Extreme Construction Game to Improve the Teaching of Extreme Programming Practices

Barbara Weber

Department of Computer Science

Univ. of Innsbruck, Austria

1. Introduction and Problem Statement

Teaching software development methods is a challenging task. My experience from previous semesters shows that it takes several weeks until students fully understand and live the taught method. Some of the students even never apply the method correctly. In the software engineering classes I am teaching (i.e., bachelor program of Computer Science at the Univ. of Innsbruck and Management Center Innsbruck in the program of study “Management and IT”) the agile software development method Extreme Programming (XP), which is based on four values and consists of 12 practices (cf. Section 2.1), is taught.

As I was not completely satisfied with how XP was applied by the students in the past couple of semesters the question arose whether the learning of XP and its 12 practices can be improved by performing a simulation of XP (called Extreme Construction Game – cf. Section 2.2) early in the course. As part of the Extreme Construction Game a mini-project (lasting for 3 hours to one day) is performed where most of the XP practices are applied to a non software project. As one of the reasons for not fully applying XP with all its practices could be a lack of understanding of the method, applying the Extreme Construction Game seems to be promising. The experiences made in the simulated project might help the student when working on a real software project later on in the course.

To evaluate the impact of the Extreme Construction Game on the learning of XP, the students had to self-assess their level of expertise regarding the 12 XP practices before and after as well as at the end of the course. The results of the experiment show that playing the Extreme Construction game leads to a better understanding of XP by the students. They further show that missing understanding was not the only reason for not fully applying XP, but missing technical skills hinder the application of XP significantly as well.

The subsequent paper is structured as follows. Section 2 provides some background information on Extreme Programming and the Extreme Construction Game. Section 3 then describes the experiment design, while Section 4 describes the experiment and its results in detail. Finally, Section 5 provides a summary and draws the conclusion.

2. Background Information

This section provides background information which is required for the further understanding of this paper. Section 2.1 gives a brief introduction to Extreme Programming (XP). Section 2.2 describes the Extreme Construction Game and Section 2.3 shows which XP practices are reflected by the Extreme Construction Game.

2.1 Introduction to Extreme Programming

Extreme Programming is an agile software development method and is based on four values: simplicity, communication, feedback, and courage. It consists of 12 practices, which are *Planning Game*, *Small Releases*, *Metaphor*, *Simple Design*, *Test-Driven Development* including *Customer Tests*, *Refactoring*, *Pair Programming*, *Collective Ownership*, *Continuous Integration, Sustainable Pace*, *Whole Team* and *Coding Standards* (cf. Fig. 1) [Jeff01].

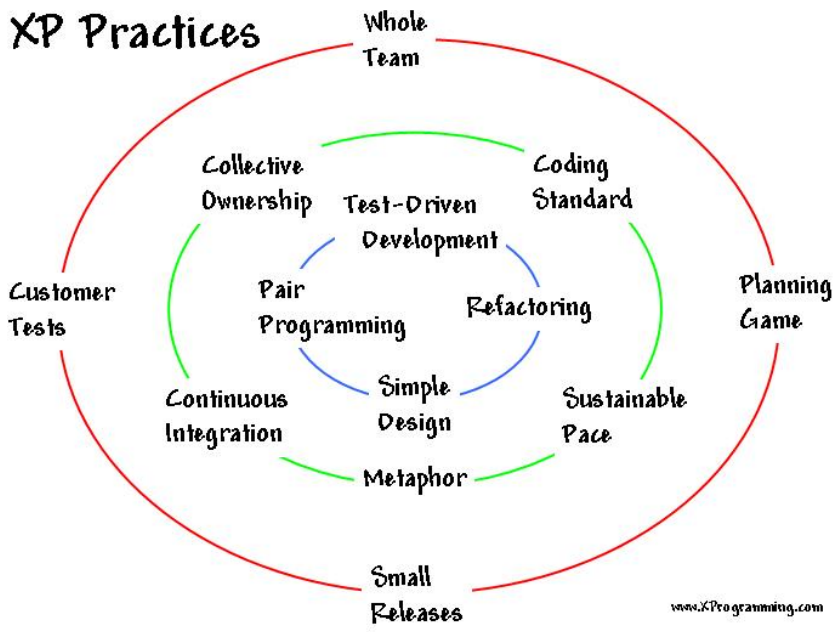

Figure 1: Extreme Programming Practices [Jeff01]

- **Core Practice: Whole Team**

As described in [Jeff01], every contributor to the project is an integral part of the "Whole Team". The team forms around a business representative called "the Customer", who sits with the team and works with them daily.

- **Core Practices: Planning Game, Short Releases**

Extreme Programming teams use a simple form of planning to decide what should be done next and to predict when the project will be done. Focused on business value, the team produces the software in a series of small fully-integrated releases.

- **Core Practices: Simple Design, Pair Programming, Test-Driven Development, Refactoring**

Team members work together in pairs, the whole team focuses on simple design and extensively tested code, improving the design continually through refactoring to keep it always just right for the current needs passing all Customer Tests too.

- **Core Practices: Continuous Integration, Collective Code Ownership, Coding Standard**

The project team keeps the system integrated and running all the time. All developers code in a consistent style so that everyone can understand, change and improve any part of the code as needed.

- **Core Practices: Metaphor, Sustainable Pace**

The Extreme Programming team shares a common and simple picture of what the system will look like. Everyone works at a pace that can be sustained indefinitely.

2.2 Overview of the Extreme Construction Game

The Extreme Construction Game is a simulation of Extreme Programming, which was developed by Josef Bergin and Fred Grossman at Pace University [Berg04a]. During the game a mini project is performed (e.g., the construction of a zoo, an airplane or a bridge) by applying many of the XP practices. In the following a brief description of the Extreme Construction Game is given.

After a quick introduction to the values and practices of Extreme Programming, the instructor creates the groups, assigns roles (i.e., customer, developers and monitor) and describes their responsibilities (cf. Fig. 2). Each group should have between 8 and 12 members.

Figure 2: Customers, Developers and Monitors: List of Responsibilities

After being established, each team works in short cycles to create the artifacts to the specification of its customers on the teams. In each cycle the individual customers write many features of the desired object on 3 by 5 cards, one feature per card (e.g. a lion or a restaurant for a zoo) (cf. Fig. 3a).

Figure 3: a) Writing Story Cards (Customer) and b) Estimating Stories (Developer) [Berg06]

These "story" cards are given to the developers on the team who estimate (in minutes) how long it might take them to build that element independently of other things. These estimates are written on the cards and the cards are given back to the customers (cf. Fig 3b). The customers then prioritize the cards in any order they wish, taking the cost (time needed) into account. The developers give the customers a Velocity[1] for the cycle (i.e., for the iteration). The customers then hands the team the selected cards which in total take less or equal time to implement than the estimated velocity (cf. Fig. 4a).

[1] In the first iteration the velocity is calculated as the iteration length * number of developers / 2. In subsequent iterations the velocity is the sum of estimates of the completed and customer accepted stories in the preceding iteration.

Figure 4: a) User Stories Are Selected by the Customers and b) Implemented by Developers [Berg06]

The team then starts to build those features, but no others (cf. Fig. 4b). Developers always work in pairs when constructing. At the end of the 15 minutes iteration the constructed objects are delivered to the customer for acceptance or rejection. As soon as a feature (e.g., a penguin) is accepted by the customer it is integrated into the overall project (e.g., zoo) (cf. Fig. 5).

Figure 5: Product is Delivered to the Customer [Berg06]

This continues for as many cycles as possible, with a 10 minute planning session and a 15 minute construction session in each cycle. After each construction session a short project retrospective is performed where the team discusses what worked in the last cycle and were improvements should be made. At the end of the Extreme Construction Game there is a detailed discussion about what has been learned during the Extreme Construction Game.

2.3 Coverage of XP Practices by Extreme Construction

Table 1 shows to what degree the different Extreme Programming practices are addressed by the Extreme Construction Game [Berg04b].

The higher the number of Xs in the table, the higher the coverage by the simulation.

Table 1: Coverage of XP practices [Berg04b]

1. The Planning Game	**XX**
2. Short Releases	**XX**
3. Metaphor	**-**
4. Simple Design	**X**
5. Testing	**X**
6. Refactoring	**X**
7. Pair Programming	**X**
8. Collective Ownership	**X**
9. Continuous Integration	**X**
10. Sustainable Pace	**-**
11. Whole Team	**X**
12. Coding Standards	**-**

3. Designing the Experiment

This section provides background information on the course in which the experiment was performed (cf. Section 3.1) and gives an overview of the structure of the experiment (cf. Section 3.2).

3.1 Overview of the Course

The experiment was performed during the course “Agile software development and project management” in the program of study “Management and IT” held at the Management Center Innsbruck. The course was hold together with Werner Wild. The course was attended by 45 students, which were divided in two classes (22 and 23 students). The goal of this course is to teach students the concepts of agile software development and project management. On the one hand the students are provided with theoretical input about agile methods, which should help them to get an understanding of the method, on the other hand the students should apply the thaught knowledge in a small software development project.

3.2 Overview of the Experiment

In order to answer the question whether the Extreme Construction Game improves the learning of XP and its practices, a controlled software

engineering experiment was conducted. As part of the experiment the Extreme Construction Game was performed with the students at the beginning of the semester (cf. Fig. 6).

Before the experiment the students were given a short introductory lecture on XP by Werner Wild. In addition, project groups for the semester's project were built and the students have chosen their project topic and gathered some initial requirements.

Figure 6: Course Structure

This was then followed by a brief lecture giving an introduction into agile software development methods in general and their background (3 hours for each class), which was taught by myself. After this half-day lecture the Extreme Construction Game was performed (3 hours for each class). Then another theoretical part followed. The rest of the semester was then devoted to the project. The students were coached during the course of the project and had meetings every two to three weeks with Werner Wild.

In order to assess the impact of the Extreme Construction Game on the learning of eXtreme Programming it would have been best to divide each class into two parts and to perform the Extreme Construction Game only with half of the students. This would then have allowed to compare the performance (i.e., grades for the project, exam) of both groups (with and without Extreme Construction Game).

Unfortunately, this was not feasible for this lecture and the Extreme Construction Game was performed with all students. Alternatively, the students were asked to do self-assessments regarding their current level of expertise in respect to the 12 XP practices before and after the Extreme Construction Game and at the project end (cf. asterisks in Fig. 6).

4. The Experiment

This section discusses the different phases of the experiment in detail. Section 4.1 covers the results of the self-assessment which was performed before the Extreme Construction Game. The Extreme Construction Game as it was run is described in Section 4.2. Section 4.3 and 4.4 describe the self-assessments after the Extreme Construction Game and at the project end.

4.1 First Self-Assessment

4.1.1 Results of the First Self-Assessment

At the end of the lecture "Introduction to Agile Methods" (cf. Figure 6) the students were asked to self-assess their level of expertise in respect to the 12 XP practices. For this, the students had to provide a rating for each of the 12 practices. The rating had to be between 1 and 5, where 1 reflects the lowest level of expertise and 5 the highest level of expertise.

Table 2: Results of First Self-Assessment (45 students)

Overall Level of Expertise	
Name of XP Practice	Average
Coding Standards	2,89
Planning Game	4,00
Short Releases	3,67
Metaphor	1,11
Simple Design	2,78
Test Driven Development	2,86
Refactoring	3,33
Pair Programming	4,22
Collective Code Ownership	2,56
Continuous Integration	2,50
Sustainable Pace	3,67
On Site Customer	3,17

The results of the self-assessment are shown in Table 2 and show that only two out of twelve practices, i.e., *Planning Game* and *Pair Programming*, have a rating higher than 4. This is not surprising as these are the only two practices which have already been applied in practice by the students. 4 practices got an average rating (i.e., Short Releases, Refactoring, Sustainable Pace and On-Site customer). The remaining 6 practices got ratings lower than 3.

4.1.2 Observations by the Instructor

Observations made during the aforementioned introductory lecture and during this first self-assessment, led to the impression that the ratings are very optimistic. The discussions with the students at this point in time showed that several of them mix up practices or have a wrong understanding of what a particular practice really means. This was especially true for the practices *Simple Design*, *Collective Ownership*, *Continuous Integration* and *On-Site Customer*. Interestingly, the ideas behind the technical practices like *Test-Driven Development* and *Refactoring* were very well known; however, practical experience was still missing. The discussion with the students further showed that all of the groups are motivated to apply XP, but they have the feeling that they need more information (than provided by the lectures so far) before they can get started with their software project.

4.2 The Extreme Construction Game

4.2.1 Introduction and Group Building

Before the experiment could be conducted the needed materials had to be prepared (see Appendix D) and project teams had to be created. In the first class 3 project teams with 8 to 10 members each were created, while in the second class only 2 project teams with 8 and 10 members participated.

After having set up the groups the different roles relevant to the Extreme Construction Game (i.e., Customer, Developer and Monitor) were described and their responsibilities were explained to the students. In addition, each student got two pages with the rules for the game (see Appendix A and B). Each of the groups had to assign two members to the role Customer, two members to the role Monitor, the rest of the team members were assigned to the role Developer. Each of the groups got the task to build a zoo using XP, following the procedure described in Section 3.2. The detailed time schedule for the Extreme Construction Game can be found in Appendix C. Figure 7 and 8 show some of the results of the Extreme Construction Game.

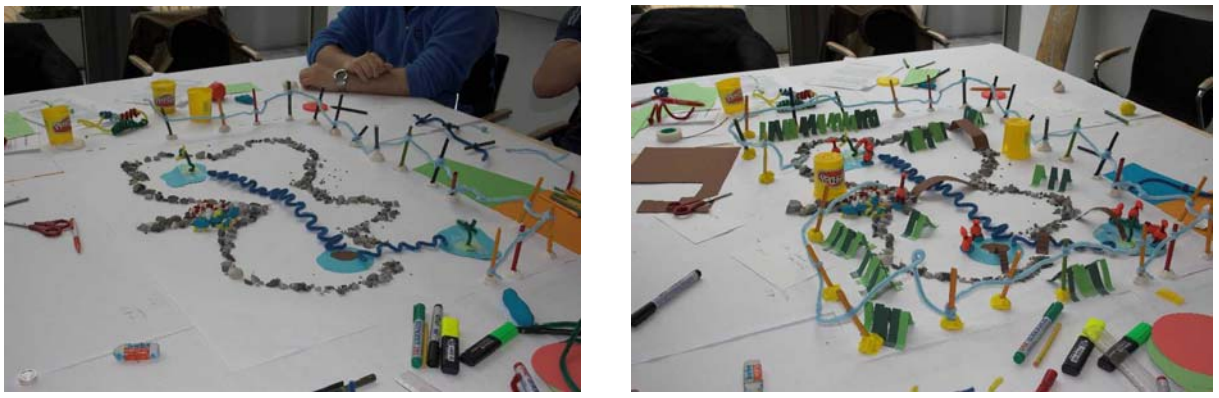

Figure 7: a) Results of the First and b) Second Iteration

Figure 8: a) Team Working on Third Iteration and b) Final Presentation of the Project

4.2.2 Observations of the Instructor

During the whole experiment the students were very motivated and worked together as a team. All zoo projects showed that productivity increased significantly with each iteration. Especially in the third iteration the productivity was very high. The communication among the team members worked very well (between Developers and between Developers and Customers), which is one of the crucial success factors for any software development project. Also the On-Site Customer practice worked very well for all teams. The responsibilities of the different roles were clear to the students. Collective ownership and Continuous Integration was done by all teams. Pair programming was not done by all developers, for those who did pair programming it worked very well. Most of the teams had to do at least one refactoring during the project. Testing was done by all of the teams, however, towards the end of the project some of them stopped to draw a sketch before building the features.

4.3 Second Self-Assessment

4.3.1 Results of the Second Self-Assessment

After the Extreme Construction Game the students were asked again to perform the self-assessment. Table 7 shows that the ratings for all 12 practices are higher than in the first self-assessment. 9 of 12 practices got ratings higher than 4. Three practices got ratings under 4. In the case of *"Coding Standards*" and *"Metaphor"* this is not surprising as the game does not cover these practices explicitly. However, what does surprise is the fact that although these practices were not covered by the game explicitly they got better ratings than they did before the game. The ratings for *Test Driven Development* increased, but not as significantly as the other practices covered by the exercise. The reason for this might be that the simulation of this practice is not as adequate as it is for the other practices. Discussions with students confirmed that the game helped them to understand the purpose of the practices and clarified many things. However, it did not provide them with information on how to apply them directly to software development.

Table 3: Results of Second Self-Assessment (45 students)

Overall Level of Expertise		
Name of XP Practice	**Average 1st assessment**	**Average 2nd assessment**
Coding Standards	2,89	3,50
Planning Game	4,00	4,80
Short Releases	3,67	4,80
Metaphor	1,11	3,53
Simple Design	2,78	4,10
Test Driven Development	2,86	3,30
Refactoring	3,33	4,20
Pair Programming	4,22	5,00
Collective Code Ownership	2,56	4,00
Continuous Integration	2,50	4,50
Sustainable Pace	3,67	4,60
On Site Customer	3,17	4,90

4.4 Third Self-Assessment

4.4.1 Results of the Third Self-Assessment

At the end of the semester, and after completing their software project, the students were interviewed again to assess the potential benefits of the Extreme Construction Game. This time most of the ratings were lower than in the second self-assessment (9 out of 12). Only 3 practices got

better ratings. 1 practice got a rating lower than 3, 6 practices got ratings between 3 and 4 and 5 practices got ratings higher than 4.

Table 4: Results of Third Self-Assessment (45 students)

Overall Level of Expertise			
Name of XP Practice	**Average 1st assessment**	**Average 2nd assessment**	**Average 3rd assessment**
Coding Standards	2,89	3,50	3,22
Planning Game	4,00	4,80	3,44
Short Releases	3,67	4,80	4,67
Metaphor	1,11	3,53	2,89
Simple Design	2,78	4,10	4,67
Test Driven Development	2,86	3,30	3,44
Refactoring	3,33	4,20	3,56
Pair Programming	4,22	5,00	4,56
Collective Code Ownership	2,56	4,00	4,11
Continuous Integration	2,50	4,50	4,22
Sustainable Pace	3,67	4,60	3,00
On Site Customer	3,17	4,90	3,89

After having applied XP in practice the students were able to assess the usefulness of the Extreme Construction Game in a more realistic way than before. Right after the Construction Game the students had the feeling that they were already equipped with the necessary information and were fully motivated for the project. During the semester the students were confronted with technical difficulties and the project turned out to be more complicated than expected (not uncommon in software development!).

4.4.2 Observations of the Instructor

During the project the students were very motivated. However, the learning of XP was hindered by lacking technical programming skills of the students, which turned the projects into real challenges for them. Consequently, the students were not able to draw their full attention to the software development process and only employed parts of the XP practices. As the results from the self-assessments show, the Extreme Construction Game helped the students to understand XP, its real life application, however, was only partially successful.

4. Summary and Conclusion

As part of this paper a controlled software engineering experiment was performed evaluating whether the Extreme Construction Games improves the learning of XP practices. The experiment was conducted in the course “Agile Software Development and Project Management” in the winter term

2006/07 at the Management Center Innsbruck. The results of the experiment provide evidence that the Extreme Construction Game helps the student to better understand XP and its practices. Observations during the project however show that the students did not fully employ XP during the project. Especially missing technical skills made the projects really challenging for the students, drawing their focus from the software development process to more technical issues.

Motivated through the very positive feedback regarding the Extreme Construction Game, the game will be performed in the next semesters again. In order to be able to improve the adoption of XP practices as well, the students need more technical education in the previous semesters. This problem was discussed with the head of the program of study “Management and IT”, who promised to consider this in the planning for the next semesters. Next semester I will hold a similar course at the Univ. of Innsbruck, were students usually have a much higher technical programming knowledge. This will allow me to further evaluate the relationship of technical knowledge and adoption of XP practices

References

[Berg04a] Joseph Bergin: Extreme Programming - An XP Game. http://csis.pace.edu/~bergin/extremeconstruction/index.html, 2004, Accessed 2006/10/10.

[Berg04b] Joseph Bergin: Coverage of XP Practices by Extreme Construction Game. http://csis.pace.edu/~bergin/extremeconstruction/XPValuesPractices.html, 2004, Accessed 2006/10/10.

[Berg04c] Joseph Bergin: Rules for the Planning Phase. http://csis.pace.edu/~bergin/extremeconstruction/planningrules.html, 2004, Accessed 2006/10/10.

[Berg04d] Joseph Bergin: Rules for the Construction Phase. http://csis.pace.edu/~bergin/extremeconstruction/constructionrules.html, 2004, Accessed 2006/10/10.

[Berg04e] Joseph Bergin: Time Schedule for the Extreme Construction Game. http://csis.pace.edu/~bergin/extremeconstruction/constructionrules.html, 2004, Accessed 2006/10/10.

[Berg06] Joseph Bergin: Extreme Construction – The Movie, http://csis.pace.edu/~bergin/extremeconstructionMovie.htm, 2006, Accessed 2007/02/10.

[Jeff01] Ron Jeffries: What is Extreme Programming? http://www.xprogramming.com/xpmag/whatisxp.html, 2004, Accessed 2006/10/10.

Appendix A – Rules for the Planning Phase

A page with the following rules for the planning phase was given to the students before starting with the Extreme Construction game, according to their assigned role [Berg04c].

Customers:

- Prioritize your estimated stories as you wish. Take time estimates into account, but choose the things you value highest.
- When given a velocity by the constructors, choose stories with time estimates up to this velocity for building in the current cycle. Give only these story cards to the constructors to build in the next construction phase.
- Write additional stories for future cycles as you desire. Give them to the constructors for estimation.
- Don't build anything yourselves.
- Answer all questions about the stories and make sure you have a common understanding about the overall nature of a story with the Constructors to aid estimation.

Constructors:

- Give the customers a "velocity" figure that is no greater than the length of the next cycle. This is a promise to do a certain amount of work in the next cycle as determined by the estimates you gave. For example, if the construction phase is to last 15 minutes, the velocity might be 10 or 12 minutes. As you get experience, both your estimates and the velocity will get more accurate.
- Estimate additional stories as necessary. For each story give a time estimate (ideal constructor minutes) that represent ideal time assuming no delays and assuming the story will be built independently of any others. If something seems impossible give it a very large or infinite estimate. The estimate is how long it will take the team to build the story if it has only that task.
- Consult with the customers as needed on the meaning of the stories. Don't assume you know what is wanted. Don't make decisions on behalf of the customers. Make sure the customers understand what you can do and NOT do. Make suggestions about possibilities. Explore alternatives as appropriate.

Monitors:

- Keep everyone communicating.
- Don't let the customers estimate.
- Don't let the constructors specify, prioritize, or assume what the customers might want.

Note: The first planning phase is longer than the others to allow for some initial stories to get written and estimated.

Appendix B – Rules for the Construction Phase

A page with the following rules for the construction phase was given to the students before starting with the Extreme Construction Game, again according to their designated role [Berg04d].

Customers:

- Be available for consultations with the constructors. They may need clarification on the meaning of the current stories.
- Write additional stories as you desire for future cycles. Give them to the constructors for estimation. If you don't like the estimates you may split your story into smaller units or otherwise modify them. Estimates are NOT negotiable.
- Do not participate in construction, though you may help in sketching.
- Do not try to change the stories in the current construction cycle. You may drop a story at any time, though.
- If asked by the constructors for additional work, you will get a maximum time. Choose high value stories with estimates up to the given time to return to the constructors.
- At the end of the phase you will have to accept or reject what is done. If you reject it, write additional stories to make it right.

Constructors:

- Construct the stories in the current cycle, with a pair of constructors choosing a story and implementing that story only. Nothing is built unless it is done in a pair. Other pairs may be doing the same with other stories. Make sure you can integrate your work.
- Before beginning the construction, make a *sketch* of the thing to be done. Use this later to verify that you have built what is wanted. The sketches should be shared with the rest of the team including the customers. You should also write down implications of the stories that need to be verified after the build. For example, if you are building an airplane model, does it have to fly? How far? Use these also to verify you have built what is needed.
- Don't assume you will **ever** get a card not in your current set of jobs. Just because you have estimated a story does not mean it will ever get built. Don't anticipate the future.

- If you finish the stories in the time allotted you may ask for additional work. Give the customers a new "velocity" up to the time remaining in the cycle.
- Ask for clarification from the customers whenever you like.
- Make sure the customer understands what you are able to do and NOT do. Suggest possibilities.
- If the customers give you additional new cards for estimation, have someone on the team provide an estimate and return the card. Write the estimate on the card itself.
- Deliver what you have done to the customers at the end of the phase. They will either accept it or reject it.

Monitors:

- Continually check with each Constructor to see if they can complete their tasks in the time remaining. If not, call a 1 minute STAND UP meeting. Inform the Customers of the problem. See if others can help out on this task without affecting their own. If not, let the Customers decide which tasks to drop and which are most important to complete.
- Keep everyone communicating. Don't let intimidation occur between customers and constructors.
- Don't let the customers build or estimate.
- Don't let the constructors specify, prioritize, or assume what the customers might want.

Note: Only the customers can decide if they are satisfied with what was built. However, the monitors will judge whether the constructors built what the customers *said* they wanted.

Appendix C: Time Schedule for the Extreme Construction Game

The time schedule proposed by [Berg04e] was adapted to meet the time constraints of this course. The following agenda with an adapted time schedule was used for the experiment.

Initial Story Writing (20 minutes)

Customers decide what they want and begin to develop feature cards--one feature per card. Keep the constructors informed. Give them the cards as they are written. Keep your features simple and independent.

Constructors self organize and discuss things with the Customers. When you get a card, estimate its time in ideal minutes/seconds. Write the estimate on the card in "ideal pair minutes". Give it back to customers. If something seems impossible, give it a large (infinite) estimate. The estimate is the ideal time it would take two people working together to build the feature described, assuming no interruptions, meetings, etc.

Planning (10 minutes)

Constructors announce your *velocity* for the next period. This is a number up to 15 minutes times the number of pairs you have in your team. You will get cards with total estimated times up to your velocity. Your velocity is an estimate of how much you can do within a time box. Be conservative with this the first time.

Customers choose the most desirable cards/features up to the time limit (velocity) announced by constructors.

Developing (15 minutes)

Constructors build the features on the cards and consult with Customers as needed. Before building anything, draw a sketch of it. These sketches can be (should be) developed with the Customers. When you build, always do so in pairs. Constructors must always work in pairs to accomplish their tasks. In a pair, continuously discuss what to do and how to do it. Try to have several pairs working independently, but don't forget to integrate what you do into the overall project. When done with a feature, verify with Customer that it is as described on the card and looks like the sketch you made. Try to switch partners for different tasks. Monitors will ask you if

you are on schedule to complete your tasks. If not s/he will hold a one minute meeting.

Customers develop new cards as desired. Consult with the constructors when they have questions. Don't change cards in the current cycle, but if you see you asked for the wrong thing based on what is being built, write more cards.

Retrospective (10 minutes)

Instructor and students discuss about what has worked and what did not work in the first iteration.

Planning (10 minutes)

Constructors estimate new cards if any and decide on a new effective time (velocity). Adjust your velocity based on the previous cycle. Use "yesterday's weather".

Customers choose new features as before.

Developing (15 minutes)

As above. Constructors may re-build any parts that they think they can do better. They can modify (refactor) things built earlier, but no additional time is allocated for this.

Retrospective (10 minutes)

Instructor and students discuss about what has worked and what did not work in the second iteration.

Planning (10 minutes)

As above.

Developing (15 minutes)

As above.

Discussion / Debrief by instructor and students (30 minutes at least)

Appendix D– Material Needed for the Construction Game

The following material is needed for conducting the Extreme Construction Game [Berg04a]:

- About 50 3 by 5 cards for each team along with pencils/pens for everyone

And a variety of children's art supplies such as

- Pipe Cleaners (important)
- Construction paper, both white and colors
- Glue suitable for paper, adhesive tape
- Scissors/ hole punch/rulers/french curve, compass, protractor
- Paint, colored markers, pastels
- Colored stickers/ ink stamps
- String/yarn/ribbon
- Packing material (peanuts, shredded paper…)
- Modeling clay/Play Doh
- Popsicle sticks, toothpicks, dowels