic evolution and mobile agent, so that the system can customize the corresponding application software according to the customer's needs, and dynamically evolve its architecture with the change of requirements and environment, so as to improve the user's satisfaction.

(3) Entity layer: entity layer is the basis of control layer, including connection module based on mobile agent and software entity with distributed, autonomous and heterogeneous characteristics in open environment.

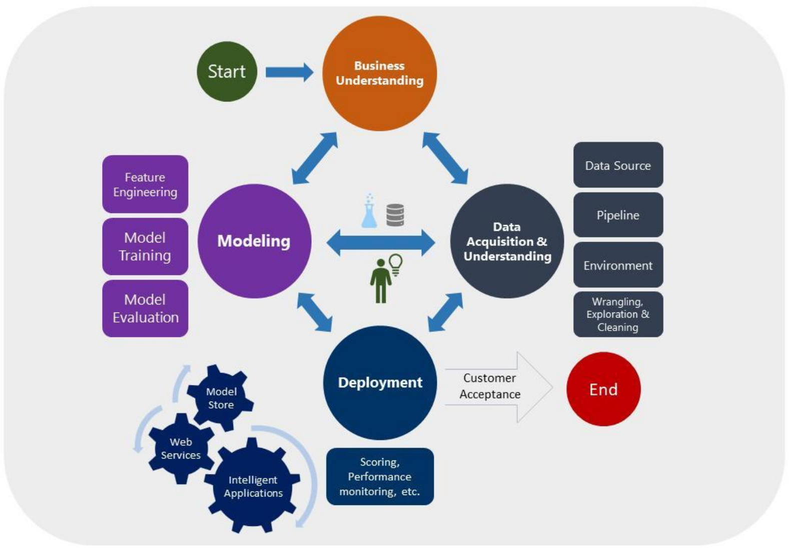

Fig 4: Data driven software collaboration framework

IV. Conclusion

This paper first introduces the characteristics of general collaborative system in general environment, and on this basis, according to the requirements of Internet environment and the design concept of network architecture software, it explains the design requirements of software collaborative framework for network architecture software and the specification definition of framework architecture. Referring to the common collaborative system model, this paper analyzes the software collaborative system for network architecture software from multiple perspectives, and designs the data-driven collaborative model and the overall architecture of the collaborative framework.

References

[1] Lu Zhengding, Huo Xiaoli. Research on Intelligent Software Collaborative Development Based on Agent Technology. Computer Science, 2007, 34 (003): 208-210

[2] Wang Tao, Yin Gang, Yu Yue. Method and Practice of Software Development Collectivization Based on Swarm Intelligence. Chinese Science: Information Science, 2020 (3): 318-334

[3] Lan Wenfei, Lu Jiguang. Application of Intelligent Agent in Component Library System. Computer Engineering and Design, 2007,28 (017): 4089-4090

[4] Chen Tongyang, Han Yuhe, Gu Xinlei. Intelligent Garbage Cleaning System Based on Bdi. Software, 2012, 33 (9): 61-62

[5] Ni Hongmei. Design of Cooperative Intelligent Question Answering System Based on Multi Agent. Journal of Yangtze University (self Science Edition), 2009, 000 (01x): 208-209

[6] Chen Liuyang, Han Benshuai, Liu Ningning. Development of Intelligent Statistical Software for Communication Optical Cable in Substation Automation System. Electrical Technology, 2014, 4: 27-28

[7] Zhao Xinpei, Li Mingshu, Chen Zhenchong. a Negotiation Based Software Process Collaboration Method. Computer Research and Development, 2006 (02): 314-320

[8] Zhou Mingjun, Xu Lishuang, Tian Feng. Research on Collaborative Pen Based User Interface Development Tool. Acta Sinica Sinica, 2008 (10): 304-312

[9] Xie Xinqiang, Yang Xiaochun, Wang Bin. a Multi Feature Fusion Software Developer Recommendation. Acta Sinica Sinica, 2018, 29 (008): 2306-2321

[10] Bergalais Industrial Automation Shanghai Co., Ltd. Using Automation Studio 4 to Realize Intelligent Engineering Design. Automation Expo, 2015 (07): 97-99