International Journal of Computer and Communication Technology

Volume 6 | Issue 2 Article 5

April 2015

# THE AWARENESS NETWORK OF MONITORING AND DISPLAYING ACTIONS OF SOCIAL NETWORKS

B.DEENA DIVYA NAYOMI
*Sreenivasa Engg College,Kurnool*, deena.divya20@gmail.com

FAROOQ MOHAMMED
*Sreenivasa Engg College,Kurnool*, Mohd.farooq.a@gmail.com

V. SANDEEP
*KTMC College,Kurnool*, v.sandeep531@gmail.com

TAMKEEN FATIMA
*NTR Govt Degree College Women, Mahaboob NGR*, tamkeen.fatima.501@gmail.com

Follow this and additional works at: https://www.interscience.in/ijcct

Recommended Citation
NAYOMI, B.DEENA DIVYA; MOHAMMED, FAROOQ; SANDEEP, V.; and FATIMA, TAMKEEN (2015) "THE AWARENESS NETWORK OF MONITORING AND DISPLAYING ACTIONS OF SOCIAL NETWORKS," *International Journal of Computer and Communication Technology*: Vol. 6 : Iss. 2 , Article 5.
Available at: https://www.interscience.in/ijcct/vol6/iss2/5

This Article is brought to you for free and open access by Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

# THE AWARENESS NETWORK OF MONITORING AND DISPLAYING ACTIONS OF SOCIAL NETWORKS

**B.DEENA DIVYA NAYOMI[1], FAROOQ MOHAMMED[2], V.SANDEEP[3] , TAMKEEN FATIMA[4]**

[1&2] Sreenivasa Engg College,Kurnool
[3] KTMC College,Kurnool
[4] NTR Govt Degree College Women, Mahaboob NGR
Email:deena.divya20@gmail.com1, Email:Mohd.farooq.a@gmail.com2, v.sandeep 531@gmail.com3
, tamkeen.fatima.501@gmail.com

**Abstract :**The concept of awareness plays a pivotal role in research in Computer-Supported Cooperative Work. Recently, Software Engineering researchers interested in the collaborative nature of software development have explored the implications of this concept in the design of software development tools. A critical aspect of awareness is the associated coordinative work practices of displaying and monitoring actions. This aspect concerns how colleagues monitor one another's actions to understand how these actions impact their own work and how they display their actions in such a way that others can easily monitor them while doing their own work. we focus on an additional aspect of awareness: the identification of the *social actors* who should be monitored and the actors to whom their actions should be displayed. We address this aspect by presenting software developers' work practices based on ethnographic data from three different software development teams. In addition, we illustrate how these work practices are influenced by different factors, including the organizational setting, the age of the project, and the software architecture. We discuss how our results are relevant for both CSCW and Software Engineering researchers.

***Keywords*****:** *Computer-supported cooperative work, organizational management and coordination, programming environments, programming teams, tools.*

## 1. INTRODUCTION:

SOFTWARE development, being a human activity, is challenged by human limitations. There are individual cognitive challenges and social collaborative challenges. The collaborative challenges are what we are concerned with in this work, having observed teams of software developers working together to deliver their target products. Collaborative challenges were identified very early in the nascent field of software engineering. Brooks observed that software development was "a complex interpersonal exercise." The seminal work by Curtis et al. recognized that breakdowns in communication and coordination efforts constituted a major problem in large-scale software devel- opment. Later, Staudenmayer recognized that good coordination of teams of developers was correlated with high team performance. Finally, Herbsleb et al. documented how software development tasks performed in distributed contexts took longer than similar tasks performed in collocated ones due to the cost of coordinating developers in different geographical locations. Over the years, software engineering practitioners have proposed a large number of strategies to facilitate the collaboration required of software development efforts, including tools (e.g., CVS), approaches (e.g., software process models), and techniques (e.g., pair programming).Many of the problems faced by software developers are the same as problems faced by professionals in other domains: communication breakdowns, coordination problems, lack of knowledge about colleagues' activities, and so on.

## 2.RESEARCH SITES AND METHODS

To identify and understand the strategies used by software developers to handle software dependencies, we conducted two qualitative studies at different large software development organizations. The first field study was conducted during the summer of 2002, when the first author was an intern in the MVP team, and the second one was performed during the summer of2003. We adopted participant and non-participant observation and semi-structured interviews for data collection. Data analysis was conducted by using grounded theory techniques. Details about each team as well as the methods used are described below. For confidentiality reasons, the names of the teams and organizations are not their real names.

## 3.MVP

The first team studied develops a software application called MVP ,a nine-year old software composed of ten different tools in approximately one million lines of C and C++ code. Each one of these tools uses a specific set of "processes." A process for the MVP team is a program that runs with the appropriate run-time options and it is not formally related with the concept of processes in operating systems and/or distributed systems. Running a tool

means running the processes required by this tool with their appropriate run-time options.The software development team is divided into two groups: the verification and validation (V&V) staff and the developers. The developers are responsible for writing new code, for bug fixing, and adding new features. This group is composed of 25 members, three of whom are also researchers that write their own code to explore new ideas.

## 4.MCW

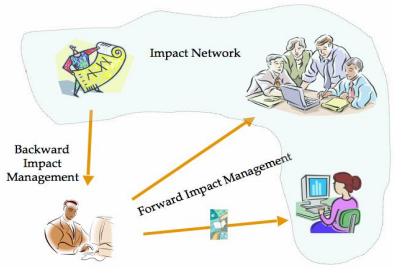
The second field study was conducted in a software development company named BSC. The project studied, called MCW, is responsible for developing a client-server application. The project staff includes 57 software engineers, user-interface designers, software architects, and managers, who are divided into five different teams, each one developing a different part of the application. The teams are designated as follows: lead, client, server, infrastructure, and test. The lead team is comprised of the project lead, development manager, user interface designers, and so on. The client team is developing the client side of the application, while the server team is developing the server aspects of the application. The infrastructure team is working in the shared components to be used by both the client and server teams. Finally, the test team is responsible for the quality assurance of the product, testing the software produced by the other teams.

## DATA ANALYSIS

After the second data collection, datasets from the two different teams were integrated into a software tool for qualitative data analysis, MaxQDA2. After that, the data collected was jointly analyzed by using grounded theory,. This technique does not require a prior theory about the data, that is, a set of hypothesis to be tested. Instead, the goal of grounded theory is precisely to generate theory grounded exclusively on the existing data. In other words, it aims to develop a theory or explanation about what is going on in the field, or more specifically, what is available in the data collected. Grounded theory proposes three major steps. The first step is called open coding, in which data (in this case, interviews and field notes) are micro-analyzed (line-by-line) to identify categories.

## IMPACT MANAGEMENT

One of the reasons why software development is difficult is the intricate web of dependencies among artifacts and software development activities. The work required to manage these dependencies can be seen as impact management. Impact management is defined as the work performed by software developers to minimize the impact of one's effort on others and, at the same time, the impact of others into one's own effort. Viewing dependency management as impact management draws attention to the developers' concern about being impacted by changes made by and impacting their colleagues during software development efforts.It illustrates how one orients himself toward his colleagues so that both can get their work done.




the distinction among backward and forward impact management is analytical: they are iterative, interwoven among themselves and among other developers' practices. Furthermore, these aspects are also complementary: whenever a developer is performing forward management, he is facilitating the backward management to be done by others, and vice versa. For instance, whenever a MVP developer decides to postpone his or her check- in's, he or she is allowing other developers not to worry about deciding whether to recompile their code.The focus of this is on the strategies of the MVP and MCW teams.

## CONCLUSIONS

In the field of Computer-Supported Cooperative Work, the term awareness is used to describe a range of work practices by which social actors coordinate their work through (i) the display of their actions to their colleagues, and (ii) the monitoring of actions from their colleagues. Recently, this concept has been explored by software engineering researchers in the design of collaborative software development tools. Most empirical studies related to awareness focus on the identification of these coordinative practices and assume settings in which the social actors who display and monitor actions do not change often. However, the practices of displaying and monitoring actions associated with awareness are useful only to the extent that social actors *know who they should monitor and to whom they should display their actions*. In collocated settings, this information is intrinsic. However, there are settings where this information is not as clear, e.g., distributed software projects. Previous studies have largely overlooked the identification of these actors.

## REFERENCES:


[1] C.R.B. de Souza and D. Redmiles, "On the Roles of APIs in the Coordination of Collaborative Software Development," J. Compu- ter Supported Cooperative Work, vol. 18, nos. 5/6, pp. 445-475, 2009.

[2] C.R.B. de Souza et al., "Guest Editors' Introduction: Cooperative and Human Aspects of Software Engineering," IEEE Software, vol. 26, no. 6, pp. 17-19, Nov./Dec. 2009.

[3] P. Dourish and V. Bellotti, "Awareness and Coordination in Shared Workspaces," Proc. Conf. Computer-Supported Cooperative Work, 1992.

[4] K. Ehrlich, "Locating Expertise: Design Issues for an Expertise Locator System," Sharing Expertise: Beyond Knowledge Management, M.S. Ackerman, V. Pipek, and V. Wulf, eds., MIT Press, 2002.

[5] K. Ehrlich and K. Chang, "Leveraging Expertise in Global Software Teams: Going Outside Boundaries," Proc. IEEE Int'l Conf. Global Software Eng., 2006.

[6] B.Deena Divya Nayomi "It Monitors and Displays the actions of systems and collegues of systems by Administration"Asst Prof in Sreenivasa Engg College,Kurnool,Aug2012.