

R·I·T

myCourses

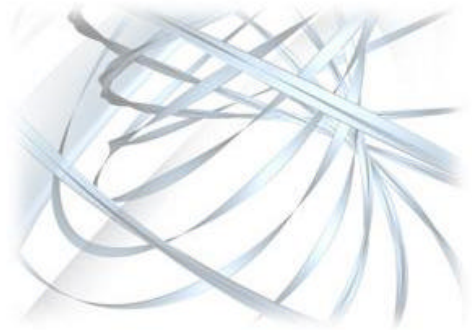
Login

Username:

Password:

Login

Welcome



Why CSCW Applications Fail

Class: CSCW and Groupware
Section: 4002-892-90
Project: Groupware Paper 1
For: Professor Rozanski
By: Robert M. Pufky
Due: 4/14/2005

Table of Contents

Table of Contents.....	2
Introduction.....	3
Why Applications Fail	3
Research to Succeed.....	6
Suggestions for Future Development	7
Appendix 1: Outlook calendar – personal and collaborative work.....	9
Figure 1: Personal Calendar	9
Figure 2: Collaborative “Groupware” Calendar	10
References	11

Introduction

Computer Supportive Collaborative Work (CSCW) has gone through many iterations in the past decades to improve user interactions and overall system acceptance; but a vast majority of these systems still fail in real-world applications. There are many reasons behind these rejections causing CSCW applications to fail; as well as various suggestions and demonstrations that improve system acceptance. By understanding the deficiencies in the entire CSCW process and the suggestions on how to improve them, superior CSCW products can be designed from the ashes of the generations beforehand.

Why Applications Fail

Grudin stated that there were three main problems with CSCW applications. The first of these problems is that “The application fails because it requires that some people do additional work, while those people are *not* the ones who perceive a direct benefit from the use of the application” (Grudin, 1988). This idea is also supported by Andrew Cockburn with “Additional Work – While some people (often managers) may benefit from the introduction of CSCW facilities, others (usually lower in the organizational hierarchy) have to do additional work to support those facilities” (Thimbleby, 1991). This implies that with typically constructed CSCW software, the benefactors are not the ones who experience the extra workload – which can even be seen on the RIT campus with such CSCW programs as MyCourses. In the College of Business, professors typically assign all of the updating and managerial work for the CSCW to graduate assistants. They enjoy the “ease” at which information is distributed to students, while the graduate students become frustrated at the complexity and enormous load of work they must do. “Extra work no matter who does it or how it relates to benefit – can sometimes be reason enough for abandoning technologies or certain courses of action” (Bowers, 1994). Indeed, this complexity is one of the reasons that RIT is switching over to the Desire2Learn

suite, which is supposed to be much easier to manage. This work imbalance must be corrected if CSCW applications are to succeed.

A second problem that Grudin believes why CSCW “design process fails because our intuitions are poor for multi-user applications” (Grudin, 1988). He believes that “application development projects rely heavily on intuition” (Grudin) and that “Intuition may be a far more reliable guide to single-user applications” (Grudin). With single user applications, “It is relatively easy to bring a single user into a lab to be tested” (Grudin), which allows for quick development of single-user systems. Trying to match this intensity of user research for a typical, if that is even possible, group of CSCW users is staggeringly expensive, and all the dynamics will still not be covered. Grudin points out that “a typical CSCW application will be used by a range of user types -- ... *all of whom* may have to participate in one way or another for the application to succeed.” Although there is no clear-cut way to get a viable group test result back, strides must be made to at least mimic the expected user base in hopes to better identify flaws in the software.

A final point that Grudin believes is a problem is that “we fail to learn from experience because these complex applications introduce almost insurmountable obstacles to meaningful, generalizable analysis and evaluation” (1988). In other words, CSCW applications can only be measured on whether they were a success or failure, whether they worked or did not. The complex nature of CSCW lends itself to making it hard to identify what exactly went wrong with the system, how users have trouble using it and what should be improved. “Establishing success or failure will be easier than establishing the underlying factors that brought it about” (Grudin). For CSCW software to succeed, these user difficulties need to be tracked and responded to.

Another problem mentioned in various articles is organizational and technological determinists. An organizational determinist is one who rejects “the technology because of how business cases were done” (Bowers, 1994), whereas a technological determinist uses “the technology to try and encourage new forms of business cases” (Bowers). Depending on how CSCW is deployed, either type can succeed or fail based on an idea called the gradient of resistance. The gradient of resistance means “the more radical a change proposed to adapt a CSCW system, the more resistance there is to accepting it” (Rogers, 1994). When applied to technological and organizational determinists, the ones with the least amount of force change typically wins. In business, this typically means that the organizational determinist will force the technology to adapt, or have it removed. “But it *turned out* that the former [organizational] was the stronger force” (Bowers). By building CSCW software that is not adaptable to organizations needs and wants, it can be said that the CSCW software will either be used sub-optimally or for personal matters (Rogers). These apparent differences in software and organizational dynamics need to be addressed for CSCW to be thrive.

Finally, organization cultures also have a great affect on whether CSCW will be successfully implemented in a business setting. A company that has a competitive culture is more likely to fail at implementing CSCW. This is simply because the roots of CSCW are founded in cooperation from all users on that system. When users become competitive, information tends to be held back, defeating the purpose of the software in the first place. In one example, “The reason for this failure was attributed primarily to the company having an inherently competitive culture that was not geared towards engendering cooperation against its employees” (Rogers, 1994).

Research to Succeed

Research has also provided some basic answers in the search for understanding how to improve CSCW applications, one of these being the use of champions. Champions are commonly found in the business area pushing their projects, usually self-selected, and passionately assuring that everyone's part is critical to its success. In a CSCW context, "these mediators adapt a new collaborative technology to a context, modify the context as appropriate to accommodate use of the technology, and support ongoing changes to the technology and context over time" (Okamura, Orlikowski, Fujimoto, & Yates, 1994). They assist, change and support other users on the CSCW system. By having this extra level of assistance readily available, overall acceptance and usage of the system should increase. "Mediators may be particularly important in increasing the effectiveness with which CSCW applications ... are adopted, implemented, and used over time" (Okamura, Orlikowski, Fujimoto, & Yates). One assumption of the champion model is that the champions have the ability to change and modify the CSCW as they see fit, which implies that the CSCW application must be extremely flexible in the way it can be setup to display information for users of that system.

Another improvement that has been suggested, and actually tested widely yet unknowingly through most of the internet using population is the use of a single-user application in a multi-user environment. This research suggests that applications designed for ease-of-use for single users can be adapted to work with multiple users. This is done through familiarity and predictability. With familiarity, "reduced disparity between the techniques used in personal and group work, the burden of learning and remembering these techniques is correspondingly reduced." (Cockburn & Thimbleby, 1991) Combined with predictability, which is "having had experience with a system in personal work, the user knows its particular quirks and need not risk

the embarrassment that might occur if an unfamiliar interface to the collaborative environment has unexpected side effects” (Cockburn & Thimbleby), they have created a positive CSCW experience. This positive experience “combined with the ability to use personally favoured tools, could serve to encourage group participation” (Cockburn & Thimbleby). Although this idea seems fairly complicated, a manifestation of this type of CSCW application already exists – e-mail. In everyday experiences, both within and outside of CSCW applications such as MyCourses, e-mail is standard. The general layout, the addresses, and the options available (although not exactly the same in each application) are in familiar formats and layouts for both personal and collaborative work. See **Appendix 1** for screenshots of this type of CSCW application. This implies that some portions of CSCW applications can be designed for single-users and moved over to a multi-user environment successfully.

Suggestions for Future Development

Considering all of the problems with the current CSCW implementations and the reasons why they fail, as well as the successful improvements, traits for successful CSCW implementations can be shown:

1. All users of a CSCW system should do equal amounts of work
2. Benefits of the CSCW system should be visible to all users
3. Use a non-homogenous testing population for CSCW development
4. User problems, difficulties, and program errors need to be accurately tracked
5. If CSCW fails, the technology is typically replaced, the organizations are rarely restructured
6. CSCW applications need to be built such that organizations can customize a majority of the application to their needs
7. Organizations should be internally non-competitive

8. Single-user design then multi-user implementation will help the success of CSCW implementations through familiarity, predictability and encouragement

These guidelines will allow for the creation of superior CSCW applications – not in the technological sense, but acceptability and usage rates. These next-generation CSCW applications will in turn reduce the rate at which they fail in the real-world and increase the overall acceptance of these systems.

Appendix 1: Outlook calendar – personal and collaborative work

Figure 1: Personal Calendar

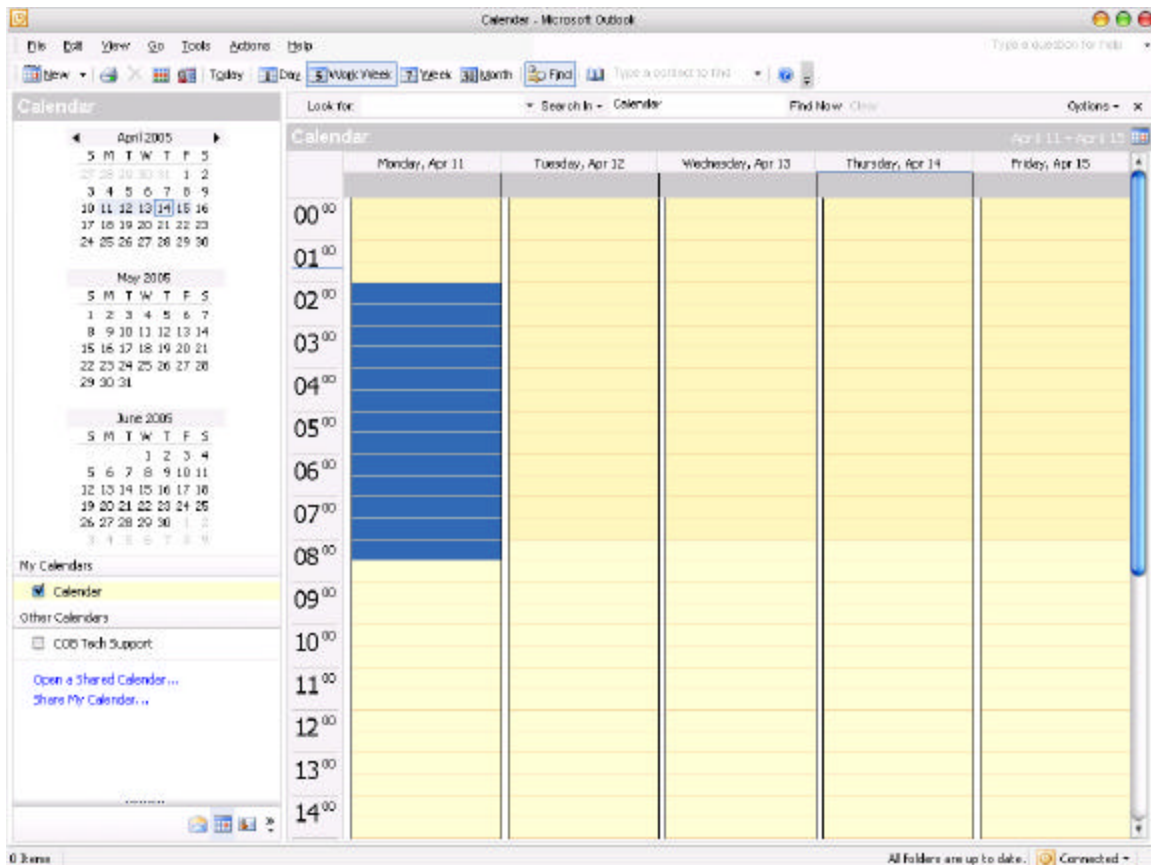
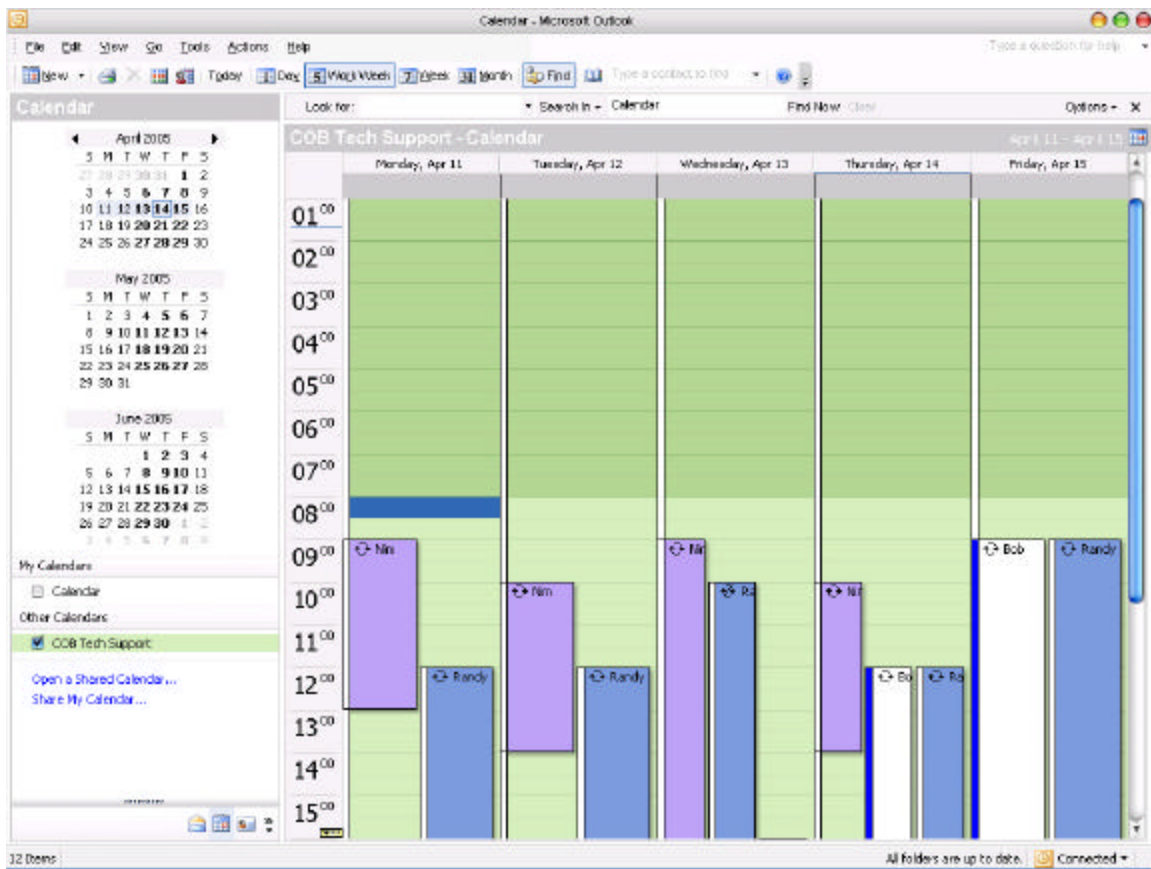


Figure 2: Collaborative “Groupware” Calendar



References

- Bowers, J. (1994, October). The work to make a network work: Studying CSCW in action. *Proceedings of the 1994 ACM Conference on Computer supported Cooperative Work*, 287-298. Retrieved April 1, 2005, from ACM Digital Library database.
- Cockburn, A. J., & Thimbleby, H. (1991, July). A reflexive perspective of CSCW. *ACM SIGCHI Bulletin*, 23(3), 63-68. Retrieved April 1, 2005, from ACM Digital Library database.
- Grudin, J. (1988, January). Why CSCW applications fail: Problems in the design and evaluation of organization of organizational interfaces. *Proceedings of the 1988 ACM Conference on Computer-supported Cooperative Work*, 85-93. Retrieved April 1, 2005, from ACM Digital Library database.
- Okamura, K., Orlikowski, W. J., Fujimoto, M., & Yates, J. (1994, October). Helping CSCW applications succeed: The role of mediators in the context of use. *Proceedings of the 1994 ACM Conference on Computer supported Cooperative Work*, 55-65. Retrieved April 3, 2005, from ACM Digital Library database.
- Rogers, Y. (1994, October). Exploring obstacles: Integrating CSCW in evolving organisations. *Proceedings of the 1994 ACM Conference on Computer supported Cooperative Work*, 67-77. Retrieved April 1, 2005, from ACM Digital Library database.