An Empirical Study of Using Planning Poker for User Story Estimation

Nils C. Haugen
Objectnet as
Ruseløkkveien 14
0251 Oslo, Norway
nch@objectnet.no

Abstract

Group estimation of user stories is an important part of Extreme Programming (XP), used for both planning releases and iterations. Research has shown that although group estimation in many cases is superior to individual estimation, there is still room for improvement. For instance, group estimation performance can be reduced by dominant personalities and anchoring effects. Through the analysis of 101 user story estimates, made by an XP team for release planning, we investigate whether the introduction of the planning poker estimation process improved the estimation ability of the team. The results show that planning poker improved the team's estimation performance in most cases, but that it increased estimation error in the extreme cases.

1. Introduction

A recent review of estimation surveys [1] documents that little progress has been made in the area of estimation performance over the last decades. This is a major concern for the software industry, as lack of estimation performance often causes budget overruns, delays, suboptimal resource allocation, lost contracts or poor quality software. Because of these rather dramatic consequences, there is a high demand for improved estimation processes.

Agile software processes try to minimise the impact of insufficient estimation accuracy by ensuring that the most important functionality is developed first. This is achieved through a flexible development process with short iterations. However, there is still a need for accurate estimates, as these are the basis for staffing, planning, prioritisation, bids, and contract negotiations.

In the agile software development method Extreme Programming (XP) [2], a user story is the unit of which software features are estimated and developed. A user story describes a feature of the software, and is composed of three aspects: a written description, conversations, and tests [3]. Each story is formulated in one or two sentences in the language of the customer, and typically written on an index card. Rather than conveying all details about the feature, details that are likely to change during the project, the cards serve as reminders for conversations to be had about the features. The details are then fleshed out in the conversations, and conveyed and documented in the form of tests.

User stories are estimated for release and iteration planning [4]. The goal of release planning is to let the customer choose the stories to be implemented in the next public release. The release planning game typically involves estimating 2-3 months worth of stories. The estimates are important, as they are the basis for planning the next release in terms of prioritising features and staffing the development team. In some situations, the estimates are also used for contract negotiations and bidding.

In the iteration planning, user stories are broken up into development tasks. Responsibility for each task is assigned to a team member. Estimates at this level are detailed, but as weekly or biweekly iterations are most common, the number of stories involved in each planning session is limited. In addition to providing new estimates for the user stories that can be used for more detailed planning, the re-estimation process provides a setting for the team to discuss design decisions and implementation strategies.

As user stories are commonly estimated using group processes, user story estimation may enjoy improved estimation accuracy compared to individual estimation processes [5]. However, with unstructured group estimation processes (which is often the case), factors such as company politics, group pressure, anchoring, and dominant personalities, may reduce estimation performance.

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

One approach that may reduce such problems is the semi-structured planning poker estimation process [3, 6, 7]. In this process, the customer first explains each user story to the group of developers. The developers then discuss the work involved in implementing the user story to the point where everybody feels they have enough information to estimate the required effort. All the developers then estimate the user story independently, and reveal their estimates simultaneously. Next, the developers with the lowest and highest estimate justify their estimates, and the group continues the discussion in order to decide on a collective estimate, possibly by conducting one or more additional rounds of individual estimating.

Planning poker ensures that all developers participate in the estimation process, and that everybody's opinion is heard, regardless of whether they are among the loudest or most influential people in the group. With unstructured group estimation, there is always the possibility that some developers will refrain from participating, for example if they perceive themselves as not having good enough estimation skills, or feel that they do not have enough information about the task at hand. A study on expert estimation of web-development projects has shown that such issues can hinder estimation performance [8]. The study found that the people with most knowledge about how to solve a task, not necessarily are the best at estimating the required effort. The same study also showed that people with less knowledge in the area employed different estimation strategies that provided different perspectives on the required effort, improving the group estimate.

Planning poker can also reduce the potential problem of so-called anchoring. Anchoring is the impact someone putting forward an estimate or expectation on cost has on subsequent estimates, even if the estimate/expectation is unrealistic. So in unstructured group estimation, the first estimate suggested for a task can impact all subsequent estimates made by the other developers. Planning poker avoids this by revealing all individual estimates simultaneously.

The goal of this study is to empirically examine whether introducing the planning poker process into the user story estimation performed as part of the release planning improves estimation performance compared to unstructured group estimation.

The reminding parts of this paper are organized as follows: Section 2 describes the study design, Section 3 presents the results, Section 4 discusses the results, and Section 5 concludes the paper.

2. Study

The study investigates whether a development team that applied the planning poker estimation process achieved improved performance compared to when they used unstructured group estimation. This is investigated in a retrospective case study of estimates made for a software system for taking and maintaining orders for home broadband service.

Data was collected from four subsequent releases of the system. An unstructured group estimation process was used for the first two releases in the study, and the semi-structured planning poker estimation process was used for the latter two. Each of the four releases was developed as a separate project. The system went live approximately 7 months before work with the first of the four releases in the study started. There had been three major releases of new functionality during that time, for which unstructured group estimation had been used.

Planning poker was introduced before the third release in the study. The goal was to make the estimation sessions more effective and to get more involvement from the whole team during estimation sessions. The team immediately took a liking to the new estimation process, and used it for all subsequent releases.

The author participated in the team and was responsible for introducing the planning poker estimation process.

The reminding part of this section describes the environment in which the projects were conducted, the two estimation processes that were used, and how data were colleted.

2.1. System and team

The projects were developed by a team responsible for maintaining a set of applications for registering and managing orders for home broadband service. The applications are supporting four sales channels; the public web site, the customer web site, dial-up customer services and a CD-ROM installation software distribution.

The system was developed in Java and JavaScript on a BEA WebLogic platform with a Sybase database server. A large number of open source libraries and frameworks were part of both the runtime and the development environment.

The development team consisted of a mix of customer employees, consultants from international consulting companies and independent contractors. The team size varied slightly between the releases in the study, counting between 15 and 20 people, with the

number of developers varying between 8 and 11. A total of 19 different developers where involved in the four projects, and of these, 10 had been involved in the initial development of the system. All were experienced developers that were familiar with Java and agile methods, and all had more than three years of professional experience.

2.2. Development process

The team was using Extreme Programming with user stories, release planning game, weekly iterations, daily stand-up meetings, pair programming, test-driven development, collective code ownership, continuous integration, and automated acceptance tests.

Each release included between 6-8 weeks of development time and was kicked off with a release planning game meeting where the customer and business analysts presented all user stories to the team. In this meeting, the stories were estimated by the developers with the customer, business analysts and project manager present to answer questions and verify assumptions. User stories were written on index cards, and the estimates were noted in the lower right corner of each card.

When estimation of all user stories was completed, the user stories were organised into iterations. The order of implementation was decided by the team and was based on multiple factors. The customer and business analysts would focus on getting the user stories with the highest business risk or priority done first, developers would want to start with the user stories with highest technical risk or stories that served as the foundation for others, while testers would argue for getting the user stories that were most difficult to test done in the early iterations.

User stories were developed in a test-driven fashion using pair programming. Usually, one pair would stick together to see the story through, but occasionally developers would re-pair one or more times per story.

When the developers had completed the implementation and automated tests for a story, this was demonstrated to the tester responsible for testing this user story. If the tester spotted something that needed to be either fixed or clarified during the demonstration, the developers would resolve the issues and do a new demonstration once completed. When the tester was happy with the demonstrated feature, the story card would get a blue sticker to signal development complete, and the tester would start manual testing of the story. At this point the number of days used in development was written on the story card next to the original estimate. A box would be drawn around this number, i.e. the actual effort, to distinguish it from the circled estimate.

After testing, the tester would put a green sticker on the story card if no bugs were found. Otherwise the card would get a red sticker, and the bugs would be registered in the Jira bug tracking system. Time taken to fix the bugs was not included in the actual effort for each user story. Instead, each release planned with having a number of developers working on bug fixing in the final iterations of the project.

This process was used for all releases in the study. An unstructured group estimation process was used for the first and second release in the study, while the planning poker estimation process was introduced and used for the third and fourth release.

2.2. Unstructured group estimation process

All user stories were estimated in the initial release planning game meeting. The developers estimated the user stories, and the customer, business analysts and the project manager were present in the estimation session to answer questions. Each user story to be estimated was first discussed in plenum to identify the work involved.

After the initial discussion, one or more developers would volunteer an estimate. Consensus was then sought among the rest of the team by asking each developer present whether they agreed. In case of disagreement, the effort would be discussed further until consensus was reached.

If the developers believed the effort for a story exceeded four ideal pair days, or identified shared functionality between two or more stories, these stories were split into several tasks. Each task was then estimated separately using the same process.

When all the user stories had been estimated, the team would plan the stories into weekly iterations. The same productivity was assumed for all pairs of developers.

From observing the unstructured group estimation process it was clear that not all developers present in the session participated equally in the discussion and estimation. A few were very vocal and involved in the discussion, and when it was time to suggest an estimate, it was the same few who eventually suggested an estimate first. Some of the others would offer their suggestions voluntarily, while others remained silent until asked specifically.

It was obvious that the process of first suggesting an estimate was quite slow. Even the most vocal developers were a bit reluctant to sticking their head out and suggest a number.

As a consequence of these observations, the planning poker estimation process was introduced to the team.

2.3. Planning poker estimation process

With the planning poker estimation process, each developer was given index cards with the numbers 0.5, 1, 2, 3, 4 and >4 written on them. After the initial discussion that lasted until the developers felt they knew enough about the user story to estimate the required effort, all developers would estimate individually and simultaneously show the card corresponding to their estimated number of pair days for this story.

If the estimates differed, the developers with the lowest and highest estimate would justify their estimate. A short discussion would follow, where the developers sought to agree on a final estimate for the task. Usually the developers would reach agreement fairly quickly. In the few cases where they did not, the project manager, playing the role of moderator, would make a decision.

Initially, a second round of individual estimating was performed after the justification of the highest and lowest estimate, in order to get a new baseline for discussion. This was quickly perceived as being unnecessary, and was replaced with the developers verbally stating their new estimate during the discussion towards consensus.

If the developers estimated a story to more than four ideal days, the story would be broken into tasks that in turn would be estimated and implemented separately.

Developers who felt they did not have any knowledge of what was involved in implementing a specific user story, could choose to refrain from estimating. This very rarely happened. Due to pair programming and collective ownership, all developers had to some degree participated in all the different types of tasks and been exposed to most of the code base.

The team immediately took a liking to the planning poker estimation process, even without knowing whether the process improved their estimation ability. Although nothing was measured in this respect, planning poker was perceived as being more effective, thus reducing the time spent discussing and estimating the effort required for each user story. The process also eliminated the problems of unequal participation in the estimation session that was observed for the unstructured group estimation, and also removed the uncomfortable situation of coming up with the first suggestion for an estimate. The team found planning poker to be a fun way of carrying out the estimation session.

After the introduction, planning poker was used for estimating the user stories of all subsequent releases.

2.4. Data collection and measures

The data used in this study is based on the recording of estimates and actual effort written on the story cards. Estimates and actual effort are in pair days. All estimates are of type most likely estimates (that is, with no contingency allowance, etc). There was 0.5-day contingency added per iteration, and 1-2 iterations reserved for bug fixing, late requirements and changes at the end of the project.

Smaller changes in functionality and changes in implementation strategy was quite likely happening in all the projects. Major changes in scope and functionality after the planning game was added as new tasks and estimated separately. There is no indication of any significant difference between the projects in this respect.

We use RE (Relative Error = (actual effort - estimated effort) / actual effort)) and MRE (Magnitude of Relative Error = |actual effort - estimated effort| / actual effort) as measures of estimation error.

2.5. Threats to validity

Two major threats to the validity of this study are inaccurate recordings of actual effort and learning.

Due to the informal way actual effort was collected, there is a risk for inaccurate recordings. Developers might have forgotten to write up the actual right away, and when writing it up later failed to recollect the actual effort precisely and thereby settling for a number that they felt was roughly right. The team lead was responsible for checking that actual effort had been noted, and reminding everybody to do it before the end of the weekly iteration.

As iterations were short, and there is no indication that failure to record actual effort immediately was a bigger problem for either the projects using unstructured group estimation or those using planning poker, we do not believe that this has biased the results of the study.

There is also a chance that multiple user stories occasionally were implemented simultaneously by the same pair, and actual effort being signed to only one of the stories or divided roughly between the stories in question. We have no indication that this was worse for either of the projects in the study.

The way actual effort was measured in the study did not include the taken to fix bugs that were discovered

after a feature was successfully demonstrated to the testers. Although actual effort did include development of automated acceptance tests for the feature, bugs were of course still discovered in exploratory testing. Unexpected bugs and faults can be an important source for effort overrun. However, the estimates in this study did not include time for fixing bugs discovered by exploratory testing. The projects handled this by adding 1-2 iterations for contingencies to the total project time. Outstanding bugs did not delay the release date for any of the projects in the study.

It is possible that estimate precision changed as a consequence of other reasons than the new process, like better knowledge of the system, more experience with a wider range of tasks, and in general better practice in estimating user stories.

The results of this study should be validated by experiments where learning can be controlled, for instance by randomized allocation to treatment.

3. Results

Results from the two projects using unstructured group estimates and the two projects using planning poker were initially analysed separately, but are grouped in the analysis in order to improve readability.

This section presents the data reduction and descriptive statistics for the results, identifies impacting factors, and isolates the studied factor.

3.1. Data reduction

Three cards were removed from the analysed unstructured group data set because the stories did not involve any development work. Consequently both the estimated and the actual effort for these cards were 0 pair days.

In addition there were 21 story cards for which information about actual effort was not reported. Actual effort was collected mainly for planning purposes. Developers sometimes forgot to update the story card with actual effort. Usually, they were reminded to do this at the end of the weekly iteration. However, some story cards slipped through this control, perhaps especially in the final iterations, when overrun no longer represented a big threat to the project. We do not suspect that story cards were left without actual effort because developers did not want it to be reported, for example because the overrun was particularly high.

Five story cards were removed from the planning poker data set because the estimated effort was based on assumptions that were not fulfilled, two were removed because the story did not involve any development work, and hence were estimated to 0 pair days. Eight story cards were left out because actual effort was not reported.

3.2. Descriptive statistics

Descriptive statistics for the two sets are displayed in Table 1 and 2. For both sets, effort was measured in pair days.

Table 1. Descriptive statistics

		Median		Mean	
Estimation process	**n**	**RE**	**MRE**	**RE**	**MRE**
Unstructured group	51	0.00	0.33	-0.08	0.35
Planning poker	50	0.00	0.25	-0.26	0.48

Table 2. Estimation error distribution (MRE)

		# tasks in interval		
Estimation process	**n**	**< 0.20**	**0.20 – 0.39**	**≥ 0.40**
Unstructured group	51	16	16	19
Planning poker	50	23	5	22

Straightforward analysis of the results suggests that the planning poker estimation process is somewhat better for the common case (median MRE) with an estimation error of 0.25 vs. 0.33 for the unstructured group estimation process.

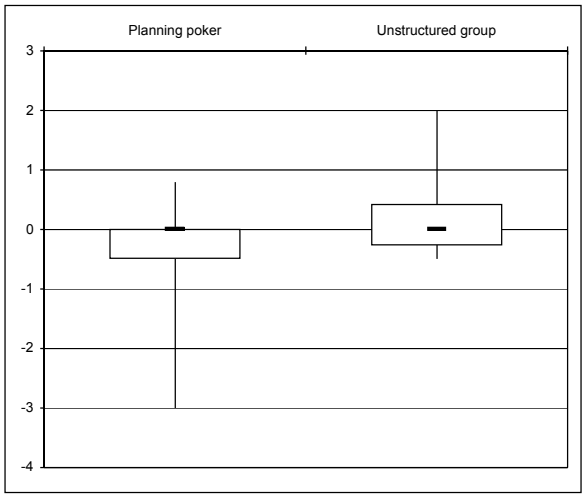

Figure 1. Box plot

The results further suggest that planning poker has larger errors in the extreme cases (i.e., higher Mean MRE), and that the direction of the error points towards overestimation (i.e., negative Mean RE). This is also illustrated in the box plot in Figure 1

3.3. Identification of impacting factors

Many other factors than estimation ability can impact estimation error (e.g., skills in cost control). In order to make a proper analysis of the estimation ability of the planning poker process, we need to isolate the studied factor (in this study, estimation ability), as systematic differences in the factors not studied might otherwise lead to flawed conclusions.

The analysis in this section is based on the framework for analysis of estimation error by Grimstad and Jørgensen [9].

We are studying the estimation ability of the unstructured group and planning poker estimation method. This means that we need to identify other factors that can impact the results, i.e. factors that systematically differ between the two projects using unstructured group estimation and the two projects using planning poker. Table 3 presents the factors listed in [9], as potentially impacting estimation that we believe is unlikely to impact the results in this study.

Table 3. Factors unlikely to impact the results

Factor	Comment
Skills in use of estimation model	Both estimation processes are intuitive, and we observed no misuse of either model.
Skills in selection of estimation model	Not relevant.
Project managers' ability to control cost	There were different project mangers, but we observed no large differences in the ability to control cost.
Project member skills	All project members were experienced developers, and new team members were rapidly introduced to large parts of the system through pair programming.
Client and subcontractor performance	Same client for all projects, and there were no subcontractors.
Completeness and certainty of the information the estimates were based on.	We observed no significant differences, but a few tasks have been excluded from the analysis because they were based on assumptions that were not fulfilled.
Project priorities	All projects had a strong focus on time-to-market, and they all had some flexibility in scope.
Terminology and measures	All estimates are most-likely estimates, and included the effort required to implement the functionality together with automated functional tests and unit tests.
Recording of data	From our observations of the projects we believe actual effort was honestly recorded, and because of the coarse granularity of time used for recording estimated and actual effort with a resolution of 0.5 days, we believe the accuracy of the numbers is decent. In any case, we have no indications of any systematic biases in either direction.
Selection of projects	The projects represent the two projects immediately before and after the introduction of the planning poker estimation process.

Ignoring these factors leaves us with the following three factors that we believe might impact results:

- Size. Estimated effort spans from 0.5 to 7 days, although the majority falls in the 0.5 to 2 days range. Smaller tasks are easier to estimate. There is a chance that there was a systematic difference between the projects in this respect, requiring this factor to be isolated.
- Experience with similar tasks. Most user stories were familiar to the developers, but for a few there were no former experience in working with the particular type of functionality or technology relevant for the task. This means that there might have been a systematic difference between the projects, and this factor needs to be isolated in the analysis.
- Estimation experience from this system. Although all projects in the study were performed by the same team, there were changes in the team between the projects that might have impacted the results as a change in expert judgement skills. About half of the consultants that had been involved in initially building the system were being replaced with customer employees and independent contractors. Although the level of general software development experience was the same for the people coming on to the team, they did not have the system experience of the people rolling off. The level of system experience for each of the teams should therefore be taken into consideration in the analysis.

3.4. Isolation of studied factor

Our analysis of impacting factors suggests that three factors need to be especially considered in the analysis; task size, experience with similar tasks and expert judgement skills. We chose to isolate the first two by grouping the data according to task size and task experience. Grouping is obviously not an option for the third factor, as estimates were made by the entire team. Unfortunately, data cannot be easily adjusted for

differences in system experience between the teams. Our chosen strategy for this factor is therefore to investigate the level of system experience for each of the team members participating in the four projects, in order to identify any significant differences that should be taken into account when interpreting the results.

Along the size dimension, estimated and actual effort for each task was divided into those with an estimated effort of less than 2 pair days (small) and those with an estimate of 2 pair days or more (large). Along the prior experience dimensions, data was grouped according to whether the team had experience with the technology or type of functionality from similar tasks or not. The results after grouping are presented in Tables 4-7.

Table 4. Small tasks with prior experience

Estimation. process	n	Median		Mean	
		RE	MRE	RE	MRE
Unstructured group	30	0.00	0.42	0.12	0.39
Planning poker	21	0.00	0.25	-0.40	0.50

Table 4 displays the results for small tasks where the development team had prior experience from similar tasks. The majority of data fell into this group. The results show that the planning poker estimation process has better precision in the common case, with an estimation error of 0.25 compared to 0.42 for the unstructured group estimation process (median MRE).

Table 5. Small tasks without prior experience

Estimation process	n	Median		Mean	
		RE	MRE	RE	MRE
Unstructured group	4	0.50	0.50	0.58	0.58
Planning poker	7	0.67	0.80	0.13	0.70

Table 5 indicates that the estimate precision for small tasks decrease when the development team has no prior experience with the type of functionality or technology estimated. The planning poker process seems to be more impacted than the unstructured group process (0.50 vs. 0.80). Notice however that the sample is quite small in this category, especially for the unstructured group data set, so the results should be used with care.

Table 6. Large tasks with prior experience

Estimation process	n	Median		Mean	
		RE	MRE	RE	MRE
Unstructured group	14	0.00	0.25	0.05	0.23
Planning poker	16	0.00	0.00	-0.16	0.21

Table 6 suggests that the planning poker estimation process is superior for estimating large tasks where the development team has experience from similar tasks (0.00 vs. 0.25 for the unstructured group process, median MRE).

Table 7. Large tasks without prior experience

Estimation process	n	Median		Mean	
		RE	MRE	RE	MRE
Unstructured group	3	-0.40	0.40	-0.24	0.30
Planning poker	6	-0.20	0.58	-0.43	0.87

There are few large tasks where the developers have no experience with this type of task, especially for the unstructured group data set. We see in Table 7 that the results for the planning poker process have higher extremes than the unstructured group process (mean MRE), and also that the unstructured group process performs better than planning poker with an estimation error of 0.40 vs. 0.58 median MRE.

Table 8. Team system experience

Project	Estimation process	n	# devs with 0-6 months system exp.	# devs with >6 months system exp.
1	Unstructured group	14	2	8
2	Unstructured group	37	3	8
3	Planning poker	12	2	4
4	Planning poker	38	5	3

As discussed earlier, differences in the level of system experience for the developers involved in the estimation process might have impacted the results. Table 8 displays the number of developers with various level of system experience for each of the four projects. We see that there were more developers with long system experience on the projects using the unstructured group estimation process than on the projects using planning poker, with 8 developers with more than 6 months system experience for both the projects using the unstructured group process vs. 4 and 3 for the projects using planning poker respectively. This means that the results might be biased towards a too positive view of the unstructured group estimation process.

4. Discussion

The results suggest that 1) planning poker is more accurate when the team has previous experience from similar tasks, 2) planning poker is possibly less

accurate when there is no previous experience from similar tasks, and 3) planning poker increase extremes. Each of these results is discussed below.

Planning poker is more accurate when the team has previous experience from similar tasks

The results indicate that the estimation ability of the planning poker process is best when the development team has prior experience with the technology and functionality involved in the user stories. The results are better than suggested in the straight-forward analysis; estimation error of 25% vs. 33% for planning poker and the unstructured group process respectively, compared to 25% vs. 42% for small tasks and 0% vs. 25% for large tasks after blocking.

We believe the better performance of planning poker can be explained by the following observations:

a) In the estimation sessions using planning poker, we observed more questions and more information sharing regarding the user story and work involved compared to the sessions using unstructured group estimation. This is an effect of all developers participating in the estimation, and we believe it is quite likely that this helped the team uncover more information relevant to the estimate, thus leading to better estimates.

b) Technical competence usually induces a bottom-up, construction based estimation strategy, something that not always lead to better estimation performance [8]. In planning poker, all developers offered an individual estimate, regardless of how well they knew the technical requirements for the task. It is possible that estimators without this detailed implementation knowledge apply a different estimation strategy than those who do have this knowledge, thereby contributing different perspectives to the estimate and enhancing it's precision. For example a top-down approach can be more likely to take historical data into account than a bottom-up approach.

c) Displaying all individual estimates simultaneously, and thereafter justifying both the highest and lowest estimate introduces compensating anchors, and this would prevent the individual estimates from being biased by the first estimate suggested.

d) The estimates better reflect the team's average ability to solve the tasks. Because the estimate is decided by taking the opinion of all developers into account, the team's ability to implement the desired functionality is factored into the estimate [5]. In the unstructured group estimation process, only a few developers would usually suggest estimates. Individual estimates are often based on how long the individual developer thinks she would need to implement the story, and not how long she thinks the team in general would take to do it. So if the developers suggesting estimates in the unstructured group process happened to be experts, their estimates could be over-optimistic. As there was nothing encouraging developers with less experience to suggest an estimate independently, there was no mechanism for correcting this effect.

e) The whole team feels ownership to the estimates, and as a consequence takes more responsibility for solving the tasks within the estimated time. Because everybody is sharing his or her individual estimate, and the final estimate is made on the basis of the opinion of all the developers in the team, we believe it is likely that all developers feel a personal stake in the estimate. Before planning poker was introduced, it is possible that some developers could have quietly disagreed with the estimate that was put forward, and when later made responsible for implementing that particular story, they would believe that the estimate was unreasonable and thus not feel as obliged to solve it "on time".

f) An important part of estimating is making assumptions about the context in which the estimates are valid. These assumptions will then need to be fulfilled in order for the estimates to be realised. We believe that the team discussion inherent in the planning poker process is better at uncovering and clarifying more assumptions in the estimates. As the team becomes more aware of these assumptions, they will also work better together to fulfil them.

Planning poker is possibly less accurate when there is no previous experience from similar tasks

The results indicate that the planning poker process is less accurate when the development team do not have prior experience with the task at hand. However, this scenario represents the smallest sample in the study, so this result must be interpreted with caution.

Possible explanations to why planning poker fails in this respect, is:

a) If the developers happen to suggest similar estimates for a story where they have no prior experience, they might feel a false sense of security that the estimate is reasonable.

b) The opposite could also be true, that the developers become more cautious, taking the various eventualities mentioned by all developers into account, and therefore overestimating the complexity of the task.

Planning poker increase extremes

Another interesting observation is that with planning poker, it is more likely to go totally wrong. This is evident from the extremes being consistently high for all planning poker results.

The higher extremes for planning poker results could be a consequence of the difference in system

experience between the teams using unstructured group estimation and the teams using planning poker. With more experienced developers participating in the estimations sessions, we can assume that the total system knowledge were higher. So for the projects using planning poker, even if they were familiar with the tasks at hand, they were not as familiar as the teams using unstructured group estimation, and consequently their estimates would sometimes be off by quite a bit.

Group polarisation is a well-known effect of group discussions. After participating in a discussion, members tend to advocate more extreme positions than individuals who did not participate in any such discussion. This effect offers a possible explanation to why planning poker increases extremes, as planning poker generated more group discussion than the unstructured group estimation process.

5. Conclusion

This study suggests that planning poker can improve estimation performance. Analysis of 101 real-world estimates, from four projects, made by the same agile development team, suggests that estimation performance is better with planning poker than for an unstructured group estimation process. However, the results indicate that planning poker can reduce estimation performance when the team lacked experience with estimation of similar tasks, and also, extreme values seem to be more extreme with planning poker than for unstructured group estimation.

This study is limited to one team within one company, and there is an unknown effect of system experience. Despite these limitations, we believe that the study indicates that using planning poker for user story estimation might provide more realistic release planning, and more optimal resource allocation. However, more research is needed.

6. Acknowledgements

I would like to thank Stein Grimstad and Dr. Kjetil Moløkken-Østvold at Simula Research Laboratory in Oslo, Norway, for suggestions and critical comments.

7. References

[1] K. Moløkken and M. Jørgensen. “A review of software surveys on software effort estimation”, *International Symposium on Empirical Software Engineering*, Rome, Italy, 2003.

[2] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed, Addison-Wesley, Boston, MA, 2004.

[3] M. Cohn, *User Stories Applied: For Agile Software Development*, Addison-Wesley, Boston, MA, 2004.

[4] K. Beck and M. Fowler, *Planning Extreme Programming*, Addison-Wesley, Boston, MA, 2000.

[5] K. Moløkken-Østvold and M. Jørgensen, “Group Processes in Software Effort Estimation”, *Journal of Empirical Software Engineering*, 9(4):315-334 (2004).

[6] J. W. Grenning, *Planning Poker*, 2002, http://www.objectmentor.com/resources/articles/PlanningPoker.zip. Accessed 17th April 2006.

[7] M. Cohn, *Agile Estimating and Planning*, Prentice Hall PTR, Upper Saddle River, NJ, 2005.

[8] K. J. Moløkken-Østvold and M. Jørgensen, “Expert Estimation of the Effort of Web-Development Projects: Are Software Professionals in Technical Roles More Optimistic Than Those in Non-Technical Roles?”, *Journal of Empirical Software Engineering*, 10(1):7-30 (2005).

[9] S. Grimstad, M. Jørgensen, "A Framework for the Analysis of Software Cost Estimation Accuracy", submitted to ISESE 2006.