Open access • Proceedings Article • DOI:10.1109/ADC.2005.1

A case study on the impact of scrum on overtime and customer satisfaction

C. Mann, Frank Maurer

Institutions: University of Calgary

Published on: 24 Jul 2005 - Agile Development Conference

Topics: Customer satisfaction, Scrum and Overtime

A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction

Chris Mann
University Of Calgary
mannc@cpsc.ucalgary.ca

Frank Maurer
University Of Calgary
maurer@cpsc.ucalgary.ca

Abstract

This paper provides results, and experiences from a longitudinal, 2 year industrial case study. The quantitative results indicate that after the introduction of a Scrum process into an existing software development organization the amount of overtime decreased, allowing the developers to work at a more sustainable pace while at the same time the qualitative results indicate that there was an increase in customer satisfaction.

1. Introduction

Now that agile methods are crossing the chasm into the mainstream of software organizations, the need for evidence as to how well these processes work in industry is increasing. The current results available are derived mainly from anecdotal experience reports from early adopters. These experience reports are usually very specific to a certain company or environment and often do not provide quantitative support for the qualitative observations. Experience reports usually do not provide enough context information to compare them with other reports.

In order to add to the body of knowledge of agile methods in industry and explore future directions of research, we conducted a two year exploratory longitudinal case study to look into the effectiveness of agile methods and practices in a small team industrial environment. This case study is being done as part of a cooperative effort between the University of Calgary and Petrosleuth Inc. (PetroSleuth). Petrosleuth is a software development firm located in Calgary. The company develops both MS Windows and web based applications for the oil and gas industry. This paper reports results on the impact of Scrum project management practices on overtime and customer satisfaction.

There are two motivations for this case study. The first motivation is to increase the body of knowledge about Scrum and to see what patterns emerge when Scrum is used in an industrial setting. The second came from the company itself. The company wants to increase the quality and speed at which it could deliver its products. The company believed that Scrum could be beneficial based on anecdotal reports but it didn't know how to adopt Scrum into their environment.

In our study, we compare the results of software development before and after the introduction of Scrum and other agile practices (pair programming, unit testing, continuous integration). We decided to conduct a study over two years to get an understanding of long term patterns and trends instead of seeing only short term effects. In this paper, we report on qualitative and quantitative results on Scrum practices. The paper will focus mainly on the quantitative results of changes in overtime patterns and the qualitative results of customer and developer experiences.

The remainder of the paper is organized into four sections. Section 2 discusses related work and Section 3 provides the experimental setup for the case study. In Section 4, we present and discuss the results of the study. Finally, in Section 5 we summarize our findings and elaborate on future work.

2. Background and Related Work

The research approach taken for this study is a combination of qualitative and quantitative research methods. The study is a long term longitudinal study of approximately two years where the researcher is embedded part-time in the workplace context for the duration of the study. A longitudinal study is used so that we can get an understanding of the long term rather then the short term effects of introducing agile methods. The embedding of the researcher in the workplace context is done so that the day to day events before, during, and after the introduction can be

observed and recorded. This gives a better understanding as to what goes on during the introduction and provides contextual information throughout the data generation period, which can be used to provide additional insight into the results. Embedding researchers within the subject of the investigation is based on ethnographical research approaches.

Ethnography: Ethnographic research methodology comes from the area of anthropologic studies but can also be applied to information systems research [1]. Its main goal is to describe the cultural context of a situation. One impact of ethnographic research is that since the researcher is part of the environment he is studying, he will have a direct impact on that environment and the understanding of the environment will have a subjective component.

Scrum: There are many experience reports such as [2,3] that describe success stories of introducing Scrum into a company. Most of the knowledge about Scrum in industry is contained within such experience reports. Two problems, though, are that these reports usually do not provide enough context about the experience and usually do not provide quantitative results with the observations. Our study is different in that it provides data over a longer period (2 years), provides a lot of study context and provides quantitative results together with qualitative observations.

3. Experimental Setup of the Case Study

In this section, we will present the data that has been collected as part of the study and the methodology of how it is collected. In addition we will describe the context in which the study took place.

3.1 Data Collection and Methodology

For the study, data has been collected from three sources: office hour time records, questionnaires, and workplace observations. In this section we will describe each data source and the method of data collection.

Office Hour Records: Office hours (or: work hours) are hours that are spent at the workplace except for lunch. This means that on a regular working day, if the developer is at the workplace from 8 am until 4:30 pm, the number of office hours should be 7.5 hours of office time if they took a 1 hour lunch break (8.5 total hours – 1 hour for lunch). The office hours reflect the expected amount of time the developers have to develop software. To record the office hours the developers were asked to use an internally developed application called TimeTracker. TimeTracker is a Windows application that allows the developers to enter their office hours into a database. The hours entered into the database are used by the accountant for billing purposes and are, thus, considered relatively accurate. The tool gives the developers the ability to categorize their time into a few different categories: Administrative (sick days, vacation, and statutory holidays), time working on client hardware support, and time spent working on software development tasks. For the purpose of the study the Administrative days such as sick days and holidays (both statutory and ones taken by the developers as part of their vacation time) are filtered out during the data collection process. This means that any time entered against sick days, vacation, etc are not expressed in the results. The reason for removing the administrative hours is because the developers are not actually supposed to be working during those hours. The frequency as to when the developers enter their office hours into the system varies with each developer. Some developers enter their time daily while others record the hours in day planners and enter the time on a weekly basis. Regardless of which way the developers choose to enter their time, all time must be entered by the end of the month. This is when the office hours are required by the company's accountant. We retrieved the office hour records for analysis form the database.

Personal observations: Since September 2003, one of the researchers (Chris Mann) was at the Company two days per week, usually on Mondays and Thursdays. His role was to assist in the build management process and the installation and maintenance of the tools introduced as part of the study within the build process. He introduced pair programming, test driven development and continuous integration into the company as part of the study but not Scrum. Some of the results obtained were through observations and conversations with developers and in some cases opinions based on the context of the observations. Since the study is using principles from ethnology, Chris's observations were recorded daily in a research note document. The research notes document was started in June 2004. Prior to this date no written personal observations exist.

3.2 Case Study Context

This section will use categories based loosely on the context factors found in the XP-Evaluation Framework 1.4 [6] as they provide an easy to understand format to convey contextual information.

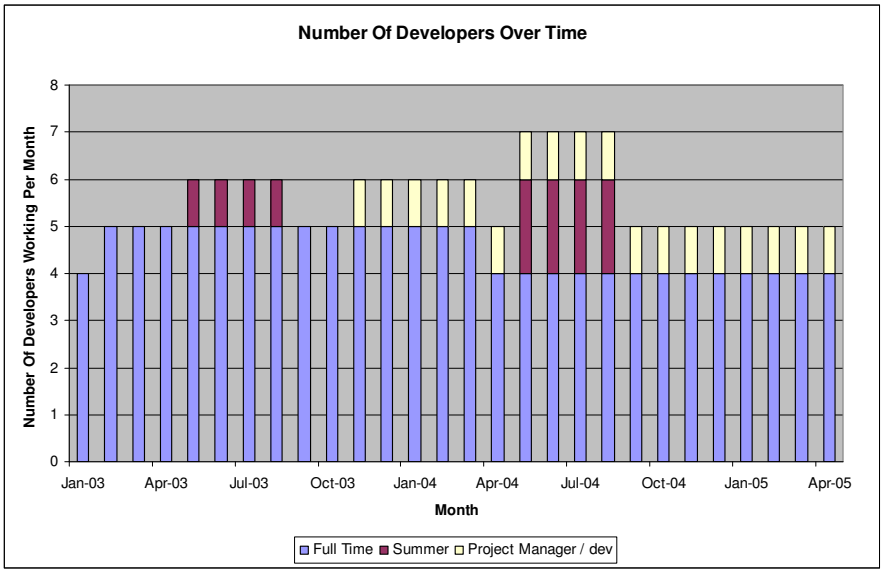

Figure 1: Team Size Over Time

Team Size: Figure 1 graphs the team size changes over the course of the study. Starting in January 2003, there were four full-time developers, with the president of the company acting as the project manager. In February of 2003, a developer was hired out of university. I again worked as a summer student In May of 2003 until September 2003 when I started this research and ceased being a software developer. In October 2003 one of the senior developers was given the role of project manager. He, however, also remained as a part time developer by splitting his time 50% development and 50% project management. In November 2003 a developer from PetroSleuth's client was added to the team. In March 2004 one developer left the company. In April 2004 one developer left the company and one developer was hired. In May 2004 two computer science summer students started to work for PetroSleuth. These summer students worked until the end of August 2004. In November 2004 a developer left the team and was replaced the same month (November 2004).

Highest Degree Obtained: Table 1 outlines the highest degree obtained

Table 1: Highest Degree Obtained

Degree	Number Of Developers
B.Sc Computer Science	4
B.Sc Plant Sciences	1
B.Eng	1
MCSE	1
None	1 + 2 summer students

Experience Level: The following table outlines the amount of software industry experience the development team had.

Table 2 Software Industry Experience

Years of Experience	Number of Developers
<5 years	4 + 2 summer students
15 years	1
>20 years	2

Most of the team is comprised of summer students and developers just out of university.

Language Expertise All of the developers who worked at the company from January 2003 until the present (March 2005) had less then five years of expertise using C# (This is the primary development language in the company). All but one had less then five years of SQL development experience. One developer had greater then 5 years of SQL experience.

Agile Experience: Table 3 outlines the agile experience of the development team.

Table 3: Agile Experience

Agile Experience	# of Developers
Tried unit testing at home	1
Pair Programmed in school	2
Class used Extreme Programming	2
No Agile Experience	3

Project Manager Experience: The person acting as the project manager from October 2003 to the present was both a developer and the project manager. His experience can be classified as medium as he has

approximately 10 years of project management experience and a PMP certification.

Geographical factors: Both the customers and the offices of the company are co-located in the same building. All of the software developers are on the same floor while the primary customer is partially on the same floor and the floor below.

Previous Software Development Methodology: Before the introduction of Scrum, the software development process could be considered an ad hoc approach. There was little actual planning involved when developing the software. There were no planning meetings per say. Every so often, when the president of the company wanted to get a status update as to the progress everyone was making, he would call a meeting. The meetings were usually held every Monday morning. Sometimes during these meetings, as part of the update, the team would talk about what they were working on but the meeting was not seen as a place to do planning. In addition to the usual Monday meetings, the president would sometimes call impromptu status meetings. A major problem with these impromptu status meetings was they were called without much warning and though they were intended to be only a few minutes, they took much longer, sometimes 45 minutes or more. In addition to both the Monday meetings and impromptu status meetings, there were also weekly meetings on Wednesday with the stakeholders of the product.

During the weekly meetings, functionality that was completed in the past week was demonstrated. Sometimes if user functionality could not be shown it was described. If there was nothing to describe or demonstrate, the session would turn into a question and answer session about the product. During the meetings, stakeholders would make suggestions and requests improvements and additional functionality. The suggestions and requests for the software being developed sometimes changed drastically from week to week.

There were two problems associated with the rapidly changing requirements. First, there was a very poor picture of what work was going on internally and what work was planned for future completion. Internally, there was no place that a developer could go and see an up-to-date comprehensive picture of what needed to be completed and when it was expected in relation to other pieces of functionality. One method, used to try and reduce this confusion, was to have a meeting every so often to discuss what was needed to be finished and when it was supposed to be completed. These meetings helped in the short term, but did not address the core issue of controlling the ever increasing number of requirements for the release. The second problem associated with the rapidly changing requirements was there was no control put in as to what the developers were working on. If a stakeholder suggested something in the weekly meeting it was very likely the developer would add what was requested to what they were working on without consulting anyone.

Even though the team was communicating with the customers on a weekly basis and showing off what they had done, the process was still out of control in terms of controlling what features went into the product at what point.

At the end of January 2004 - before Scrum was introduced - pair programming was introduced at the company. A month after Scrum was introduced (June 2004), unit testing with continuous integration was introduced. These practices were introduced to deal with the actual development of the software itself rather then the managing of the software requirements and timelines. We will now describe how the Scrum [4] process was introduced at the company and the modifications made to the process as part of the introduction.

Scrum had been talked about as a way to improve the software process at the company by Chris for over a year but there was not enough management support to implement it. Scrum finally gained acceptance when the project manager went to a Scrum presentation at a VS Live conference and saw the Scrum process as part of a presentation. When he returned, he championed the use of Scrum as the software development methodology to be used in the future. A month later, the Scrum process was put in place at the company.

The company adopted many of the techniques of Scrum such as: Daily meetings, Sprint Planning Session, 30 day fixed sprints, Sprint review, Sprint Retrospective, Prioritized Product Backlog, and Sprint Backlog. Modifications were made to the process to have it fit into the environment the company was operating within. The first modification was to the daily Scrum meetings. In the Scrum process, these meetings are supposed to be short 10-15 minute standup meetings. The practice at the company is to have sit-down meetings. In the beginning, it was explained to the people attending the meeting that the reason the meetings are standup is to prevent the meetings going on for more then 15 minutes but the developers and the customers who come still prefer to sit down at the meetings as it is more comfortable for them and although the meetings are sit-down they are held at the same time in the morning each day in the same room. Another modification to Scrum is the scope

of the sprints is not totally fixed. The reasoning is because the company has to do maintenance on the current software out there and if a critical bug pops up and needs to be fixed it is slotted into the sprint. If the bug has to be slotted in, the customer is informed that something will have to be dropped and the customer makes the decision what they can live without. In addition, if some developers finish early they go to the customer and ask what is next on their priority list even though there is a prioritized backlog.

4. Results and Discussion

In this section, we will provide a description of both the qualitative and quantitative results from the case study.

4.1 Quantitative Results and Discussion

Sprint Timelines: The sprint start and end dates were retrieved from the VersionOne [5] sprint planning tool. Originally, the sprints were decided to be as close to 30 days in length as possible. The first sprint took 29 days. The second sprint ended up being 57 days in length. The second sprint which started on June 4th was originally planned to end on July 5th. It was extended to July 16th to accommodate more work to the sprint. Near July 16th a problem with input data (provided by an outside source) was discovered and the sprint was extended to July 22nd to try and correct the problem. Eventually it was determined the problem could not be fixed quickly so the sprint was extended to July 30th. The team knew that the problem was not going to be fixed by July 30th but the decision was only made to stop the sprint after the second extension to allow the team to regroup and plan how to attack the discovered problem. The other reason for stopping the sprint was because the series of extensions seemed like the way software was developed before Scrum was introduced and no one wanted that to happen again.

The third sprint which took 53 days also encountered a problem near its completion. For the third sprint, the developers finished everything a week early and they went to the customers to see what they wanted done to fill in the remaining time. The customers came back with a request for work that would take the team two weeks to complete. The Scrum master at the time was one of the developers as the regular Scrum master was on vacation. He decided that since they could not get the work done in the week remaining that they would extend the sprint to allow the work to be completed. The decision not to end the sprint and start a new one was taken because the developers did not want to "waste" a day doing the review and planning sessions again. After the Scrum master returned, the sprint was extended again to allow for the work that had been promised to the customer to be completed. During the third sprint retrospective both the customers and developers agreed that there would be no more extended sprints as they had caused too many problems in terms of the developers not knowing what they needed to do. The extensions can be seen as part of learning how to use the Scrum process from books rather then experienced consultants. It was decided that for all future sprints if the work could not be completed in the sprint it would either be trimmed down or moved to the next sprint. The table below reflects the start and end dates for each of the sprints completed so far at the company.

Table 4. Sprint Timeline

Sprint	Start Date	End Date
1	May 3rd 2004	May 31st 2004
2	June 4th 2004	July 30th 2004
3	July 31st 2004	September 21st 2004
4	September 21st 2004	October 21st 2004
5	October 22nd 2004	November 20th 2004
6	November 21st 2004	December 23rd 2004
7	December 23rd 2004	January 31 2005
8	January 31st 2005	March 1st 2005

Background information for time charts: In this section, we will provide some timeline background information for the quantitative and qualitative information to be presented. Before January 2003, the software team released a small Windows application. It was fairly small with little functionality. Between January 2003 and October 2003 the team was both maintaining and enhancing the Windows application they had developed previously. There were very few official release dates for this application that was being maintained and enhanced, with releases being more of a hot fix to user problems. The hot fix that was deployed was usually whatever code had compiled on the developers machine that made the fix in the first place.

In the period of October 2003 until the end of February 2004, the team had developed a website application that was a large scale project with a very aggressive deadline. They did not have experience doing a website before so most technical aspects of the project had to be learned as the team developed the

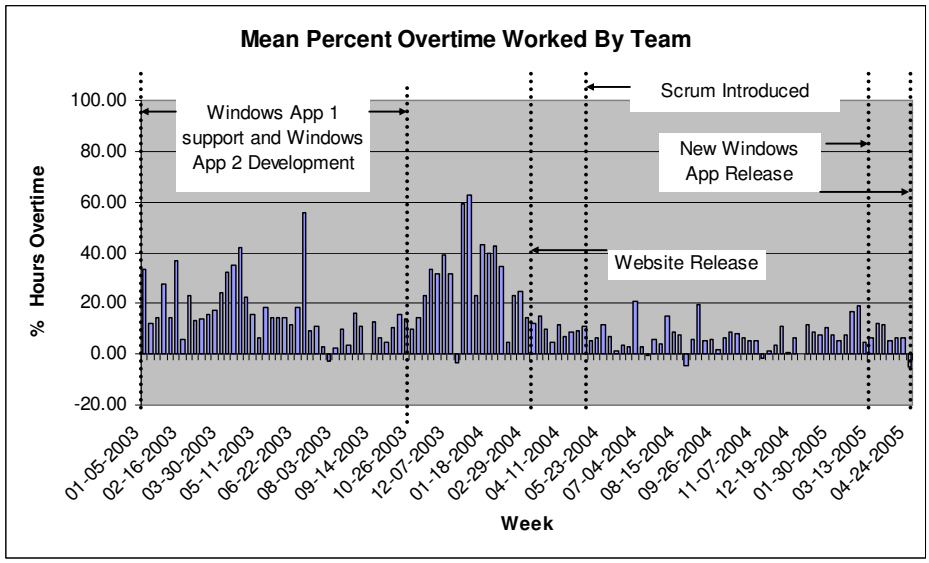

Figure 2: Mean percent overtime worked by team

software. March and April 2004 were spent on cleaning up and maintaining the website. In May 2004, Scrum was introduced. During this month the team decided that the previous windows application could no longer be extended so they started a period of researching and developing a new system architecture and a new windows application.

Mean Percent Overtime Worked by Team: Figure 2 outlines the mean percent of overtime worked by the software developers as a team on a weekly basis [see Appendix A for the metric definition]. This means that a given percentage is the percent overtime per developer per week. The use of a weekly basis instead of on a per sprint basis was done to allow the comparison of pre-Scrum introduction overtime amounts with post-Scrum introduction overtime amounts.

From the above chart you can see that in the period before Scrum was introduced there were periods of overtime spikes, both during a release such as in November 2003 to March 2004 and development and maintenance such as between January 2003 and October 2003.

There are two fairly noticeable spikes before the introduction of Scrum. At the end of June 2003, the team was transitioning from .net 1.0 to .net 1.1 and it broke a lot of the application the team was supporting, therefore there was a large spike of overtime during this period. The other spike was for the development of the website.

After Scrum was introduced there are still some spikes, although substantially smaller. These spikes almost perfectly correspond to dates where the team had to run a data transformation process that was created before the website, that never has really worked the way it is supposed to and there is always a lot of clean up of the output when it is finished. In the following table we will compare the mean percentages before and after Scrum was introduced.

Table 5. Before and after statistics

	Before Scrum	After Scrum
Mean percentage overtime worked	19	7
Standard Deviation	14	5
F Test for variance: F= 9.11 Dfn=68 DFd=51 p= < 0.01 (one tailed)		
T Test (unequal variance): DF=87 p= < 0.01 (one tailed)		

To compare the periods before and after the introduction of Scrum an F-Test was performed with the following hypothesis:

H_O: there is no difference between the two variances

H_A: The larger standard deviation in overtime before Scrum was introduced is significantly different than the smaller standard deviation in overtime after Scrum was introduced.

The F-Test showed the standard deviation before Scrum was introduced was greater than the standard deviation after Scrum was introduced indicating that there was more stability in terms of overtime worked

after Scrum was introduced. A T-test was also performed on the following hypothesis:

H_O: there is no difference between mean percentage of overtime worked before and after Scrum was introduced

H_A: The mean percentage of overtime worked before Scrum was introduced is greater than the mean percentage overtime worked after Scrum was introduced.

The T-test assumed that the variances of the period before and after the introduction of Scrum were not equal as shown by the F-Test. The T-test shows the mean after Scrum was introduced was less than the mean before. Indicating that the team worked less over time after Scrum was introduced.

Overtime Discussion

Here we will discuss the quantitative results obtained.

The mean Percent of Overtime worked by the team hours in the period before the introduction of Scrum is much higher then the overtime after Scrum was introduced. In the case of the Mean Percent of Overtime between the before and after there was almost a three times decrease in overtime from before to after. The F-Test showed that after Scrum was introduced the variance in overtime decreased, this indicates a more stable work environment. Also the T-Test showed that the mean percentage of overtime was smaller after Scrum than before Scrum. The influence of Scrum is supported by our data but would have to be tested in future experiments and/or case studies to see if it is actually valid in other environments. However there are some limitations to the results provided. The two time periods discussed have several differences between them even though they are at the same company.

The first difference is that the software developers are not the exact same software developers who worked after Scrum was introduced. There are many points in time where developers are added (summer students) then they leave after a few months. There are also instances of developers leaving the company. Another difference between before and after is that there were differing pressures to deliver the software. Between May 2003 and October 2003 the pressure fluctuated almost daily. The reason was that the development team was under pressure both to develop new feature and to fix the many bugs that kept on appearing in the previously deployed software. From October 2003 until March 2004 there was a tremendous amount of pressure to deliver the website as the company was seeking to use it as a revenue source. After the website was delivered, there was very little pressure to deliver anything new but there was pressure to figure out a better software process than before. Finally from April 2004 until present, there has been a lot of pressure to deliver the new application again as the company looked to use it as an additional revenue source but when looking at the mean overtime during this period there was no large spike in overtime. A third difference is the complexity of the projects. Unfortunately at this time we have no quantitative way of comparing the complexity of previous projects to the current one. In informal discussions with the developers they say that the website is of similar complexity to the new Windows application deployed in March, though they do say that they are much more experienced then when the website was developed and they are also now using pair programming, unit testing and continuous integration. In the future we may be able to look at features developed or code created. One key result that can be seen in the data is after the Scrum process was introduced there seems to be much more emphasis put on working at a sustainable pace over the long term rather then working at a frantic pace over the short term. Even though there are other practices in play that may have helped the developers do their work faster, it was still the Scrum process that helped keep the amount of work and the amount of time to do it reasonable from both a developer and customer perspective.

4.2 Qualitative Results and Discussion

Developer and Customers Opinions: Here we will present some qualitative results from developer and customer questionnaires. The questionnaires were run to get some feedback from the customer and developer perspective about the Scrum process and to get some idea of the differences before and after the introduction of Scrum.

Customer Opinions: The overall feedback we got from the customer group was positive. All three of the customers said that they would recommend using Scrum in the future, though some would like to see some modifications to it. Of the current set of customers, some were involved with the previous releases of software in terms of testing and verification but were not very involved with the decisions relating to functionality or usability. This situation was very common prior to Scrum where the customers were only used to check the results of the development rather than making decisions as to what they wanted to see for development and how they would use it.

When asked how satisfied they were about the software developed before Scrum was introduced the customers said that they weren't really part of the process and did not really care for the software produced. One customer said that they were "*Ambivalent,* [the] *product was alright, not great"*. Other customers had stronger opinions. "*The release of the website was quite honestly a nightmare...*". The overall theme from the customers was that they were not very involved with the software produced before Scrum and in some cases were not satisfied with what was produced. Since the introduction of Scrum the customers have been more satisfied with the software developed. When asked how satisfied they were with the software produced after Scrum had been introduced one customer was "*Very satisfied"*. Other customers responded to the same question by mentioning other benefits Scrum had brought: "*I believe there has been far greater consistency, transparency and coordination since the implementation of Scrum*". Another customer mentioned that they were much more involved in the process than before, *"…The initiation of the Scrum process has lead to our being more involved in the daily review and discussion. This has lead to us being more aware, and being held accountable earlier in the process for any changes and concerns that have or had to be considered.*"

The customers said that the Scrum process changed how they interact with developers. Some customers gained more respect for the software developers, "*I have a greater respect for the software developers and understand how easy it is for expectations and results to differ without clear instructions and regular communication between all parties"*.

The customers said that they like the sprint planning meetings. One customer had this to say about the planning meetings, "*Superb forum for planning; the whole team is involved and thus everyone knows what is required from them*". Another customer mentioned how they think that the planning meetings prevent problems later on: "*Although the day as a whole can be a very tiring process, I have found that the time spent in the planning meetings has lead to less misdirected development and a more clear understanding of both the requirements and the limitations of the development process by both the customers and the developers*".

The customers also liked the sprint reviews and retrospectives, "*While the Scrum process has often made much of the accomplishments in a sprint to be known before a sprint review, it has helped us as customers* [to] *see visually the product, and we are able to see these earlier in the process again. This has lead to the ability to "tweak" and change the product in a more timely fashion. We as customers have found it difficult to visualize the product ahead of time, and some concerns have only arisen when the demonstrations have been shown*". One customer linked the review and retrospective to accountability by the software developers for what work they take on. "*This is a great opportunity for the programmers to demonstrate the accomplishments and leads to proving their own accountability on their tasks that they took on*".

Since many of the customers did not get very involved with the software development decision process with much depth until after Scrum was being used by the development group, many of them either did not comment on how the transition went or said that they came on board after Scrum was introduced. One customer was part of the previous process and commented that the previous way of developing software was so bad that they were willing to try anything. "*I believe the complete frustration with the petrocube website release made it clear that a more responsive system was needed and most were ready for anything that would have made the system better*".

The customers found that using the Scrum master / project manager in a business analyst role between sprints helped them out a lot to be more prepared for the planning meetings. "*It has lead to a more timely completion of the planning stages earlier. In the first few sprints, we were never prepared when it came to being ready for the next planning session. But with the project manager actively starting the planning and long term visions ahead of time has made this planning sessions easier*".

The customers also found that the daily Scrum meetings allowed them to be kept up to date on the progress of the software development and to be informed on issues as they happen rather than at the end. "*Good forum for hearing progress updates and what issues/problems are lurking*". Another customer commented that "*They have helped us as customers stay in the loop and have a better idea of when I should expect questions*"

When asked if there were any difficulties with using the Scrum process one customer said that it was "*too ridged*" while another said that sometimes it was difficult for him to understand what tasks the developers were doing at times.

Developer Opinions: In this section we will present the developer views and opinions about different aspects of the Scrum process.

The developers found the Scrum process very beneficial. From the questionnaire every developer would recommend using Scrum on projects in the future. There was a range of satisfaction about the software before Scrum was introduced. Some developers were satisfied with the software while others were very unsatisfied with it: "*Some amazing work was done before the Scrum process was introduced. Not necessarily the best code*" while others said "*I was very unsatisfied with the product developed before the Scrum process and would not like to be associated with any of the products produced at that time*".

After the Scrum process was introduced, the developers were more satisfied with the products being produced: "*I am very satisfied*" and another developer said "*Very satisfied with the software product(s) being developed*".

The developers saw the Scrum process had fostered more customer involvement and communication: "*It promotes better communication with the client*" and "*It is useful to see customer representatives everyday, since there are always questions that could be asked…and it makes me more confident in what we're doing because customer always has up-to-date information about the progress*".

The developers also found the Sprint planning, the review and the retrospective meetings helpful to them: "*…These meetings are useful since we can choose the scope of work, although guessing on the time for each backlog it is not always easy and time estimates do not always come out right*". Although the developers found the planning meetings useful they did notice some room for improvements: "*The only negative is that these* [meetings] *take a little too much time and it's hard to concentrate*". Another developer noted the customers were not as prepared as they should be "*…I feel that our Scrum master/project manager and customer are not prepared enough for them* [meetings]*, so the meeting drags on*".

This feedback was noticed in previous sprints so for Sprint 8 the project manager worked with the customer during the sprint. The developers had this to say about the project manager working with the customers: "*Someone (being the project manager or a BA [business analyst]) needs to help the client formalize their thought on what they want. The development team should not decide for the client on what needs to be done. The client needs to make the decision. In order to have the client make a thoughtful decision all alternatives need to be explained in the client's language as well as pros and cons for each option. So I guess my answer is that it did help. There is more work to do on that aspect, but generally the last sprint planning was much better that way*"

The developers found the review and retrospectives useful. One developer liked the review because it was a place to show off what they had done and get feed back on it: "*Useful, because not only the customer can see what we have achieved so far, but all the developers can see how it all works (or doesn't work) together. Plus we can hear some useful questions/suggestions for the improvement*". Another developer compared the review meeting to the previous process where there were weekly meetings: "*It replaces a weekly meeting and is more efficient*".

For the most part the developers like the retrospectives but there was still the problem of the meeting not being as focused as it should be. "*Sometimes we don't keep focused and it goes on too long, but it's a good concept*".

The developers found the daily sprint meeting useful but again found that sometimes the meetings can drag on if the team is not focused on doing them quickly. In addition to commenting on the process the developers commented on their confidence in the software they are producing after Scrum was introduced: "*…before Scrum I was not as confident since nothing was reviewed by the manager who gave the work and I had no exact deadline. The Scrum process makes me more confident since I know exactly what I'm doing and when it is due (and that it is probably doable in the time given)*". Another developer said this about their confidence: "*The Scrum process is giving me confidence that we are developing the software that the customer wants…*"

The developers were asked about the problems they found in transitioning to the Scrum process. The developers found that the customers not knowing what they want, and not being able to describe what they want to them was a large problem.

Customer Opinion Discussion: The customers liked the Scrum process and the changes it had brought. They found that the daily Scrum meetings keep them up to date and the planning meetings were helpful because they reduced the confusion about what should be developed. The customer's attitude toward the software changed from one of ambivalence to becoming involved and invested in what was being developed for them. The customers noted that in the beginning they were not prepared for the Scrum planning sessions. The solution to this problem was to have the project manager work with the customers between sprints. Since this was introduced, the customers have been more prepared and the Scrum

meetings have been more useful to both the customers and the developers. In addition, customer responses indicate that the customer's satisfaction has increased over previous products developed at the company. An important lesson learned from their responses is that the customers should also be trained in the Scrum process not just the developers so that they can understand the new expectations imposed upon them.

Developer Opinion Discussion: The developers found the introduction of Scrum helpful. They mentioned how it helped them have a better idea of what they were working on and when requirements needed to be completed. The developers noticed in previous sprints that the customer was not prepared for the Scrum planning meetings. The meetings would drag on without much being accomplished because of the lack of preparation. Therefore, the developers saw the need for having someone work with the customers between the sprints to help formulate their ideas with enough detail that the developers could use for estimates.

5. Summary and Future work

There are two main contributions in this paper. The first contribution is the presentation of empirical results from the case study showing that after Scrum was introduced the customer satisfaction increased while at the same time overtime for the developers decreased (allowing the developers to work at a sustainable pace). The second contribution is the formulation and testing of a hypothesis based on observed industry information. In the future, we would like to test the presented hypothesis in additional case studies.

6. Acknowledgements

The authors would like to thank the individuals on the PetroSleuth Inc. development team for participating in the study. PetroSleuth Inc. and the Natural Sciences and Engineering Research Council of Canada supported the project.

7. References

[1] Klein, Heinz K. and Michael D. Myers, **A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems**, *MIS Quarterly* 23 (1): 67-94 (1999).

[2] K. Schwaber, ***Agile Project Management with Scrum***, Microsoft Press, Redmond, WA, 2004

[3] J. Sutherland, **Inventing and Reinventing SCRUM in five Companies**, http://www.agilealliance.org/articles/articles/InventingSCRUM.pdf , Accessed March 10 2005

[4] M. Beedle, K. Schwaber, **Agile Development With Scrum**, Prentise Hall, 2001

[5] http://www.VersionOne.net

[6] http://www4.ncsu.edu/~lmlayma2/xpef.html

Appendix A

The weekly mean percent overtime is calculated as follows:

% Overtime = (Actual Hours/Expected Hours) -1

Where

Actual Hours: Sum of hours worked from (Sunday to Saturday) of a given week

Expected Hours: The number of hours the developers should have worked as a team in a given week.

We will now discuss how the expected hours were calculated. To calculate the expected hours we first have to calculate how many person days should be included in the week. We start out by assuming that all the developers will work every day of the week,

From this perfect week, we then subtract the administrative days recorded as the developers are not expected to work on those days. In the case of the salaried employees these administrative days are coded as sickdays, vacation, courseday, flexday, compassionate leave.

For persons on contract, they do not enter administrative days into the system. If they have a sick day or want to take a holiday then they would not enter time for the day. For this paper, we will assume that if the contract employee did not enter a day then it will be counted as an administrative day and will be subtracted from the perfect work week.

The final expected hour calculation is as follows:

Expected Hours = Expected Person Days *7.5

Expected Person Days = PerfectWeek – Administrative Days