Email Communication Pattern in Global Software Development

Paulus Insap Santosa and Ridi Ferdiana, Electrical Engineering and Information Technology Department, Universitas Gadjah Mada, Yogyakarta Indonesia

The most challenging aspect in global software development is a communication between team members. This problem become more challenging since the team member is separated geographically. In this case, the team should have same vision and mission to achieve the goal of the project. The project goal can be achieved when the team has same cognitive level. Email is one of the most used indirect communications tool to construct same cognitive level. This paper will describe how cognitive perception can be achieved by providing additional information in an email. This research is developed based on several case study in global software development. The result of this research will propose partial solution for the GSD team to use an email as a better communication tool to deliver better software.

Software Engineering, Communication Effectiveness, Global Communication, Electronic Mail, Collaborative Work

I. Introduction

Global Software Development (GSD) defines as software development that uses teams from multiple geographic locations [5]. According to Sangwan [5], GSD is implemented because some of the following reasons:

1) Finding low cost human resources in others region which has lower gross income.
2) Collaborating with a person who has excellent talent but live in different region.
3) Expanding the local market through many people work in distributed are that the software is targeted.
4) Expanding the software support for local user through creating local resources that also build the system.
5) Decreasing the investment cost by hiring the person who has low gross income.

These reasons are well established because some of motivations to do so like [1], which are:

1) Limited trained workforce in technologies that are required to build today's complex system.
2) Differences in development costs favoring dispersing team geographically
3) A-shift based work system facilitated by time zone differentials allowing for shorter times to market
4) Advanced in infrastructure like internet connection capable the team works remotely with sufficient facility.
5) A desire to be "close" to foreign and local market

Although every company has different reason and motivation to implement GSD, one characteristic that remain constant is the difficulty of coordination. The difficulty of global software development is in overall lifecycle, started from its initial requirements until deployment execution. In collocated software development, the osmosis model is great to implement. Architect tells the designer or developer about the current architecture by using direct communication, during the direct communication between peers, intuitive and idea to build the system is lives collectively in the head of the teams rather than in paper. This approach is somewhat impossible in global software development since the member of the team is separated geographically. GSD team will face coordination challenge, less responsive, different culture and languages.

Software development collaboration is limited by phone, instant messaging, or email. According from research by Jackson [2], 56 percent business users using email as communication model which more used than phone or direct communication. However, some glitches also come which are:

1) Handling of email inefficiently because less of email education.
2) Overwhelmed because of the volume.
3) Forgetting or losing important items.
4) Feeling pressure to responds quickly and answers incorrectly in language manner.

Despite of the challenge of an email, an email still becomes the most used communication nowadays, including in software development. The idea of this paper is to enhance email experience in global software development by knowing the effective pattern in the term communication. Finding the email communication pattern in GSD will guide the team member to use email effectively in their daily activities.

II. Related Work

This paper is inspired by previous research that is escalating the email communication in several areas. Lee is working to improve the email communication in business environment through a symbol called avatar. Avatar provides an emotional enhancement in email and argues as cost-effective business communication improvement [3]. Vollmer [6] research makes an effort to improve the quality of email communication through a rating model, which is developed through a concept called reflection. Reflection concept is focused in feedback and analyze model in specific email communication need. Others related work in improve email communication is done

by Murshed, et al [4]. Murshed deeply explore the pattern about disintegration in organization by using email communication analysis. The result of the research provides some communication patterns to improve cohesive organization which might be disintegration or distributed.

Based on previous works we see that still no specific enhancement email communication in global software development (GSD). Furthermore, the employing of an email as secondary communication in software project leads us to investigate better patterns to improve the email communication in global software development.

III. Communication Problem in GSD

When two or more people are separated geographically then the interaction between them is limited. Global Software Development provides a better concept in adopting software development in geographical context. GSD concept tells us about the process, the artifact, and the project management. However, its shortage of communication plans. The lack of communication problem is unavoidable since every team in every world has their own way. In developing country, the lack of good internet connection makes the mobile phone, or SMS is preferable as a main communication tool. On the other side, a country that has enough broadband internets prefers communication tool like a video conferencing, VOIP, or even unified communication platform.

The divergence of communication tools that used in certain country will ascend a problem in communication collaboration. For example, when developed country that has sufficient bandwidth collaborate with developing country who has very limited bandwidth. The mainstream communication will be limited and the most used communication in both levels will be an email. The usage of the email is come from several reasons such as:

1) Email is cheaper than phone call, video conference, or VOIP
2) Email is richer than SMS, MMS, or Fax.
3) Email is low cost in infrastructure mind. In example user can get email as free services.

However, the problem still exists since indirect communication still fiddling some of problem such as:

1) Email is slow than communication through phone or face to face.
2) Email is need cognitive level of perception to make the receiver has same knowledge with the sender.
3) Email is threaded in conversation but separated in view. Therefore, a lot of user lost the information in email because of lot email information for one topic is separated in many emails.

These problems are more challenging when global software development is executed. In a global software development project, indirect communication takes a lot of part. For example, team member can send email to the others, and wait the peers to answer the email. Indirect communication in GSD gives an impact such as follow:

1) Increasing software development lifecycle. For example, member is sending an email to the peer and waiting the peer to response the email. This problem is more escalated when members in different time zone. Information update is slow and sometime several team members are losing information update.
2) Different culture in GSD make cognitive level in email is decreased dramatically. For example people from one region has different culture in communication, so there is opportunity when both peer doesn't know the right meaning of the email.
3) Inefficiency information archiving. Communication in GSD is multimodal; each member can communicate through phone, SMS, email, and chat. The problem for all of them is unmanaged information is separated

Our hypothesis to handle those problems is by creating a discipline and a model that can be guidance for GSD team. We investigate the discovering mechanism through three case studies. We focused the case studies to discover the model and discipline that related directly with email usage.

IV. Email Usage in Global Software Development

We capture three different models in Global Software Development. It is done in order to precise the email usage pattern. The three case studies namely Alpha, Beta, and Charlie. Alpha project is a case student when the team and client is separated geographically but still in one country. For example, the client is in Detroit and the development team is in New York. Beta project is a case study when the team is separated in some tier and the client. For example, the client team is separated in two regions, and client is separated, however, some of the team member exists collocated with the client. Charlie project is a case when the client and the team are separated in over the world, in different time zone.

When selecting those three cases, we chose projects which have similarity in resources, budget, and time. Those three projects have same resources that are consist of six peoples, have budget equal in $5000, and have a development time in 2 months. As mentioned, the difference between three cases is the geographical location that is the main point of GSD process.

Direct communication is the best way to achieve same cognitive level. Direct communication is also the effective way to declare a product vision, project planning, and agreement contract between team and the client. A process framework is used in Global Software Development. Sangwan [5] categorizes Global Software Development Framework in four main processes that are:

1) Requirement Engineering. This process has two main inputs which are product vision and business goal. This process provides an output as requirement specification.
2) Architecture Design. This process use requirement specification as an input as well as legacy system in order to create model and architecture specification. This process provides and output as architecture and module

specification.

3) Project planning. This process uses product vision, requirement engineering and architecture design as the input in order to create scalable project plan as an output.
4) Software Development. This process use project plan, requirement specification, and architecture specification. This longest process provides an output called as software product.

Based on those steps, this paper captures the email usage in each step of process framework. We compel the result to create a generalization model as communication pattern in email usage. The pattern will be adopted as proposed knowledge for people who want use Global Software Development.

Global Software Development is a process that is the implementation depends on software methods. In this research, Alpha project is using RUP (Rational Unified Process) method, Beta project is using ICONIX method, and Charlie project is using Extreme Programming method. Every method has its own artifact, and the artifact itself related directly with the step of software method. For example, RUP is the most detail method which has many artifacts such as UML diagram, Requirement Engineering document, etc.

The number of artifact will reflect the number of email which is might be sent through team member. For example, since Alpha project has more artifact than Extreme Programming or ICONIX. Table I provides the artifacts for each case study.

TABLE I ARTIFACT LIST FOR EACH CASE STUDY

Case Study	SE Method	Artifact
Alpha	RUP	• Requirement Artifact Set • Analysis and Design Set • Implementation Artifact • Test Artifact • Deployment Artifact • Configuration Artifact • Project Management Artifact • Environment Artifact • Business Modeling
Beta	ICONIX	• Requirement Analysis • Preliminary Design • Detail Design • Implementation Artifact
Charlie	XP	• Story Card (User Story) • Task Card (CRC) • Implementation (The Bullpen)

In our case studies, we capture 324 email communications for Alpha project, 188 emails for Beta project, and 296 emails for Charlie project. These results somewhat anomalies, since in normal situation Charlie project should be less than the beta project, but is not. We strongly consider that these anomalies happen because of geo-graphical and team structure.

In order to proof our consideration and find the email communication pattern in GSD, we capture the detail information flow in their case studies. We capture three main interesting points which are:

1) Email distribution.
2) Information architecture in email.
3) Cognitive preference in email.

Email distribution shows the demographics email communication in global software development. We discovered that email distribution is depend on two main factor which are the method they are used and technical experience in team member. We map the email distribution in four conventional processes step in software engineering that is requirement, analysis-design, development, and deployment. Figure 1 shows email distribution for our three case studies.

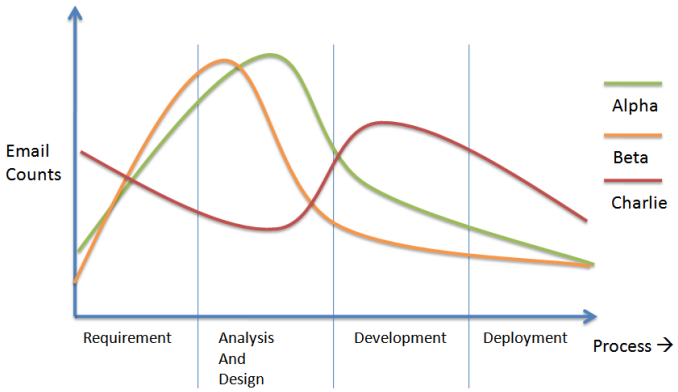

Fig. 1. Email Distribution Chart

By looking the result, we discover that email distribution is majorly happen in analysis and design model for every project except in Charlie project. Charlie project uses XP method that has lot communication activities like pair programming, testing discussions, and small release. The identification of the email distribution is come from the content and the subject of the email. In a common sense requirement document is sent in Requirement phase, and binary code is sent in development phase. This result influences us to create an implicit labeling for subject or an attachment in every email communication.

Email distribution labeling gives us a way to identify a phase for the project. Although, in some circumstances there is an overlapping possibility, it still can be identified by the email information architecture. Email information architecture contains four elements that are:

1) Recipient. It provides information about the non-technical or technical audiences. For example, manager usually receives email which has project management content, and technical worker receive email which has technical content. By identifying the recipient we can easily define first filter of the email whether technical or management email.
2) Subject. It identifies precisely the meaning of the email. For example, an email subject with Requirement development should have related information about requirement phases. Although we found that some email has unrelated subject like empty subject. A lot of email has common subject about the artifact that discussed in the email like “architecture model”, or specific problem subject like “deprecated API”.
3) Email body. It is identifies two things, the first one is as additional message accompanied the attachment and the

rest provides a discussion message. Both message type is randomly exist in every phases of software development

4) Attachment. The real artifact can be classified through the content of the file or its name. Attachment in every email can describe versioning of the artifact.

Through the email information architecture, we can identify an email and its content as a part of information flow in GSD. The email identification process is depending on the cognitive level of the team. According to the cases, we see the cognitive level of a person is depending on several factors that are:

1) Experience and creativity, we see that certain people who has experienced collaborate with the same team can learn the meaning of an email by seeing sequentially the subject, the sender, and its attachment. This cognitive perception can guide a people to obey the email or vice versa
2) Language, we see that a same origin language in software engineering communication provide effective cognitive process.
3) Attention, we see that in certain time, team member have a selective or divided attention when receiving update through email. Selective attention exists when the team member in sudden death situation like a deadline. Divided attention happens when team member in day-by-day situation.

According to the case studies, we investigate that these factors can be influenced with several things like collective intelligence, visualization infrastructure, and communication symbolic. Collective intelligence provides a good way to integrate each of member experience to share the knowledge in their project. Visualization infrastructure has a means to empower the team through communication technology like desktop sharing, video conference, or groupware. Communication symbolic is another uniqueness of the form communication. Some of the communication has unique symbol to enhance the cognitive level. For example, a manager in Alpha has a unique way to labeling the email by flagging it into high importance email and specific subject like "Important". Another story is in Charlie project which has a different way to give a symbol for an email communication, they use a label for each subject to identify the severity of an email like 'call to action', 'for your attention', and 'for your information'. Table II provides a list of action that every team do which is equal in as the influenced factor that showed.

TABLE II. COGNITIVE INFLUENCED FACTOR IN GLOBAL SOFTWARE DEVELOPMENT CASE STUDIES

Influenced Factor	Alpha Case	Beta Case	Charlie Case
Collective Intelligence	N.A.	Book sharing and discussion	Web reference, and discussion
Visualization	Image screenshot	Screenshot, instant Messaging, demo site	Screenshot, LiveMeeting, demo site
Communication Symbolic	Severity flag email	Email labeling	Email Labeling

Table II show a unique value in influenced factor of global software development. We see an action like discussion email, image screenshot, and email labeling is done by the most company in our case studies.

V. EMAIL INFORMATION PATTERN IN GSD

Pattern in software engineering is defined as a general reusable solution to a commonly occurring problem in software engineering. Three case studies in this paper indirectly tell some of the pattern that happens in global software development which is related with the email communication.

Starting with believes from the case studies, we do analysis for commonality and variability in email communication. Commonality provides information that is.

1) The similarity of email flow based on software method
2) The similarity of email labeling technique

A. Email Flow Based Software Method Pattern

We see that every email flow is depending on software engineering method that is applied to the engineering discipline that team used. For example, there will be no email with a subject "Re-factoring Case", if the team is not using Extreme Programming. The means that email communication is depend on the software engineering method. There are three kinds of email that is depending on software method which are.

1) The number of email.
2) The discussion in the email content.
3) The content of the email attachment

These three kinds of information are not only depending with the software method. For example, number email is depending also with number of the team member, technical difficulties, and team experience. Table III shows the email flow based pattern in context, problem, and solution.

TABLE III. EMAIL FLOW BASED SOFTWARE METHOD PATTERN

Pattern Key	Descriptions
Context	Email communication in Global Software Development
Problem	Email Information in Global software development is sporadic and not tide in the matter of flow
Solution	- Align the email communication with the software method lifecycle that they used in development - Integrate logically the document artifact with the email attachment in communication flow - Mapping the information architecture for email with the implemented of software method

B. Email Labeling Pattern

Email labeling is useful technique to give a metadata for an email. By adding the metadata through labeling, team can

easily identified to:

1) Respond the email in the matter of urgency by just seeing the email label
2) Rapid recognition about the information contain in the email.
3) Grouping the email in thread flow through labeling
4) Searching improvement through label.

Therefore, email labeling is the most recommended pattern that used to improve global software development environment. It provides an efficient way to improve communication in email. Table IV provides the email-labeling pattern.

TABLE IV. EMAIL LABELING PATTERN

Pattern Key	Descriptions
Context	Email communication in Global Software Development
Problem	Inefficient email processing, searching and responding in global software development
Solution	- Labeling the email subject with severity status, artifact flag, or software method lifecycle - Using unique subject labeling technique which can be understand by the team member - Aggregate the same subject in some way to provide better searching and threading

C. Putting it All Together

In order to implement the pattern, we are doing synthetic sample case. This sample case is developed in a project which has five people. One has a role as a client, and the rests are have a role as a team member. In this case, we are using Extreme Programming (XP) method. Based on the first pattern, we map the software engineering method to create email flow template. Email flow template is a flow template that can be used by the team to easily mapping the current communication with current step of software lifecycle. Based on XP lifecycle we identify the email flow template as a follows:

1) User Stories
2) Spike
3) Release Planning
4) Iteration
5) Acceptance Test
6) Small Release

Email flow template guides the team to create and discuss the project in an engineering manner. In example, when members send an email about iteration, a team can easily understand where the email belongs in software development phase.

The template can be a tag for the email subject. Member can send email by giving a subject like [User Stories]. This subject element can be joined with the name of project and additional tag. For example, we use a project called Whitten, therefore the subject email become "[Whitten] [User Stories] additional discussion". For better and clear tagging, we identify the tag by giving square bracket "[]". We are using three tags to identify the severity, project name, and project flow such as "[CTA] [Whitten] [User Stories] additional subject". Some severity common tags are:

1) CTA, call to Action
2) FYI, for your interest
3) FYA, for your attention

By adding the following severity tag, the user can more easy to identify the severity of the email. Those tags can be combined or filtered with a kind of system so user can see clearly the information architecture in global software development.

These hierarchies will help the user to search, categorize, and memorize the email in software engineering method manner. Rather than using manual search, the taxonomy model will help the user to identify the email in way that is more efficient.

VI. DISCUSSION AND FUTURE WORK

The focus of the research is capturing the email communication model in Global Software Development. Through the three case studies, we discovered two important patterns; the first one is aligning the email communication model with software engineering method and the second one is giving a label for each email to provide better information in the email.

Both ways provides sufficient model to enhance the communication way in global software development through email. However, this approach need further investigation and research such as:

1) The comparison email communication research between using the pattern and not using the pattern
2) The tag implementation in taxonomy model to provide sufficient categorization, and searching model.
3) The supporting tool that is developed to help user develop a tag based email communication.

REFERENCES

[1] Herbsleb, D.J, Paulish, J.D., and Bass, M. 2005. Global Software Development at Siemens: Experience from Nine Projects. ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA.

[2] Jackson, W.T., Burgess, A., and Edwards, J. 2006. A simple approach to improving email communication. Communication of ACM. Vol 46. USA.

[3] Lee, Y., Kozar, A.K., and Larsen, R.K. Does Avatar Email Improve Communication. Communication of ACM. Vol 45. USA.

[4] Murshed, H.S., and Hossain, L. 2007. Exploring Interaction Patterns of Cohesive subgroups during Organizational Disintegration.CHINZ 2007, July 2-4, 2007, Hamilton, New Zealand.

[5] Sangwan, R., Bass, M., Mullick, N., Paulish, J. D., and Kazmeier, J. 2008. Global Software Development Handbook. Aurbach Publications. Florida

[6] Vollmer, G., and Gabner, K. 2005. Quality Improvement of Email Communication in Work Groups and Organizations by Reflection. GROUP'05, November 6–9, 2005, Sanibel Island, Florida, USA.

Paulus Insap Santosa. Insap was born in Klaten, 8 January 1961. He obtained his undergraduate degree from Universitas Gadjah Mada in 1984, master degree from University of Colorado at Boulder in 1991, and doctorate degree from National University of Singapore in 2006. His research interest including Human Computer Interaction and Technology in Education.

Ridi Ferdiana. Ridi was born in 1983. He got his doctoral degree at Universitas Gadjah Mada in 2011. He earned his master degree from the same university in 2006. In his professional area, he holds several professional certifications such as MCP, MCTS, MCPD, MCITP and MCT. In his daily research activities he really enjoys to learn about software engineering, business platform collaboration, and programming optimization.