CoxR: Open Source Development History Search System

Makoto Matsushita, Kei Sasaki and Katsuro Inoue
Graduate School of Information Science and Technology, Osaka University
1-3, Machikaneyama-cho, Toyonaka-shi, Osaka, 560-8351 Japan
{matusita,k-sasaki,inoue}@ist.osaka-u.ac.jp

Abstract

In typical open source software development, developers use revision control systems for product management, mailing list systems for human communications, and bug tracking systems for process management. All of these systems store development histories of the products that show significant information of problems during the development.However, it would be a hard job to retrieve useful information related to a current problem faced by developers. In this paper, we describe a software development supporting system CoxR that is capable of crawling the development histories. CoxR creates software development information web which consists of developers, emails, and program deltas, and provides an interface to search, navigate, browse, and retrieve past development results. Through a case study, we confirmed that CoxR helps developers to solve their problems by making it easier to search development history.

1. Introduction

In a open source software development, developers all around the world use revision control system for managing their products, mailing lists for communication among developers, and bug tracking system that manages what issues are running. These systems store development histories of the products that the developers developed, and the transmitted e-mails at the individual archive. The archive information include much information that 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 assume that developers have "knowledge" and the knowledge is stored to repositories mentioned above. However, as those information become immense volumes, it is hard to retrieve the information from the stored information that developers want.

So, we purpose the CoxR[9] that extracts information from existing repositories by integrating supporting system for source code modification (CoDS)[11] and software products cross reference system(SPxR)[7]. Also, we applied this system to real data in open source development for applicable experimentation. Next, we implement CoxR as a web system as CoDS and SPxR does, and evaluate with actual data from existing open-source software development. As a result, we confirm that CoxR makes developers possible to get useful information from archives by web navigation.

In section 2, we will briefly overview the CoxR. In section 3, we will present the implementation of CoxR. In section 4 we also show an example of using CoxR. In section 5 we discuss related works of CoxR, and finally we will conclude our works with a few remarks in section 6.

2. CoxR

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

(1) Search by code fragment they have; similar codes and its modification histories in repositories could be found, and it would be useful for future development.

(2) Search by file name or directory name; highly related files could be found.

(3) Search by keywords; information that is associated with the keyword could be found.

(4) Navigate development histories web; information structures are shown as it is, and tracking information easily just like web navigation.

(5) Search additional information again using search results.

2.1. Knowledge

We define the word "knowledge" that is development information accumulated to the repositories, such as revision control system CVS, email archive, and bug-tracking system GNATS. The knowledge consists of "intention" (the reason why such development is occurred), "task" (actual processes of development), and "entity" (knowledge attributes for identification).

The "knowledge" is extracted from the archive contents; "intension" comes from each CVS commitlog, each email subject and body, and each GNATS entry's attributes (category, type of bug, etc), "Task" comes from each CVS code delta, and each GNATS revision histories and PR status, and "entity" comes from each CVS filename, revision number and date, each email message-ID and date, and each GNATS filename, bug-ID and date.

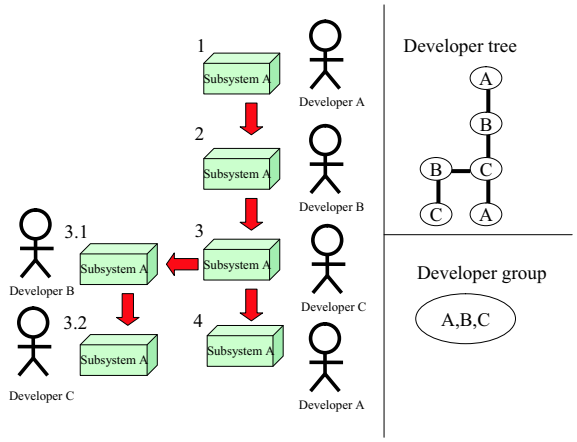

Figure 1. development process on subsystem A

2.2. Repositories Analysis

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

We define three analysis 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.

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.1. Three developers develop subsystem A in the order as indicated fig.1. 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.

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 related discussion previously. Therefore, we think user can get related topics by such develop discusser.

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.

2.3. System Structure

CoxR consists of **CoDS module**, **SPxR module** and **main module**.(fig.2) 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),(4)** And, CoxR main is interface which connects previously cited modules and developer. User use search results as next query word.**(5)**

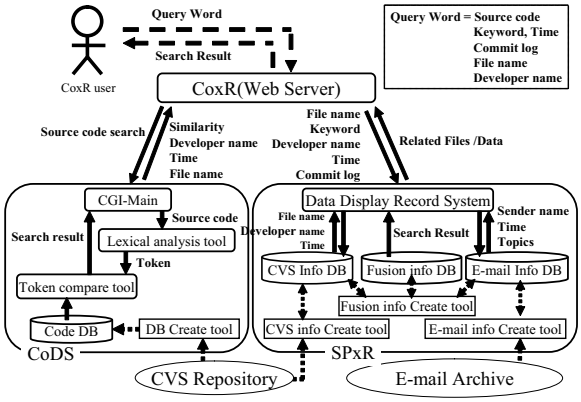

Figure 2. CoxR overview

3. CoxR Implementation

3.1. Design Policy

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

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. 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 implemented 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.

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 development. 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 similar development which same development and developer involved in.

4. Example

```
if (i != 0)
    error("Permission denied, please try again.");
password = read_passphrase(prompt, 0);
packet_start(SSH_CMSG_AUTH_PASSWORD);
packet_put_string(password, strlen(password));
memset(password, 0, strlen(password));
xfree(password);   ;; ;;
packet_send();
packet_write_wait();
```

Figure 3. Input Source Code

```
Make password attacks based on traffic
analysis harder by requiring that"non-
echoed" characters are still echoed
back in a null packet, as well as pad
passwords sent to not give hints to the
 length otherwise.
```

Figure 4. Commit Log

This section shows usage example of CoxR.

```
    error("Permission denied, please try again.");
    password = read_passphrase(prompt, 0);
    packet_start(SSH_CMSG_AUTH_PASSWORD);
    ssh_put_password(response);
*   memset(password, 0, strlen(password));
    xfree(password);    ;; ;;
    packet_send();
    packet_write_wait();
```

Figure 5. Similar source code

4.1. Using FreeBSD as an example data

We gather the development data from FreeBSD, and create pseudo software development environment. Actually, following FreeBSD's data is used:

- CVS repository
  Fully duplicate FreeBSD CVS repository (51379 files, total 511054 revisions).

- Email archive
  Duplicate email archives as of year 2001, that includes 49736 emails.

4.2. Searching Data

Assume that a developer maintains OpenSSH that is bundled with FreeBSD core system. At some time, a vulnerability is found that while password is sent via socket, password length is also sent; if attacker tried to sneak, it is easily unveiled the actual password length. The developer tried to fix the vulnerability with CoxR — the starting point is a code fragment that may have a vulnerability, shown as figure 3.

A developer uses CoxR to search similar code in repositories. Figure 6 is a search result. There are several candidates to look, but it seems that the file sshconnection.1 revision 1.8 (figure 5, and its commitlog is figure 4) is a good candidate for further investigation since its commit log denotes "pad passwords sent to not give hints to the length." and it is very similar to current problem. However, the deltas for this revision only suggests that yet another function should be used (`packet_put_string(response_stren(response)` to `ssh_put_password(response))` but no description about the new function `ssh_put_password()`.

So next will be a related information search. There are several approaches for that:

- by developers name (here, "green")
- by time (here "2001/03/20 02:06:40)
- by email related to the revision (here, revision 1.8)
- by keyword "openssh"

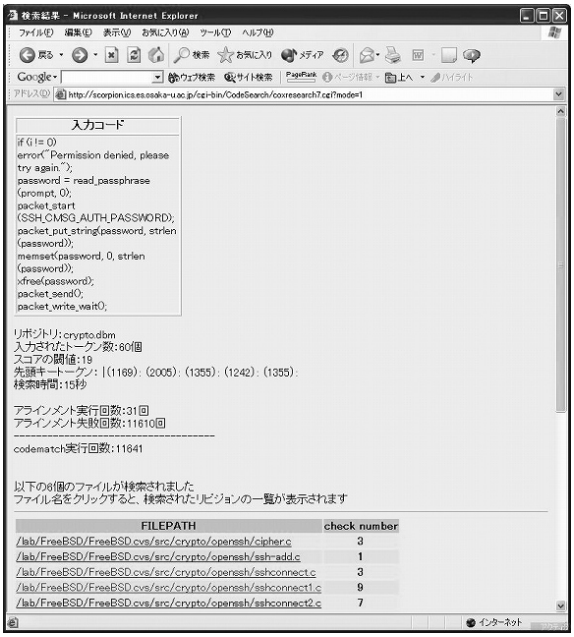

Figure 6. Search result

It would be easy to search all four search, just clicking a link associated with a name, time, revision, and put a keyword to the textbox. In this case, searching by time makes a good result; sshconnection.c is modified by the same time (see figure 9 and 10), and it has the function `ssh_put_password()`'s body (figure 8).

4.3. Discussion

In the example above, a developer is easily retrieve a key function to fix the vulnerability, starting with just a code fragment. Even though there are some try-and-error processes, it is simple and no worries what should search next.

However, there are some problems (maybe an open problem) in CoxR. For example, "keyword search" approach is highly depends on the contents of commitlog and email, it may not work iff S/N ratio is too low. Search results may have some noises (not the all information developer wants), so searching iteration process may lead developers to the wrong direction, and they will get no result. There are some rooms to improve keyword retrieval from the archive.

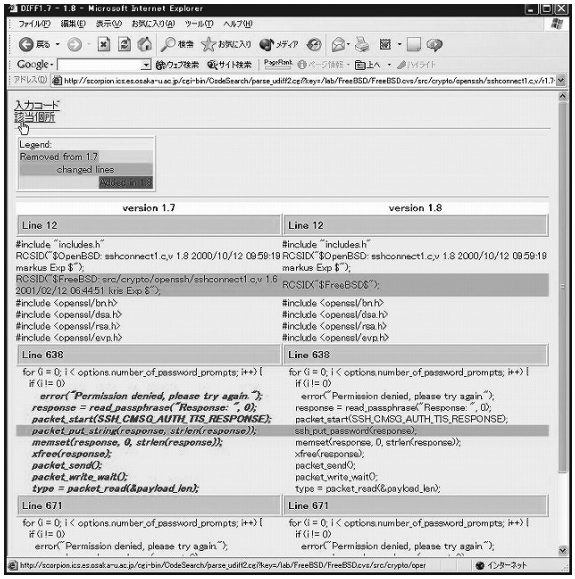

Figure 7. Code delta

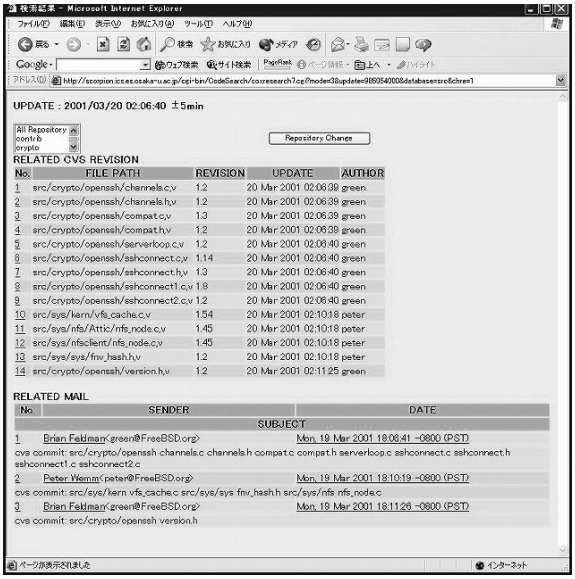

Figure 9. Search by date

```
void
  ssh_put_password(char *password)
  {
    int size;
    char *padded;

    size = roundup(strlen(password) + 1, 32);
    padded = xmalloc(size);
    memset(padded, 0, size);
    strlcpy(padded, password, size);
    packet_put_string(padded, size);
    memset(padded, 0, size);
    xfree(padded);
  }
```

Figure 8. Function ssh_put_password

5. Related Works

5.1. Repository Analysis

Mockus[8] analyse the aim of the change, and its time/size based on word appearance frequency of development log-file. Harald[5] extracts logical coupling among classes, files, and functions, and provide information to the users.

5.2. Development Community Analysis

Hipicat[2] is known as a community analysis system. In the Hipicat, “group memory” based on the relationship between CVS, BTS, and email information is extracted. Also they propose group memory to use future user’s works. However, this system presents pre-extracted relationships to users; different users must get the same results. Since searching something is maybe a creative work so it must have flexibility for users to choice yet another option.

6. 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] Agrawal, H., DeMillo, R.A., and Spafford, E.H.: “An execution backtracking approach to program debugging”, IEEE Software, pp.21–26(1991).

[2] Cubranic, D. and Murphy, G.C., “Hipikat: Recommending pertinent software development artifacts”, In Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), pp.408–419 (2003).

[3] Feiler, P.H., “Configuration Management Models in Commercial Environments”, CMU/SEI-91-TR-7 ESD-9-TR-7 (1991).

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

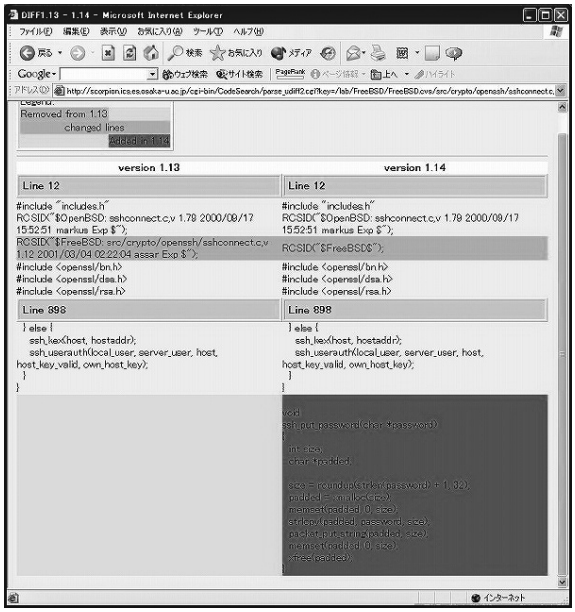

Figure 10. Search by ssh_put_password

[5] Gall, H., Jazayeri, M., and Krajewski, J.: "cvs release history data for detecting logical couplings", in International Workshop on Principles of Software Evolution (IWPSE 2003), pp.13–23, Helsinki, Finland (2003).

[6] Gusfield, D., "Algorithms on Strings, Trees, and Sequences", Cambridge University Press (1997).

[7] 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).

[8] Mockus, A. and Votta, L.G., "Identifying reasons for software changes using historic databases", In Proceedings of International Conference on Software Maintenance (ICSM 2000), pp.120–130 (2000).

[9] 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).

[10] Smith, T. and Waterman, M., "Identification of Common Molecular Subsequences", J.Molecular Biology, 147, pp.195–197 (1981).

[11] 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).