Educational Experiences from a Global Software Engineering (GSE) Project

Maryam Purvis, Martin Purvis, Stephen Cranefield
Department of Information Science
University of Otago
PO Box 56, Dunedin, New Zealand
{tehrany,mpurvis,scranefield}@infoscience.otago.ac.nz

Abstract

Technological advancement in the Internet and other areas of communication have made the idea of collaborative projects with other people who might be physically located on separate sites more feasible. There are many benefits that can result from the wider scope of interactions that is afforded by these developments, such as taking advantage of available expert resources that might exist in various parts of the world – for example in emerging IT centres in India and China. Also, a well-integrated team made up of a cross-cultural group can be more equipped to meet demands associated with a heterogeneous market. However there are some problems in getting these disparate teams to work together effectively. In this paper, we discuss some of the issues that were encountered during our collaboration with the Technical University of Munich in running a joint software engineering project that involved the development of a multi-player online game environment running on multiple platforms on a peer-to-peer wireless networking environment. We also discuss lessons learned and provide some recommendations for improving the processes associated with a global software engineering project.

Keywords: Global Software Engineering, Distributed team, Virtual team collaboration.

1 Introduction

Both from the economical and educational point of view it seems appealing to take advantage of the resources and special skills that might be present in a larger geographical domain by forming virtual teams whose members are physically located in different centres. It is evident that a well integrated software development team, in which the members can easily share information with each other, is an important factor in the success of any software engineering project. Communication, always a key issue, is even more important when the team members are not co-located. To facilitate a well-integrated virtual team, an appropriate set of guidelines and processes and a useful set of tools needs to be provided, which can help to overcome the communication difficulties that might be caused by a distributed set of team members. Previous research that has been done in this area [Kobylinski et. al 2002, Damian and Zowghi 2003, Layzell 2000, Grinter 1999] indicate that a good portion of the problems associated with these virtual teams are social rather than technical. Due to limited knowledge that the team members have about each other, it is more difficult to get involved in casual interactions, and therefore they have less frequent information exchange about relevant issues during software development. According to these previously published studies, this problem leads to latency in the information exchange, which results in problems being revealed much later in the development cycle, and this compounds the problem.

To facilitate team members becoming more familiar with each other, attempts have been made [Favela 2001] to request each participant to come up with a personal portfolio and make it available to others. Although this activity might help make people more familiar and comfortable with each other, the biggest separation is not just superficial acquaintance, but can also involve deeper issues in terms of the differences that might exist in the respective backgrounds, cultures, and languages [McCarthy and McCarthy 2002]. Overcoming this latter type of separation is more difficult. Perhaps the first beneficial step that one might take is to become more aware of the existence of such differences in order to be able to make the appropriate adjustments..

It is interesting to observe that many of the open source projects overcame the problems mentioned above. In some cases, such as the Linux project [Linux 1994], a complex system has been successfully developed by a large community of people who were mostly not personally familiar with each other. The question that comes to mind is, what are the important characteristics of the successful open source projects that overcomes the physically distributed nature of the participants? Some of these factors from the point of view of the authors will be discussed in this paper.

The goal of this paper is to describe our educational experience with the collaboration involved in running a global software engineering (GSE) project as part of separate academic papers (courses) offered at both at a University in New Zealand (UO), and at the Technical University of Munich (TUM) in Germany. In the remaining sections of this paper, first, a brief history about the software project is given. Second, a brief description of the characteristics of both teams is provided. Third, the infrastructure that was set up to facilitate the collaborative work is discussed. Fourth, some of the issues and problems associated with this joint effort are discussed. Finally, a set of recommendations and guidelines to improve the process is presented.

Figure 1. The Sharp Zaurus SL-C700 personal digital assistant (PDA)

2 The Global Software Engineering project

The software project undertaken involved providing a generic framework, called ARENA, that can be used in developing various real-time multi-player online games using a peer-to-peer, wireless communication architecture. Initially a game called SWORD [ARENA 2003], which is a simplified version of the popular games such as Quack [Quack 1999] and WarCraft [WarCraft 2003] was specified to be developed in order to test the functionality of the underlying generic game-playing support framework, which is called FRAG [Nagel 2003]. One of the requirements of the system to be developed is to be able to run on various platforms, including the Apple iBook [iBook 2003] laptop and PDA hand-held computing systems, such as the Sharp Zaurus (SL-5600 and SL-C700) [Zaurus 1991]. To accommodate this requirement, the Java language, which is essentially platform independent, was selected as the development language (although there are different versions of Java that run on the Apple and the Zaurus computers).

An earlier version of this system, running only on Apple laptops, was developed by the students at TUM in the previous year. The goal for this year's project was for the two teams to operate as follows:

1. The TUM students were to extend and improve the existing prototype system associated with the SWORD/FRAG system. In particular, they were interested in adding 3-D graphics to what was originally only a 2-D graphical set-up.
2. The students at UO were required to make sure that the existing prototype system would run on the Sharp Zaurus PDA platform. This meant that the system was to be customized appropriately so that the smaller screen and the limited keyboard and pointer capabilities on the Zaurus were taken into consideration (see Figure 1).

The system was originally designed with a layered architecture, as shown in Figure 2. In this diagram the bottom layer deals with the network, object management and the interaction between the objects across different hosts. The second layer deals with some generic functionality required for these types of online games, in terms of general game concepts and the management of graphical objects (2-D or 3-D) . Finally the top layer is concerned with a particular game application functionality (i.e the SWORD game).

3 The characteristics of the two teams involved

The group at TUM comprised 29 students, while the group at UO was much smaller, with 5 students on the project. The language of inter-team communication was English, although the native language for most of the TUM students was German.

As is normal in software development, there were considerable variations in the skills and abilities of the students involved, even among the co-located students. At UO the students had a broad range of backgrounds: a few of the students had significant industry experience, while others had more conventional academic backgrounds and had limited experience. There were also two students on the UO team who had come from Germany to take part in a one-year course of study at UO. Although the authors of this paper are less familiar with the backgrounds of the TUM students, we observed from our communications from them that they had good technical backgrounds.

The academic terms of the project courses offered in Germany and New Zealand were not synchronous, and this present some coordination difficulties. The course offered at the TUM was a one semester paper, while the one at UO was a full year paper. In addition, the start and the end of the academic semester differed for both universities. The total collaboration period was about 3.5 months which included two weeks of a mid-semester break period for students at UO.

Both groups had a common textbook [Bruegge and Dutoit 2003] whose authors were the course coordinators of the software engineering project course at TUM. This provided some common ground in terms of software engineering concepts and principles used during this collaboration.

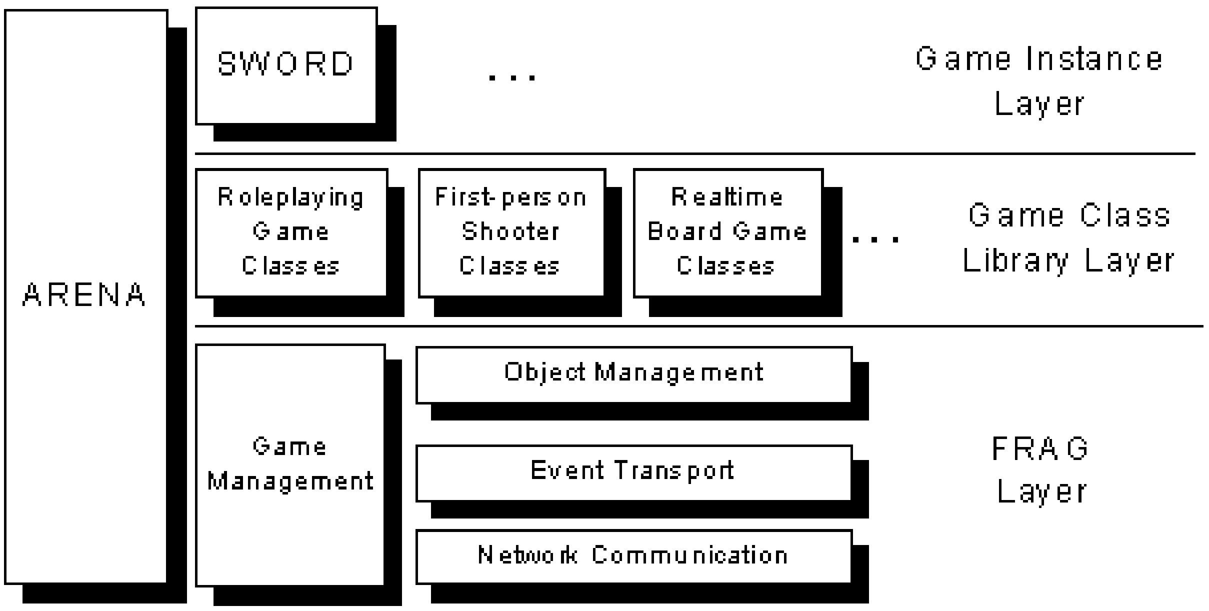

Figure 2. The original ARENA/FRAG system architecture [Nagel 2003]

4 The collaborative environment and process set-up

The collaboration was performed primarily by using the Lotus Notes-based bulletin board [Lotus Notes 2003] that was set up by the staff at TUM. At UO, we had Web access to this bulletin board. All the source code and documentation was available through this Web interface. In addition, there were a number of discussion threads for various aspects of the project, with labels such as ‘UI’, ‘Network’, ‘Architecture’, and ‘Documentation’. There was also one thread for ‘Off-Topics-Discussion’, which could be used to discuss general topics and announce social events.

Most of the discussion was structured around IBIS (Issue-Based Information System) concepts [Kunz and Rittle 1970]. In this context, an issue concerning some aspect of the project can be raised, and a number of positions can be put forward by either the issue poster or other participants. Each position in turn can be either supported or opposed by a number of arguments. The winning position is the one which has the strongest arguments put forward for it. An issue is resolved when there is a clear winning position for that issue identified. In this way, a record of all the discussions that were considered was kept for all the stakeholders of the system. Both sides were familiar with the basic ideas associated with this concept

In order to facilitate collaborative work and generate some of the required documents, such as the requirements and design documents, the REQuest tool [Dutoit and Paech 2001] was used. This tool provides the capability to use both use-case models and the IBIS model to capture rationale for decisions made with regard to specifying the requirements and the design of the system. The participants at UO were initially reluctant to employ this tool– perhaps due to lack of familiarity with the basic functionality and usage of the tool. However, since this was in use at TUM, UO users learned to use it in order to review the information that was put forward by the TUM participants.

Initially, the participants at UO were using a locally set up bulletin board called Blackboard [Blackboard 1997], which is used for most of the papers offered at UO. This bulletin board is used to access certain course repositories as well as to access various online discussions. The Blackboard is normally administered by the staff who have higher privilege for setting up the required infrastructure and the documents/information posted. To give more flexibility and control to the students in terms of creating their own Web pages and editing and commenting on each others work, WiKi pages [WiKi 2003] were also used. However, after some experimentation and trialing of these various media, it eventually became difficult to keep up with the discussion on all these forums, and it was decided to use only one bulletin board: the Lotus-Notes bulletin board set-up at TUM. There was a discussion thread added to this bulletin board for the UO-specific topics that were unlikely to be of relevance to the students at TUM. It turned out that the Web access to the bulletin board set-up at TUM was rather slow, since the server was local to TUM. Work was then begun on setting up a local Lotus Notes server at UO, but due to fire-wall and security issues this work was not completed until much later in the second semester, when the TUM project term was at an end.

In addition to the above infrastructure for the collaborative work, other communication mechanisms such as email, telephone, and video-conferencing were occasionally used. In particular there were several DVDs produced at TUM, which recorded scheduled project review meetings at various stages of software development, and these were posted to New Zealand. The UO team also had made a video for the purpose of introducing staff and student members and presenting a brief statement about their expected roles. This video was made available on the project web site. In general most of the interaction was carried out asynchronously, although there were some video conferences arranged. However the eleven-hour time difference made it difficult to find a time that was convenient for all parties.

5 The role of each group in the shared project

The students at TUM were responsible for designing and implementing the SWORD/FRAG, system as specified in the problem statement document. There was a prototype of the system already available due to the effort associated with the previous class’s work. The task for the current students was to extend the functionality of the prototyped system. The platform that was used during this development at TUM was the Apple iBook computer.

The group at UO were responsible for delivering a version of the SWORD/FRAG system that would run on the Sharp Zaurus computer. One of the overall goals of this system was to design the architecture of the system so that it would run on multiple platforms (although the specific issues associated with this portability had not been examined in detail at the beginning of the project). The assumption was that the basic framework at the lower layer (the FRAG architecture depicted in Figure 2) would address all the cross-platform issues. Using this basic framework, the specific modular components needed to support special constraints associated with each platform were to be inserted as plug-in components.

6 Issues and problems observed about the joint effort

6.1 Difficulties in having a shared schedule

Due to different academic schedules (in terms of the start and the end) and different project sub-goals, it was difficult to come up with a shared schedule for various milestones. Although the initial requirement was to develop a system that could be run on various platforms, the initial architecture had been developed without consideration of the constraints associated with small, hand-held devices, such as the Sharp Zaurus.

6.2 Architectural issues due to lack of clear understanding of the requirements at the outset

The students at UO, during the early examination of the prototype system, observed that the architecture of the existing system need to be re-factored in order to reduce the coupling between various components and also to isolate the device-dependent modules. After some discussion, the diagram in Figure 3 was proposed as an alternative to the original architecture shown in Figure 2.

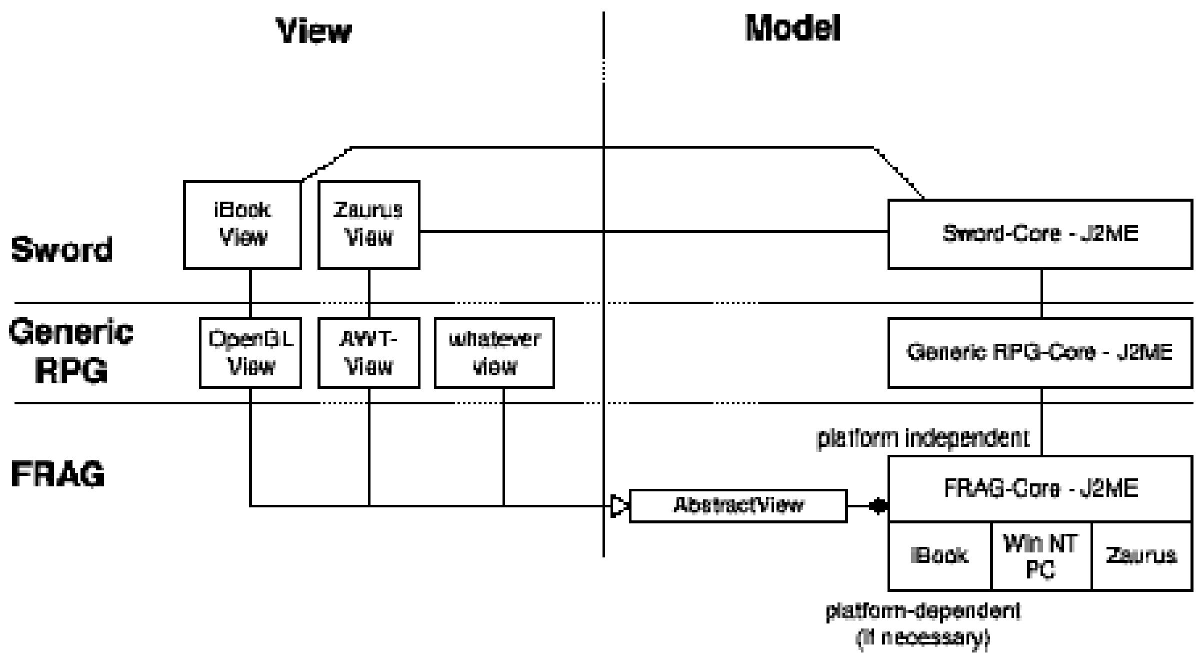

Figure 3. The revised system architecture for the ARENA/FRAG system. In this diagram RPG stands for Role Playing Games. J2ME is a light version of the Java language that runs on the Zaurus platform.

This diagram (Figure 3) shows an attempt to make the required platform-dependent modules more explicit and specify how these modules should be interfaced with the core system. For example at the FRAG layer there are modules that deal with iBook or Zaurus platform-specific issues. Dependencies at this level were reflected at the higher level as well. For example, due to the limited processing power of the Zaurus computer, it was not possible to provide a 3-D representation of the graphical objects in the game. Therefore there had to be a 2-D representation of any 3-D objects. Also, the Zaurus operating system platform had access only to a subset of the user interface functionality provided for the Java language on standard platforms. Therefore a different user interface module had to be written that only accessed this subset of functionality.

It became clear to the students at UO that without these changes to the existing architecture, it would not be possible to build a system with a shared framework. After some joint discussion with both parties (TUM and UO), a compromise architecture was developed which maintained some of the modules in the original architecture (due to the fact that the TUM-end of the project had already developed some code that supported that earlier architecture) and also contained some additional modules to facilitate the requirements for the platform independent requirement.

However, due to academic schedule constraint, it was not possible to take full advantage of the newly evolved architecture. The students at UO were required to produce a prototype of the system running on the Zaurus by the end of the first semester, and the architectural redesign had consumed some time. Therefore the UO team adopted a more short-term approach to meet the schedule requirement: take the existing system as it was in the original (TUM-produced) prototype and attempt to port the system to the Zaurus platform by eliminating some code that did not apply and provide some patches to make the minimal set of functionality work on this smaller platform.

During the same period, on the other hand, the students at TUM, who had more time for that phase of the development, went ahead with the development of the newly proposed architecture.

During the course of development at UO, the team identified more limitations with the Zaurus platform, which caused the group to modify further their path. One of the crucial limitations of this platform was the small amount of RAM memory that was available for application use. It was decided that running a generic game-playing framework such as ARENA/FRAG might not be suitable for a small device platform. So the group at UO decided that they needed to rewrite a FRAG-like framework (FRAG being the lower-level, more platform-specific aspects of the ARENA/FRAG architecture) designed for a small device, where the resource usage had to be carefully designed to minimize the memory consumption. Although the group had to come up with a new design, they still felt they had benefited from being exposed to the original architecture of the FRAG system.

During this phase of the development, both groups showed some frustration due to some changes that had to be made in order to accommodate the requirements at both ends. But due to semester and schedule incompatibilities these changes could not have been easily accommodated within the originally specified time frame.

If we make a comparison with Open Source projects, we will see that in an Open Source project the basic architecture/infrastructure of the system is somewhat layed out at the outset. It is this shared understanding of the skeleton of the system, that holds the group together. In our case, the initial structure provided a common ground for the discussion, but due to some deficiencies in this architecture, different paths had to be taken.

Another constraint that the Open Source community does not have to deal with is the time constraints that a software engineering project in an educational context have to comply with. Due to these fixed deadlines that exist in these contexts, the nature of decisions made during these projects are impacted.

6.3 Conflict Over Teams' Responsibilities and Authorities Concerning Shared Components

It was difficult to partition some of the teams' responsibilities concerning common components. If the common components were not suitable for a particular platform, what plan of actions should be taken? Should both teams deviate from the schedule and take time to re-design and re-implement the common framework, or should the affected team divert from the joint path by creating another framework that is more suitable for the targeted platform? Clearly, not all these issues can be predicted at the outset. But providing mechanisms to deal with these contingencies is something we believe needs to be considered at the earliest stages of development.

One possible mechanism to address this issue would be to schedule a regular online meeting where all the concerned parties are required to attend in order to discuss any critical issues that might exist. In these meetings when required possible alternative paths agreeable to both sides could be negotiated.

6.4 Different Culture and Conventions Associated with the Software Development Process

Obviously, each team has a different background. In the context of education, each instructor who leads a software engineering team has a different focus. In our experience, it was noted that the group at TUM were more rigorous in documenting their requirements and design documents and their processes in this regard were more strict than those that were followed at UO. The smaller team at the UO, on the other hand, were more concerned with the implementation of the final product and less concerned about the documentation standards. This was partly due to the nature of the work that was undertaken at UO, where a first implementation prototype had not yet been produced. Developing a complex system on a new platform (Zaurus) required more feasibility analysis in order to identify earlier some of the constraints associated with the platform. This had forced the team at the UO to take more of a bottom-up approach than their counterpart team at TUM.

7 Recommendations and Lessons Learned

7.1 Identify the role of each group member and their responsibilities with regard to the counterpart team.

This provides each team member with the basic knowledge about the structure of each group. For example it is important to know who (on the counterpart team) is involved in the networking aspect of the project, in case a networking issue is identified that requires immediate attention. By clearly identifying the responsibilities, the stakeholders know what to expect from each other. For example, if there is a guideline with regard to responding to a posted message within certain limits of time, this could reduce some of the anxiety whether someone should respond to a request or not.

By identifying some of these responsibilities at the outset, the members of both groups can negotiate on a common set of policy which is satisfactory for all involved.

7.2 Adopt a set of tools acceptable to all groups for communication and development.

Various people are comfortable with different tools with which they are familiar. It is difficult to impose the use of a tool that is not in their common tool set if there has been no training or adequate information available. The rationale for using such tools should be well explained and the people should be given adequate time to become familiar with the details of using them effectively. In the context of GSE, when other tools are used locally, the result of the work should be recorded in a commonly accepted or agreed-upon format.

7.3 Provide a set of guidelines for communication protocols and the development process

In order to reduce some of the anxiety associated with working with a remote site, it is valuable to have a set of general guidelines that provide some common vocabulary and protocols to be used. The fact that both of our teams were familiar with the textbook was a major factor in establishing a common culture from the software development process point of view. Also being familiar with IBIS concepts which the discussion board was centred around, was beneficial. However, not being comfortable with using a bulletin board based on a remote server and the use of REQest tool was more problematic.

7.4 Be flexible and adaptable when new requirements are identified.

We believe that it can often be advantageous to follow the Agile [Highsmith 2002] process for software development, in particular when a project involves a high element of risk due to unknown factors involved. In other words, in the circumstances of GSE it is valuable to be as flexible as possible and to be able to alter pre-planned schedules and choose more adaptable processes. In our experience, once it was identified that due to memory constraints we could not use the same common framework (the FRAG layer for the peer discovery and object management), we had to make a major decision in choosing to develop our own FRAG-like framework for small devices. We still benefited from our joint collaboration, since we had learned some of the key issues involved in creating such a framework: in particular the critical issues concerning efficient use of memory by re-using or re-cycling required objects.

8 Conclusion

We think this has been a very valuable experience from educational point of view. The software produced through a GSE effort can be better quality software. This is because there is more motivation to make the components more modular and make the interfaces more clearly defined, since the success of each team depends on the ability of their counterpart team to be able to use their modules effectively. In a GSE environment, the members are more motivated to produce a better quality of documentation promptly. In addition, it is more likely that the system would go through more careful considerations in terms of quality of design and implementation, since these artifacts are more public commodities. Similarly with Open Source systems, the chances of identifying bugs earlier on and finding solutions for them by various stakeholders involved are increased. Perhaps this is associated with the goal and effect of improving one’s status and getting recognition from a larger peer group [Raymond 1997].

By following the above recommendations, the GSE groups can reduce some of the frustrations that might be caused in this process and still enjoy the benefit of working in a satisfying and versatile team environment and have a chance to produce a good quality product.

Acknowledgement

The authors would like to thank all the staff and the students who participated in this project in particular, we appreciate the work done by Mark George, Geoff Hay, Lars Ehrler, Noel Garside, Benjamin Herrman, Jonathan Lingard, and Rodney Tamblyn.

9 References

Damian, D., Zowghi, D., An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations, in Proc. of the 36th Hawaii Conference on Systems Sciences (HICSS'36), Hawaii, January 2003.

Grinter, R., Herbsleb, J., Perry, D., The geography of coordination: dealing with distance in R&D work, Proceedings of the international ACM SIGGROUP conference on Supporting group work, p.306-315, November 14-17, 1999, Phoenix, Arizona, United States.

Kobylinski, R., Creighton, O. , Dutoit, A.H., Bruegge, B., Building Awareness in Distributed Software Enginering: Using Issues as Context , International Workshop on Distributed Software Development, International Conference on Software Engineering. Orlando, Florida, May 21, 2002.

Layzell, P.J., Brereton,P., French, A., Supporting Collaboration in Distributed Software Engineering Teams, APSEC2000: The Asia-Pacific Software Engineering Conference, 5-8 December 2000, Singapore, IEEE Computer Society Press, 2000, pp.38-45.

ARENA http://wwwbruegge.in.tum.de/projects/arena/doc/documents.html Accessed Sept. 2003.

Blackboard http://www.blackboard.com/ 1997.

IBook http://www.apple.com/ibook/ Accessed Sept. 2003.

Linux http://www.linux.org/ 1994.

Lotus Notes http://www.lotus.com/products/product4.nsf/wdocs/noteshomepage?OpenDocument&cwesite=notes Accessed Sept. 2003.

Quack http://www.q3arena.com/ 1999.

Raymond, E., The Cathedral and the Bazaar, http://catb.org/~esr/writings/cathedral-bazaar/

WarCraft http://www.blizzard.com/wow/ Accessed Sept. 2003.

WiKi http://www.zshwiki.org/cgi-bin/wiki.pl?HomePage Accessed Sept. 2003.

Zaurus http://www.zaurus.co.il/ 1991.

Kunz, W., Rittle, H. Issues as Elements of Information Systems. Technical Report Working Paper No. 131, Institute of Urban and Regional Development, University of California, Berkeley, Berkeley, CA (1970).

Nagel, M. FRAG: A Java Framework for Peer-to-Peer Games, A thesis report from the Technical University of Munich, 2003.

Bruegge, B. Dutoit, Allen H., *Object-Oriented Software Engineering: Using UML, Patterns and Java*, 2nd Edition, Publisher: Prentice Hall, Upper Saddle River, NJ, 2003

Bruegge, B. Dutoit, A.H., *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*, Prentice Hall, Upper Saddle River, N.J., 2000.

Dutoit, A.H., Paech, B., Rationale Management in Software Engineering. In S.K.Chang (Ed.)*Handbook of Software Engineering and Knowledge Engineering*. World Scientific, 2001.

Highsmith, J., *Agile Software Development Ecosystems*, Pearson Education, 2002.McCarthy, J., McCarthy, M., Software for your Head, Addison Wesley, 2002.

Favela, J., Feniosky, P., Experiences in Geographically-Distributed Collaborative Software Development, *IEEE Software*, March 2001.