J. Software Engineering & Applications, 2009, 2:34-39
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

Call for Implementation: A New Software Development Mode for Leveraging the Resources of Open Community

Weiping Li[1], Weijie Chu[1], Ying Liu[*]

[1]School of Software & Microelectronics, Peking University, Beijing 100871 China, *IBM China Research Lab, Beijing 100094, China
Email: wpli@ss.pku.edu.cn, chuwj@ss.pku.edu.cn, aliceliu@cn.ibm.com

Received December 31st, 2008; revised February 6th, 2009; accepted March 10th, 2009.

ABSTRACT

With the growth of the internet and open software, there are additional software developers available from the open community that can participate in the development of software application systems. Aiming to leverage these resources, a new development model, CFI (*call for implementation*)*, is proposed. The basic idea of CFI is to publish some part of a software project to the open community, whole or part, in certain phases of the software development lifecycle to call for implementation. This paper discusses the basic concept and method for a software development process in CFI mode. Two different modes of CFI with different granularities are analyzed. And one of the CFI modes, fine-granularity-CFI mode, is thoroughly discussed including the main methods and basic steps. To verify the ideas a pilot project, an online store system, is built up with the CFI development process. The online store system takes the traditional Model-View-Control architecture and some common technologies such as Struts, Hibernate, Spring are used. The result shows that this new kind of software development mode is feasible though there are many problems that are still requiring further study.*

Keywords: *Call for Implementation*, *Software Development*, *Open Community*, *Integration*

1. Introduction

Open community, which is composed of students, programming fans, SOHO (small office/home office), etc., is a virtual development resource pool [1]. With the development of open source software, open community is now booming, which contributes many new ideas and solutions to some frameworks and common solutions.

Currently there are many open source software programs available in many web sites such as Apache, SourceForge, and many others. The Apache Software Foundation provides support for the Apache community of open-source software projects [2]. SourceForge.net provides free hosting to open source software development projects with a centralized resource for managing projects, issues, communications, and codes [3]. Besides the open source software, open community provides additional software development resources that, when properly compensate financially, can do the coding work for certain software components. For example, TOP CODER [4] provides a platform for the open community developers on which some software components are available for implementation. There are even small-size project provided on ScriptLance [5]. On SXSoft [6], a Chinese website, both small software components and small sized projects are available. And this new mode of development occurred only one or two years ago.

As we can see, there already exist some practices on using the resource of open community for coding. Though currently the components that done by the open community developers are small size and the main tasks are coding, it is really a new phenomenon which inspires us to leverage the rich resources of the open community. To follow this idea and practice, some companies are going beyond outsourcing small components to outsourcing the whole project. In this whole project outsourcing mode, we need some new methods and technologies to manage the whole project. We call this new mode is Call for Implementation (CFI). Fortunately there are some research efforts on outsourcing [7] and open source software development model [8]. And there are also some previous research efforts and practices on CFI at IBM China Research Lab, where some researchers finished a CFI project in the popular SOA architecture [1,9].

Sponsored by IBM China Research Lab, we, Peking University, have been working on a joint study CFI project, in which some methods for requirement analysis, designing and partitioning the project have been studied. This paper describes the basic ideas and methods for CFI and a practice that verifies the ideas in a pilot project.

The rest of this paper is organized as follows: Section 2 describes the basic ideas of CFI and the two different modes for CFI as well as some extra work which the project owner should take into account. Section 3 depicts in details the ideas, steps, and methods for one of the CFI

model, i.e. fine-granularity-CFI mode. Section 4 puts forward a pilot project finished in the fine-granularity-CFI mode. Section 5 concludes the paper and presents the future work.

2. CFI Overview

2.1 The Concept of CFI

CFI is a new kind of development style that is different from traditional software development within an organization. Under the CFI mode, usually the owner of a certain software development project will publish some tasks on their website, or a specific platform, to call for implementation. The task here means a piece of work with a reasonable granularity such as a java class, a component, or a JSP page that can be implemented by a single developer. And usually a task is a package that is composed of different kinds of the artifacts. After finishing the tasks the developers will submit them back to the owner. The project owner then continues the testing and integration work. After all the necessary work done, the whole system is built up. The conceptual model of CFI is depicted in Figure 1.

Given this idea, we can tell the differences between CFI development and conventional development. In the normal software development lifecycle, there are the basic tasks such as requirement analysis, general design, detailed design, implementation, testing, and maintenance. Usually the process used in a certain software system development is one of the following models: Waterfall Model, Spiral model, RUP [10], etc. Whereas in the CFI model some extra work should be added to implement this new kind of development, namely, publish and submission. These two tasks can be added into certain phase of the process according to the policy of CFI. In the traditional process it is relatively easy to get feedbacks from different roles and the iterative process can be used. But in the CFI situation, it is hard to get any feedback and accordingly hard to take the iterative process.

There are two basic modes for CFI process, coarse-granularity-CFI and fine-granularity-CFI mode, which can be seen in Figure 2 and 3.

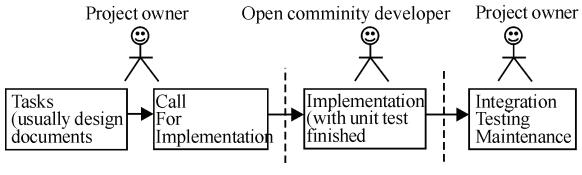

Figure 1. The conceptual model and process of a CFI project

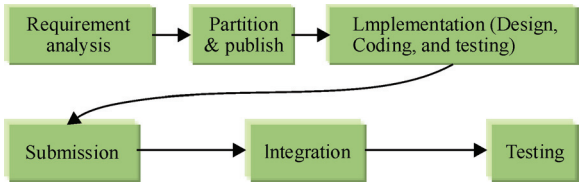

Figure 2. The coarse-granularity-CFI development process

Figure 2 shows the coarse-granularity-CFI mode, in which the publisher, i.e., the project owner, just publishes the requirement to the open community while the developer will perform further tasks such as design, coding, and testing. In the coarse granularity mode, usually the project owner should firstly design the structure of the database and the architecture of the software and determine the technologies used in the project. The developers should work under these constraints in order to achieve a smoother integration among the individual tasks submitted by the open community developers. For each task, the implementation includes the design, coding and unit testing in this mode.

Figure 3 shows the fine-granularity-CFI mode, in which the publisher will finish the designing work and publish the tasks which contain some part of the design artifacts. The implementation in this mode includes coding and partial unit testing.

2.2 The Main Challenges of CFI

As compared with the traditional software development process, there arise some challenges for the CFI development process that the project owner should balance among them. The main objective of CFI is to leverage the development resources in the open community. When taking advantage of this potential resource, the project owner has to face some extra effort for testing, integration, and potential quality decline. So the following issues should be taken into account in managing for CFI project.

• Information concealment: When publishing the tasks to the public, all the information that is not related to a certain task should be concealed. Only the needed information about the task itself will be published.

• Granularity: Fine granularity means the number of tasks will be increased accompanied by the difficulty of integration test, while the coding task itself is easy to implement. On the other hand, relative coarse granularity means single task will be hard to implement while decreasing the difficulty of integration and testing.

• Testing policy: Partitioning the whole project into parts creates another challenge: testing. And the main difficulty comes from the unit testing. Usually a class in a certain task may link with other class(es) in other task(s) and different tasks will be done by different people. So it is impossible for the classes in different tasks to run together until at the integration phase done by the project owner. The developer can write the unit test cases but it is hard to run them.

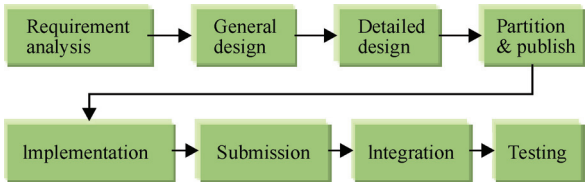

Figure 3. The fine-granularity-CFI mode development process

• Integration: The codes are from different developers and those developers are usually not available in the integration phase. So how to ensure the code quality is another big challenge, which the project owner and the core team have to face the challenge.

3. The Partition Methods for CFI Project

There are two CFI modes according to the granularity of tasks. Currently we are studying the fine-granularity-CFI mode and the corresponding partition method and rules for CFI development. Also some pilot projects have been done to verify the ideas of CFI. The study for the coarse granularity mode remains to be study.

For the fine-granularity-mode, the project owner should firstly finish the detailed design and then partition the artifacts into small pieces of tasks that will be implemented by different people. Suppose the designing work is as usual. To publish the tasks on the platform in this mode, we start from the design document, and focus mainly on the class diagram, which provides the foundation for further partitioning of the whole project. With the relationships among classes the big picture of the project can be obtained. To assure that all the development tasks can be included into the big picture and ready for further partition, the GUIs of the project, e.g. JSPs, should be included into the big picture, which is actually an extended class diagram.

The question now is how to partition the big picture, i.e. the extended class diagram, into tasks. For this purpose, some rules can be introduced to help the partition. Next we will use an example to describe the methods. Figure 4 shows a very simple big picture with the Java Script Pages (JSPs) included as classes. This class diagram follows the common Model-View-Control architecture. The nodes in the figure are classes or JSPs and the arcs indicate the relationship among the nodes. There

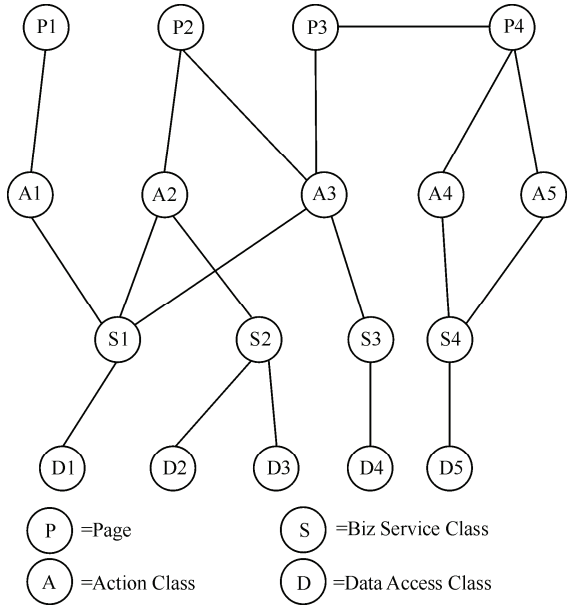

Figure 4. The extend class diagram for CFI partition

are four kinds of nodes, namely, JSP, Action, Business Service, and Data Access Object (DAO). JSP stands for the GUI. Action stands for the Action Servlet in the Struts framework. Business Service represents the classes that fulfill the real business processes and functions. And DAO is the class for accessing the database. Each node gets a flag that indicates its type, e.g. JSP, ACTION, SERVICE, or DAO.

To split the whole picture into parts, namely tasks, some rules will be followed at the beginning of the partitioning process by the project manager or the architect. There exist some basic rules:

• A node (with certain type defined) can be placed into one task with the node(s) (with another type defined) linking to it.

• A node (with certain type defined), although having relations with others nodes, can be partitioned into one task.

• For the isolated nodes that have no arcs linked with any other nodes will be naturally in a task.

• A node can be placed into ONLY ONE task.

Besides these rules, when in actual situations, some other rules can be introduced when needed.

For the class diagram in figure 4, let's define the following rules:

1) The nodes with type JSP and nodes with type ACTION that are linked together will be grouped into one group.

2) The node with type SERVICE or DAO forms one single group.

Conforming to these rules, the class diagram can be grouped into thirteen groups, i.e. tasks. They are:

(P1, A1), (P2, A2, A3), (P3), (P4, A4, A5)
(S1), (S2), (S3), (S4)
(D1), (D2), (D3), (D4), (D5)

Because both A2 and A3 are linked with P2, they are in the same group. Although A3 is linked with P3, P3 itself forms a group because A3 is already in another group. Once all the groups are determined, they are ready to be published to the open community as tasks.

Before publishing these tasks to the open community to call for implementation, there is still some extra work that should be done. For example, usually the database design, some common interfaces and classes, the Cascading Style Sheets (CSS) of JSPs and other necessary building blocks need to be delivered to the open community developers.

4. A Pilot Project

To verify this new development mode, we started a pilot project using the CFI methods. The project is to develop an online store application system, using the fine-granularity-CFI mode. There are seven modules in this application system and we select on of the simple modules as

the example to explain the CFI mode. Before this pilot project, a platform supporting CFI development was already developed by another team at Peking University under Professor Yu Lian [11]. The platform provides the necessary functions for the CFI development such as task publishing, payment, project management and so on. We use this platform to partition the tasks, to publish and receive the finished tasks, and to monitor the project.

4.1 The Online Store Project Overview

There are two teams involved in this project, one core team and one development team. The core team works on requirement analysis, detailed design, partitioning, publishing tasks, integration, and testing, while the development team works on coding and unit testing. There are eight persons in the core team: two teachers act as project managers, one senior graduate student as the architect and five junior graduate students as designers. A total of 45 students are in the development team that works on coding and unit testing. The 45 students here simulate the open community developers. Currently the integration is finished and next work is to test the system.

The time table is as follows:

- Oct. 15-Nov. 15, Requirement Collection,
- Nov. 19-Dec. 10, Detailed Design,
- Dec. 11-Dec. 12, Partition and Publishing,
- Dec. 13-Dec. 21, Coding and Unit Testing,
- Dec. 24-Dec. 29, Integration
- Feb. 18-.Mar. 30, Testing

There are seven modules in this online store application system, i.e. Customer Management, Shopping Cart, Sales, Procurement, Inventory Management, Consignment, and System Management. The system adopts the Model-View-Controller architecture and is developed in Java. The development toolkit is Rational Software Architect, Rational Application Developer 6.0 and the server is WebSphere Application Server 6.0. The main technologies are Struts, Hibernate, Spring, JUnit, etc.

4.2 An Example for Partition

Let's take the procurement module as an example to demonstrate the partitioning process in CFI. Procurement module is a simple module that is responsible for the procurement of the merchandise to be sold on the online store. This module follows a standard procurement process, including creating a purchase requisition, examining and approves a purchase requisition, creating a purchase order, examining and approving a purchase order, getting manifest, and warehousing. The class diagram is shown in Figure 5. There are thirteen JSPs, seven Actions, three business services and three DAO class in the extended class diagram.

Rules used

There are four rules used in this pilot project.

1) Every business service will be in a single group

2) Every DAO class will be in a single group

3) A JSP and its linked Actions will be grouped together.

4) If one action class has more than one linked JSPs then this action class will be grouped with one of the JSPs randomly.

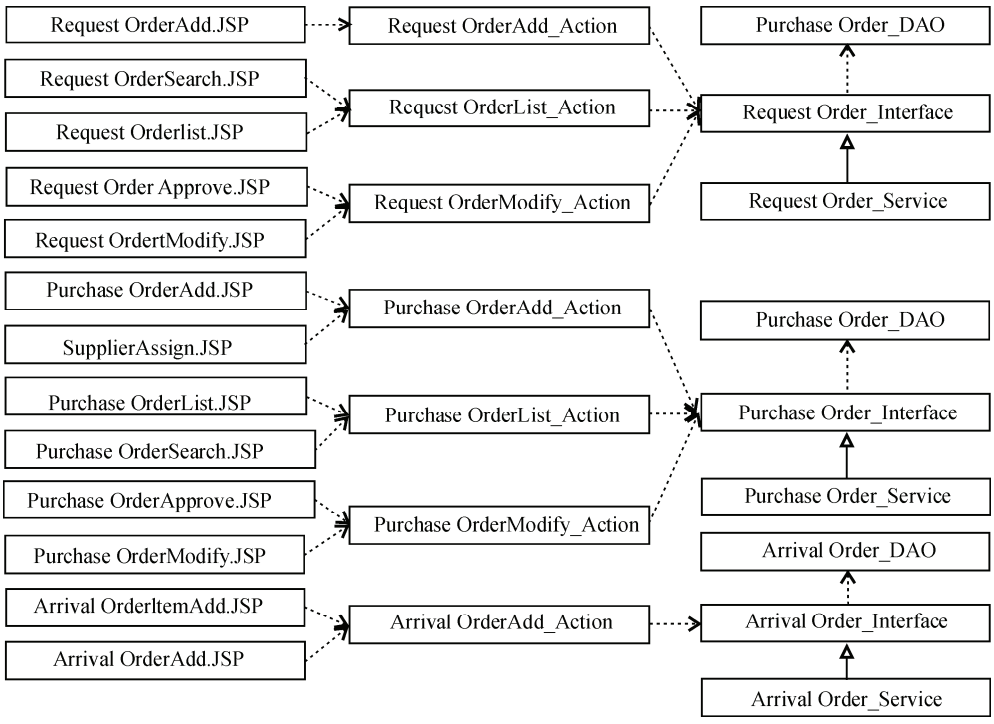

Figure 5. The extended class diagram for procurement module

The partition result

According to the rules the procurement module was divided into 16 groups, i.e. 16 tasks, see table 1. The three business services are grouped as single tasks according to rule No. 1, see task No. 14, 15, and 16. All the other JSPs and Actions are divided into the other 13 groups.

From Figure 5 we can see that ArrivalOrderAdd_ Action links with both ArrivalOrderAdd.JSP and ArrivalOrderItemAdd.JSP. According to rule No. 4, they are placed in group 12 and 13 respectively.

The DAOs in Figure 5 can be generated automatically by the Hibernate tool so they are not included in the tasks. That means they are finished by the core team. Also there still some other classes those were not included in the tasks and hence not published to the open community.

Before publishing the tasks on the platform, two extra efforts should be made. One is to package the right parts of design document with the tasks, which will help the developers to understand the design. Another is to build up the project framework and the some components, such as DAOs mentioned above, will be provided in the development toolkit, i.e. Rational Application Developer. The components in the project include the package and class name, the jar files that are imported into the project, some common interfaces and tools, and so on. This would provide all the developers the same working environment.

Additionally, the database design should be finished by the core team before the publishing or at least before the integration phase.

4.3 Analysis of the Practice

Compared with the traditional software development process, the CFI mode does have following different characteristics.

Integration

The integration is done by the core team. Before publishing the tasks the framework of the project is already built so the integration is relatively easy. Integration is simply put all the code on the right place, i.e. right package and right directories. The code may be finished by the open community developers or by the core team members.

Table 1. The partition result

1	Purchase 0rderAppmve JSP	9	Request 0rderModify JSP Request 0rder Modify_Action
2	Request 0rderModify JSP Request OmderModify_ Action	10	Request 0rdzer List JSP
3	Purchase 0rderList JSP	11	Purchase 0rderSearch JSP Purchase 0rderList_ Action
4	Request 0rderSearch JSP Request 0rderList_Action	12	Arriwal 0rderItmAdd JSP
5	SupplierAssign JSP	13	Arriva0rderAdd JSP Arrival0rderAdd_Action
6	Request0rderAdd JSP	14	Reques0rder_Service
7	Reques0rderAppnove JSP	15	Purchase0rder_Service
8	Purchase0rderAdd.JSP Purchase0rderAdd_Action	16	Arrival0rder_Service

The configuration file used in the project, namely Spring and Hibernate configuration files, shall not be open to all the developers for the purpose of information concealment. But usually there exist one configuration file for the whole project. In this project we cut the file into pieces according to different tasks' needs. This raised the extra work for cutting the file and reuniting the pieces into one integrated file.

Testing

The developers have finished the coding and, accordingly, have written the test cases for unit test using JUnit methods. But mostly unit test cases cannot run until the integration is finished because every single developer can only get a task, which may have to call other interfaces that will be implemented by other developers. Although they can use surrogate(s) to help finish the testing, it must be retested after integration. The core team will continue to complete the test wok. During this period, it will be lucky if we can find the developers to correct any errors, which we suppose won't do it because they have been already paid. Even if we are lucky, we can find the right persons, it is still hard to correct some errors which have relationships with other modules.

Code quality

Usually the code style is hard to control. The project owner may define the code style for the project. But it is hard to ensure all the open community developers abide by it well. Because the unit test can not be complete before the integration it is also difficult to ensure the quality of the code.

The code quality is deeply depended on the skill and experience of the programmer. This is a risk for the success of the project.

Surely the quality of the design document will impact the quality the code. And the partitioning work adds extra difficulty to the design documents.

5. Conclusions

This paper analyzes the basic ideas of Call for Implementation, which can help an enterprise to leverage the resources in the open community. The basic methods for CFI are introduced. Especially the partition method for dividing the class diagram into tasks is introduced. Based on this new idea and corresponding methods, a pilot project has been developed in the so called fine- granularity-CFI mode aiming to verify the methods.

From the pilot project we have found some interesting points that differentiate CFI from traditional software development. Some extra work is needed for the CFI development such as partitioning the design work, publishing the tasks, integrating the artifacts, and preparing the project framework.

1) In CFI mode, the coding work is finished by the open community programmers while the unit test will be

done by the core team. And there arise the potential risk of lowered quality and even more workload for correcting the errors.

2) Iterative development is regarded as a useful process. But in CFI mode it is hard to send any feedback to the developers. So it is hard to use this kind of process.

3) In CFI mode every task is a group of classes that may have some natural relationships with other task(s). So the information of the project is hard to be concealed. The developers must know the interfaces that this task calls. If someone purposely collects all the tasks in some way then he will discover the functions, interfaces, and even the architecture of the project.

4) Based on the previous practice we find two useful principles for partitioning, which we believe can improve the quality of the code. One is that the nodes in the extended class diagram that fall into one use case will be divided into one task. In this way the whole business process of a certain function will be encapsulate into one task so it will be done by one single programmer. This eliminates the communication among different programmers and hence improves the quality of the code. The other is that the controller classes in terms of Model- View-Control, namely the main process, shall be assigned to the core team. This will help conceal the business process and also improve the quality of the code.

Still some issues are not covered in this pilot project such as the information concealment and quality assurance, which need further study and practice. Or you can find some valuable information and analysis in paper [9].

6. Acknowledgement

This research is funded by the IBM China Research Lab and the China 863 project (No. 2007AA04Z150).

REFERENCES

[1] Y. Liu, C. H. Feng, W. Zhao, H. Su, and H. H. Liu, "A case study on community-enabled SOA application development," IEEE International Conference on Service-Oriented Computing and Application, Newport Beach, California, June 19−20, 2007.

[2] http://www.apache.org/.

[3] http://sourceforge.net/.

[4] http://www.topcoder.com.

[5] http://www.scriptlance.com/.

[6] http://www.sxsoft.com/.

[7] S. L. Huff, "Outsourcing of information services," Business Quarterly, pp. 62−65, 1991.

[8] T. Oreilly, "Lessons from open-source software development," Communications of the ACM, Vol. 42, pp. 32−37, 1999.

[9] X. Zhou, Y. Liu, *et al.*, "Case study: CFI-enabled application development leveraging community resource, 2008 international conference on Service Science, April 17−18, Beijing.

[10] P. Kruchten, "Rational unified process-an introduction," Addison-Wesley, 1999.

[11] A private discussion with Professor Yu Lian from Peking University.

