Supporting Dynamic Communications with Development Histories

Kei Sasaki†, Makoto Matsushita†, Katsuro Inoue†
†Graduate School of Information Science and Technology, Osaka University
{k-sasaki,matushita,inoue}@ist.osaka-u.ac.jp

Abstract

In a open source software development, developers use revision control system for efficient management of product, and mailing list for communication among developers. These systems store development histories of the products. Developers can obtain a deeper understanding about former development by reviewing an archive. In addition, we think that developers have "task" and "knowledge" in their development histories. But, as those information become immense volumes, it is not easy to retrieve the information from the stored information that developers require.

In this paper, we purpose software development supporting system CoxR for searching development history information. Additionally, focusing "developer", we aim to extract effectively "task" and "knowledge" as "Dynamic Community". Then CoxR will enable developers to reduce communication costs and improve software quality.

1 Introduction

In a open source software development, developers use revision control system for efficient management of product, and mailing list system for communication between developers. These systems store development histories of the products which the developer developed, and the transmitted E-mails at the individual archive. The archive information include much information which is useful to future development. Developers can obtain a deeper understanding about former development "task and knowledge" by reviewing an archive in the software development, and it is expectable that they help developers. In addition, if new developer wants to use or develop the existing system, he/she must refer the history information or question for Developer's community. Stated another way, we think that developers have "knowledge" and "task" in their development. We call the group of these "task and knowledge" to " Dynamic Community", because task and knowledge are floating for each developer which needs history information. However, as those information become immense volumes, it is not easy to retrieve the information from the stored information that developers require.

So, we purpose the CoxR[6], extracting information that developers require and displaying pertinent data by integrating supporting system for source code modification (CoDS)[7] and software products cross reference system(SPxR)[5]. Also, we applied this system to real data in open source development for applicable experimentation. Next, we purpose how to design new supproting system extending CoxR. We focus "developer history" of CoxR. Finally, we aim to extract the "task" and "knowledge" effectively . Then CoxR will enable developers to reduce communication costs and improve software quality.

In section 2, we will briefly overview the CoxR. In section 3, we will present a "Supporting Dynamic Communications with Development Histories". In section 4, we will conclude our discussions with a few remarks.

2 CoxR

In this section, we briefly explain the software development supporting system CoxR. CoxR aims at realizing the following 4 functions.

(1) Search the source code similar to the source code which the users are owner of .

(2) Search file name or directory name

(3) Search keyword of history information.

(4) Search additional information again using search results.

2.1 CoxR Structure

CoxR consists of **CoDS module**, **SPxR module** and **main module**.(fig.1) User can use query word as input data.In CoDS module, User can search similar source code.**(1)** In SPxR module, user can search useful history information from CVS and E-mail archive.**(2),(3)** And, CoxR

main is interface which connects previously cited modules and developer. User use search results as next query word.**(4)**

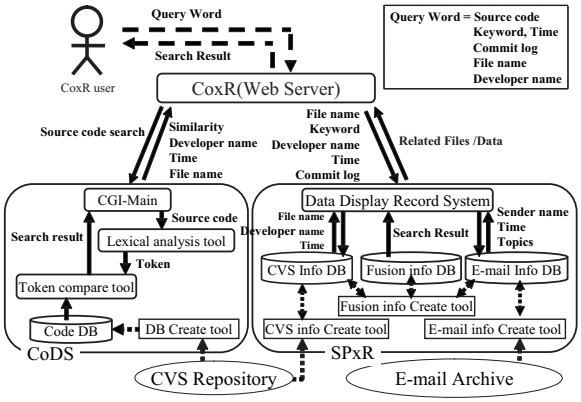

Figure 1. CoxR overview

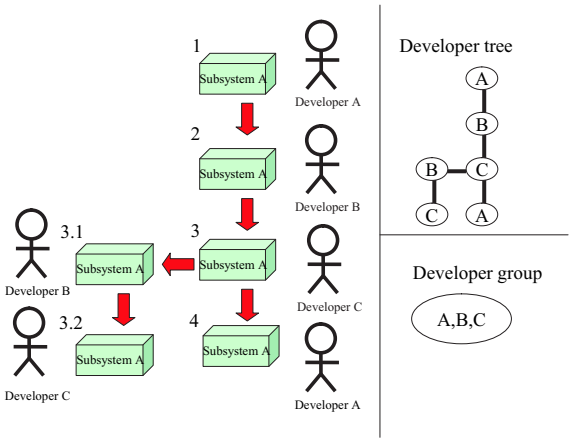

Figure 2. development process on subsystem A

3 Supporting Dynamic Communication System

In this section, we explain how to design new supporting system extending CoxR.

3.1 Analysis of Dynamic Community

We define three analysys patterns as follows. We have realized to extract "developer history" of former development in the CoxR. We use these history information to analyze former development.

3.1.1 Process Analysis

We analyze the tree of developer history. Does the group of developers develop the system in similar development process?

- **Building up developer tree**
  We analyze who are often development group together from development history. We assume same development member develop together in some project. We analyze on a root directory-to-root directory basis. Then, we consider the groups form tree of the developers. an example shows fig.2. Three developers develop subsystem A in the order as indicated fig.2. Then, developer tree and group became as indicated.

  And, we assign developers to past development by past development history.

- **Predicting future process**
  We predict future development process statistically, then using the prediction data, we know who develops in a special project.

3.1.2 Topic Analysis

We analyze E-mail and commit log. Who send the topic about "discussion X" ? What does "developer Y" write in the mail and Commit log?

- **Automatic classification**
  We decide keyword sets at each topics. We sort keywords into each topics. Then, we analyze who transmitted what topics. we sort developer groups by each topic.

- **Extract discusser**
  We call graph of mails among developers "thread". We search E-mail archive by using some animator of a discussion's E-mail address in same mail-thread. Developers often develop in special field. So, we think that developers which discussed activity have a high probability that they have relaited discussion previously. Therefore, we think user can get related topics by such develop discusser.

3.1.3 Role Analysis

How do developers develop the system? How do they share system development?

- **Using bug tracking system**
  When we analyze the tree of developer history, we use Bug tracking system in addition to CVS and E-mail archive. Bug tracking system keep track of individual

request for changes. To analyze request, we know who completed the modification request? and What the developer modified ?

- **Using analysis results**
  Using described previously analysis method, we consider developer role in each project history.

3.2 Implementation of the system

3.2.1 Design Policy

Web interface It is most effective approach that creating our proprietary tool.

Cooperate with CoxR User can use this system and CoxR at same time without being aware.

Depend on the user Every user may want different knowledge or task. We can't decide analyzing result to each user. So, our system gives dynamic result from each knowledge and each task which user wants, and user must judge dynamic result which user wants.

3.2.2 System Structure

the system consists of three modules. the modules are "Analysis", "Database" and "System Control" as follows.

- **Analysis module**
  We handle each analysis in this module. We analyzes about developer history information, using CoxR database and Bug tracking system database.

- **Database module**
  After described three analysis, we store result data. Database consists "**Process database**", "**Topic database**' and "**Role database**'. Process database is stored "what group developed what product?", Topic database is stored "who have what knowledge?", and Developer database is stored "Who has what role in each project?". these database return the search result to database query of system control module.

- **System control module**
  We implemente system control by GUI interface, and accept search requests from user. This module passes search requests to Database module as database query. After database search, GUI indicates Search result from database. User can use the result as next search request.

We envisage the creation of Supporting environment for Open Source software development using "CoxR" and "Supporting Dynamic Communication System".(fig3)

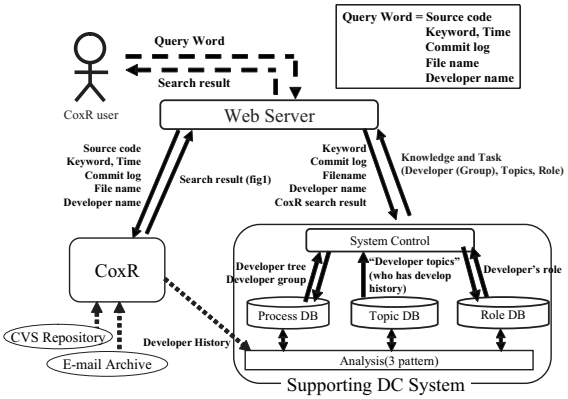

Figure 3. Supporting Dynamic Community system

3.3 Search strategy

We can use this system to break the ice of development problem. First, To search CoxR, we can sight valuable development history about problem which user face. Secondly, to use search result of CoxR to this system, we think user can understand detailed knowledge and task about the deveopment. User can choice every search results of CoxR as next query word to this system. To use such information, user can obtain knowledge and task of development group from various sources, and topic about simillar development which same development and developer involved in.

4 Conclusion

In this paper, we have explained software development supporting system CoxR's overview which we established. Then, we have proposed the implementation design of Supporting Dynamic Communications with development histories by extending CoxR.

The major characteristics of this system is that the analysis of developer history of CoxR. Any more, We will implement system based on a detailed design, and We will verify about the system validity to use the system actually .

References

[1] H. Agrawal, R. A. DeMillo and E. H. Spafford: "An execution backtracking approach to program debugging", IEEE Software, pp.21-26(1991).

[2] Peter H. Feiler, "Configuration Management Models in Commercial Environments", CMU/SEI-91-TR-7 ESD-9-TR-7(1991).

[3] Karl Fogel, “Open Source Development with CVS”, The Coriolis Group(2000).

[4] Dan Gusfield, “Algorithms on Strings, Trees, and Sequences”, Cambridge University Press(1997).

[5] Ishikawa, T., Yamamoto, T., Matsushita, M., and Inoue, k.: “Design of Communication Supporting system with Revision Contorol System”, IPSJ Technical Report, 2001-SE-133, pp.23–30(2001).

[6] Sasaki, K., Matsushita, M., and Inoue, K.: “E-mail and Source Code Revision Information Retrieval System for Open Source Software Development”, IEICE Thechnical Report, SS2003-9, pp.19-24(2003).

[7] Tahara, Y., Matsushita, M., and Inoue, K.: “Supporting Method for Source COde Modification with the changes of Existing Software”, IPSJ Technical Report, 2002-SE-136, pp.57-64(2002).

[8] Temple Smith and Michael Waterman, “Identification of Common Molecular Subsequences”, J.Molecular Biology, 147, pp.195-197(1981).