

# Analyzing Communication Patterns in Software Engineering Projects

**H. Keith Edwards ,Robert R. Puckett**

University of Hawaii at Hilo  
Department of Computer Science  
200 West Kawili Street  
Hilo, HI 96720, United States  
{hedwards , puckett}@hawaii.edu

**Art Jolly**

University of Hawaii at Hilo  
Department of Geology  
200 West Kawili Street  
Hilo, HI 96720, United States  
{ajolly}@hawaii.edu

**Abstract** - *Effective communication is a key to project success. This is particularly true in the area of software engineering projects where specialized team members must communicate highly technical material to one another in a consistent and timely fashion. In this paper, we analyze the communications of three student software development teams engaged in a year long project in order to understand patterns of communication, the type of information communicated, and how team role impacted communication within the group. We then looked at the impact of these factors on project success or failure, and how management can unobtrusively monitor communication patterns (rather than actual content) to look for warning signs in the software development process.*

**Keywords:** Communication, Software Engineering, case study.

## 1 Introduction

Effective communication is recognized as a critical component of project and career success [2,10]. The absence of consistent and valuable communication will doom any project to certain and abject failure. This is particularly true in the realm of software engineering, and the Capability Maturity Model [9] discusses several areas of the software engineering process (e.g. requirements elicitation and process development) where effective communication is required to progress through the model's five levels.

In this study, we look at the consistency and content of group communication as a predictor of project success or failure. In particular, we analyze the group communications from three year-long software engineering projects to determine what aspects of group communication plays the greatest role in project outcomes.

This research tracks the messages from the class project bulletin boards to examine the following factors and to correlate them with project success.

- Consistency of Messages over time
- Length of Messages
- Content of Messages (e.g. Files)
- Who communicated and their Role

## 2 Related Work

There is a large body of related work in the area of team communication in a project environment. In particular, much of the work seems to be devoted to soft factors in the software development process as evidenced by [6], which presents a synopsis of a workshop held on human and social factors in software engineering. The workshop focused a large amount of attention on communications and provides a good initial categorization of the work in this area. In particular, the paper partitions the research into social factors in process improvement, empirical and qualitative approaches, and tool based communications. The following subsections examine a similar set of domains as they relate to our work.

### 2.1 Communication Methods and Processes

Grace Tai [12] discusses a communication architecture developed at the Siemens Corporate Research center for a rapid prototyping environment. The paper outlines the typical higher level roles, artifacts, and modes of communication. The communication architecture divided the people into knowledge domains representing the software engineers, client, user interface designers, and usability specialists. From each domain a leader provided an interface to the other domains. Communications artifacts, such as storyboards or prototypes, fall between knowledge domains and are noted as facilitating communication and valuable feedback between different roles. Tai admits the need for an adaptive communication medium and attests that their communication architecture aided their group in meeting stringent deadlines, and satisfying customers. However, the paper does not discuss a

method for gauging if the communications are contributing to the success of the project. While communication architectures could set the ground for a successful communication strategy, this method may not be wholly applicable to smaller programming environments. The architecture also relies upon the need for identifying strong communicators for key roles as well as maintaining smooth group communication.

In his paper [7], Horberg explains how technical communicators which gain experience in software engineering can provide a valuable service. Such communicators may shorten development time and allow software engineers to perform their jobs more effectively. He explains areas where technical communicators can improve their skills to make them more valuable to software engineering and where they could be most effectively incorporated into such projects. Clients can exhibit difficulty in communicating specifications and are unaware of constraints on the software developers. On the other end, software engineers must contend with evolving specifications and resolve misinterpretations. As such, effective communication is essential for maintaining a smooth development process.

However, Horberg cites evidence that software engineers may not be the best at such communication as they prefer to obtain quantitative system requirements. Additionally, information collected from engineers may not necessarily be quality information. Surveys of engineers and clients demonstrated divides of satisfaction with software quality, cost, and project time. Horberg characterizes professional communicators as being “staunch user advocates” which could ensure engineers design from a user-centered perspective. Professional communicators are portrayed as playing a role as a mediator between clients and engineers as well as a user advocate during product design. However, professional communicators provide an additional overhead to a project and may not be appropriate for small design teams. As such, agile methods may not benefit as much from incorporating a professional communicator if the person does not add sufficient design and software engineering skills.

## **2.2 Tools for Analyzing and Assisting Communication**

Polack-Wahl provides an overview of the Blackboard CourseInfo as a tool in aiding collaboration for computer science student group projects [8]. Prior to using Blackboard, students experienced problems scheduling meetings, bottlenecks and setbacks in information dissemination

as well as version control issues. Blackboard augmented existing communication methods and provided a centralized communication medium which included file drop box, discussion board, and chat interface. She polled students to determine the perceived frequency of use and effectiveness of communication of courses without Blackboard, courses with Blackboard as well as the popularity of individual communication tools or mediums. Results showed increased perceived frequency of use and effectiveness of communication with Blackboard as well as preference for the digital drop box and discussion board. Similarly the University of Hawaii at Hilo computer science department employs WebCT which provides similar capabilities to Blackboard. However, although Polack-Wahl provides an overview to the benefits accrued from using a communication tool, she does not correlate the use of the tool to any metrics for success or failures of projects.

Another approach to assisting communication is to examine the theoretical underpinnings of small group communication such as done in [11]. Herein, Sutcliffe creates a modeling approach for small groups based on complex adaptive theory and discusses generic requirements for incorporating social interaction into groupware systems. This research takes a slightly different approach in that we are primarily interested in an empirical approach to understanding communication paradigms.

Finally, Gaffar, et al. [3] examine discuss how HCI patterns can be represented in XML to effectively support their dissemination and assimilation in a programmable environment. While, developing a theoretical model for the communication of HCI aspects and standardizing this portion of the software development process is certainly needed, our work takes a more holistic view of the process in order to understanding broader issues relating to the development of quality software.

## **2.3 Case Studies on Communication in Software Engineering**

D. Woit and K. Bell utilized surveys of students with varying experience in a non-face-to-face (NFtF) distributed learning environment for software engineering [14]. The survey of their case study addressed perceptions of respect, frustration, and connotations of silence. Students were asked to respond from both the point of view of face-to-face (FtF) communications as well as NFtF. From the analysis of the survey they performed a correlation analysis between the facets of communication in the study. The study found that the lack of non-verbal

cues, presence of silence in discussions, and perceived disrespect contributed to greater frustration with NTfF environments. This study proposed constructive and yet seemingly common sense suggestions for improving such communication mediums. However, this study investigates communication from a social perspective and relies highly upon surveys of participants. They do not address mixed FtF and NTfF environments. Additionally, they do not correlate the quantity or quality of communications to any metrics of success for the software engineering projects.

Gibson and Kelly propose a model by which software engineering concepts provide a framework for understanding problem-solving and the learning process [4]. In their case studies they observed and compared the problem solving process of both young children and first-year computer science students. Elements of problem solving examined include problem visualization, refinement, and abstraction as well as reuse of solutions. Students at all levels demonstrated algorithmic understanding whereby they developed a repeatable solution and could identify steps to solve a given problem. A major difference between the age groups lay in their skill in communication. Young children compensated for small vocabularies through “combined speech, drawing, sounds, and gestures.” Although this paper provides some insightful analysis of the learning process, it does not correlate communication as a determining factor of developing solutions.

A further case study by Grisham and Perry [5] examines the social aspects between programmers and clients in an extreme programming environment. The authors examine the role that extreme programming played in several software development projects that took place at diverse organizations such as the United States Postal Service and Daimler-Chrysler. In particular, the paper seems to use the XP wiki’s to gain an understanding of conditions that lead to customer satisfaction with the extreme programming process. While similar to our work in the fact that it examines communication aspects of the project, the focus is squarely on customer and development team communication as opposed to communication that takes place solely within the development group. The paper also eschews a quantitative examination of the communication data.

### 3 Problem Description

The subject of software engineering is taught as a capstone course in the University of Hawaii at Hilo (UHH) Computer Science Department. For the 2004-2005 year projects, the UHH Computer Science Department approached the Center for the Study of

Active Volcanoes (CSAV) with a concept of providing useful software engineering solutions. CSAV’s mission goals include assisting the Hawaii Volcano Observatory (HVO) in its training, monitoring and outreach efforts in Hawaii, the Pacific Basin and the developing world. Towards these mission goals, CSAV identified three projects where UHH software expertise could be applied. The projects included: 1) a web-based teaching and testing module (team 2), 2) a web-based earthquake location teaching module (team 1) and, 3) an earthquake database web-based query module (team 3). Projects 1 and 2 were identified as areas that could improve CSAV’s training and outreach capabilities with volcano observatories located worldwide and in developing countries. Project 3 was identified as an area which could improve HVO’s ability to monitor seismicity associated with volcanic activity. The three projects were designed to be completed by the three teams over an academic year.

For Project 1, Team 2 was asked to develop an interface that would allow a CSAV instructor to develop course content online including text and images. The project also required Team 2 to develop a web-based testing utility which would allow the CSAV instructor to assess the progress of any CSAV student viewing the instructor’s content. The project required expandability of content and system security such that the instructor could monitor CSAV student progress.

For Project 2, Team 1 was asked to develop a web-based earthquake location tool which allowed a CSAV student the ability to measure earthquake arrival times and amplitudes from seismic waveforms, and then determine an earthquake location and magnitude. The project required that the team provide the ability for CSAV students to test alternate arrival times and Earth velocity models. CSAV student feedback from the instructor was an added requirement.

For Project 3, Team 3 was asked to develop a web-based query module to access the HVO earthquake database containing several thousand earthquakes. The project required rapid access to several database parameters (including but not limited to spatial and temporal earthquake parameters), the ability to accommodate future expansion of the database, and allow the user to save data output to a text file.

The instructional tool WebCT was utilized as part of the software engineering course. According to the University of Minnesota [13], WebCT is “Online management software that aids students in their

classes by creating, managing, organizing, and housing a Web-based learning environment. On this site, professors can post lecture notes and information, grades, past quizzes, and a chat area and bulletin board.”

In the software engineering class, students naturally gravitated toward the use of WebCT message boards, which were set up for each team. We used these message boards to perform a post-mortem analysis of the communication patterns for the various groups in order to determine whether the nature and amount of communication played a role in project success.

## 4 Analysis

Of the three teams, only team 3 experienced a large degree of success during the final phases of the project, both in terms of customer satisfaction and in terms of a working system. In this research, we sought to analyze the communication patterns of the teams to see whether any insight might be gained into the role that communication plays in project success or failure.

In order to analyze the communication patterns that took place within the groups, we first sorted all the messages from the bulletin boards by date. Since the default is to show the messages according to the thread, it can sometimes be difficult to determine when communication took place.

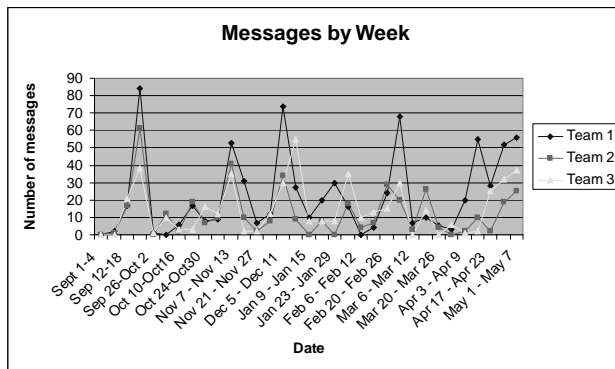


Figure 1 – Total Messages by Week (1<sup>st</sup> Semester)

Deliverable	Date
Project Proposals	September 23, 2004
Initial Prototype (Feasibility)	November 9, 2004
2nd Prototype & Formal Requirements	December 7, 2004
Formal Specifications / Test Plans	March 1, 2005
Implementation	May 2, 2005

Figure 2 – Project Deliverables and Dates

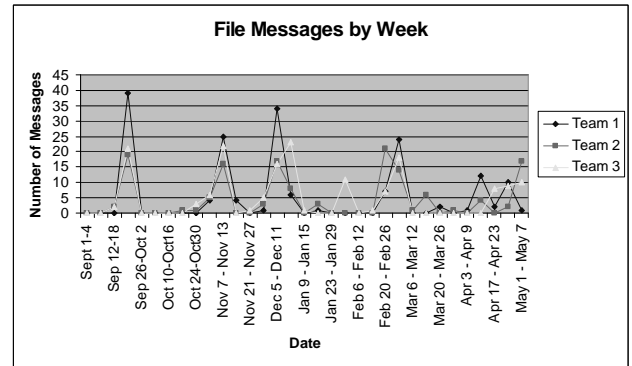


Figure 3 – File Messages by Week (Entire Project)

After sorting the messages by date, we counted the total number of messages for each group, and then counted the total number of messages posted by each group for each week of the term. Figure 1 shows the total number of messages by week for each group.

When examining the graph of total messages by week, one can readily discern a pattern of high peaks followed by low valleys. This can be attributed to the fact that most of the groups postponed working on the project until just before the deadlines (see Figure 2). If one assumes that the messages correspond to work taking place, then this results in a work practice that is precisely the opposite of the one recommend in the Rational Unified Process [1]. By monitoring just the number of communications that are taking place, management can detect this type of warning pattern without snooping through the contents of individual messages.

One item of note is that graph for team 3 is slightly offset from the other groups. This slight offset shows that the members of team 3 discussed the project and potential revisions immediately after the deliverable. Since team 3 experienced some moderate success with the project, communications that take place immediately after the deliverable may be a positive indication toward project success.

Figure 3 shows the total number of file attachments for each team over the course of the project. In this figure, we can see similar same peaks and valleys that characterize the graph in Figure 1.

Criteria	Team 1	Team 2	Team 3
Total Messages	754	409	484
Mean	24.32	13.19	15.61
Std. Dev	23.63	13.96	14.39
90% Alpha	6.77	4.00	4.12
Low 90% Conf	17.56	9.20	11.49
High 90% Conf	31.09	17.19	19.73

Figure 4 – Synopsis of Messages by week

Figure 4 shows data on the total number of messages by each group. Surprisingly, the team with the most number of messages did not experience the greatest success with the project. While team 1 had the largest number of messages, the client was somewhat dissatisfied with the results that they achieved. There is also a high standard deviation for each group, but even with this high standard deviation, it is clear that team 1 had more communication than team 2 at a 90% confidence level. We also note that team 2 and team 3 had a lower standard deviation, indicating somewhat more uniform communications.

Team Role	Fall Total	Spring Total
Project Leader	69	43
Assisting Programmer, Tester	34	36
Database Designer	13	17
Security Analyst	39	17
Web Specialist, Documentation	44	28
Database Designer, Tester, Writer	25	18
Database Design, Electronic Specialist	8	18
Team Role	Fall Total	Spring Total
Project Management	78	84
Software Engineer	60	83
Software Engineer	52	47
Software Engineer	65	61
Financial Officer	33	55
Sr. Software Engineer	19	24
Sr. Software Engineer	41	52
Team Role	Fall Total	Spring Total
Project Management	87	71
Technical	35	37
Writing	55	76
Technical	28	25
Technical	6	7
Non-technical	23	21
Database	6	6

Figure 5 – Messages by Team Role

Figure 5 shows the communications for each group as broken down by the role of the communicator. In each group, the project manager made the largest number of communications. One interesting difference between the groups is that team 3 had less specified roles and a greater number of people in support positions as opposed to technical positions. In particular, team 3 had the benefit of having a designated technical writer and a person to handle non-technical matters, which most likely assisted them in their project communications.

Upon further analysis, we found that team 3 also had more emails from their non-technical people than from their technical people. For team 3, the ratio of non-technical to technical communication was about 2.2 to 1 whereas teams 1 and 2 had ratios of 0.61 and 0.55 respectively. Hence, one can hypothesize that

non-technical communication is one of the keys to project success.

## 5 Conclusions

The most distressing aspect to this data is that our results showed precisely the inverse of the suggested amount of work for each phase in the rational unified process. Increased team communication centered solely on deadlines and technical information can be ineffective in achieving quality software products. However, management can use this fact to monitor communication for peaks and valleys without being overly intrusive as to the nature of the messages.

Secondly, the teams that performed slightly better on the project spent time discussing the project after each deliverable and had moderately more consistent communications throughout the course of the project.

Finally, the team role had an impact on the amount and type of communication within the group. In particular, the team that had less defined roles and more roles specified for communication and for non-technical matters had greater success. The number of non-technical communications also seemed to have an impact on the quality of the project.

## 6 Future Work

More frequent deliverables in a rapid prototyping environment may have a leveling effect on the nature of team communication. In particular, subsequent efforts should have an increased number of deliverables in order to determine if this results in a more uniform distribution of communications and a corresponding increase in software quality.

## 7. References

- [1] Booch, G, Rumbaugh, J, Jacobson, I. The Unified Modeling Language User Guide: The ultimate tutorial to the UML from the original designers. Addison-Wesley Publishing. Menlo Park, California. ISBN 0-201-57168-4. [1997].
- [2] Currie Little, J., Granger, M. J., Boyle, R., Gerhardt-Powals, J., Impagliazzo, J., Janik, C., Kubilus, N. J., Lippert, S. K., McCracken, W. M., Paliwoda, G., and Soja, P. 1999. Integrating Professionalism and Workplace: Issues into the Computing and Information Technology Curriculum: Report of the ITiCSE'99, Working Group on Professionalism. In *Working Group Reports From ITiCSE on innovation and Technology in Computer Science*

*Education* (Cracow, Poland, June 27 - 30, 1999). ITiCSE-WGR '99. ACM Press, New York, NY, 106-120.

[3] Gaffar, A., Seffah, A., and Van der Poll, J. A. 2005. HCI pattern semantics in XML: a pragmatic approach. In *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering* (St. Louis, Missouri, May 16 - 16, 2005). HSSE '05. ACM Press, New York, NY, 1-7.

[4] Gibson, J. P. and O'Kelly, J. 2005. Software engineering as a model of understanding for learning and problem solving. In *Proceedings of the 2005 international Workshop on Computing Education Research* (Seattle, WA, USA, October 01 - 02, 2005). ICER '05. ACM Press, New York, NY, 87-97.

[5] Grisham, P. S. and Perry, D. E. 2005. Customer relationships and Extreme Programming. In *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering* (St. Louis, Missouri, May 16 - 16, 2005). HSSE '05. ACM Press, New York, NY, 1-6. DOI=<http://doi.acm.org/10.1145/1083106.1083113>

[6] John, M., Maurer, F., and Tessem, B. 2005. Human and social factors of software engineering: workshop summary. *SIGSOFT Softw. Eng. Notes* 30, 4 (Jul. 2005), 1-6.

[7] Horberg, J. K. 1994. A future for professional communicators in software engineering. In *Proceedings of the 12th Annual international Conference on Systems Documentation: Technical Communications At the Great Divide* (Banff, Alberta, Canada, October 02 - 05, 1994). SIGDOC '94. ACM Press, New York, NY, 76-87.

[8] Polack-Wahl, J. A. 2001. Enhancing group projects in software engineering. In *Proceedings of the Sixth Annual CCSC Northeastern Conference on the Journal of Computing in Small Colleges* (Middlebury, Vermont, United States). Consortium for Computing Sciences in Colleges. Consortium for Computing Sciences in Colleges, 111-121.

[9] CMMI Product Team. Capability Maturity Model Integration Continuous Representation – Version 1.1 (March 2002). Software Engineering Institute. Pittsburgh, PA. [2002].

[10] Sommerville, Ian. Software Engineering: Seventh Edition. Addison-Wesley Publishing. Menlo Park, California. ISBN 0-321-21026-3. [2004].

[11] Sutcliffe, A. 2005. Applying small group theory to analysis and design of CSCW systems. In *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering* (St. Louis, Missouri, May 16 - 16, 2005). HSSE '05. ACM Press, New York, NY, 1-6.

[12] Tai, G. 2005. A communication architecture from rapid prototyping. In *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering* (St. Louis, Missouri, May 16 - 16, 2005). HSSE '05. ACM Press, New York, NY, 1-3.

[13] University of Minnesota. University of Minnesota Parent Website. <http://www.parent.umn.edu/> [2006].

[14] Woit, D. and Bell, K. 2005. Student communication challenges in distributed software engineering environments. In *Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education* (Caparica, Portugal, June 27 - 29, 2005). ITiCSE '05. ACM Press, New York, NY, 286-290.