Effectiveness of Elicitation Techniques in Distributed Requirements Engineering

Wesley James Lloyd
Hewlett Packard
wlloyd@acm.org

Mary Beth Rosson
Virginia Tech
rosson@cs.vt.edu

James D. Arthur
Virginia Tech
arthur@cs.vt.edu

Abstract

Software development teams are often geographically distributed from their customers and end users. This creates significant communication and coordination challenges that impact the effectiveness of requirements engineering. Travel costs, and the local availability of quality technical staff increase the demand for effective distributed software development teams.

This research reports an empirical study of how groupware can be used to aid distributed requirements engineering for a software development project. Six groups of seven to nine members were formed and divided into separate remote groups of customers and engineers. The engineers conducted a requirements analysis and produced a software requirements specification (SRS) document through distributed interaction with the remote customers.

We present results and conclusions from the research including: an analysis of factors that effected the quality of the Software Requirements Specification document written at the conclusion of the requirements process and the effectiveness of requirements elicitation techniques which were used in a distributed setting for requirements gathering.

1. Introduction

Software quality is often reflective of the quality and maturity of the software development process of the organization. Pedagogical software process models such as the Waterfall, Spiral, and rapid prototyping all include Requirements Analysis activities. It is generally accepted that the ultimate quality of the delivered software depends on the requirements upon which the system has been built. [1] [7] Studies performed at many companies have measured and assigned costs to correcting defects in software discovered at various phases of the project lifecycle. Generally the later in the software lifecycle a defect is discovered the more expensive it is to rectify. Research suggests that correcting software defects can require nearly two hundred times the effort if the correction is implemented in the maintenance phase versus the requirements specification phase of a software lifecycle. [6]

Software development organizations are often geographically distributed from their customers and end users. High travel costs, and the local availability of skilled technical staff increase the demand for distributed software development efforts. Frequently face-to-face meetings with end-users are not realistic.

This paper reports the results of an exploratory empirical study conducted to study the effectiveness of requirements engineering in a distributed setting. Forty-six participants role-played as either customers or engineers to form six separate groups in the project. At the conclusion of the project, the participants who role-played as software engineers wrote a Software Requirements Specification (SRS) document using only the knowledge gained from their remote collaboration with their customer.

This empirical study had three primary goals. The first goal was to identify what factors led groups to write high quality SRS documents. A second goal was to evaluate the effectiveness of the groupware (software for collaboration) used to enable distributed requirements engineering. And finally, a third goal was to assess the effectiveness of various requirements elicitation techniques when used in the distributed mode. This paper focuses on the first and third goals of the study, presenting observations, results and conclusions.

2. Background

Distributed software development projects have become practical because of technological improvements in communications structure, bandwidth and performance. Global business ventures, strategic partnerships, and joint

ventures, all share a need in supporting distributed software development efforts. [14]

Organizational factors contribute to the need for distributed software development. Many reasons for conducting distributed software development have been identified including: project members that are unwilling to travel, lack of skilled workers in a geographical area, high travel and relocations costs, and the physical location of specialized hardware. [3]

The many factors motivating distributed software development do not automatically constitute gains in productivity and quality. Microsoft has long valued the ability for software engineers to meet informally face-to-face to resolve development problems. [3] However software engineering literature has identified advantages for distributed work. [2] [8] [11] [12]

The use of email has been identified to be a rich communications tool for supporting telecommuting and asynchronous collaboration. Some notable advantages of email include: broadcasting capability, management of communication, access to information resources, low communication cost, file transfers, and temporal and spatial flexibility. [2]

In [8] a software design experiment was conducted to compare the quality and creativity achieved using two different meeting environments. Forty-one groups carried out a design process using either distributed asynchronous meetings or synchronous face-to-face meetings. The results suggested that the distributed asynchronous groups produced more creative and possibly higher quality software designs than the face-to-face groups. The study attributed the better performance of distributed groups to the benefits available with using computer support tools to mediate the process.

At Fujitsu corporation a study found that the development cycle time was reduced 75 percent by applying the use of concurrent software development methods. When team members were distributed in different locations, software development could proceed around the clock across time zones.

A large number of requirements elicitation techniques exist which help requirements engineers gather requirements from customers and end-users. [5] Some of these requirements elicitation techniques may work very well in a distributed setting, where others may fail due to the limited informational bandwidth of distributed interaction. For example, studies of computer-mediated communication have documented many of the problems that arise when less "rich" communication media (e.g., text versus videoconferencing) are used to support remote interaction. Lower-bandwidth channels seem to be particularly problematic for interaction that involves negotiation or persuasion [17]. This in turn suggests that elicitation techniques that involve a back-and-forth negotiation may be more difficult to enact in a distributed setting. This exploratory study seeks to identify which requirements elicitation techniques work best in the distributed mode. This knowledge can lead to ideas about what would constitute a distributed requirements engineering process.

The literature suggests that there is motivation for desiring to perform distributed requirements engineering. Some studies have identified performance gains with a distributed work model. Research issues and questions arise which lead to the desire for experiment. Next the design of this empirical study is presented, followed by a presentation of observations and results.

3. Empirical Study of Distributed Requirements Engineering

This empirical study simulated a distributed requirements engineering project. Groups of computer science graduate students from Virginia Tech role-played as participants in a requirements engineering effort. All group interaction in the study was distributed, and was supported by a set of groupware tools that enabled both synchronous and asynchronous collaboration. The customer and engineer participants never met face-to-face to perform any negotiations or discussion related to the project. The intent was to observe the process and effectiveness of gathering and refining requirements in a distributed setting. This was an exploratory study in which we sought to learn as much as possible about techniques, strategies, and outcomes of distributed requirements engineering.

3.1 Project Overview

Students from a graduate Software Engineering course played the role of the software engineers. The engineers' customers were students from a second graduate class on Computer Supported Cooperative Work (CSCW). Each customer was given a role description that described his or her role in a hypothetical company (e.g., vice president, technical staff, administrative staff); each hypothetical company was represented by a group of 3-5 employees. These groups selected a name for their company and effectively acted on behalf of that company for the duration of the experiment. Each company had the same general description, "a multi-discipline engineering firm

with approximately 100 employees". Each firm was described as employing engineers, project managers, secretaries, accountants, and administrative personnel, which were housed in a modest 2-story office building.

Requirements engineering was supported using a set of collaborative tools. These tools are referred to as the groupware tools. The tools consisted of: Centra Symposium [13], which supports real time virtual meetings, MOOsburg [15], to facilitate file sharing and informal impromptu meetings, and email, for file sharing and asynchronous discussions.

The software system specified was a personal planner and scheduling system for the virtual company. The virtual company was described to have a fixed number of meeting rooms and audio/visual equipment in their two-story office building. The desire was to develop a software system to help schedule shared company resources as well to aid the personal planning of all company employees. Being a small company both manpower and physical resources are limited, and an automated scheduling system has been identified as a way to efficiently optimize finite resources to increase the productivity of the company. Additional details can be found in [16].

A meeting scheduler system has been established as a benchmark problem for research in requirements engineering. [9] There is an existing two-page requirements document describing the scheduling problem. [10] Determining and negotiating requirements for a meeting scheduling system is challenging and problematic. This problem has enough complexity to make it a useful sample problem, raising issues similar to real world problems. The problem also requires no special domain knowledge of persons playing the customer role. We could not assume our student/customers would have special knowledge of the requirements for any particular software system, so the scheduling system seemed to be an ideal problem for this empirical study.

We used Lamsweede's two-page requirements document describing the scheduling problem [10] as a basis for our more elaborated problem statement that included details about customer roles and specific requirements. Our extension of the original specification also requested that the teams develop a state-of-the-art technological solution. Customer roles were defined, including hypothetical conflicts between customers over system requirements (e.g., secretaries wanted to maintain control over scheduling versus engineers who wanted extensive automation, see [16]). The problem specification was given to the customer groups but kept from the software engineers for the duration of the experiment.

Each software engineering group conducted a distributed requirements analysis to develop and write a Software Requirements Specification (SRS) document that specified the requirements of the corporate scheduling system. A number of constraints were placed on the engineers to help structure the project.

Each requirements engineering group conducted a series of four planned virtual meetings with the customer groups. Each meeting was allowed to take up to ninety minutes. The software engineers and customers met using Centra Symposium, a point-to-point audio conferencing and meeting support tool [13]. (Video conferencing was not used in this study.) The software engineers needed to plan an effective agenda to make optimal use of the limited time in virtual meetings.

The requirements engineers were also allowed to conduct additional meetings using MOOsburg, a place-based collaborative environment [15]. MOOsburg does not support point-to-point audio conferencing, so these meetings were limited to text chat or document exchange. Group email distribution lists were also available to complement the four real-time meetings. Open issues or questions could be clarified by sending email using these group email aliases.

To assist the software engineers in requirements analysis, process guidelines were provided for planning the activities of each virtual meeting. The set of guidelines and requirements are shown in the table below. The list provided a benchmark so software engineers could understand the types of activities that should occur at different points within the requirements engineering lifecycle. A general requirements engineering process was also presented that could be used if desired during the project.

Virtual Meeting 1	Capturing User Requirements
Virtual Meeting 2	Analysis, Prototyping, Modeling
Virtual Meeting 3	Analysis, Productivity
Virtual Meeting 4	Walkthrough verification of the specific

Table 3-1 - Virtual Meeting Activities

3.2. Requirements Elicitation Techniques

One of the research goals of this study was to analyze how well requirements elicitation techniques work in a

distributed setting. We hypothesize that some requirements elicitation methods are better than others for distributed requirements engineering. Thus, the software engineering participants in the study were introduced to a number of popular requirements elicitation methods as part of their participant training for the experiment. This enabled us to study the use and effectiveness of the various techniques as part of the overall process.

The requirements engineers were not required to use any particular elicitation techniques for requirements engineering. They were trained on a variety of techniques and were then free to select techniques most appropriate for the situation. The participants' experience with requirements engineering and their use and reaction to individual techniques was then assessed by survey. A few of the techniques were particularly emphasized as part of the instruction in the software engineering course.

The following requirements elicitation techniques from [5] were presented to the participants and evaluated in the study:

- Question and Answer Method
- Customer Interviews
- Brainstorming and Idea Reduction
- Storyboards
- Prototyping
- Questionnaires
- Use Cases
- Requirements Management.

3.3. Meeting Facilities and Software

Since this empirical study studied distributed work, it was expected that participants might desire to work on the project from a variety of locations. MOOsburg was provided to support distributed work outside of planned meetings. Participants were able to use MOOsburg from any computer with an internet web browser having the proper plug in. MOOsburg was intended to support both planned and impromptu synchronous and asynchronous meetings throughout the project. Participants could log on to MOOsburg from their home computer as needed for collaborative work.

Use of the virtual meeting system Centra Symposium was restricted to on-campus computer labs during assigned meeting times. Although there were no technical issues preventing students from accessing the software from remote locations, they were required to use the assigned computer labs for all virtual meetings. This allowed for a controlled environment and for easy observation of the participants. The software engineers met in a large computer lab with approximately thirty networked computers on the third floor of the building. The customers met in a small computer lab that was located on the first floor of the same building. Observations were made from both sides throughout the experiment. Due to the physical layout of the building the customer and engineering participants would typically use different doors to enter and exit the building. This reduced the likelihood of them encountering each other.

4. Process Assessment and Results

In order to assess the effectiveness of distributed requirements engineering in this study a combination of survey and observational methods were used. After the software engineering groups had produced their software requirements specification (SRS) documents, a set of metrics was applied to assess document quality. Correlations between these quality measures and the survey and observational data were examined to investigate the effectiveness of the distributed requirements engineering processes used.

4.1. Evaluation Methods

The empirical study used surveys to determine the software and requirements engineering experience level of participants. Survey data and observations were also collected at each planned virtual meeting. Meeting sessions were recorded. At the conclusion of the project, participants completed an extensive online survey consisting of 89 questions. Each project group created a collaboration space in MOOsburg to share documents and to act as a virtual place for impromptu meetings. These group spaces were examined thoroughly and artifacts were archived. All email communication between customers and software engineers was monitored and the messages were examined (for specific details about evaluation methods mentioned above see [16]).

4.2. Assessing SRS Document Quality

Four different metrics were applied to evaluate the overall quality of the SRS documents. Each metric produced a positive decimal value from 0 to 1. An equally weighted average of all the metrics taken together produced a numeric result that reflects the overall quality

of the SRS documents. For extensive detail about the metrics used see [16].

- **SRS Document Grade**

  The course professor generated the SRS Document grade by combining the results from a qualitative student assessment and the professor's qualitative impression of the SRS document quality.

- **Measurement of Requirements Evolution**

  All requirements identified were classified as either Original (O), Original with Evolution (OE), Evolved (E), or Unclassifiable (U). Documents with the most requirements that were OE or E were considered to exhibit a high degree of requirements evolution. One challenge with conducting requirements elicitation is determining when all of the requirements have been identified [5]. It is expected that mature SRS documents should exhibit a higher quantity of evolved requirements. Herela states in [5] that requirements are a negotiated product generated through a collaborative requirements process. Having an elaborated, detailed and rich SRS document ("product") suggests that more rich negotiation ensued in the process producing the SRS document, than in groups, which produced lower quality documents.

- **Requirement Errors**

  Each Requirement in the SRS documents was identified and classified as either defect free, ambiguous, incomplete, inconsistent, not traceable, or not verifiable. The metric value was then calculated based on the percentage of defect free requirements compared to requirements with defects.

- **Original Requirements Supported**

  There were seventeen original requirements provided to the customers at the start of the study. This metric was based on the percentage of the original requirements supported in the final SRS document.

4.3. Factors Influencing SRS Quality

Based on the application of the metrics described above performance scores were calculated as percentages for each of the six groups in the study. The groups were then classified as either High Performance or Low Performance. Results were as follows:

High Performance Groups		**Reduced Performance Groups**	
Group 1	79.34%	Group 4	69.11 %
Group 2	77.32%	Group 6	66.85 %
Group 3	76.44%		
Group 5	75.96 %		

Table 4-1 - Group Performance Scores

Requirements engineers were asked about their perception of customer participation. For the survey a rating scale was used to rate the quality of customer participation. A weak positive trend was seen between ratings of customer participation and overall SRS quality. Requirements engineers were asked "What factors made this simulation seem unrealistic?" The two groups classified as reduced performers tended to complain more often (56% of the group members) about customers not participating enough in the requirements process, where groups classified as higher performers complained about customer participation much less (12% of the group members).

Informally, we observed that some of the software engineers became frustrated with the customers during the study. They expected the customers to explicitly provide the requirements to them in a ready-to-use form. This coincides with the naïve view that requirements are preexisting "immutable facts", and that the process of requirements analysis should be a one-way transfer of information from the customers to the engineers. [4]

Requirements engineering experience is an additional factor that one would expect to impact SRS quality. We discovered a rather weak but positive relationship between a group's average requirements engineering experience and the quality of their' SRS documents (r[df=4]=.56, NS). (Note that this study included only six groups, making the power of our group-based analyses rather weak. However, this study was exploratory in nature, so even though many group-based correlations are not statistically significant, we report them in the interest of stimulating further work.)

After the conclusion of each virtual meeting the software engineering team members rated the contribution and participation of their peers for both the virtual meeting and the project related work leading up to the meeting. A value was calculated for the average perceived peer participation per group (APPPG). This value described the overall impression of the participation that each team thought they were putting forth into the project. A weak

positive relationship was seen between overall SRS quality and APPPG (r[df=4]=.60, NS).

A marginally significant *negative* relationship was observed between requirements elicitation technique effectiveness (RETE) and overall SRS quality (r[df=4]=-.74, p<.09). RETE was the calculated average effectiveness score for all requirements elicitation methods. This correlation implies that groups who produced high quality documents tended to report that elicitation methods were ineffective in general. Closer investigation suggests that this relationship is largely due to perceived value of Prototyping and Questionnaires.

In Summary, in this distributed requirements engineering experiment, groups appeared to produce higher quality SRS documents if they reported having more experience with requirements engineering. Groups who perceived their teammates as contributing to the group seemed to produce higher quality SRS documents. Groups produced lower quality SRS documents if they used questionnaires as an elicitation technique (an effect explained further in section 4.4).

4.4. Effectiveness of Requirements Elicitation Techniques

One goal of this empirical study was to identify which requirements elicitation techniques are most effective when used in distributed requirements engineering. Through survey we found that the study participants had varying degrees of experience using requirements elicitation techniques. Ultimately the selection of techniques to use was influenced by technique experience and instruction in the course.

By the nature of their inherit characteristics, some elicitation techniques may be more well suited for use in a distributed setting, while others may function poorly in the distributed mode. We tabulated the average perceived effectiveness ratings for each requirements elicitation technique introduced to the software engineers. A five point scale was used, with 5 corresponding to "outstanding effectiveness" and 1 corresponding to "no effectiveness". The graph of results appears at the top of the next column.

As mentioned in the previous section there was a negative relationship between requirements elicitation technique effectiveness (RETE) and overall SRS quality (r[df=4]=-.74, p<.09). However, an investigation of the individual methods ratings reveals that only Prototyping (r[df=4]=-.70, p<.12) and Questionnaires (r[df=4]=-.56, ns) were negatively related to overall SRS quality.

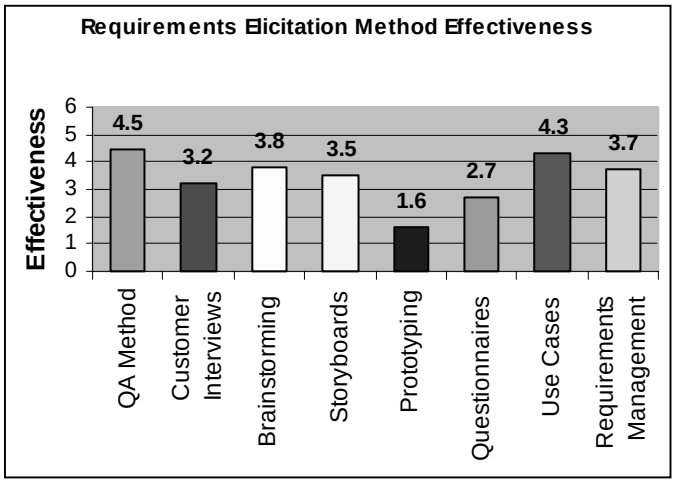

Figure 4-1 –Elicitation Method Effectiveness

The ratings regarding Prototyping are problematic because the requirements engineers in this study did not use this method. They used a lower-fidelity technique known as storyboarding. Because the software engineering participants did not actually build prototypes, it is not clear how to interpret their assessment of this elicitation technique's effectiveness (e.g., it may simply reflect that they chose not to use it).

In contrast, study participants did employ Questionnaires at times. Questionnaires are an asynchronous elicitation technique, for instance a written lists of questions given to end users to obtain information about system requirements. In this experiment such questions were typically delivered and answered by email, or posted to a group's collaboration space in MOOsburg. For the use of questionnaires there was a trend between the rated effectiveness of questionnaires and the level of customer participation outside of the scheduled meetings. The engineers tended to see questionnaires as a better elicitation technique when they felt that their customers had a high level of participation outside of the virtual meetings (r[df=24]=.30, p<.14). (Of course, this could simply reflect that customers were more likely to participate when questionnaires were used.)

As reported above, Questionnaires tended to be viewed as less effective for the groups with high quality SRS documents. This may mean that the lower-performing groups may have relied on asynchronous elicitation techniques more heavily. For example, there was a weak negative relationship between the degree to which email was used versus overall SRS quality (r[df=4]=-.64, p<.17).

These relationships between asynchronous techniques and overall SRS quality raise several interesting questions:

Was the interaction available in the planned virtual meetings insufficient for the lower-performing groups? Were email questionnaires and asynchronous work needed as a supplement to real-time communication? The data suggests that groups producing high quality SRS documents tended to avoid using email and asynchronous methods for requirements elicitation. Perhaps groups using the asynchronous techniques were doing so because they were failing to obtain the necessary information from the scheduled meetings (e.g., perhaps they prepared more poorly and obtained poorer quality results). As mentioned before, lower performance groups had more negative comments with respect to perceived customer participation.

In contrast, there is a positive correlation between the perceived effectiveness of the Question and Answer method and reported customer participation ($r[df=24]=.41$, $p < .05$). Question and Answer is an ad hoc elicitation technique where the engineers ask the customer/end user questions and the path of dialogue is then allowed to vary based on user feedback and participation. Software engineers' ratings of this technique were higher when they also reported active participation by the customers in the meetings.

In summary the Question and Answer method, Use Cases, Brainstorming, and Requirements Management were the reported as the most effective requirements elicitation techniques in the experiment. The impact of specific requirements elicitation techniques on overall SRS quality is inconclusive. There is some suggestion that synchronous collaboration in the requirements process in this study was possibly more effective than asynchronous collaboration.

5. Conclusions

The results from this study suggest that distributed requirements engineering is more effective when stakeholders, in our case customers, participate actively in synchronous activities of the requirements process. The richness of the synchronous collaboration in this study that was supported with voice conferencing and additional tools seems to have delivered higher informational bandwidth than the asynchronous tools email and MOOsburg.

Groups who obtained adequate requirements information from the planned virtual sessions had better success writing high quality SRS documents. Their processes captured greater numbers of the original requirements, exhibited more requirements evolution in the SRS document, produced fewer errors in the SRS, and consequently received better SRS grades.

It may be that the open-ended and loosely specified nature of the problem in this study favored the use of synchronous collaboration. If the initial fixed set of requirements had been more detailed, asynchronous communication might have been sufficient for the requirements gathering. But in the real world, when are software requirements fixed? What if the customers were not participating actively outside of virtual meetings simply because they were not required to? Attendance at planned meetings was mandatory but there was no strict enforcement of participation outside of meetings. Is this a similar problem with real projects? Busy customers may find it easy to ignore or postpone responding to requests for information from email or other sources.

6. Future Work

Further empirical studies such as reported here will serve to expose additional information about the problems and strategies of distributed requirements analysis. With only six groups, our observations and conclusions about document quality must be seen as relatively informal and speculative. Additional trials of this experiment are needed to lend strength to the exploratory findings reported here, and to identify which trends are truly representative of problems in distributed requirements engineering. Nonetheless, even at this early stage, we have identified a number of interesting trends and problems that can form the basis of more targeted research projects (e.g., the relative impact of synchronous and asynchronous elicitation methods on outcome).

Another obvious extension would be to contrast group distributed requirements teams with a control condition of co-located face-to-face meetings. This would lead to a better understanding of how the distributed setting influences the process and result. Perhaps a future study could conduct the entire requirements analysis just with co-located teams and then those results could be compared to the distributed team results obtained here.

The location of the groups is not the only interesting variable we could investigate in future studies. We could consider controlling the specific elicitation methods that groups were allowed to use. By fixing the methods we could have sharper contrasting views on the effectiveness of specific requirements elicitation methods. This study was an exploratory study where we presented an array of options and did not explicitly specify which elicitation methods groups were to use. We wanted the groups to have some freedom to choose their approach merely for

the sake of seeing what they would do, and how this would affect SRS quality in the end.

Our findings seem to suggest that asynchronous tools and methods may not be as effective at supporting distributed requirements engineering. What if we had a control group of engineers that only could perform requirements analysis with asynchronous techniques? Imagine an Asian software development firm conducting a requirements analysis for a product with the customer based in the United States. Whenever a customer's workday does not have overlapping hours with the engineers this is a possibility. In this situation asynchronous collaboration may be the norm with few opportunities for synchronous meetings.

Distributed requirements engineering, and distributed software engineering in general are large research areas for the future. With the increasing quality of communication and the decrease in communication cost it only makes sense that more distributed collaboration will be the norm in the future. We are already beginning to have trouble remembering a time when we didn't perform distributed work in software development. Continued research on groupware to support such distributed work will help better enable this future.

7. Acknowledgements

This work formed part of a Masters Thesis prepared by the first author while at Virginia Tech. We want to express our thanks to the Faculty Development Institute and New Media Center at Virginia Tech. They provided the computing facilities necessary to carry out the distributed meetings. Special thanks also to the Spring 2001 students of CS 5704 Software Engineering and CS5734 Computer Support Cooperative Work who volunteered to participate as subjects in the study.

8. References

[1] Brooks, F.P., "No Silver Bullet: Essence and Accidents of Software Engineering," IEEE Computer, 20, 4 (April 1987), 10-19.

[2] Higa, K. Understanding Relationships Among Teleworkers' E-Mail Usage, Email Richness Perceptions and E-Mail Productivity Perceptions Under a Software Engineering Environment. *IEEE Transactions on Engineering Management* 47, 2 (May 2000), 163-173.

[3] French, A. A Study of Communication and Cooperation in Distributed Software Project Teams. In *Proc. International Conference on Software Maintenance*. (1998), 146-154.

[4] Herela, D., Greenberg, S. Using a Groupware Space for Distributed Requirements Engineering. In *Proc. Seventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises WET ICE '98.* (1998), 57-62.

[5] Leffingwell, Dean and Don Widrig, *Managing Software Requirements: A Unified Approach*, Addison Wesley., Boston, MA, 2000.

[6] Davis, Alan M. *Software Requirements: Objects, Functions, and States.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[7] Hrones, J., Jedrey, B., Zaaf, Driss. Defining Global Requirements with Distributed QFD. *Digital Technical Journal* 5, 4 (Fall 1993), 36-46.

[8] Ocker, R., Hiltz, S.R., Turoff, M. Fjermestad, J., Computer Support for Distributed Asynchronous Software Design Teams: Experimental Results on Creativity and Quality. In *Proceedings of the 28th IEEE International Conference on System Sciences*. (1995), 4-13.

[9] Potts, C., Takahashi, K., Antón, A. Inquiry-Based Requirements Analysis. IEEE Software 11, 2 (March 1994), 21-32.

[10] van Lamsweerde A, Darimont R, and Massonet Ph. The Meeting Scheduler System: A Problem Statement. Available via ftp: //ftp.info.ucl.ac.be/pub/public/92/MeetingScheduler.ps

[11] Aoyama, M. Agile Software Process Model. In *Proceedings of the 21st IEEE International Computer Software and Applications Conference COMPSAC '97* (1997), 454-459.

[12] Gorton, I., Hawryszkiewycz, I., Ragoonaden, K., Chung, C., Lu, S., and Randhawa, G. Groupware Support Tools for Collaborative Software Engineering. In *Proceedings of the 30th IEEE International Conference on System Sciences.* (1997), 157-166.

[13] Centra, "Centra Products and Services – Centra Symposium" available from: http://www.centra.com/products/symposium/info.asp Internet accessed 10 October 2000.

[14] Karolak, D. *Global Software Development: Managing Virtual Teams and Environments.* IEEE Computer Society, Los Alamitos, CA, 1998.

[15] Carroll, J.M., Rosson, M.B., Isenhour, P.L., Van Metre, C., Schafer, W.A., & Ganoe, C.H. 2001. MOOsburg: Multi-user domain support for a community network. *Internet Research, 11*(1), 65-73.

[16] Lloyd, W. J., Tools and Techniques for Effective Distributed Requirements Engineering: An Empirical Study. Masters Thesis, Virginia Tech, 2001. Available at: http://scholar.lib.vt.edu/theses/available/etd-07262001-110924/

[17] Dennis, A.R., Valacich, J.S. 1999. Beyond media richness: Towards a theory of media synchronicity. In Proceedings of HICSS'32, New York, IEEE.