Review Article

iMedPub Journals
www.imedpub.com

American Journal of Computer Science and Information Technology

2021
Vol.9 No.10:116

Collaborative Problem Solving and Groupware in the Software Development Domain

W. Ajayi[1], Lucky Samuel E[1*], Segun Sofoluwe A[1], Olomola Babatunde T[1]

Department of Computer and Information Science, Lead City University, Nigeria

***Corresponding author:** Samuel EL, Department of Computer and Information Science, Lead City University, Nigeria, E-mail: slucky@babcock.edu.ng

Received date: October 29, 2021; **Accepted date:** November 12, 2021; **Published date:** November 19, 2021

Citation: Ajayi W, Samuel EL, Sofoluwe AS, Babatunde TO (2021) Collaborative Problem Solving and Groupware in the Software Development Domain. Am J Compt Sci Inform Technol Vol.9 No.10: 116.

Abstract

Problems are at the core of much of what people accomplish at work on a daily basis. The difficulties you face can be vast or small, simple or complex, easy or tough, whether you're addressing a problem for an internal or external client, helping people who are solving problems, or identifying new problems to solve. The different approaches and ways in which the challenges we meet in software development can be decreased or perhaps eliminated permanently will be demonstrated using problem resolution strategies in software development. Understanding and assessing a set of requirements for an issue, devising a solution, and executing that answer on the computer, together with tests that prove the program fulfils its goals, would all be part of the programming.

Keywords:

Gathering Information; Models for collaborative problem solving; Groupware

Introduction

In software development, problem solving is the process of attempting to solve a problem domain by applying theoretical knowledge and research, best practices, and putting ideas to the test. After all, the software you're creating should be useful. What are your goals, and how can you break them down into features, user cases, stories, and processes. After all, what good is software if it doesn't make things easier, faster, more efficient, or cheaper, or if it doesn't solve a problem like employees deviating from a process?

So, in software development, problem solving is the use of logic and process thinking combined with creativity to solve a software challenge. The technique for resolving software development problem.The process of fixing a software development challenge, in my opinion, may be broken down into four steps:

- Determine the problem.
- Gather information
- Iterate on possible solutions.
- Put your solution to the test.

Method of collaborative problem solving and groupware in the software development domain

In software development, problem solving is critical. Many of the fundamental software development processes, from requirements analysis, specification, and design through testing and verification, may be seen as conventional problem-solving methods [1]. As the complexity of software development has increased, a new aspect has emerged: collaboration. Indeed, the increasing complexity of applications has demanded the usage of teams or groups to develop software, as individuals are unable to develop huge software systems with sufficient speed or quality. This review will focus on collaborative or group problem solving research and development in the field of software development, with the goal of identifying major outstanding topics and possibilities for both theory and practice advancement [2].

The modern computing professional works in an environment where programs can be thousands or millions of lines long, are frequently adjusted and maintained rather than built, are edited in a tool-rich environment, and work is almost always a collaborative endeavour [3] computer scientists are unprepared for today's environment since their pre-professional training typically concentrates on the creation of little programs (programming-in-the-small) and provides little expertise in sophisticated software development. Large-scale system development, on the other hand, necessitates a collaborative effort, and the more complex the problem, the greater the team required to tackle it. The fact that domain-specific expertise is typically localized and geographically spread is another element that contributes to the requirement for team development. According to studies, the ability to deploy good groupware is important to the success of such developers, especially when they are distributed [4]. Collaboration in system development has become a requirement, not just a theoretically feasible alternative, as a result of these causes. Fortunately, the advent of the World Wide Web has made geographically distributed collaborative systems technologically practical in ways that were before difficult or impossible. The word "groupware" will be

used to describe the software environments required to support a team whose members cooperate via a network [5]. Groupware solutions are designed to give a team a shared workspace even if they are geographically and temporally distant. The use of groupware or collaborative solutions can help to alleviate the logistical challenges that come with using distributed skills. Indeed, the future generation of development processes is projected to place a premium on the efficient integration of scattered knowledge.

Collaboration has been shown to have a favourable influence on both experienced and rookie programmers' experimental trials. Wilson and Nosek [6] conducted research to see if prior cooperation experience could help novice programmers with problem-solving and programming tasks. The findings supported the idea that collaborative efforts could increase the problem-solving skills needed for programming assignments. The study compared a control group of beginner programmers who worked alone on a software challenge to a group of programmers who were free to talk with one another. The results showed that even simple collaboration improved the beginner programmer's problem-solving abilities. The study also discovered evidence that an individual's ability had minimal overall impact on team performance, phenomena they argue occurs because teamwork compensates for individual flaws. The study also found that the collaboration gave the programmers more confidence in the answer and made the problem-solving process more enjoyable for them. Beginning programmers appear to benefit from collaborative interactions, which appear to aid in the analysis and modelling of problems, as well as the mastery of the analytical abilities required for such activities. Other controlled experimental investigations show that including collaborative activities into problem solving and programming instruction is beneficial even at the early stages [7]. Collaboration helps the problem-solving process, according to experiments with experienced software developers [8]. Indeed, all of the study's team projects outperformed equivalent independently completed projects, while team members were more personally satisfied with their work and had more confidence in their answers.

The overall goal of this study is to find strategies to make the software development process more efficient through collaboration. The review will focus on four areas: group problem solving, individual problem solving, groupware, and group psychology/sociology, including group and individual problem solving models and tools, groupware systems, group cognition, and team dynamics in the software development domain. With the goal of identifying a study topic that will represent an improvement in the state of the art, we will highlight contributions and remaining issues in group problem solving and group software development.

Models for collaborative problem solving

By definition, a group engaged in collaborative problem solving produces a plan for building a solution to tackle an existing problem. Collaborative groups appear to be better at dealing with complicated tasks than individuals, in part because groups have a wider range of skills and abilities than individuals [9]. Regardless, research show that group problem solving is more difficult than solo issue solving [10]. It can present group-specific issues such as an interaction environment that limits free expression of ideas [11], according to Hohmann participation biases, disputes resulting from interpersonal issues, or obstacles coming from the group's structure. Overall, however, the advantages of problem-solving teamwork greatly outweigh the drawbacks [12]. One prominent benefit of cooperation is the ancillary improvement of human capital that occurs as a result of individuals learning from the talents and abilities of others in the group [13]. The requirement to communicate designs, critiques, and arguments to other members of a group also improves a person's technical, critical, and interpersonal abilities [14].

A collaborative problem solving model is a strategy for facilitating collaborative problem solving that is clear. Not only would a comprehensive model include generic problem-solving procedures, domain-specific tasks, and essential cognitive skills, but it will also incorporate the communication and coordination activities that a collaborative setting necessitates. It's possible that the collaborative problem-solving approach is similar to the solo problem-solving approach. Indeed, Hohmann observes in his key work on group software development that collaborative problem solving can be done using the same problem solving methodologies as individual problem solving. While it is vital for a group to clearly choose and adopt a problem-solving approach, and while group members should be familiar with the method, according to Hohmann, the method does not have to be devised specifically for group issue solving. Despite this laissez-faire attitude toward the problem-solving method of choice, Hohmann observes that the way a team adapts such a strategy in a collaborative context differs significantly from the way an individual applies the same method.

Groupware

A groupware system can be classified as synchronous, asynchronous, or a combination of the two. Synchronous groupware systems operate in real time and facilitate group communication and collaboration through the use of tools like instant messaging. An electronic meeting system for brainstorming is an example. Asynchronous technologies, such as email, allow users to access saved messages or transmit messages to be seen later. A system having a message board and a chat feature is an example of a system with both asynchronous and synchronous features [15].Investigated how synchronous and asynchronous techniques affect communication behaviour differently. Asynchronous systems, for example, have extensive conversations with several, concurrent discussion threads, whereas synchronous systems have participants focusing on a single issue at a time [16]. Make a new distinction between synchronous and asynchronous systems, based on how tasks and information are shared rather than the temporal features of interactions. Asynchronous systems are defined as groupware systems in which tasks and choices are assigned individually and not shared until they are completed. Synchronous systems, on the other hand, provide a completely shared workspace that is always available to all users, where work products are generated

and evaluated in a collaborative environment with little task separation, and then merged by joint team decisions.

Groupware software

Some of the basic features of groupware functionality have been expanded into a number of settings. This section will go through some of the collaboration tools, platforms, and settings, as well as collaborative problem solving and software development. The review will be selective rather than thorough, as it is primarily designed to demonstrate the types of systems that are accessible. We'll start with a look at some common groupware systems, then move on to systems intended expressly for collaborative problem solving and/or software development. Finally, we shall focus in greater depth on the features of several important collaborative systems including Lotus Notes, Groove, and Rational Rose [17].

Conclusion

Software development groupware's ultimate goal is to improve the software development process. To date, such applications have placed a greater emphasis on processes and technologies than on people. Current systems have similar limits, as well as corresponding chances for improvement. These shortcomings stem from "not comprehending the particular demands this class of software imposes on developers and users," according to a notable researcher in the field. The aim is to convert these flaws into research opportunities.

The general result of our analysis of the literature is that merging perspectives and issues from collaborative problem solving, psychology, sociology, and collaborative software development can significantly advance the state of the art. Our overall goal will be to design a collaborative problem-solving model that takes into account a collaborative software development group's problem-solving cognitive processes as well as psychological and sociological elements that influence teamwork. The model will specifically handle a group's communication, cooperation, and coordination needs.

References

1. Deek FP (1999) The software process: A parallel approach through problem solving and program development. Comput. Sci Educ 9: 43-70.
2. Deek FP (1997). An Integrated Environment for Problem Solving and Program Development. Dissertation, New Jersey Inst Techno.
3. Mulder M, Haines JE, Prey JC, Lidtke DK (1995). Collaborative Learning in Undergraduate Information Science Education", Papers of the 26th SISCSE technical symposium Comput. Sci Educ 400-401.
4. Nunamaker JF (1999). Collaborative computing: The next millennium. Comput. Sci Educ 32:66-71.
5. Zwass V (1998). Foundations of Information Systems, Irwin McGraw-Hill, Boston, Massachusetts E-comm.
6. Prey JC (1996). Cooperative learning and closed laboratories in an undergraduate Computer Science curriculum. Comput. Sci Educ Bulletin. 28: 23-24.
7. Wilson JD, Hoskin N, Nosek JT (1993). The benefits of collaboration for student programmers. Comput. Sci Educ Bulletin. 25: 160-164.
8. Sabin RE, Sabin EP (1994). Collaborative learning in an introductory computer science course. Comput. Sci Educ Bulletin 12: 304-308.
9. Nosek JT (1998). The case for collaborative programming. Communications of the ACM Digital Lib 41: 105-108.
10. Finnegan P, Mahony LO (1996). Group problem solving and decision making: An investigation of the process and the supporting technology. J Inf Technol. 11: 211-221.
11. Hoffman LR (1965). Group problem solving. Advances in experimental social psychology. Am Psychol Assoc2: 99-132.
12. Hohmann L (1997). Journey of the Software Professional: The Sociology of Computer Programming. Prentice Hall J Softw Profes.
13. Guzdial M, Kolodner J, Hmelo C, Narayanan H, Carlson D, et al, (1996). Computer support for learning through complex problem solving. Communications of the ACM. 39: 43-45.
14. Hiltz SR, Turoff M (1985). Structuring computer-mediated communication systems to avoid information overload. Communications of the ACM. 28: 680-689.
15. Jarzabek S, Huang R (1998). The case for user-centered CASE tools. Communications of the ACM. 4: 93-99.
16. McGuire EG, Randall KA (1998). Process improvement competencies for IS professionals: a survey of perceived needs. In Proceedings of the 1998 ACM SIGCPR conference on Computer personnel research 1: 1-8.
17. Grudin J (1994). Groupware and social dynamics: Eight challenges for developers. Communications of the ACM. 37: 92-105.