



PACIFIC NW
28TH ANNUAL
SOFTWARE
QUALITY
CONFERENCE

OCTOBER 18TH – 19TH, 2010



ACHIEVING
QUALITY
IN A COMPLEX
ENVIRONMENT

*Conference Paper Excerpt
from the*
CONFERENCE
PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Lessons Learned About Distributed Software Team Collaboration

Raleigh Ledet
Apple Inc.
ledet@apple.com

Kal Toth
Portland State University
ktoth@cs.pdx.edu

Abstract

This experience report summarizes the conduct and outcomes of a practicum project [1] completed under the auspices of the Oregon Master of Software Engineering (OMSE) Program at Portland State University completed during the winter and spring of 2010.

Software engineers and learners are increasingly working in teams that are widely dispersed, often globally. Operating in distributed teams significantly increases the complexity of already complex software engineering tasks. Challenges amplified because of team distribution include resolving time-zone and cultural differences, addressing the need to become familiar with the tools and the human processes being used, and dealing with conflict within and across teams. This paper describes a practicum project conducted in a hybrid learning mode that studied one particular aspect - namely, the problems and properties of distributed collaboration processes used by software engineering teams.

The overall goal of this practicum project was to provide advice and guidance to distributed software teams, and to offer suggestions for further work in this field. The primary objective was to identify, specify and evaluate selected software engineering processes adapted to support collaborative software teams. The project was aimed explicitly at processes over tools - the software tools being the means rather than the ends for learning about such team collaboration. The OMSE students conducting this practicum project leveraged their industry experience and the software engineering principles and processes they had studied in the OMSE program. They progressively evolved a practical framework for defining and evaluating a specific core group of distributed software collaboration processes.

The ultimate outcome of this practicum project was that the students learned a number of critical lessons about how various distributed team processes and tools fit with the challenges of collaborating in teams. The students also confirmed that when working in a distributed team, be that as a practicing software engineer or part of a group learning exercise, the types of processes and tools that they use day-to-day are virtually the same. The practicum project described in this paper is an example of how software engineering practice informs learning as well as how software engineering education informs practice.

Biographies

Raleigh Ledet is a software engineer at Apple, Inc., operates a small shareware company called Mage Software, and holds a Master of Software Engineering degree from Portland State University. Previously he was with Wacom Technology Corp. of Vancouver, WA and DOGPAW, a non-profit organization.

Kal Toth is the Executive Director of the Oregon Master of Software Engineering Program (OMSE) and Associate Professor in the CS Department at PSU. He holds a Ph.D. from Carleton University in Canada and has worked for Hughes Aircraft, CGI Group, Intellitech, and Datalink Systems Corp.

1. Introduction

This experience report of the winter/spring 2010 practicum class [1] was performed under the auspices of the Oregon Master of Software Engineering (OMSE) Program at Portland State University. Offering traditional in-class software engineering courses since 1998, OMSE committed to offer all courses online in 2006. All OMSE courses are now available in either online or hybrid (face-to-face + online) delivery modes. However, some improvements are still needed.

Recognizing the common challenges of online learning and distributed team collaboration, several distributed team projects have been proposed to students for their (two-term) practicum projects over the last two years. OMSE practicum students have enthusiastically explored this area of “e-collaboration” [6], [7], [8] because they encounter such problems all the time on the job when working in dispersed teams. Some OMSE students have also published and presented papers related directly to the topic of distributed teams at PNSQC 2009 [4] and [5]. OMSE adjunct professor Milhauser in [3] explored several fundamental issues about distributed team collaboration some of which were addressed by this year’s practicum teams. And Hines in [2] discusses the problem of conflict in distributed teams which informed this year’s practicum team and therefore this experience report.

During this practicum the students explored distributed team processes through a combination of individual, online, and collaborative study and experimentation. They reviewed relevant literature, drew on what they learned about software engineering processes, and worked as a collaborative distributed team, the purpose being to reflect on and learn from their experiences. This enabled them to validate and criticize their various assumptions, draw reasonable conclusions, and make practical recommendations to others.

This year’s practicum cohort consisted of eight students split into two cooperating sub-teams with slightly different distribution characteristics that adopted distinct, but related, project responsibilities. Similar team-of-teams of varying sizes are not uncommonly found in practice. This thereby offered a unique opportunity to realistically emulate day-to-day activities such as project management meetings, specification document reviews, and code inspections.

The emergent and evolutionary nature of the project informed the targeted results. More specifically, the students experimented with different collaboration processes and tools during the specification and planning phase of their work which fed the implementation phase. This focused their attention on some of the distributed team collaboration building blocks – both base processes and representative foundation tools. Some of these were rejected, while others were kept and incorporated into their emergent processes. In other words, the students “ate their own dog food” in an evolutionary fashion as they progressively evolved towards their final work products.

2. Establishing Teams

The project started with a clean slate without limitations as to which processes would be explored and defined, or which tools would be used. The joint team was tasked to develop a project specification and development plan to meet the practicum objectives, and utilize tools to shape and evaluate the fruit of their work.

The eight students in the class are all working software professionals with considerable hands-on experience developing software products and services. Two of the students were obliged to participate remotely as they lived and worked further away than commuting would allow. The furthest student lives in central California. Additionally, for the first three weeks, another team member participated from Italy while on a work assignment for their company.

The team members shared many traits including similar social and ethnic backgrounds, language and time-zone - other than the 3-week Italian connection. However, there were also several differences in corporate software engineering and organizational culture, work roles, responsibility levels, experience,

application domains, and, of course, personalities. Some of the students had worked together on class projects in the past. However, several had not.

One of the first things this practicum cohort accomplished was to organize into two teams of four persons each. Given that two students were obliged to attend remotely, it was decided to form a “local team” (Team A) and a “remote team” (Team B). One of the students would have had to make a long commute to campus and therefore elected to be on the remote team. Another member volunteered to work with the remote team. It was acknowledged that even the “local team” members would need to communicate among themselves electronically - they too were a distributed team when collaborating from their homes or workplaces. The team make-ups were set.

3. Deciding Project Goals

The practicum instructor proposed that the practicum students tackle a project related to distributed team collaboration. A software engineering assessment project was also on the table. Soon the students decided to consider the area of distributed teams. They were given considerable latitude in shaping the exact nature and scope of their project. After some brainstorming, the top topic choices were “assessing distributed collaboration tools” and “studying distributed collaboration processes”.

When deciding on projects, each OMSE practicum team normally decides on their own project and the various practicum teams work independently of each other. This year, with the instructor’s approval, the two teams decided to combine their efforts on the same over-arching project. There would still be two teams, but they would collaborate to divide the work and accomplish a combined project. As it turns out, this led to an unexpected distributed team collaboration challenge – namely some “conflict” - much like that which one would encounter on the job. Fortunately, the conflict experienced was very civilized and issues were resolved for the benefit of everyone involved.

4. Working with the Existing Distributed Infrastructure

OMSE courses are typically conducted in a hybrid delivery mode where face-to-face sessions are audio-video (a/v) recorded so that students taking the course online, or those who have missed a class for personal or business reasons, can stream the sessions to their desktops. Course content and class collaboration is supported by email and an online learning management system (Blackboard). The streaming video has approximately a ten-second delay. For regular lecture-style classes this is not a problem as online students do not view the stream live, but instead watch it at time convenient for them. Mind you, they do miss out on the opportunity to interact with the instructor and their class-mates.

A practicum class is not conducted like a lecture. The students need to regularly interact with each other in real time as well as with the instructor, but on a less frequent basis. This was the first practicum class that included remote participation and some modifications to the lecture tools were therefore in order. The first attempt used a telephone conferencing bridge with speakers and microphones in conjunction with the existing a/v streaming system. The stream was used to visually share content, namely, “talking heads” of the instructor and the students in the classroom as well as shared documents (like specs and plans). However, the video was of marginal quality and the delay was particularly frustrating. The telephone bridge also presented some challenges. The audio from the remote participants was routed into the room’s PA system. Although there were microphones in the classroom it was often difficult for both remote and local participants to understand what was said. Background noise would sometimes interfere with meaningful information exchange and drop-outs were sometimes experienced.

Over the subsequent few weeks, the team experimented with various setups and types of microphones, however, the improvements were not particularly satisfactory. Everyone was highly motivated to improve the collaboration environment and tools so that effort could be focused on the collaboration processes rather than the infrastructure.

5. First Collaboration Attempts

In an effort to give the newly formed teams an initial workout, and to come up with a tentative artifact as an anchor for discussion, both teams took a week to generate a straw-man requirements document.

Even though Team A was slated to be the local team, they soon decided to work in a distributed fashion and only meet face-to-face once a week right after class. They used Skype for team discussions and initially communicated by exchanging Word documents via email. This was a familiar and practical approach for collaborating as a team.

Team B experimented with Google Chat, Google Wave and Google Documents. While some members of the team were not familiar with instant messaging or Google Chat, none of the team members had ever used Google Wave or Google Documents before. This resulted in some pain while learning how to use these new tools and determining their limitations and trying to achieve some progress. Team B alternated between synchronous editing / discussion sessions and asynchronous editing sessions. Email was only used for notifications of changes for review or to setup a synchronous session.

When the two teams met again for the regularly scheduled Tuesday night class, they decided to use Team B's document as the basis for further discussion on the project goals. Each team shared what tools they used and how well they worked. Team A preferred synchronous voice meetings while Team B had a preference for instant messaging / text chats. Interestingly, Team B did not favor Google Wave or Google Documents. The main complaint with Google Wave was that the formatting capabilities left much to be desired. While Google Documents allows simultaneous document editing, the open sharing approach implemented by this tool allows someone to inadvertently throw away someone else's changes which proved to be problematic.

6. Project Challenges

As the project moved forward, the team refined the classroom setup, stopped using the streaming video capability and started using Google Documents as a shared Document medium. One person could share a Google Document and make edits while the document was displayed on the projector in the classroom or in the remote participant's browser. Unfortunately, the team members were at the mercy of Google Documents deciding when to synchronize all the viewers. Often this did not work and they would have to manually refresh the document. Google Wave did a much better job with live synchronization, but the lack of strong formatting capabilities hampered its use. The teams agreed that Google Documents was a great improvement over the delayed streaming video approach used initially. However, the team yearned for better tools while experimenting to determine the internal processes that work best for each team.

While struggling with the classroom tools, the teams were also struggling with the exact nature of project. They didn't have a clear, concise vision at this stage. After each class some form of consensus seemed to have been reached, yet attempts to work on the requirements document were mired in confusion, uncertainty, and disagreements on what had been decided in the previous class meeting. Getting eight mostly distributed individuals to arrive at the same understanding of the concepts being described is difficult at the best of times, and remained a challenge throughout much of the project.

7. Writing a Specification, Making a Plan

During the next class the team spent the entire three-hours working on project scoping and requirements. On this occasion, the team wrote a one-line description of the project's overriding goal which was:

To provide a set of processes to support software engineering throughout the SDLC [software development lifecycle] in a distributed collaborative environment and provide guidance for process selection for specific teams, project types and methodologies. [1]

The joint team also reviewed the current requirements specification and decided on guidance to be followed for each section and overall project scoping. Team A was assigned the requirements specification and Team B the Project Management Plan (PMP).

Being distributed 6 out of 7 days during the week continued to delay the formation of a shared context [1] for the class. The guidance that the team had spent three hours developing was found to be inadequate because it relied on a shared context that was not defined or available for consistent interpretation. At this point, the teams still had no leadership structure in place, that is, individual(s) responsible for clarifying, mediating, and arbitrating critical issues to facilitate and ensure that project momentum and progress would be maintained. The task conflicts similar to those anticipated in [1] occurred, evidenced by the “message churn” on the email distribution list. At this early juncture, the teams even disagreed over what constitutes consensus and the teams were still debating project goals.

The result of all this was that there was much uncertainty about what should go into the requirements document. Nevertheless, a first rough draft of a generic project management plan was developed recognizing that this could not be finalized without nailing down the requirements.

The practicum instructor strongly suggested that the teams rotate roles as the project progressed. The Project Management Plan (PMP) therefore defined four roles: Team Lead, Project Manager, Quality Assurance and Architect. Both teams implemented these four roles. The PMP defined one planning iteration and three work iterations and it was agreed that members of both teams would rotate into each role once.

During the subsequent class session, the class focused on developing clear definitions of the project goals and deliverables. The joint team agreed that the project should develop a set of processes designed to guide software practitioners of a typical product development firm when collaborating as a distributed team. The practicum teams would deliver a set of processes with usage guidance packaged in a form of an off-the-shelf product.

The following is a sampling of the processes that were addressed by the project:

- Synchronous Communication
- Synchronous Artifact Collaboration
- Asynchronous Communication
- Asynchronous Artifact Collaboration
- Artifact Review and Feedback

The teams also ratified the role-rotations and the iteration schedule including the assignment of team leads for the remainder of the planning iteration. It was agreed that the leads would work together to resolve project issues as they arose and the rest of the project management plan was reviewed and updated. With an established leadership structure in place and a plan of action to develop the requirements specification and refine the project management plan, the two teams were able to begin working independently on their assigned tasks and artifacts.

8. Collaboration Environment (tools)

Up to this juncture, the teams had not explicitly agreed on which tools to use to support the project. Team A had begun to use Google Documents for collaboration. Each team member was assigned a section of a document and team members worked asynchronously when refining a given document. Team A meetings continued to be held face-to-face after the joint team meetings, followed by sparse email and Skype communication during the week. Team B meanwhile continued to use Google Documents, text chat and email eschewing the use of Google Wave.

The practicum class had at its disposal an OMSE server supported by the CS department IT support group running a “Trac” wiki installation. Trac is a project management system that includes an open source project repository with lightweight project collaboration features including a generalized defect tracking system (a.k.a. “ticket system”) and a source code control system (Subversion) integrated with Trac. It addressed two project needs: the ticketing system provided a central location to retain project tasks and artifacts including feedback about them; and Subversion provided a storage and revision control and tracking capability. These features satisfied many project requirements as well as the OMSE program’s commitment to maintain project archives for future practicum students. It was well-understood that this asynchronous revision control capability would enable a much more reliable process than attempting to coordinate project artifacts by way of email. However, to continue using Google Documents would require exporting project artifacts to documents that the Trac wiki could store. It was therefore decided to drop Google Documents for a more familiar tool, namely, Microsoft Word.

However, the team still did not have an adequate tool to support distributed synchronous collaboration for the purpose of sharing and reviewing project documents. Fortunately, an alternative was introduced late in the winter term of the project. Portland State had acquired a cross-platform web conferencing tool called Elluminate that satisfied this critical project need. With simultaneous screen sharing, chat, and audio conferencing capability, Elluminate completely replaced the old streaming and telephone conferencing systems. The screen-sharing feature of Elluminate allowed the teams to easily share real-time updates of virtually any type of document being edited with all the formatting prowess of familiar native desktop applications. Though only the person sharing their PC screen could modify the document, this turned out to be much better for synchronous document collaboration than Google Documents or Google Wave.

All team members agreed that Elluminate was a marked improvement, however, certain issues remained to be resolved. Microphone echoing issues and displaying documents on the classroom PC and projector did not work seamlessly, marginally dulling the team’s enthusiasm for Elluminate. To resolve these issues, the local students started bringing their own computers to class and joined the Elluminate conferences using their own headsets. Unfortunately, hearing the others in the classroom a second before hearing them again on their headsets proved to be distracting. Eventually, the most trouble-free solution was achieved by having everyone attend the class remotely from each other using Elluminate. Finally, the team discovered an approach whereby everyone in the class was able to collaborate effectively.

9. Introducing and Refining Collaboration Processes

As the Project Management Plan evolved, the teams decided to define processes and guidelines for conducting class meetings including creating agendas, keeping minutes and capturing action items. The minutes eventually made it into the Trac wiki, and the Trac ticketing feature was used to manage action items. Interestingly, the teams continued to use Google Documents and Google Wave despite some of their shortcomings.

Because an agenda is transitory and never meant to be formally captured, Google Documents was used to share agendas before meetings and allow team members to update them as required. Agendas needed some formatting features of Google Documents that were easier to use than those of Google

Wave. The minutes, on the other hand, did not need any formatting. However, to participate in their creation and updating them in real time, Google Wave fit the bill quite nicely. After the meeting, the scribe would deposit the minutes into the Trac wiki and create new tickets to capture action items. As the project progressed and mutual trust among members grew, the team discovered that they really didn't need to use Google Wave to capture and share the minutes - this task could be safely delegated to the assigned "scribe". Nevertheless, it became accepted practice (habit perhaps) to use Google Wave for this purpose. Once Elluminate became available, this tool's document sharing feature enabled sharing and annotating agendas and minutes as required during the meetings – which proved to be a very productive process.

By the end of the planning phase, Team A continued to use Google Documents for brainstorming and initial drafting and editing. They observed that assigning sections to various team members resulted in a document that lacked continuity and flow as the writing style changed from section to section. To help overcome this problem, they modified their workflow to allow each team member to work anywhere in the document. Instead of modifying someone else's work, they would add a comment in the document and allow the original author to make the modification (comments could be in both substance and style). Both teams encountered difficulties with Google Documents when trying to work simultaneously. They soon decided to avoid working simultaneously on their Google Documents.

Team B was somewhat frustrated with Google Documents. Google Document's formatting capabilities were sufficient but the team found it difficult to set up and maintain the consistent formatting they desired. Furthermore, concurrent editing resulted in lost work, and conversion to and from Word documents failed to meet expectations. Soon Team B switched to Microsoft Word retaining documents in Subversion for team sharing purposes. Since Subversion cannot auto-merge Word documents, Team B relied on the locking features of Subversion to ensure that only one person could edit a given document at a time. Generally, one person was assigned to write an initial straw-man document, and then the team would meet using Elluminate for a synchronous editing session. One team member was assigned the role of editor and made all the discussed changes to the document while the rest of the team observed via Elluminate's screen sharing feature, providing feedback when needed. Trac tickets were issued at the end of the meeting to assign unresolved fixes for later asynchronous editing.

Team B kept in constant communication during the week via email, instant messenger and Trac tickets. Team A held synchronous meetings using Elluminate to resolve outstanding issues with a given document assigning someone to convert the document for storage into Subversion.

By the end of the first practicum term (winter), the Project Management Plan was base-lined and both teams were beginning to follow its management processes. Along with the defined team structure, the teams were better able to re-solve their task assignments and the occasional conflict that would inevitably arise. Both intra-team and inter-team shared contexts were strengthening and the class became increasingly productive. The processes that the teams were organically discovering would become the basis for the processes that they would start executing during the next (spring) semester. In retrospect, the first term of the practicum was an essential and worthwhile process discovery and team-building exercise.

10. Iteration 1: Full Steam Ahead – A Few Bumps on the Road

After a short break between the winter and spring semesters, Teams A and B met up again to start working on the plan to develop the deliverables that were defined the previous semester in the specification document. There were still a few major decisions that needed to be made that the newly appointed team leads tackled the first week back after the break. The teams planned to both write and evaluate the collaboration processes to be written. The first draft of the document defining the process evaluation criteria had been created during the winter semester. This document needed some major rework, review, and ratification by the teams. Given the practicum team had eight process authors, they also needed a template document that would define their look and feel to ensure consistency across all of the processes.

In addition to the evaluation criteria and process template documents, each team constructed one process using Microsoft Word consistent with the process template. However, with over thirty processes in the requirements specification, some team members started to become concerned about maintaining consistency using Microsoft Word to write these processes. What if some aspect of the process template specification was changed? Would it be possible to consistently apply those changes to all existing processes? Emails were exchanged and a couple of solutions were suggested. Team A suggested the use of the Eclipse Process Framework (EPF) that was used on one of the OMSE classes. A member of Team B decided to build a web-based prototype tool that would take xml fragments and generate web pages that matched the look-and-feel of the defined process template document.

During the second class meeting of Iteration 1, the teams debated these two tools. EPF is powerful, free and generates attractive HTML output. However, there is a significant learning curve to achieve proficiency. And, even though EPF is based on the cross-platform Eclipse project, EPF itself only works on Windows and Linux. Two of the students had Macintosh computers and could not run EPF without a virtual machine (VM) installation which was considered cost-prohibitive, and some members have had sub-par experience with free VM tools. Meanwhile, the proposed custom tool was simple to use and easily customizable, but it was only a prototype at this stage. And there was concern that the team member proposing it would not have the time to create it or support it – after all, the project focus was to create processes, not develop a process-writing tool.

When it came time to decide on which tools to use, all members of one team voted for EPF and the other team opted for the prototype tool. The class was divided along party lines. The team did agree, however, to follow the leadership plan adopted earlier to resolve conflicts. The two team leaders met, discussed the issues, and finally resolved the deadlock with a coin toss. The custom tool was chosen.

The custom tool implementer was able to crank out the initial version of the tool with documentation within a few days allowing everyone to start writing their processes. By the end of the week the tool was generating a deliverable website, and the teams had written an additional eight processes and converted the two already existing ones using the custom tool format.

The following week was devoted to evaluating the processes that had been written. The plan was that each student would be assigned to lead the evaluation of a process written by the other team. Along with two to three other students, the evaluation team would role-play a scenario using the process which ensured that that each student would evaluate at least three processes. This turned out to be much more work than it took to create the processes, and there was a fair degree of push-back from the team.

11. Iteration 2: Quality over Quantity

The beginning of Iteration 2 gave the teams an opportunity to take stock of what they had accomplished and make necessary adjustments to the project plan. They flagged the following two major issues and discussed them at length.

The first issue was the question of quality versus quantity. Over thirty processes were defined in the requirements document and the teams had successfully written ten processes during the previous iteration. If the current pace was maintained, all of the allocated processes could be specified – at least in theory. However, it became clear that the quality of the written processes was seriously lacking. To write all the defined processes, the teams would not be able to spend much time on each process much less revise or fix any defects. Furthermore, it was agreed that the current pace could not be maintained. All agreed to modify the plan by limiting project scope to a small set of well-polished processes. The processes defined in the requirements document were divided into the following groups: “general”, “project management”, “requirements” and “development”. It was decided that the “general” processes would be the best place to focus the team’s efforts as these are referenced by processes in the other groupings. In addition, the ten general processes had already been drafted (though lacking) during Iteration 1. The new agreed-upon plan was to iterate over the processes already written to elevate them quality as much as possible.

The second issue was the evaluation criteria and process that had been defined. It was now recognized that even with a reduced number of processes to work on, role-playing the processes was not feasible. And the quality of the results that could be achieved from such an exercise was challenged by more than one team member. This issue was resolved in two ways. First, the formal role-play aspect was dropped and evaluations became an intellectual exercise performed by each assigned individual. However, the requirement to have the evaluation of a process performed by a member of the opposite team was kept. Second, the formal role-playing activity was replaced by an informal “eating your own dog food” strategy. This was possible because the general processes the teams were defining were exactly the same types of processes that were being used in the conduct of the project - for example, processes to synchronously or asynchronously conduct meetings and edit artifacts. The teams agreed to follow the fledgling processes during their collaborative project activities while updating them in parallel. This provided another means for team members to identify defects in their processes and record tickets for later resolution.

With this modified action plan the teams went to work polishing their processes. Each team member took one process per week and reviewed and revised it within their team before turning it over to the other team for evaluation. Both teams used the Trac ticketing system to assign work but otherwise continued to work as they had at the end of the winter term. Team A continued to use Google Documents to create draft processes which they converted to text files, placed in Subversion, and processed through the custom process specification tool. Meanwhile, synchronous meetings were held with Elluminate for desktop sharing; and Skype was also used because some team members felt that Skype provided better quality of audio than Elluminate. Extensive use of Elluminate, email, instant messaging, Subversion and the custom tool was made during this iteration by Team B.

12. Iteration 3: Finishing Up

The beginning of Iteration 3 provided a final opportunity to take stock of what had been accomplished and make necessary adjustments to the project plan. This time, the team identified four critical issues that needed to be addressed.

Applying the custom tool and process template document enabled all of the processes to exhibit a common look. However, at the end of Iteration 2 there was one lingering consistency issue. Each process was characterized by a workflow of re-usable work tasks. A work task may or may not take some artifact as an input; and after performing the steps of the work task some output artifact is created. In essence, a work task is a foundation process element that cannot be broken down into smaller chunks, and a process is a unique ordering of work tasks with some additional supporting text. Team A and Team B differed on when an activity should be expanded into a complete work task, and when it could be written as supporting text of the process. The team leads considered the issue and decided that this inconsistency in the processes was not great enough to warrant changing at this point in the project.

While the custom tool happily created a web site composed of a collection of process pages, it was missing standard document trappings such as an introduction and a usage guide that would bind the pages into a cohesive deliverable. A minor unofficial effort to resolve this issue was undertaken during Iteration 2 and two team members were officially tasked to finish this in Iteration 3.

Along with delivering a collection of processes, the practicum team planned to deliver a document that mapped which processes were best-suited for the various stages of the software development life cycle. It was decided to deliver this document even if this implied devoting less time to polish existing processes.

The teams also needed to allocate effort and schedule to create a number of deliverables during this final iteration to meet the requirements of the practicum project including a final presentation and a final report which incorporated the project's specification, project management plan, process evaluation criteria, and process deliverables. This last iteration required a lot of cross-team collaboration to generate these project deliverables and prepare for the final presentation.

The class decided to use the Elluminate tool, together with other tools they had used, to conduct a distributed team presentation of their project accomplishments. Conducting a dry-run a week before the final presentation proved to be invaluable. The team used the experience it had gained with the collaboration tools and distributed collaboration processes to setup the classroom such that they would avoid many technical and process problems during the final presentation.

The final presentation was performed using Elluminate's document-sharing and white board features to display PowerPoint slides summarizing the project. A PSU classroom on campus was secured and potentially interested parties were invited to attend. This included current and past OMSE students, faculty, and industry guests. Team A was present in the classroom together with the instructor and some of the invited guests. The Elluminate session was projected on a large screen and the audio was piped into the classroom PA system. The local team also had their laptops linked into the Elluminate session through the WiFi network. Team B and several other invited participants attended remotely via their PCs and headsets. The order of speaking and audio had to be carefully controlled by the team, each member having a critical role to play in both controlling and making the joint presentation.

13. Summary

This practicum was a hybrid online learning experience conducted in a university setting. Interestingly enough, the challenges faced by the students were virtually the same as those encountered by distributed teams in the real world. In both educational and industry settings one would ask questions like: "How does a distributed team organize a meeting?" "How do team members work on and review project artifacts like specs and plans?" "How does one team interact and work with another team?" "How do team members communicate facts and issues to other members?" And, "How do distributed teams ensure that everyone on the team has the same interpretation of project information?"

In general, the practicum teams found that they could communicate effectively in an informal fashion when the team was small (e.g. four). However, joint meetings of all eight participants required a more organized and systematic approach. In addition, getting eight team members to adopt and follow the same vision for the project did not turn out to be a simple matter. The lack of an empowered subset of the team with a clear understanding of the vision and good communication skills undermined early achievement of goals for the team. Once a leadership structure was put in place decisions were faster and progress was more tangible and visible to the entire team. The planned iterations facilitated positive forward progress because they allocated specific effort and milestones to achieve clearly articulated iteration goals. The teams were able to continue to make progress following an imperfect plan knowing that unresolved issues would be addressed at the beginning of the next iteration.

However, certain aspects of the team's process made it more challenging to keep everyone motivated. The lack of face-time and non-verbal communication cues made it difficult to tell when someone was no longer engaged. Furthermore, it was found that when such cues were missing from the communication medium, messages were easily misinterpreted potentially leading to misunderstandings and even ill-feelings amongst team members. And such disconnects tended to go un-noticed for longer periods across distributed teams than collocated teams. Addressing them as soon as possible is always a good idea.

Team building and team cohesion strategies were not explicitly investigated. However, it soon became clear that distributed teams need to seek out ways to build trust and camaraderie to overcome interpersonal communication problems. The practicum teams overcame some of these issues to a degree by encouraging each team to explore how they best worked well together.

Finally, when it comes to distributed collaboration tools, to no one's surprise, it was verified that no one tool fits all requirements. A distributed team needs to use multiple tools to work together effectively. Furthermore, the teams discovered that it is highly beneficial to have certain team members become proficient in their tools of choice acting as technical support and consultants to the other team members.

Both teams experienced some pain points early on which were overcome as the project progressed. After reviewing Hinds [2] p. 617-623 later in the practicum, the teams reflected on their experiences and thereby came to a better understanding of their "issues". As explained by Hinds on the topic conflict on distributed teams, a new team needs to develop a shared context for working together. Early in this practicum, some of the team members encountered a degree of internal conflict about the central focus of the project and their particular roles in this project. Their initial results were less than hoped-for with fairly sparse coverage of the topics – and they wondered why. But this could be explained by the initial lack of shared context and lack of processes in place. Once the teams began to understand their true goals and develop processes for moving forward, they jelled as a team, fell into a rhythm, and began to achieve step-wise successes. Although they constructed an initially over-reaching specification and plan, they were able to hunker down, de-scope the project, and achieve a stellar result.

14. Conclusion

The OMSE students conducting this practicum project were well-aware of many of the challenges of working in distributed teams and were able to leverage their industry experience in this interesting problem area. They were able to combine this experience with the software engineering principles and processes they had studied in the OMSE program. And, by way of thoughtful and systematic experimentation, they were able to progressively evolve a practical framework for defining and evaluating a specific core group of distributed software collaboration processes.

The ultimate outcome of this practicum project was that the students learned a number of critical lessons about how various distributed team processes and tools fit with the challenges of collaborating in teams. The students also confirmed that when working in a distributed team, be that as a practicing software engineer or part of a group learning exercise, the types of processes and tools that they use day-to-day are virtually the same. The practicum project described in this paper is an example of how software engineering practice informs learning as well as how software engineering education informs practice.

Acknowledgement

The referenced OMSE software engineering practicum team was comprised of Thomas Carlier, Jason Cowley, Adam Kolb, Merri LeClerc, Raleigh Ledet, John McConnell, Grant McCord, and James Thompson. OMSE faculty member Kal Toth facilitated the project. The authors of this paper thank the team members for the valuable outcomes that emerged from their work as well as their review and thoughtful contributions to the creation of this paper.

References

- [1] Carlier, T., J. Cowley, A. Kolb, M. LeClerc, R. Ledet, J. McConnell, G. McCord, J. Thompson, "Distributed Software Engineering Process", Final Report, OMSE Practicum, winter and spring 2010.
- [2] Hinds, Pamela J, and Diane Balley. "Out of Sight, Out of Sync: Understanding Conflict in Distributed Teams." *Organization Science* 14 (2003): 615-632
- [3] Mihauser, Kathy, "Distributed Team Collaboration", PNSQC, October 26-28, 2009
- [4] Eby, S., B. Rydell, C. Seaton, "Distributed Requirements Collaboration Process", PNSQC, Oct. 2009.
- [5] Khan, A., R. Ramadass, L. Rosenbaum, B. Varma, K. Toth, "Distributed Collaborative XP Meetings", Poster Paper, PNSQC 2009.
- [6] Khan, A., R. Ramadass, L. Rosenbaum, B. Varma, "Distributed Collaborative XP Meetings", Final Report, OMSE SE Practicum, winter and spring 2009.
- [7] Eby, S., B. Rydell, C. Seaton, "Distributed Requirements Collaboration Process", Final Report, OMSE SE Practicum, winter and spring 2009.
- [8] Bates, K., J. Maurer, S. Wainstock, E. Wilson, "Social Networking", Final Report, OMSE SE Practicum, winter and spring 2009.