An Empirical Study of the Impact of Asynchronous Discussions on Remote Synchronous Requirements Meetings

Daniela Damian[1], Filippo Lanubile[2], and Teresa Mallardo[2]

[1] University of Victoria, Computer Engineering Department,
Victoria, BC, Canada
danielad@cs.uvic.ca
[2] University of Bari, Dipartimento di Informatica,
Bari, Italy
{lanubile, mallardo}@di.uniba.it

Abstract. Our research explores the combination of synchronous and asynchronous collaboration tools for global software development. In this paper we assess the impact of tool-mediated inspections to improve requirements negotiation meetings with stakeholders spread over different continents. We present the design of our investigation in an educational environment, in a course where the clients and developers in a software project were in geographically distributed locations. In particular, we studied the usefulness of asynchronous discussions in IBIS tool in enabling more effective requirements negotiations meetings. Our findings indicate that the requirements negotiations were more effective when the groups conducted asynchronous discussions prior to the synchronous negotiation meetings.

1 Introduction

Technical reviews and in particular software inspections and client reviews are considered among the most important software quality assurance techniques in software engineering. The software inspection process was first introduced by Michael Fagan at IBM [Fag76] with the main goal to find defects before testing starts and to reduce rework effort. Although its application was initially limited to code, as a complement of testing techniques, software inspections have been also applied to early life-cycle software artifacts [Lai00] because detecting defects close to their point of creation reduces rework [Boe81]. As requirements defects are the most expensive to correct if they are not detected soon, many researchers have subsequently conducted empirical studies of software inspection on requirements documents [Bas96, Bas99, Bif03, Lai02, Lan98, Por95, Sch92, The03]. Experiences from these studies indicate that inspecting requirements documents, other than producing information for correcting the document, leads to a better understanding of the real problems, increases confidence in the acquired knowledge, and improves communication among stakeholders.

The focus of our research is in this area of collaborative software development and in particular on processes that support stakeholders to collaboratively develop a shared understanding of the required software functionality. We regard requirements

L. Baresi and R. Heckel (Eds.): FASE 2006, LNCS 3922, pp. 155 – 169, 2006.

inspections and client reviews as powerful mechanisms not only for checking the requirements documentation for qualities such as completeness and correctness, but also for validating that stakeholders share same understanding of the requirements. Requirements inspections create the opportunity for identifying areas in which designers and customers need to further discuss and negotiate requirements issues.

Over the last years however, the dramatic trend towards developing software in geographically distributed settings has challenged the communication and collaboration processes in software teams [Dam03, Her03], and the development of tools and methodologies to support a combination of synchronous and asynchronous activities in distributed teams emerges as critical. In particular, it becomes important to research approaches that enable effective requirements inspections and negotiations in distributed software development, as they are activities that should support collaborative software engineering in remote teams as well as they do in traditional software teams. While research in requirements inspections and negotiations [e.g. Boe98] is being complemented by studies of inspections for validation of requirements negotiation models [Grü04, Hal03], there is little research into enabling effective negotiations that follow the identification of requirements issues during the inspections. These negotiations, as examples of requirements meetings that involve relevant project stakeholders, are traditionally difficult and expensive to coordinate, especially in geographically distributed teams.

In this paper we describe our research and early results of studying the usefulness of asynchronous discussions, as part of the requirements inspection process, to facilitate more effective synchronous requirements meetings in distributed teams. In particular, we studied the use of a web-based inspection tool, IBIS [Lan03], in support of the remote communication between clients and developers collaboratively developing a requirements specification.

IBIS supports remote teams during the inspection of requirements documents, and in particular supports teams through stages of issue Discovery, Collection, and Discrimination. During the Discovery stage, inspectors review individually the document with the help of checklists or scenarios, and records issues. In the Collection stage the inspection leader or the document's author collate recorded issues and eliminate duplicates. In the Discrimination stage the inspection team makes decisions about collated issues. The Discrimination stage is designed as a structured asynchronous discussion with two mechanisms: posting of messages for each issue under discussion and voting as to whether an issue is a true issue or not (false positives). In [Lan04], we investigated IBIS support to remote inspection teams and found that asynchronous discussions in the Discrimination stage were as effective as co-located inspection meetings at discriminating between false positives and true issues.

Our findings indicate positive impact on the effectiveness of such requirements meetings in resolving open issues when preceded by asynchronous discussions in IBIS.

The paper is structured as follows: Section 2 describes our research design, by introducing the educational environment as the context in which we conducted an empirical study of asynchronous discussions in support of synchronous requirements meetings. Section 3 then reports our early results of how IBIS was used and how we assessed the effectiveness of requirements meetings when preceded by the asynchronous meetings. We then discuss possible limitations and threats to validity and our plans for future research.

2 Research Design

To investigate the usefulness of asynchronous discussions to facilitate effective synchronous requirements negotiation meetings, we studied tool-supported remote inspections in six educational global project teams in a global software development (GSD) course. Each software project followed an iterative development process in which designers in collaboration with clients were to develop a requirements specification (RS): after a requirements elicitation stage, a requirements inspection of an early draft of RS involved the discovery as well as asynchronous discussion of requirements issues and was further followed by requirements negotiations and prototype demonstrations before the final draft of the RS was delivered. In this section we describe the research setting: the software development course, the use of IBIS and our research design that compared the effectiveness of the requirements negotiations when preceded by the asynchronous discussions in IBIS to those negotiations with no prior asynchronous discussions.

2.1 The GSD Course: Students, Groups and Remote Collaboration

The Global Software Development course was offered in a three University collaboration involving University of Victoria, Canada, University of Technology, Sydney, Australia, and University of Bari, Italy during January and May of 2005[1]. The course involved a total of 32 students. 12 of them were Master's and Doctorate students at the University of Victoria, 2 graduate and 8 undergraduate students at the University of Technology, Sydney, and 10 Master's students at the University of Bari.

As shown in **Table 1**, the Canadian students worked on software projects with the Australian and Italian groups as follows: the 12 Canadian students formed three groups of 4 (Gr1-3), the Australian students formed two groups of 5 (Gr4-5), and the Italian students formed two groups, of 3 and 7 students respectively (Gr6cl and Gr6dev). Each Canadian and Australian group was involved in two different projects, playing the role of client (C) and developer (D) respectively. Each of the two Italian groups was involved in only one project, either as a client (Gr6cl) or as a developer (Gr6dev).

Table 1. Project teams (PT) and their allocation to course projects

Country	Group	Project A (A1, A2)		Project B (B1, B2)		Project C (C1, C2)	
		PT1	PT2	PT3	PT4	PT5	PT6
Ca	Gr1	Client (C)					D
	Gr2		D	C			
	Gr3				D	C	
Au	Gr4	Developer (D)			C		
	Gr5		C			D	
It	Gr6cl						C
	Gr6dev			D			

[1] More information can be found on the course website: http://segal.cs.uvic.ca/csc576b

2.2 The Software Projects

There were three distinct projects in the course (A, B and C). Two global software project teams were allocated to each project, each with the client and developer group in two different countries (see **Table 1**). The project topics are briefly described in the following:

Project A (A1 and A2 in Table 1): Global software development system. A system to facilitate collaboration in GSD by supporting informal communication as well as document exchange in remote teams. Tasks supported by the tool included: displaying people's availability information, viewing changes between different versions of documents and authors of those changes, visualizing the evolution history of a particular document, and discovering who has been working on a particular document or section of a document.

Project B (B1 and B2 in Table 1): iMedia system. A "iMedia" software that will allow users to purchase movies online, organize their movie library, and play movies. One of the key requirements was that the interface be simple to use even for inexperienced computer users, without sacrificing key features.

Project C (C1 and C2 in Table 1): Virtual Realty system. A system that provides accurate and easy-to-find information to real estate agents and home buyers in the Victoria area. The system had to display an interactive map, where the end-user can zoom in, zoom out, pan, etc., and click on it to get the information of the property.

The projects were assigned to groups before group membership was determined. The project assignment was done so that each group worked with a different partner group for each of the two projects it was assigned (with the partner group always located in a different country), and so that the two projects it worked on were on a different topic.

2.3 An Iterative Process to Develop Requirements Specifications

Each project followed an iterative process of developing a requirements specification (RS) through collaboration between developers and clients over a period of 7 weeks. The RS development life-cycle (illustrated in **Fig. 1**) consisted of six phases of requirements discovery and validation, and through which the understanding and documentation of requirements was to be improved. Each of which stages included either client, developer or group tasks and ended with a project deliverable on which students were graded for the class. The final deliverable was the final version of the SRS, which reflected the shared understanding of the project that the clients and the developers built over the previous four phases. The project finished at the point where the developer group would start writing the code for the system called for by the project.

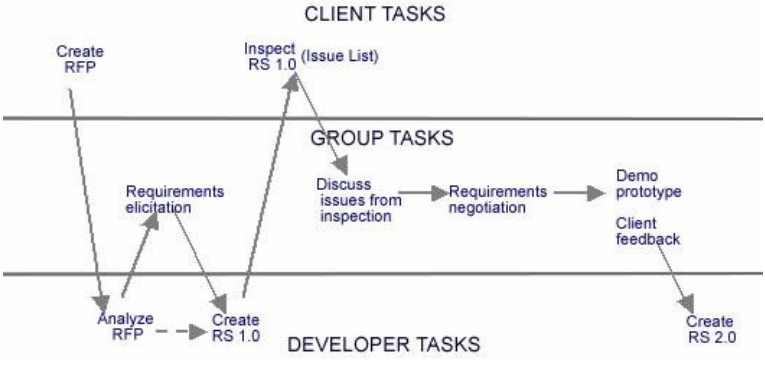


Fig. 1. An iterative process to develop the requirements specification (RS)

For each project, these six project phases consisted of:

- *Request for proposals.* Starting with the assigned project topic, the client group created a "Request for Proposals" document (RFP) which invited developers to propose their solutions to the clients' needs.
- *Requirements elicitation.* In response to the RFP, the developer group assigned to the project had three days to analyze it and come up with a list of clarifications that they needed from the clients before proceeding. The developers and the clients then held a scheduled one-hour requirements elicitation videoconference, during which the developers clarified the clients' needs and elicited more requirements. A week after the requirements elicitation meeting, the developers delivered an initial "Requirements Specification" document (RS 1.0). This document described in detail the features and scope of the project and followed the IEEE standard for requirement specification.
- *Meetingless inspection of RS 1.0 and asynchronous discussions of requirements issues in IBIS.* Upon receiving the RS 1.0 document, the clients had a week to carry on an inspection in order to identify gaps in understanding of requirements. This inspection was entirely performed online through the use of IBIS tool. With the designer team considered the authors of RS1.0, the inspection was carried out by the client team. Each member of the client team, individually, participated in the Discovery stage and read the RS 1.0 available in the system and recorded issues. The issue information contained a description of the issue found, as well as a number of issue attributes such as type severity. A course assistant collected all issues and merged duplicate issues, found by more than one client, into a unique list of collated issues. This discovery of issues was followed by a four-day asynchronous discussion. The entire project team, clients and developers, participated in this discussion using IBIS (i.e. in the Discrimination stage). The purpose of the asynchronous discussion was to come to an understanding of each issue and those issues that could be closed online (i.e. where resolution could be reached without further negotiation) or remained open issues (anything else, and which had to be further negotiated in real-time discussion). Discussants attempted to close issues by using the two mechanisms in IBIS: posting messages with respect to a certain issue, and voting as to whether it is still an open issue or is resolved and thus could be closed.

- *Requirements negotiation.* Those issues that could not be resolved during the asynchronous discussion in IBIS (i.e. open issues) were then discussed during a scheduled requirements negotiation held in a videoconference meeting between developers and clients.
- *Prototype demo.* After the requirements negotiation meetings, the developer group had one week to develop a prototype of the system to reflect the results of the negotiation. This prototype did not have to contain working code, but could consist of storyboards and paper or computer-based mockups. The purpose of the prototype was to express the developers' understanding of the project and their clients' needs, which was done through a one-hour teleconference demo. The clients could give their feedback to the developers and thus reach a consensus on the project between the two groups.
- *Create final Requirements Specification (RS 2.0).* Finally, three weeks after the prototype demos, the developers submitted a final version of the Requirements Specification (RS 2.0). This version incorporated the clients' feedback collected since the first RS draft was written, that is, through the requirements negotiation and prototype demo.

2.4 Exploring the Usefulness of Asynchronous Discussions to Facilitate Effective Synchronous Requirements Negotiations

To assess the impact of asynchronous discussions on synchronous negotiations meetings, we traced the number of open issues through the stages of each of the six projects. In particular, we studied the usefulness of asynchronous discussions prior to requirements negotiations by investigating the teams' ability to close some of the issues prior to the negotiation and focus the discussion on the issues that could not be resolved during the asynchronous discussion.

To this end, we instructed half the projects to conduct the asynchronous discussion before the negotiation, and half the projects to jump into the negotiation without asynchronous discussion. **Table 2** indicates which projects conducted the Asynchronous discussion (AD) and which did not (No AD). Then, the process variant (AD or No AD) was the main independent variable that we manipulated for experimental purposes.

When asynchronous discussions were scheduled for a project team, both clients and developers used the IBIS tool over a week, as a threaded discussion forum. The aim was to come to an understanding of each issue by exchanging messages and to an early resolution through a common agreement expressed by voting. Those open issues that could not be closed during asynchronous discussion in IBIS were then left for the synchronous negotiation meeting. For those project teams which skipped the asynchronous discussion, all collated issues were thus considered as open issues to be dealt at the negotiation.

To measure the usefulness of asynchronous discussions, we defined the following dependent variables:

- *Collated issues* = the number of open issues at the end of the inspection carried out by the client groups.

- *Closed issues during asynchronous discussion* = the number of issues for which a consensus was reached between developers and clients during the asynchronous discussion. A closed issue did not require any further discussion.
- *Open issues before sync negotiation* = the number of open issues carried over to the synchronous negotiation meeting. If there was no asynchronous discussion, open issues equate to collated issues.
- *Closed issues during sync negotiation* = the number of issues for which an agreement was reached at the videoconference requirements negotiation meeting between developers and clients.
- *Open issues after sync negotiation* = the number of issues for which an agreement has not been reached at the teleconference requirements negotiation meeting.

More specifically, to understand the impact of asynchronous discussions on the synchronous negotiations, we were interested in the variation across projects of the number of open issues resolved during the asynchronous discussions, as well as during the synchronous negotiation. To complement the quantitative data, we gathered the students' perceptions on the usefulness of the AD. In this paper we report the students' degree of agreement, based on a 4-point rating scale, to the following statements:

- "Asynchronous discrimination is useful as a preparation to the requirements negotiation meeting."
- "Reading and posting messages is effective to clear up issues."
- "Reading and posting messages is effective to develop consensus on issues."
- "Voting is effective to develop consensus on issues."

Table 2. Experimental design

project team (client/developer)	process variant
A1 (gr1/gr4)	No AD
B1 (gr2/gr6dev)	No AD
C1 (gr3/gr5)	No AD
A2 (gr5/gr2)	AD
B2 (gr4/gr3)	AD
C2 (gr6cl/gr1)	AD

3 Early Results

Here, we present the results from a preliminary analysis of the quantitative and qualitative data we collected from the IBIS database and questionnaires given to project members. We present the values on the variables we collected as well as discuss the participant's feedback with respect the usefulness of IBIS to facilitate more effective negotiations.

3.1 Effectiveness of Requirements Negotiation Meetings in Resolving Open Issues

When observed at the project level, the data sample size is too small to lend itself to statistical analysis for measuring effectiveness. Instead, we report in **Table 3** the values of the dependent variables for each of the six projects, and discuss the traces of open issues at each stage in the collaborative process as an indication of effectiveness of asynchronous discussions.

The three projects which did not conduct any asynchronous discussions (A1, B1 and C1) entered the synchronous negotiation with different numbers of open issues to be resolved: 40, 61, and 100 respectively. The number of issues that were closed during the remote meetings ranged from 26 to 47, leaving from 12 to 74 issues unresolved. At the same time, an important difference can be seen in the three projects which conducted asynchronous discussions (A2, B2 and C2); these groups entered the remote negotiation meetings with a much lower number of open issues (12, 13, 12, respectively), leading in two cases to fully resolving open issues in the meeting agenda (remained open issues 0,3 and 0 respectively). The last column in **Table 3** also shows the significant difference in the percentages of open issues after the negotiation in the projects which conducted asynchronous discussions as compared to those which did not.

Table 3. Resolution of issues from inspection to negotiation meeting

Project (cl/dev)	Collated issues	Closed issues during async discussion	Open issues before sync negotiation and percentage out of collated issues	Closed issues during sync negotiation and percentage out of collated issues	Open issues after sync negotiation	Percentage of open issues after negotiation out of open issues before negotiation
A1 (gr1/gr4)	40	No AD	40 (100.0%)	28 (70.0%)	12	30.0%
B1 (gr2/ gr6dev)	61	No AD	61 (100.0%)	47 (77.0%)	14	23.0%
C1 (gr3/gr5)	100	No AD	100 (100.0%)	26 (26.0%)	74	74.0%
A2 (gr5/gr2)	23	11	12 (52.2%)	12 (52.2%)	0	0.0%
B2 (gr4/gr3)	112	100	12 (10.7%)	9 (8.0%)	3	2.5%
C2 (gr6cl/ gr1)	23	10	13 (56.5%)	13 (56.5%)	0	0.0%

Similarly, **Fig. 2** graphically illustrates the trajectory of open issues throughout the three stages in each of the six projects, as an indication of how asynchronous discussions improved the effectiveness of remote requirements negotiations. It can be seen that all three dotted lines, representing projects with AD, finished below the three continuous lines (which correspond to projects with no AD). Particularly important is

project B2 which started with the highest number of collated issues (112) but which ends with a significantly lower number of open issues (i.e. 3), thanks to the asynchronous discussion.

To provide more insights into what actually happened during these asynchronous discussions we report in **Table 4** the intensity of message exchanging and voting, the two basic mechanisms which could be used to resolve issues before the negotiation meeting. It can be seen how participants in project B2, although with the highest number of collated issues, were nevertheless active in discussing issues (282 posted messages) and extensively exploited the voting feature provided by the tool (910 votes). We hypothesize that the intensity of the discussion made it possible to drastically reduce the number of open issues (12 left unresolved, that is 10.7%).

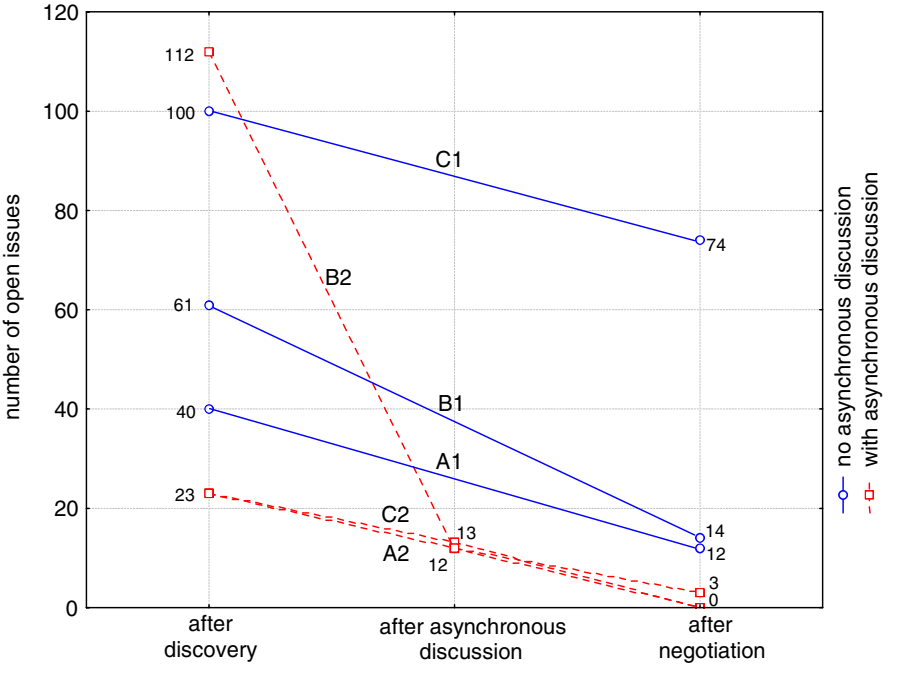


Fig. 2. Open issues at the end of three process stages

Table 4. Intensity of the asynchronous discussions

project team (cl/dev)	collated issues	participants	posted messages	messages per issue	messages per participant	votes	votes per issue	votes per participant
A2 (gr5/gr2)	23	9	131	5.7	14.6	128	5.6	14.2
B2 (gr4/gr3)	112	9	282	2.5	31.3	910	8.1	101.1
C2 (gr6cl/gr1)	23	11	72	3.1	6.5	236	10.3	21.4

3.2 Subjects' Perception

Here we present the results of the subjective evaluation of effectiveness of the asynchronous discussions. Although survey questionnaires related to the entire project experience were proposed to all students, we only considered the answers from the Canadian and Australian students because they were involved in both process variants (i.e. with and without asynchronous discussion). We analyzed answers from the Australian students (8 out of 10 responses) who experienced the asynchronous discussion as clients/reviewers and the lack of it as developers/authors. Conversely, we analyzed answers from the Canadian students (11 out of 12 responses) who experienced the asynchronous discussion as developers/authors and the lack of it as clients/reviewers.

As shown in **Fig. 3**, the great majority of students (both as clients and as developers) considered asynchronous discussion useful as a preparation to the requirements negotiation meeting. Developers, who were the authors of the requirements document under inspection, seem more enthusiastic than clients, who acted as reviewers during inspection. We believe this is due to the early feedback that developers gained as a result of the asynchronous discussion.

This is corroborated by some answers that students specified in form of further comment to the question:

"*The asynchronous discussion provided individuals the opportunity to discuss each other's issues and concerns and provide their understanding/comments on the situation in attempts for greater understanding and clarity. It helped reinforce individuals understanding of our requirements and how they work in the overall system. In this sense it helps to filter a lot of thought-to-be issues which would set the questions and agenda for the negotiation meeting*".

"*The asynchronous discussion served as an excellent platform to not only layout the issues, but also to narrow down the number of issues to be addressed in the negotiation meetings*".

Fig. 4 and **Fig. 5** show that the great majority of students appreciated to read and post messages in order to clear up issues and develop a consensus on them. Comments that provided motivation for the broad appreciation for forum-style message exchanging include:

"*Reading and posting messages during the asynchronous discussion was very effective. The question/answer style method allowed individuals to view other's interpretation of the requirements and the issues they perceive. As such this clarified much misunderstandings*".

"*By creating a written source for the asynchronous discussion, we provided the framework for the refined SRS. I appreciated the opportunity to document the issues and how they are resolved*".

However, there were also some comments highlighting limitations of asynchronous discussions:

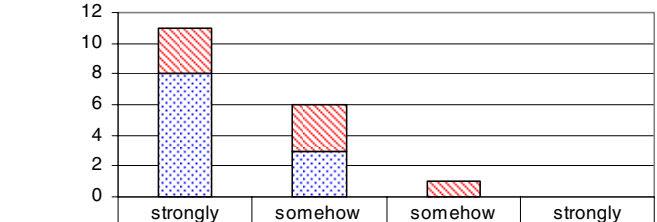
"*The messages were an effective vehicle to let both sides know where there were questions and potential disagreements. For relatively "easy" issues, it was a very effective forum. For those issue that were more involved, having many, many lengthy messages is perhaps not the best way to resolve them*".

"*This helped to understand different points of view. However, because of the time difference it took a long time to get feedback. In addition, because each item needed to be checked there was a huge amount of time spent reviewing the discussion each day*".

The other mechanism of the asynchronous discussions, voting, did not gain the same undisputed consensus as message exchanging. **Fig. 6** shows that half of the students did not acknowledge the effectiveness of voting for quickly expressing the opinion about open issues.

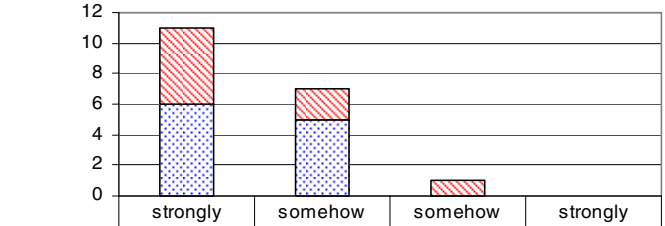
In general, students pointed out that they were asked to vote about an issue before exchanging messages. They would have preferred to vote after reading and posting messages about an issue:

"*The asynchronous discussion was useful to get everybody's opinion on the issues and to early filter out issues that all or none agreed upon. However I feel that because of time issues that we didn't get to do this stage in a properly. To have a real discussion people need to enter IBIS several times during the stage and in the setting of this project I do not know if this was done. Another problem would be that people could vote before hearing what people had to say about the issue*".



	strongly agree	somehow agree	somehow disagree	strongly disagree
client	3	3	1	0
developer	8	3	0	0

Fig. 3. "Asynchronous discussion is useful as a preparation to the requirements negotiation meeting"



	strongly agree	somehow agree	somehow disagree	strongly disagree
client	5	2	1	0
developer	6	5	0	0

Fig. 4. "Reading and posting messages is effective to clear up issues"

	strongly agree	somehow agree	somehow disagree	strongly disagree
client	1	6	1	0
developer	2	7	2	0

Fig. 5. "Reading and posting messages is effective to develop consensus on issues"

	strongly agree	somehow agree	somehow disagree	strongly disagree
client	0	4	3	1
developer	3	3	3	2

Fig. 6. "Voting is effective to develop consensus on issues"

4 Conclusions

Research advances in the area of tools and processes to support geographically distributed teams is important as developing software in global teams is becoming a fact of life nowadays. Further, empirical studies of tools and processes that support sound and proven software engineering techniques inform global teams that want to adopt practices that increase their ability to deliver high quality software. In this paper we described our research in supporting collaborative practices in geographically distributed teams, and in particular computer-mediated requirements validation techniques such as software inspections and negotiation meetings. Our strong motivation was that synchronous requirements meetings that are usually conducted during or after requirements inspections are difficult and expensive to carry out. Thus we are interested in ways in which to improve the effectiveness of such synchronous requirements meetings by allowing remote teams to discuss issues identified during the inspection in a forum of structured asynchronous discussions.

Our study has investigated the usefulness of computer-mediated, asynchronous discussions that followed requirements inspections in IBIS inspection tool, in facilitating more effective requirements negotiations that are often needed to resolve issues

from such inspections. Our findings from the observation of use of IBIS in an educational environment indicate that the teams who conducted asynchronous discussions were able to close many more issues before entering the requirements negotiation meetings. Further, the percentage of issues resolved during the synchronous meetings was much higher in those projects where the members did participate in asynchronous discussions. While these are promising research findings, we are planning a more in-depth analysis of the data we collected in this study. Directions for future research are described below, after we discuss the threats to validity in this research.

4.1 Threats to Validity

For this empirical study, we believe the following threats to internal validity that were beyond the researchers' control would make it uncertain to build cause-effect relationships between independent and dependent variables.

- *Instrumentation effects.* Instrumentation deals with the problem that differences in the results may be caused by differences in experimental material. Because in this study there were three different project topics, we cannot exclude that the topic and project complexity could have been a confounding factor.
- *Selection effects.* Results can be caused by variations in human performance. Usually, assigning subjects randomly to tasks controls this threat. In our case, selection of the participants was restricted by the practical course. For example, while Australian and Canadian students were exposed to both levels of the main independent variable, although with different roles (clients or developers), Italian students were not able to work on two projects and had the chance to choose the experimental treatment. Thus, we were not able to completely randomize the selection and participants' assignment to the different groups.

In the following we also list the most important threats to external validity, which limit the generalization of these findings to the industrial practice of distributed software development.

- *Representative subjects.* Since we involved students both as clients and as developers, they may not be representative of the population of professional stakeholders. This threat is partially mitigated by the presence of Canadian students, who were attending a specific course on global software development and then were trained on meeting protocols and negotiation techniques for requirements engineering. Some students had also previous working experience in the software business.
- *Representative artifacts.* The requirements documents inspected in this study may not be representative of industrial requirements documents. Our documents were requirements specifications for web applications while inspections are often conducted for dependable systems where quality and rework costs are perceived as critical.

4.2 Future Research

A number of important directions for furthering this research emerge as these early results indicate that the asynchronous discussions were beneficial in enabling more effective requirements negotiation meetings. To gain a more in depth understanding

of ways in which structured asynchronous discussions can support remote teams resolve open issues prior to negotiations, we are analyzing the broader context in which this causal relationship was observed. In particular, analyzing the type of issues identified during the inspections of the six teams, the complexity of those closed during the asynchronous discussion as well as negotiation meetings behavior and process will enable us to understand which factors in the computer-mediated collaborative process contributed to these results. We are conducting the analysis of the data stored in IBIS database and videotapes of the six project requirements negotiations. We hope to draw more detailed guidelines on conducting structured asynchronous discussions in support of expensive but important synchronous requirements negotiations.

Acknowledgments

We gratefully acknowledge the technical support of Luis Izquierdo and Fabio Calefato during the duration of the three-University course. Thanks also to Dr. Ban Al-Ani for her instrumental role in this collaboration and to all the students who participated to the remote projects.

References

[Bas96] V. R. Basili, S. Green , O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, and M. Zelkowitz. The empirical investigation of Perspective-Based Reading. Empirical Software Engineering: An International Journal, 1 (2): 133-164, 1996.

[Bas99] V. Basili, F. Shull, and F. Lanubile. Building Knowledge through Families of Experiments. IEEE Transactions on Software Engineering, 25(4): 456-473, July/August 1999.

[Bif03] S. Biffl, and M. Halling. Investigating the Defect Detection Effectiveness and Cost Benefit of Nominal Inspection Teams. IEEE Transactions on Software Engineering, 29 (5): 385-397, May 2003.

[Boe81] B. W. Boehm. Software Engineering Economics. Prentice Hall, Englewood Cliffs: NJ, 1981.

[Boe98] B.W. Boehm , A. Egyed, J. Kwan, D. Port, A. Shah, R. Madachy. Using the WinWin Spiral Model: A Case Study. Computer, 31(7): 33-44, July 1998.

[Dam03] D. Damian, and D. Zowghi. Requirements Engineering challenges in multi-site software development organizations. Requirements Engineering Journal, 8: 149-160, 2003.

[Fag76] M.E. Fagan, Design and code inspection to reduce errors in program development. IBM Systems Journal, 15(3), 1976, 182-211.

[Gen01] M. van Genuchten, C. van Dijk, H. Scholten, and D. Vogel, Using Group Support Systems for Software Inspections, IEEE Software, 18(3) : 60-65, 2001.

[Grü04] P. Grünbacher, M. Halling, S. Biffl, H. Kitapci, and B.W. Boehm. Integrating Collaborative Processes and Quality Assurance Techniques: Experiences from Requirements Negotiation. Journal of Management Information Systems, 20(4): 9-29, 2004.

[Hal03] M. Halling, S. Biffl, and P. Grünbacher. An economic approach for improving requirements negotiation models with inspection. Requirements Engineering Journal, 8: 236-247, 2003.

[Her03] J.D. Herbsleb, A. Mockus. An Empirical Study of Speed and Communication in Globally-Distributed Software Development. IEEE Transactions on Software Engineering, 29(3): 1-14, 2003.

[Lai00] O. Laitenberger, and J.M. DeBaud. An encompassing life cycle centric survey of software inspection. The Journal of Systems and Software, 50 (1): 5-31, January 2000.

[Lai02] O. Laitenberger, T. Beil, and T. Schwinn. An Industrial Case Study to Examine a Non-Traditional Inspection Implementation for Requirements Specifications. Empirical Software Engineering, 7(4): 345-374, December 2002.

[Lan98] F. Lanubile, F. Shull, V. Basili, "Experimenting with error abstraction in requirements documents", 5th Int. Symposium on Software Metrics (METRICS '98), pp.114-121, 1998.

[Lan03] F. Lanubile, T. Mallardo, and F. Calefato. Tool Support for Geographically Dispersed Inspection Teams. Software Process: Improvement and Practice, 8(4): 217-231, October/December 2003.

[Lan04] F. Lanubile, and T. Mallardo. A Preliminary Study On Asynchronous Discussions For Distributed Software Inspections. Proc. of the Workshop on Cooperative Support for Distributed Software Engineering Processes (CSSE 2004), September 2004.

[Per02] D.E. Perry, A.A. Porter, M.W. Wade, L.G. Votta, and J. Perpich. Reducing Inspection Interval in Large-Scale Software Development. IEEE Transactions on Software Engineering, 28(7): 695-705, 2002

[Por95] A.A. Porter, L.G. Votta, V.R. Basili. Comparing detection methods for software requirements inspections: A replicated experiment. IEEE Transactions on Software Engineering, 21(6): 563-575, 1995.

[Sch92] G. M. Schneider, J. Martin, and W. Tsai. An experimental study of fault detection in user requirements documents. ACM Transactions on Software Engineering and Methodology, 1 (2): 188-204, April 1992.

[The03] T. Thelin, P. Runeson, and C. Wohlin. An Experimental Comparison of Usage-Based and Checklist-Based Reading. IEEE Transactions on Software Engineering, 29(8): 687-704, August 2003.