# Project Replayer with Email Analysis – Revealing Contexts in Software Development

Kimiharu Ohkura*, Keita Goto*, Noriko Hanakawa**, Shinji Kawaguchi*, and Hajimu Iida*
*Graduate School of Information Science, NAIST, Japan
{kimiha-o, keita-g, kawaguti}@is.naist.jp, iida@itc.naist.jp
**Faculty of Management Information, Hannan Univ., Japan
hanakawa@hannan-u.ac.jp

## Abstract

*In many software development projects, people tend to repeat same mistakes due to lack of shared knowledge from past experiences. Generally, it is very difficult to manually find out valuable phenomena from huge data. Invisible context, which cannot be known directly from software documents or formal reports, is an important factor to these difficulties.*

*We propose a new method to find contexts based on analysis to email archives in a project repository. In this method, we first apply natural language processing to extract keywords from email messages. Next, similarities among the messages are calculated based on the extracted keywords, and the messages are classified into clusters according to the similarities. The clustering result can be presented with other information such as code growth graph or schedule charts. This method is implemented as an extension to the Project Replayer, a tool to review past project data. Pilot analysis confirms that a researcher could grasp important contexts of failures in actual projects using the Project Replayer.*

## 1. Introduction

Recently, repeated failures of software development project become considerable problem in association with increasing demand for software development in society. One of the reasons for the failures is lack of shared knowledge from past experiences.

Many analysis methods [2, 12] to capture sharable knowledge for software development have been proposed. However, it is very difficult to manually find out valuable phenomena from huge data. Moreover, both making and validating hypotheses based on the observation are difficult. Invisible context, which cannot be known directly from software documents or formal reports, is an important factor to these difficulties. Existence of unreported troubles and informal discussions are examples for invisible contexts.

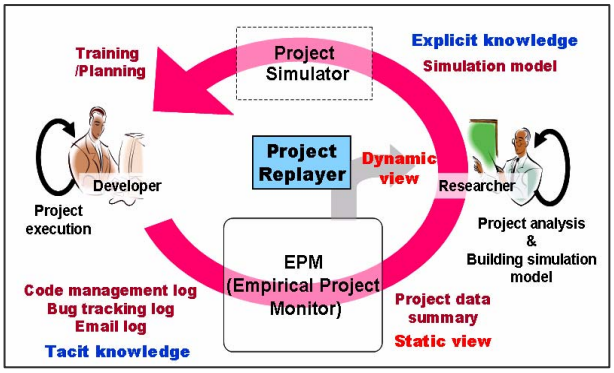


Fig.1 Knowledge Feedback Cycle

To reveal the contexts, we focus on email archives of software project. Email archives contain various conversation logs among developers such as when a meeting was held, who set up the meeting, and what the topic was. Although such information is very important source of development contexts, email archives often contain very large number of messages. Thus, summarization and categorization of emails are essential to present for a postmortem researches.

We propose a new method to find contexts based on analysis to email archives in project repositories. As summary of email archives, email clusters are generated based on the similarities among emails. This method is implemented as an extensional feature of the Project Replayer [5]. The Project Replayer is a tool to review past project data. It shows development progress of past project in dynamic way as if we replay the video records. By the extended feature, contexts are implied as generated message clusters and they can be presented with development progress information. They would imply the reasons of progress information changes.

This paper mainly describes the method of email analysis. Section 2 shows related work. In section 3, we show original features of the Project Replayer and Knowledge Feedback Cycle that is an underlying concept of the Replayer. In section 4, the methods for email analysis and clustering are described. In section 5, two pilot experiments to evaluate capability of the

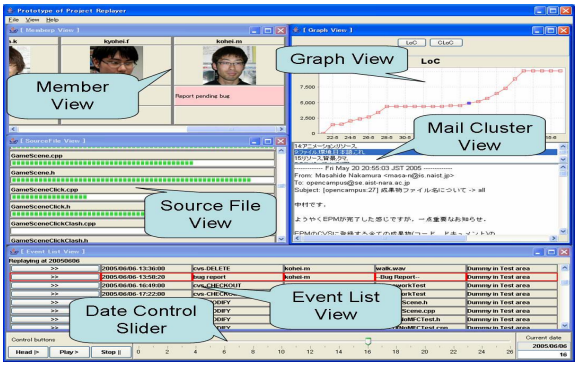


Fig.2 Project Replayer

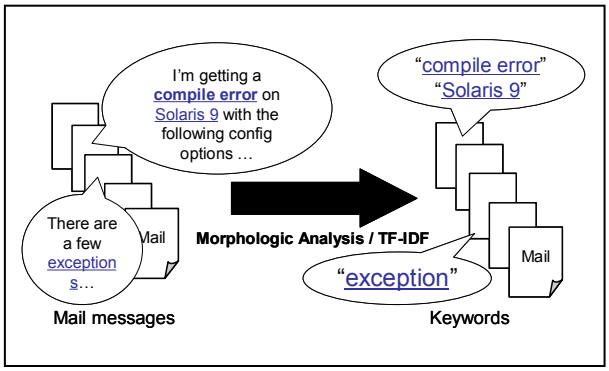


Fig.3 Extract terms

email analysis are shown. One is preliminary experiment to confirm availability and operability of the email analysis feature. The other one is an experimental application to an actual software project. We also discuss the result of these experiments. Finally, in section 6, conclusions are shown.

## 2. Related Work

Various researches related to email communication had been published in the field of software process. Multi-party deliberation structure model that Murakoshi et al. [7] proposed are given as an example of a method using natural language processing. The model can automatically construct conversational structure called deliberation trees based on linguistic clues. Several topics contained within email archives are visually represented by the deliberation trees. The model focuses facilitation of communications in cooperative work. Though this method enables deep analysis to small meeting logs, it can't be applied to conversations in large projects in long time.

There are some researches to retrieve useful information from software repositories: CVS repositories, email archives and bug tracking system. Hipikat [3] is implemented as Eclipse plug-in and automatically recommends entities related with what user editing. To determine what should be recommended, Hipikat analyze whole software repositories. CoxR [6] is web based software repository search system. CoxR supports keyword based searching and shows results with related entities. For instance, if a user searches CVS commit logs, results are presented with related emails and bug tracking entries and vice versa. They reveal various relationship among kinds of repositories however, they do not present whole picture of development process.

Bird et al. [1] and Wagstrom et al. [13] construct social networks among developers from several sources, including CVS commit logs and email archives. They use developer social networks to visualize relationship among developers or build simulation model of how developers joined and left projects. Relationship among developers is also important context. However, our current goal is presenting reasons of capital events.

Robles et al. [9] proposes identification method for several repositories. For example, it identifies each CVS user's email address. Such information may help understanding details of email messages.

## 3. The Project Replayer for Knowledge Feedback Cycle

### 3.1 Knowledge Feedback Cycle (KFC)

**3.1.1. What is KFC?** Developers are supposed to acquire new knowledge while experiencing software development projects. If such knowledge can be transferred to future projects at low-cost, it is quite valuable and helpful for the members. Authors proposed KFC (Knowledge Feedback Cycle) as a concept to circulate such knowledge from experience of past projects to future projects.

To establish such cycle, KFC employs three elemental tools; Empirical Project Monitor (EPM) [8] developed by EASE Project [4], the Project Replayer and the Project Simulator (See Fig. 1). KFC also involves two human roles – software developers and software engineering researchers. Developers utilize the KFC environment in order to acquire new knowledge from past projects while researchers utilize the KFC environment in order to construct simulation models which are embedded to the Project Simulator.

**3.1.2. Scenario.** A typical scenario in KFC would be as follows;

**Step1:** Various development data (records of code modification, bug tracking, and e-mails) is automatically captured by EPM during the project enactment (See "EPM" part of Fig. 1).

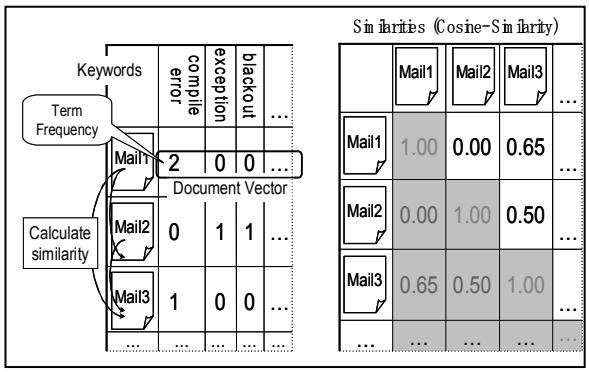


Fig.4 Vector Space Model

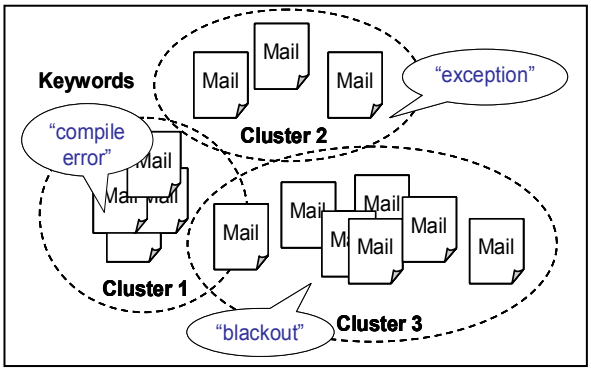


Fig.5 Clustering

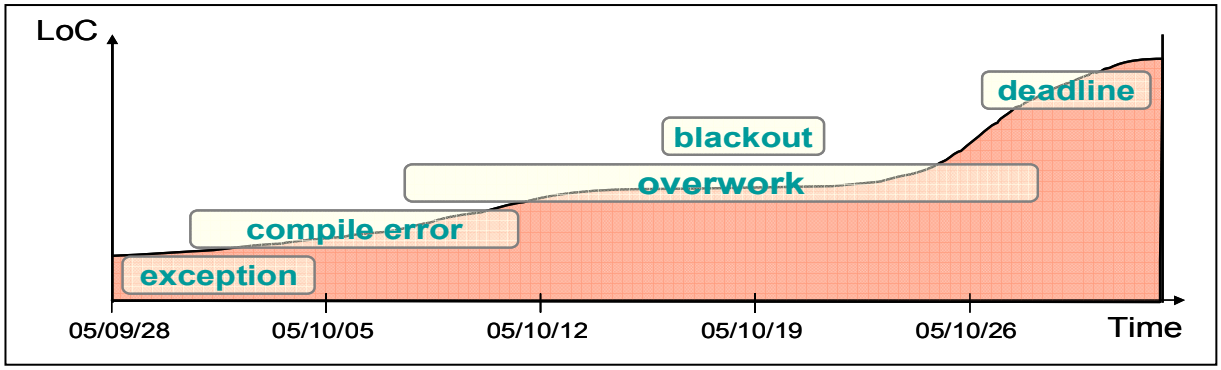


Fig.6 Example of overlapped display of mail clusters and progress chart (code size growth)

**Step2:** Researchers analyze collected data to construct various simulation models using the Project Replayer and analysis tools (See "Researcher" part of Fig. 1).

**Step3:** Using the Project Replayer, developers review past projects. Events and accidents that are not recorded by EPM are also clarified in interview with developers (See "Project Replayer" part of Fig. 1).

**Step4:** Regarding results of reviews and interviews, researchers refine their simulation models that were made in Step2. The models are embedded into the Project Simulator (See "Simulation Model" part of Fig. 1). The Project Simulator is a tool to simulate and predict actual software development projects.

**Step5:** Using the Project Simulator, novice developers learn complicated phenomena in past projects. Developers can also utilize the Project Simulator to make their next project plans. The planned project is regarded as the target of Step1 of the next cycle (See "Project Simulator" part of Fig. 1).

The whole mechanism of the KFC environment is currently under development in our group.

### 3.2. The Project Replayer

**3.2.1 Overview.** The Project Replayer is a tool to replay project data collected by EPM in order to help understanding behavior of past projects. The Project Replayer accelerates knowledge circulation by supporting both of two roles in KFC; developers can use the Project Replayer to revisit their past projects for postmortem evaluations, while researchers can use the Project Replayer to deeply understand and analyze dynamic behavior of the projects. Replaying past real project is also important for education and training because simulators sometimes provide practitioners with quite less reality that is derived from abstract and ideal models. The Project Replayer faithfully replays various behaviors of past projects. Developers are more familiar with the behaviors of past projects than behavior of the virtual projects.

**3.2.2. Features.** Original implementation of the Project Replayer has four views (*Event list view*, *File view, Graph view* and *Member view*) and a time-control bar.

*Event list view* shows various (CVS: Concurrent Versions System, bugs, and email) events collected through EPM are listed in order of time (See "Event list view" part of Fig. 2). The first column of each line works as a button to jump to the time of the event, which is indicated in the second column. The third column indicates the type of the event, the fourth column shows owner of the event, and the fifth column shows related filename.

*File view* presents source files in CVS repository (See "File view" part of Fig. 2). Each file item is shown with its name and progress bar. The progress bar shows rate of progress calculated as ratio of current CLoC (Cumulative modified Lines of Code) to the final CLoC. Graph view shows transitions of various data including total LoC (Lines of Code) and CLoC (See "Graph view" part of Fig. 2). Y-axis of the line chart indicates quantitative value such as LoC or CLoC, while x-axis indicates calendar time (days) of the project. The method of email analysis we proposed in this paper is integrated this view.

*Member view* lists project members with their role names and current actions (See "Member view" part of Fig. 2). The first row of a member item shows the member's name, the second row shows the member's portrait (or avatar), third row indicates current action s/he has performed, and fourth row shows active files that are currently being modified by the member.

The time control bar indicates the time (date) currently replaying. Moving the slider changes the time currently displayed. The bar also provides buttons such as start and pause.

# 4. Email Analysis

## 4.1. Overview

Email communication is often used to software development, and the email archives contain very important contexts for investigation of project. For example, the context means a series of relationships among various actions in order to solve a problem. However, the investigation from huge archives is very difficult and time-consuming. Though time-series graphs are frequently used to analyze the project, project context often appears very little on the graphs. By using Email Analysis based on natural language processing, the context in the software development project becomes clear. Our analysis consists following 4 steps.

**Step 1. Extract terms:** Terms are extracted from each mail using natural language processing.

**Step 2. Calculate the similarities:** Similarities among each mail are calculated.

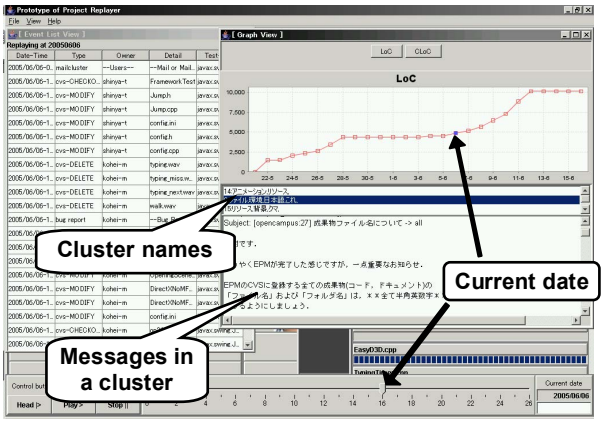


Fig.7 Execution screen of email analysis

**Step 3. Classify the emails:** The mails are classified into numbers of clusters using clustering algorithms.

**Step 4. Plot the clusters into time-series:** Each cluster is plotted on time-series, and is lapped over other time-series graphs. *Cluster chart* is integrated to the *graph view* of the Project Replayer (See Fig.7). Although current implementation is not smart enough to fully represent our concept, it can cover the minimum feature to reveal contexts.

## 4.2. Details of Analysis

**4.2.1. Extract terms.** At first, terms are extracted from each mail using a morphological analyzer (See Fig. 3). Then terms are weighted by means of TF-IDF (Term Frequency - Inverse Document Frequency) [10]. TF-IDF is basic means for term weighting. It is expressed following formula:

**Term weight** $w(t,d) = tf(t,d) \times idf(t)$

where

**TF:** $tf(t,d)$ = Frequency of term $t$ in document $d$

**IDF:** $idf(t) = \log \frac{N}{df(t)} + 1$

$N$: Total number of documents

$df(t)$: Number of documents including term $t$

Terms frequently appear such as "is", "was", and most terms of other than noun were low weighted and the each mail is characterized by terms high weighted according to TF-IDF. We call this term *keyword* for simplicity.

**4.2.2 Calculate the similarities.** Next, we calculate similarities among emails. To calculate the similarities, a keyword vector is made from the keywords of all

mails. The vector corresponds to the union set of high weighted terms in each mail. A *document vector* for each email is constructed by gathering frequencies of keywords appearing in the message (See Fig. 4).

Similarities among the document vectors are calculated by means of *Vector Space Model* [11]. Cosine-Similarity is used as a measure of similarity.

**4.2.3 Classify the emails.** The mail archives are classified by topics with Clustering algorithm (See Fig. 5). *K-Means* algorithm, featuring fast processing speed and decent precision, is used in the current implementation. Each cluster is named after the top 5 weighted keywords extracted the mails. The cluster name helps researcher to browse summary of topic in the cluster.

Although this phase costs most of the time, it needs to be performed only once because the clustering result is recorded to local files.

**4.2.4 Plot the mail clusters into time-series.** According to a date field of mail-headers, the clusters are plotted into time-series chart. By lapping over the mail clusters graph on other time-series data such as code growth graph, schedule chart, we expect various invisible contexts of the projects are revealed (See Fig. 6).

# 5. Experiment

## 5.1. Experiment Overview

We conducted two experiments to evaluate capability of the email analysis. One is preliminary experiment to confirm availability and operability of the email analysis, and the other one is experiment to apply actual software development projects. And we discuss the result of the experiments.

## 5.2. Preliminary Experiment

To examine availability of the renewed the Project Replayer added e-mail analysis, we conducted exploratory experiment. The aim of the experiments is to confirm that the feature of e-mail analysis picks out a context of software development project. The experiment was conducted following steps.

First, a researcher analyzes the project using old-version Project Replayer without e-mail analysis feature. Then, the researcher analyzes the same project using extended version of the Project Replayer with e-mail analysis feature, and confirms whether invisible contexts are picked out.

The target project is development of a typing-game program used in our open campus. The project was operated for 26 days by six developers. The developed program consists of 105 files and the final code size was 9,578 lines in total (See Table. 1 above). Though processing of the email clustering most takes a long time in this method, it was fast (31 msec). However, it is expected to cause the small archives.

## 5.3. Results and Discussions of Preliminary Experiment

Invisible contexts revealed in the preliminary

Table.1 Profile of the target projects

| (a) Open Campus Project | |
|---|---|
| **Size of mail archives** | 176KB (48 mails) |
| **Period of development** | 21 May, 2005 to 16 June, 2005 (26 days) |
| **Total LoC*** | 9,578 lines |
| **Total number of files*** | 105 files |
| **Processing time of mail clustering** | 31 msec. |

| (b) Corporate Project in a Japanese Industry | |
|---|---|
| **Size of mail archives** | 63.2MB (38,349 mails) |
| **Period of development** | May, Year-X ~ September, Year-X+1 (about 500 days) |
| **Total LoC*** | Non-disclosed |
| **Total number of files*** | Non-disclosed |
| **Processing time of mail clustering** | 2,699,987 msec. (45 minutes) |

* Total LoC and Total number of files are collected by EPM

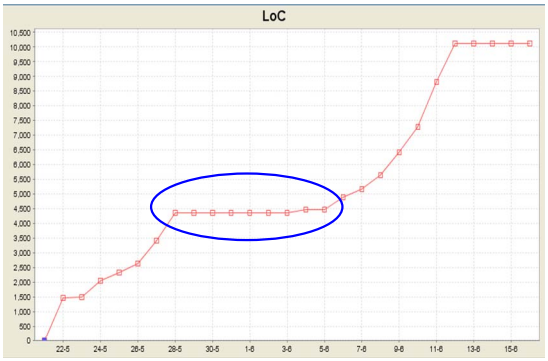

Fig.8 LoC graph of open campus project

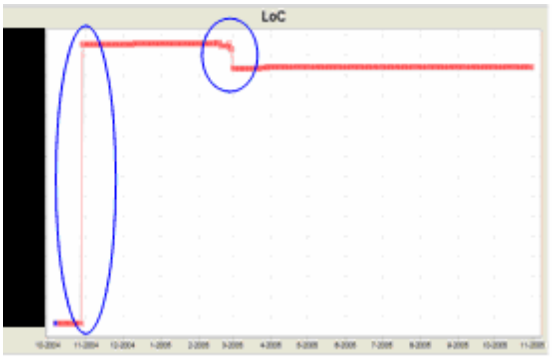

Fig.9 LoC graph of corporate project

experiment are following three points.

**1. Information about developer's meeting**

In this project, meeting about development was set up on 3 times. The meetings are all small scale. Therefore, records of the meetings did not exist.

**2. Period for development interruption**

The project had interrupted in a period between 27 May and 2 Jun. In the period (See Fig.8), CVS commit and bug report also had interrupted. This makes an impression of phenomenon of troubles for the researcher. However, a reason of the interrupt could not be revealed.

**3. Email positioning**

Email is mostly used for several notifications such as adjustment of schedule for meeting, announcement of code modification, and reporting for resource upload. The researcher could expect that most of detailed deals are made on actual meetings. And the expectation was correct.

Although the target project is very short span of time, we have revealed the several invisible contexts using our method. The result indicates that our approach is available and helpful for actual project.

## 5.4. Analysis to a Project in Industry

We conducted another analysis to huge data consisting of maintenance phases and operation phase of software development in industry. The target project is development of public online application system in Japan. The project was consisting of more than 10 developers and was operated for about 500 days. Though we can not disclose application area and actual size of product (number of files and final code size), it is a *real* project and is much larger than the first experiment project (See Table. 1 below).

Unlike in the case of preliminary experiment, the processing time of mail clustering is long (45 minutes). In this regard, we may have to refine the method in future.

Shortly after we load the project data to the Project Replayer, two prominent phenomena are found out by code growth graph. Details of these phenomena P1 and P2 are described below.

**P1: Sudden growth of LoC**

The most conspicuous feature of the LoC curve is a sudden increase of LoC on October 28, Year-X (See left circle of Fig.9). Usually, LoC increases little by little as software is developed.

**P2: Sudden decrease of LoC**

The conspicuous curve was also observed on February 28, Year X+1. It suddenly decreases (See right circle of Fig.9).

In order to reveal invisible contexts in the two phenomena, we investigated the project using the Project Replayer with email analysis.

## 5.5. Results and Discussions of Second Experiment

**5.5.1. Result and Discussion of P1.** Using only the original Replayer, we were not able to grasp what happened on October 28. Original Replayer just replayed addition of large amount of source codes, not present why LoC was increased suddenly.

By analyzing clusters of mail archives, we found out the reason of the sudden increase of LoC. The reason was that a new environment of CVS was built due to hardware crash of the previous CVS environment. The LoC data in the Replayer was replayed the CVS data of the second version environment of CVS. Therefore, on October 28, all sources were added to the new CVS environment from the previous CVS environment. In addition, although

mail archives were accumulated from April Year-X, CVS data was accumulated from October Year-X because the CVS environment was replaced in October Year-X. This is the reason why the LoC curve suddenly changes on October 28 in the Replayer.

Next, we explain how we found out the reason from the mail archives. At first, we picked up clusters including mail archives occurred on October 28. The number of the clusters is 51. However, we easily detected a key cluster using clusters' titles. The cluster title was "source, library, file, development span". The title was clearly different from the other clusters' titles. The other clusters' titles were about this project's themes such as "Registration target", "Online application", and "Registered information" (this theme is modified to avoid the target project identified). Although the key cluster included 26 mail archives, the number of mail archives that occurred from October 27 to October 29 in the cluster is only 5. After we checked the 5 mail archives, the important mail to grasp the contexts was detected. It was an announcement about the new CVS environment. In this mail, we did not understand why the new CVS environment was needed.

Therefore, we analyzed the former clusters than October 28. A cluster on October 27 included important mail archives about the replacement of the CVS environment. The cluster's title included just a term "problem". Using the cluster, we were able to trace a series of serious problems about the CVS environment. From the beginning, hard disk drive error occurred when sources were added to the CVS environment. Although members tried to recover the hard disk drive, they gave up the rescue of the hard disk drive. Next, the members tried to exchange only the hard disk drive, and they tried to rescue the version histories of sources. However, they decided to build a new CVS environment on a new computer in order to meet a deadline of next release of the software.

In this way, we were able to detect a significant problem in the project, and a reason of the problem through tracing a series of mail archives in a cluster. In addition, we found out another hard disk trouble of the CVS environment in January Year-X+1 when we were analyzing mail archives. The trouble was also detected from the same cluster including hard disk trouble in October Year-X. In short, we can say learning from past experiences is very difficult.

**5.5.2. Result and Discussion of P2.** In the same way, we investigated the phenomenon P2. As a result, we also found out the reason of sudden decrease of the LoC on February 28, Year-X+1.

First, we browsed clustering result, and confirmed that there are 46 clusters on February 28, Year-X+1. Most of the cluster names contain keywords: "Application", "Registration target", "Special target", "Environment" (this theme is modified to avoid the target project identified, too). These keywords had no relationship to the sudden decrease of the LoC. Thus, we only focused a few clusters without these keywords. Then, the reason of the sudden decrease of the LoC was found from one of the clusters without the keywords.

The phenomenon P2 had been caused by two reasons. One is change of implementation language. On 28 February, a part of the system was rebuilt in PHP from previous language. According to the change, many modules are removed in the CVS repository. The other one is clean-up of the repository. The change of system leaves many discarded files. In that day, one of the developers removed them with remaining garbage in past all together. From the results of above, we could grasp the invisible contexts behind the sudden decrease of the LoC. However, mining contexts depends on ability of the researcher, and relevance of the cluster name is hardly adequate.

## 6. Conclusion

We have proposed the method of email analysis to reveal contexts. The method is implemented to the Project Replayer. We explained details of our email analysis method. The method classifies email archives into clusters by means of TF-IDF, Vector Space Model, and clustering algorithm. Referring to date header in each mail, the clusters lap over on other time-series chart such as code growth graph. As a result, invisible contexts of software development project are revealed.

To confirm availability of our method, we conducted two experiments. One of the experiments is preliminary experiment. Target of the experiment is typing-game program developed by students for our open campus. As a result of the preliminary experiment, we confirmed a researcher could grasp several contexts such as information of developer's meeting, period for interrupt development, email positioning using the Project Replayer with email analysis.

The other one is an experiment to apply a corporate development project. Analyzing such project data is very difficult and time-consuming because of the data is huge. Using the Project Replayer with email analysis, we attempted investigation to the large scale project to reveal invisible contexts as with the preliminary experiment. Then, we immediately found out two prominent phenomena such as sudden increase and decrease curves on LoC graph. Also, we investigated the phenomena using the Project Replayer with email analysis, and could reveal several very important

contexts such as trouble of hard disk drive crash (and same trouble on another day), cleanup CVS repository associated system renewal, and change language for system development from the mail clusters. However, the part of revealing depends on ability of the researcher, and relevance of the cluster name was hardly adequate. Also, the processing of mail clustering takes long time in case of treat huge archives.

It follows from these arguments that our method is available to reveal invisible contexts. However, there is room for improvement in scalability and understandability of results. In these regards, we will constantly refine the method to enhance the processing time and precision. We also improve the tool for deep analyzing, example of analysis of relationship among the mail clusters in the future.

## Acknowledge

We would thank Dr. Makoto Sakai in EASE project/SRA-KTL/SRA for his valuable suggestions in interpreting industry data.

This research was partially supported by the Japan Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research (C) 17500024, and also by the EASE project in Comprehensive Development of e-Society Foundation Software program of the Japan Ministry of Education, Culture, Sports, Science and Technology.

## References

[1] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining Email Social Networks", In Proceedings of the International Workshop on Mining Software Repositories 2006, 2006.

[2] M. S. Conklin, J. Howison, and K. Crowston, "Collaboration using OSSmole: a repository of FLOSS data and analyses" In Proceedings of MSR2005, St Louis, Missouri, May, 2005.

[3] D. Cubranic, G. C. Murphy, J. Singer, and K. S. Booth, "Hipikat, "A project memory for software development" IEEE Transactions on Software Engineering, June 2005, 31(6):pp.446-465.

[4] EASE Project, http://www.empirical.jp/

[5] K. Goto, N. Hanakawa, and H. Iida, "Project Replayer An investigation tool to revisit processes of past project", In Proceedings of SPW/Prosim2006, LNCS, May 2006, pp.72 -79.

[6] M. Matsushita, K. Sasaki, and K. Inoue, "CoxR: Open Source Development History Search System", In Proceedings of 12th Asia-Pacific Software Engineering Conference, Taipe, Dec. 2005i, pp.821-826.

[7] H. Murakoshi, T. Yamami, A. Shimazu, and K. Ochimizu, "Construction of Multi-Party Deliberation Structure Using Mailing List in a Cooperative Work", 19th International Conference on Computer Processing of Oriental Languages (ICCPOL'2001), May 2001, pp.359- 364.

[8] M. Ohira, R. Yokomori, M. Sakai, K. Matsumoto, K. Inoue, M. Barker, and K. Torii, "Empirical Project Monitor: A System for Managing Software Development Projects in Real Time", Proceeding of ISESE2004, 2004, pp.37-38.

[9] G. Robles and J. M. Gonzalez-Barahona, "Developer identification methods for integrated data from various sources" SIGSOFT Softw. Eng. Notes 30, 4, July 2005, pp. 1-5,.

[10] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval", Information Processing and Management, Vol. 24(5), 1988.

[11] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing", Communications of the ACM, 18(11), 1975, pp.613-620.

[12] W. Scacchi, C. Jensen, J. Noll and M. Elliott, "Multi-modal modeling, analysis, and validation of open source software development processes", In Proceedings of the First International Conference on Open Source Systems, July 2005.

[13] P. A. Wagstrom, J. D. Herbsleb, and K. Carley, "A social network approach to free/open source software simulation" In Proceedings First International Conference on Open Source Systems, 2005, pp.16-23.