RC24197 (C0702-011) February 27, 2007
Computer Science

IBM Research Report

A Case Study on Community-Enabled SOA Application Development

Liu Ying, Feng Chenhua, Zhao Wei, Su Hui, Liu Hehui
IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, P.R.C. 100094

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home .

A Case Study on Community-enabled SOA Application Development

Liu Ying, Feng Chenhua, Zhao Wei, Su Hui, Liu Hehui
IBM China Research Lab, Beijing, China
{aliceliu, fengch, weizhao,suhui,hehuiliu}@cn.ibm.com

Abstract

The idea to leverage large numbers of the open community resources is straightforward to cater the expectation on reducing the development cost. Knowledge protection and quality assurance in this process are critical challenges for the overall success of such kind of software outsourcing. It is pivotal to provide methods and technologies to ensure all the goals of low cost, no knowledge loss as well as high quality while outsourcing development works to open communities. Call-For-Implementation development method put forward in this paper intends to distribute implementation tasks to the developers of open communities through partitioning a holistic design into pieces of work segments based on some knowledge protection policies. Although CFI method can be widely used for any types of applications, SOA applications are regarded exactly suitable for this method since the components of SOA applications are designed to be loosely coupled. In this paper, we present our study on conducting the CFI method on a real SOA application. Some metrics are defined for validating the hypotheses of the CFI method, including lower cost, knowledge protection, and quality assurance. Measurement result analysis of this case is presented and findings acquired are also reported.

1. Introduction

With the rapid development of IT industry, more and more large-scale applications come forth. Besides experienced analysts and architects, large numbers of developers are necessary for delivering the implementation of these large applications in time. It is of high return on invest for an enterprise to hire and cultivate a small group of experienced requirement analysts, business architects, and IT architects to deliver application designs with high quality. However, it is not always cost-effective for an enterprise to employ and keep a large number of developers no matter from management or cost perspectives if software implementation is not the core business of the enterprise. As a result, an innovative approach to helping solve the development resource problem for enterprises is of critical importance for the successful delivery of a business application.

Open community, which is composed of students, programming fans, SOHO (small office/home office), etc., is a well-known virtual development resource pool. Nowadays, a lot of open-source software [1] projects have successfully demonstrated that the open community is an unneglected channel to deliver good software. In fact, some commercial companies have realized the value of the huge resource pool of the open community, and involved in some open-source developments or incubated some open-source projects in order to share the copyrights of the open-source software. However, the commercial companies can only gain limited benefits from these ad hoc community-enabled activities. A systematic way to leverage the open community resources is becoming a great target pursued by the commercial companies.

Outsourcing [2] of software development has gained much attention of researchers as well as practitioners. It leads to lower cost of software development. We adapt the outsourcing approach to the open-community environment to put forward *CFI (Call-For-Implementation) development method*. This approach proposes to partition an application as some pieces of development work segments which can be distributed to the individual developers in the community. These pieces of work segments are specified as some semi-formal documents which are ready for implementation. These documents are published in the communities, so that the developers of the open community can apply for the implementation.

The benefit of the CFI method is three-fold. Firstly, the development cost of an application can be dramatically reduced because of leveraging the cheaper resources of the open community. Secondly, the key knowledge of the application which may dominate the businesses of to the enterprises can be protected because of the adopted knowledge protection polices. For example, the holistic design of an application is partitioned as pieces of work segments to be allocated to the different developers of the open community, which is an effective way to protect the whole application design. Thirdly, the quality of applications developed by the CFI method can be

guaranteed because of the specialized quality assurance process and some advanced testing technologies. In this paper, we validate these benefits through a real SOA application case, Digital Currency Manager (DCM).

The rest of this paper is organized as follows. Section 2 starts with the overview of the CFI method. In section 3, the research method and process is briefly introduced. Section 4 gives the background of the DCM application. The main contents of this paper, including the specific process of the CFI method performed on the DCM application and metrics acquired, are presented in section 5. Some findings through this practice are reported in section 6. Section 7 lists some related works. The final section concludes this paper and presents the future works.

2. CFI method overview

The CFI method essentially adapts outsourcing model to the open community, where knowledge protection and quality assurance technologies enable the CFI method to be a commercialized development method leveraging the open community. Figure 1 depicts the overview of the CFI method.

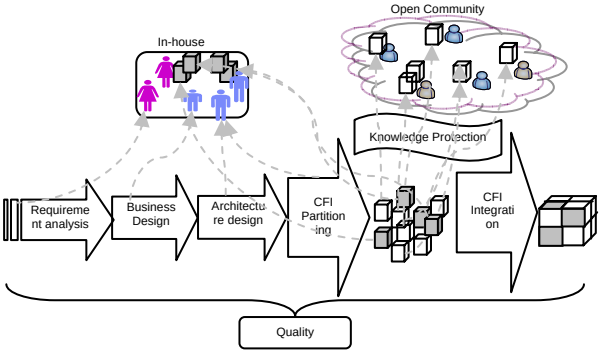

Figure 1: Overview of CFI Development Method

In the CFI method, *CFI partitioning* is one of important activities. CFI partitioning is an activity that the design of an application is partitioned as some pieces of work segments which are able to be independently implemented by different developers. Each of the work segments is called a CFI which will be distributed to an individual developer of the open community. The granularity of CFI can be very flexible according to available resources, application characteristics, and the business knowledge protection.

The participants of a CFI project are usually grouped as two teams: *in-house team* and *open community*. The in-house team includes project manager, requirement analyst, application designer, and a small number of skilled developers. The in-house team can be regarded as the owner of the project. This team is responsible for the works which require high levels of knowledge and skills, such as requirement analysis, design, CFI partition, and final integration. The open community is a virtual team which is constructed by the developers who join the project implementation based on the applied CFIs.

In the CFI method, there are two important supporting technologies, knowledge protection and quality assurance. For a commercial application, how to avoid the loss of 'trade secret' (which is called *key knowledge* in this paper) during the CFI process is pivotal to the success of the CFI method. We categorize the key knowledge of an application as five types: main function features, business architecture design, IT architecture design, key data model, and key business processes. Correspondingly, we proposed several approaches to protecting the key knowledge, such as, partitioning the overall application design into some pieces of work segments, replacing some business keywords with other domain's words or meaningless words, controlling the exposure rate of data model, and partitioning a whole business process as some small business processes.

In order to reduce the quality risk due to the implementation work partitioning and the distribution to the open community, we adapt the *quality assurance process* of RUP [3] to the CFI method by enhancing it with some key technologies and methods, such as unified test technology and rigorous code review approach. Because of space limitation, the details of the unified test technology will not be covered in this paper.

3. Research method and process

In this section, we present the research method and process employed in the study. The specific goals which we expect to acquire through performing the research method are introduced at first. According to the determined specific research goals, the descriptive study method is decided to be adopted as our research method, which aims primarily at gathering knowledge (i.e. descriptions and explanations) about the object of study but does not wish to modify the object. The target is to find out how things are, or how they have been. We will give the detailed description to the organization and process of our executed research method. In addition, the metrics employed to come up to the conclusions are also presented in this section.

3.1. Research method

The research approach which we adopted for evaluating the CFI development method is a kind of empirical study. This section presents the specific goals, i.e., hypotheses expected to be validated, of this empirical study and the organization and process of the study based on the determined goals.

Hypotheses The CFI development method is aiming at developing application with lower cost and high quality by leveraging the open community resources and assuring the necessary control on the key knowledge of the application at the meanwhile. These are hypotheses of CFI method that we are going to validate through the empirical study. We will compare the data collected from the empirical study with the statistical data in the literature. Since the hypotheses which we determined to verify through the study are to acquire the recognition of the CFI development method quantitatively to some extent just from one case, the descriptive method can be adopted for our purpose which focuses on acquiring the knowledge of the observed objects but no any modification on them.

Organization and Process Based on the method we adopt, we designed this study in detail from both organization and process perspectives. In this empirical study, there are three independent teams. One is the *in-house team*, the other is the *observation team*, and the third one is the *open community team*. As introduced in CFI method overview section, the in-house team mainly performs the CFI development approach, and the open community team is composed of 13 graduate students from universities. The observation team is responsible for defining the metrics for indicating the hypotheses, collecting raw data and analyzing them. The specific metrics can be found in the following sub-section.

The process of the empirical study consists of three main steps. In the first step, the observation team constructs the data collecting environment for each member of the execution team and gives the training to them to assure the integrity of the collected data. We use IBM Rational ClearQuest [4] for this task. The second step includes two paralleled threads of both the CFI development and the tracking on it. That is to say, while the in-house team performs the CFI method, the requested data is also tracked everyday by them. At the meanwhile, the observation team members check whether the data is updated in time and assure their validity and integrity. In the third step, after the CFI development process finishes, the observation team analyzes the collected data and presents the absolute and comparative results.

3.2. Metrics definition

Goal oriented measurement approach [5] is widely used in software industry. In our study, we use the G-Q-M (Goal-Question-Metric) method to define metrics for verifying the hypotheses of the CFI method. Considering some essential features of the CFI method, the following four goals are defined:

1. To give insight of effort allocation of the CFI development method
2. To evaluate potential cost-savings and productivity-improvement due to leveraging the open community resources
3. To check if there is any degradation in software quality due to work partitioning
4. To evaluate the effectiveness of knowledge protection through work partitioning

According to these goals, we define four kinds of metrics, including product, process, quality and knowledge protection metrics. The details of these metrics are given in table 1, 2, 3, and 4 by category.

Work Product	Metric Definition
Code	Size of code (KSLOC)
Requirement	Number of use case, Number of use case transaction Page of requirement specification
Design Model	Number of class/interface
Test Model	Number of test case

Table 1: Product Metrics

Process	Metrics Definition
Resource profile	Project staffs profile per phase / per iteration
Efforts distribution by discipline	Efforts on every discipline, including requirement, analysis & design, implementation, test, environment, configuration, project management, training, deployment, etc.
Efforts on CFI related activities	CFI partition CFI partition - review & adjust CFI specification - effort on specification CFI specification - review & adjust CFI specification - effort on mock testing CFI implementation (code, unit test) CFI - code review CFI communication/management CFI training CFI partial integration by community developers CFI system integration CFI change management CFI rework
Cost	Development cost of developers in the in-house team Development cost of developers in the open community

Table 2: Process Metrics

Quality	Metrics Definition
Defects density	Number of defects/product size
Defects distribution	Number of defects by which phase introduced/phase found Number of defects found by which test activity and defect trigger
Defects age	The age/duration from defects submitted to defects removal.

Table 3: Quality Metrics

Knowledge protection	Metric Definition
Application key functions	Keywords: total number vs. CFIed (i.e. Distributed in CFI) Key features: total number vs. CFIed
Key business processes	Key business processes: total number vs. CFIed
Data model	Number of data distributed in CFI

	Percent of Business Object(BO) fields distributed to the developers of open community
System architecture	Architecture significant module: total number vs. CFIed

Table 4: Knowledge Protection Metrics

3.3. Data collection method

The Eclipse Metric plug-in is used to calculate the source of DCM code. Other product size is collected when the work product is completed and is updated when the project is finished.

We developed a daily log tool based on IBM Rational ClearQuest to collect efforts from the in-house team and the open-community team. In this tool, each team member's activities will be categorized into disciplines (e.g. requirement, analysis and design, test, implementation, etc.) and detailed categories (e.g. CFI partition, CFI specification, CFI implementation, CFI integration, etc.). The observation team sent daily emails to reminder each team member to input their daily efforts properly. And all the effort data is validated by the observation team periodically.

For the quality metric, we also use the IBM Rational ClearQuest to manage all the defects of DCM. And the ODC (Orthogonal Defect Classification) [6] technique is used to analyze defect data.

In order to collect data of knowledge protection metrics, we design some questionnaires to collect qualitative data from developers when the developers deliver their works.

4. DCM project overview

Digital Currency Manager (DCM) is a real SOA application developed for IBM China Research Lab, which has been on-line in December of 2006. DCM includes 10 main functions, including user management, use account management, digital currency exchange rules management, digital currency transfer, transaction management, batch deduction process, food service provider management, food booking service, food orders management.

DCM aligns with SCA standard [7], and application layer and data layer are implemented as the SCA components. Overall, there are 19 components in the application layer and 21 components in the data layer. Each component has about 4 interfaces, and each interface has several operations. The development environment is WebSphere Integration Developer 6.0.1[8] and the application is deployed on WebSphere Process Server 6.0.

5. DCM practice process and data

In this experiment, we do not put UI development works in the CFI scope but only data and functional components. In this practice, interface is taken as the basic unit of CFI, i.e. each CFI includes multiple interfaces coming from different SCA components. In addition, there are also some BPEL components in DCM, so each BPEL component is also taken as the basic unit of CFI.

5.1. Experiment process

The CFI process introduced in section 2 is customized to align with SCA development features in this case study. In this section, we introduce some key activities in the DCM development process which omits some detailed steps just for the simplicity. The following several steps are key activities in this process.

- *Business Key knowledge identification*

In the DCM application, most of the SCA components are Java components. Besides, we also have several BPEL components. For those Java components, they mainly implement the data management or data processing functionalities which do not refer to the business knowledge at all. For the BPEL components, all of them consist of some confidential contents of the business. As a result, we decided not to distribute the BPEL components to the open community. Instead, they were developed by the in-house team.

- *CFI generation and documentation*

CFI partitioning is the key step of the whole CFI approach which is different from the traditional software development process. As mentioned in section 2, the granularity of CFI partition can be very flexible. In our study, we partition the project into CFIs each of which include several interfaces. In this experiment, 54 CFIs are generated, which are allocated to the 13 developers of the open community. Besides CFI document, the whole DCM project is partitioned as different pieces based on CFI documents. So each CFI has a corresponding project which only includes the necessary content related to this CFI.

A key difference of CFI partition from the distributed development is that different holders of CFIs do not share the implemented contents each other although the dependencies among the different CFIs are kept for debugging and unit testing. That is to say, if interface A depends on another interface B, the CFI of interface A includes the definition of interface B but no the implemented contents of it.

In this experiment, the CFI partitioning activity is conducted completely manually since CFI partitioning supporting tools are still under development.

- *CFI development*

The CFIs generated from CFI partitioning are distributed to the open community. Each developer of the open community receives their applied CFI packages. A CFI package includes a CFI document, the DCM partial project, and some test cases. Then these developers can import the DCM partial project into their workspace and implement them by referring to the CFI document. During the implementation, the developers can have communications with the in-house team for clarifying some unclear design points. At the same time, when the design has some changes, the in-house team updates these changes to the developers.

- *CFI integration*

Once all of the CFIs have been finished by the developers of the open community, the in-house team performs the integration work based on the overall design. The integration testing is also conducted as follows. In our study, the defects found during the integration phase are fixed by the in-house team.

In this experiment, the unified test cases are designed manually and applied to test the system in integration testing. In system testing, some main functions of DCM are implemented with BPEL (Business Process Execution Language) [9]. BPELTester [10] is applied to help generate the test cases for these BPEL processes.

5.2. Measurement results

In this section, we present the measurement results according to the pre-defined metrics.

- *Product size:*

Product size results are showed in Table 5.

Product Type	Metric Description	Unit	Size
Requirement Size	Number of use case	#	33
	Number of use case transaction	#	57
	Requirement Specification	# of page	70
Design Model Size	Software architecture description (design specification)	# of page	70
	Number of CFI	#	53
	Number of database table	#	20
Implementation Model	New developed source code (NCNB, no comment, no blank)	SLOC	25860
	Reused source code	SLOC	0
	Number of class/interface	#	281
	Number of JSP files	#	78
	Number of XML, WSDL, XSD	#	188
Test Model	Number of test case	#	211
	Page of test case specification	# of page	154

Table 5: Product Size Metrics

- *Project resource profile:*

Project resources include resources of the in-house team and the open community, where in-house team includes 5 IBM employees and the open community includes 13 graduate students from universities. The resource profile is given in Figure 2.

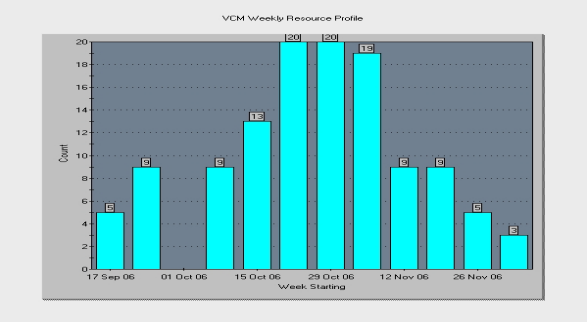

Figure 2: DCM project staffing profile *(Note: Oct. 1- Oct. 8 is the public holiday)*

- *Efforts and cost:*

DCM Efforts distribution by disciplines and CFI activities are given in Table 6 and Table 7. Cost profile is given in Table 8.

Discipline	Effort (hours)
Analysis and Design	826
Business Modeling	82
Deployment	14
Environment	126
Implementation	975
Others	102
Project Management	90
Requirement	176
Test	343
Training	32
Total:	2765

Table 6: Efforts distribution by discipline

CFI Related Activity	Effort(hours)
Conduct CFI communication	10
Develop CFI WBS - prepare	103
Develop CFI WBS - review & adjust	55
Handle CFI change request	68
Perform CFI integration	185
Perform CFI Promotion	6
Write CFI specification - effort on spec	369
Write CFI specification - effort on test mock	27

Table 7: Efforts distribution by detail CFI activities

Effort Distribution	Effort (hours)
In-House Effort	1543
Open community	1222

Table 8: Cost profile

- *Quality:*

The number and age data of defects are presented in Figure 3 and Figure 4.

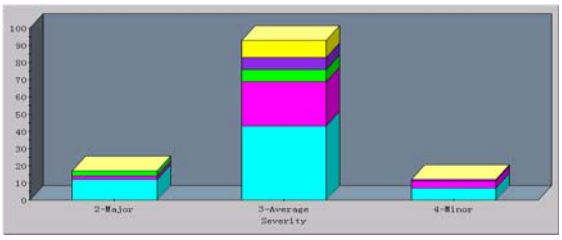

Figure 3: Defects number chart

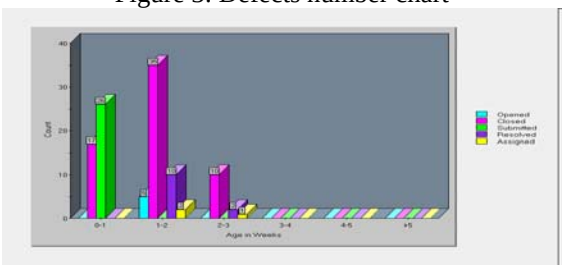

Figure 4: Defect aging chart

- *Knowledge protection:*

Knowledge protection metric	Data
Keywords: total number vs. averaged CFIed (i.e. Distributed in CFI)	40 vs. 4
Key features: total number vs. average CFIed	10 vs. 1
Key business process: total number vs. CFI	5 vs. 0
Average percent of Business Object(BO) fields in CFI	52%
Architecture significant module: total vs. average CFIed	10 vs. 1

Table 9: CFI knowledge protection data

5.3. Metrics result analysis

In this section we conduct analysis on the collected data. First, the project summary data is consolidated in Table 10. The analysis is presented afterwords.

	Total size	New developed	Reused
Product size	34727 SLOC(NCNB)	34727 SLOC	0
	754 FP (Function Point)	754 FP	0

	Effort (hours)	Effort (man-month)
Project Effort	2765	2765/6/22 = 20. 95 We assume one month has 22 working day and each day has 6 working hours

	Total calendar days	Project start date	Project close date
Project Duration	77	Sept. 18, 2006	Dec. 4, 2006

	Number of defects before shipping	Number of defects after release
Defect	122	7

Table 10: Project summary data

- *Productivity:*

34727 SLOC / 20.95Man-Month=167.85 SLOC/Man-Month

754 FP / 2765Hours*100= 27.30 FP/100Hours

Comparing the productivity of the CFI method with the data from ISBSG (International Software Benchmarking Standards Group, the average productivity from ISBSG is 12.87 FP/100Hours (196 new dev projects, java, 2000-2005) [11]), it is obvious that the productivity of DCM project is higher than the ISBSG average productivity.

- *Defect density:*

(122+7)/34.727=3. 715 defects / KSLOC

It is not enough to judge the quality only from the defect density. But from the fact that only 7 field defects (no critical/major defects) are found after DCM online 90 days, we can conclude the quality is acceptable to some extent.

- *Lifecycle effort distribution:*

The comparisons on efforts distribution by each discipline of our case study and data of typical RUP [3] are given in Table 11.

	RUP	DCM case
Project Management	11%	8%
Requirement	11%	9%
Analysis & design	19%	30%
Implementation(Code and Unit Test)	27%	35%
Test	20%	12%
Deployment	6%	1%
Environment	6%	5%

Table 11: Efforts comparison between DCM case and typical RUP

From the above comparison, we can have the following facts: the percent of analysis & design (30% vs. 19%) and implementation (35% vs. 27%) discipline is a bit higher; test discipline is a bit lower (12% vs. 20%). In the collected data, the CFI integration effort is allocated into the implementation discipline instead of test discipline. That is the partial reason why implementation is a bit higher and test is a bit lower than that of RUP profile. Through the analysis, we can conclude that more efforts in CFI method would be allocated to the analysis and design discipline.

- *CFI efforts profile:*

Figure 5 shows the CFI efforts profile.

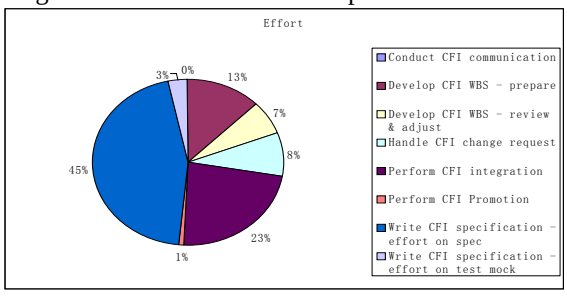

Figure 5: Efforts distribution by CFI activities

CFI related activity accounts for about 30% effort in the whole development. Among the CFI related activities, CFI specification writing, CFI integration and CFI partition are the top 3 effort-consuming activities. We are developing some supporting tools

which are expected to help reduce manual CFI activity efforts in the future.

- *Cost model:*

In the DCM project development, there are two kinds of development resources: in-house developers and the open community developers. We assume that the cost rate is *r1* for in-house developers and *r2* for community developers. The efforts contributed by in-house developers are 1543 hours and the efforts of the open community developers are 1222 hours. Cost saving rate is:

[(1543+1222)*r1–
1543*r1+1222*r2)]/(1543+1222)*r1
= 1222 *(r1-r2) / 2765*r1
= 0.44 * (1- r2/r1).

If r2:r1=1:3, the cost saving rate is: 0.44*(1-1/3) *100%= 29.48%, if r2:r1=1:2, the cost savings rate is 0.44*(1-1/2) * 100% = 22%.

- *Knowledge protection:*

We conducted questionnaire survey within the open community developers to assess knowledge protection results. The survey results showed that only the knowledge carried in the CFI documents has been grasped by the developers. All the open community developers do not know the overall project background, overall application architecture, and BPEL processes. In fact, these results are natural and intuitive because there are no communications among these developers and CFI is the only channel for them to acquire knowledge of the project.

6. Findings and lessons learnt

Through the DCM project, we have the following findings and lessons acquired from the CFI application development approach:

1.CFI method can be used for SOA application development
2.CFI partitioning is feasible to protect knowledge
3.CFI method can reduce development cost and improve productivity
4.The quality of application developed in CFI method is acceptable
5.CFI related activities account for about 30% efforts in the whole development. Some supporting tools are necessary to reduce the efforts of CFI partitioning and documentation. Specifically, we learned a lot in quality assurance, knowledge protection, and the overall method through this case study.

For quality assurance, we have the following three main findings. Firstly, the correctness and clearness of CFI document are of special importance for assuring the quality. A detailed and unambiguous specification could help avoid errors in CFI development due to misunderstanding. Secondly, the automatic testing framework is important for bug identification at the earlier stages. Thirdly, in the integration of CFIs, the test cases with boundary values are critical to reveal bugs. Through strictly following CFI specification, some logic errors and interface errors may be avoided. A lot of bugs come from the missing branches for the boundary value such as the null value of the string type. Therefore, to design test cases with boundary values would be helpful in revealing the bugs.

From knowledge protection perspective, this case illustrated that the CFI partitioning is an effective way to protect the whole design related knowledge, such as application architecture and application holistic function structure. In fact, this result is natural and intuitive because each developer only has a local view of the application. Another important purpose of this case study is to validate that each developer can correctly perform implementation works under the constraints that they only have limited knowledge of the project. This case study has illustrated this point.

From the CFI overall method perspective, this case study at least proves that SOA applications can be developed using the CFI method. As you have noticed in this paper, CFI partitioning is closely related to application types, we should not make the conclusion that CFI method can be applied to any applications only through this particular SOA case. In addition, the following five specific problems of the CFI method identified from this case are useful for the further study of the CFI method: Firstly, the open community developers have different coding styles and skills, which may lead that the overall coding style of an application is inconsistent. It is necessary to provide effective way to avoid this problem in the CFI method deep study. Secondly, how to guarantee the developers to finish their works on schedule is a big challenge. CFI implementation can be regarded as the service delivered by the open community, so the CFI method can leverage some experiences in the service level agreement (SLA) area to make contract with the open community developers. Thirdly, CFI documentation is a kind of labor-intensive work, which demands us to provide some supporting tools. Fourthly, the rigorous quality assurance process is proposed in the CFI method. However, code review and test coverage checking may spend big efforts of the in-house team. We need further study on the balance of quality assurance process and efforts control. Fifthly, the large volume of resources of the open community is a sugar for the CFI method. However, how to effectively communicate with them in CFI document clarification, CFI implementation checking, code review, and CFI

bug fixing are also great concerns. Finally, how to handle CFI change is another critical problem for the in-house team.

7. Related works

The CFI method can be considered as the combination of open source software development method and software development outsourcing model. There are lots of existing research works on outsourcing [2] and open source software development model [1]. We pay more attention to the research works on discussing knowledge management issues of outsourcing [12] and the skill barriers issue of open source software development [13]. As our survey, a lot of papers only mention these issues or give high level methods but without providing concrete technologies.

Quality assurance of the CFI method can leverage some existing works, for example, the quality assurance activities in open source model is a good reference. In face of the quality assurance problems in open source development, Adam Porter etc. [14] designed a Skoll DCQA (distributed continuous quality assurance) process to help assure the product quality in open source development, in currently, the process is applied to the development of ACE+TAO. In CFI method, to leverage development resource is one of the major targets. In fact, how to leverage this resource to disperse testing works is our future research works for the CFI method. Luyin Zhao etc. [15] found that the user participation is considered as the major quality assurance activity. In the survey of Luyin Zhao, 20% to 40% of the faults in 20% of the projects are found by users, and 44% of the respondents thought that users found “hard” bugs. This work reminds us that we should study how to allocate system and function testing works to open community.

8. Conclusion and future works

In this paper, we present the experiment process of developing an SOA application, Digital Currency Manager, using the CFI development method. Through the analysis on the experiment data, we have validated some hypotheses of the CFI method, i.e. the development cost of SOA applications by the CFI method can be dramatically reduced by leveraging the cheaper open community resources, the quality of delivered application can be assured, and the key knowledge of the application can be avoided losing because of some specific technologies provided by the CFI method, such as CFI partitioning and business keywords replacement.

In the future, more experiments will be conducted to validate that CFI method is appropriate for other kinds of applications. In addition, we will continue to do the further deep study on knowledge protection technologies, such as business keywords hiding and information exposure control. For quality assurance perspective, unified test technology is one of research keystones. How to partition function and system testing works to CFIs is another research focus.

9. References

[1] T. O′reilly, Lessons From Open-Source Software Development, Comm. ACM, vol. 42, pp. 32-37, 1999.

[2] Huff, S. L., Outsourcing of Information Services, Business Quarterly, Spring 1991, pp. 62-65.

[3] Kruchten, P., Rational Unified Process – An Introduction, 1999: Addison-Wesley.

[4] IBM Rational ClearQuest, http://www-306.ibm.com/software/awdtools/clearquest/index.html.

[5] Robert E. Park, Wolfhart B. Goethert, William A. Florac, Goal-Driven Software Measurement: A Guidebook, 1996, http://www.sei.cmu.edu/pub/documents/96.reports/pdf/hb002.96.pdf .

[6] ODC, http://www.research.ibm.com/softeng/ODC/ODC.HTM.

[7] Service Component Architecture Specifications, http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications

[8] WebSphere Studio Application Developer Integration Edition (WSAD-IE). (2006). http://www-306.ibm.com/software/integration/wsadie/support/ .

[9] Business Process Execution Language for Web Services. (2003);ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf .

[10] Zhongjie Li, Wei Sun, Zhongbo Jiang, and Xin Zhang. Bpel4ws unit testing: framework and implementation. In the proceedings of the 2005 IEEE International Conference on Web Services (ICWS’ 2005), pages 103-110. Orlando, Florida, USA, July, 2005.

[11] ISBSG, http://www.isbsg.org.

[12] L. Zhao, T. H. Yim-Teo, K. T. Yeo, Knowledge Management Issues in Outsourcing, International Engineering Management Conference 2004, pp. 541-545.

[13] Georg von Krogh, Sebastian Spaeth, Karim R. Lakhani, Community, Joining, and Specialization in Open Source Software Innovation: a Case Study. Research Policy 32(2003) pp. 1217-1241.

[14] Adam Porter, Cemal Yilmaz, Atif M. Memon, Arvind S. Krishna, Douglas C. Schmidt, Aniruddha Gokhale. Techniques and Processes for Improving the Quality and Performance of Open-Source Software. 11(2): 163-176. Software Process: Improvement and Practice, 2006. John Wiley & Sons, Ltd.

[15] Luyin Zhao, Sebastian Elbaum. Quality Assurance under the Open Source Development Model. 66(1): 65-75, Journal of Systems and Software, 2003. Elsevier Science Inc. New York, NY, USA.