

# Introducing User-Centered Design to eXtreme Programming

Anders Toxboe

Copenhagen Business School  
Department of informatics  
May, 2005

[anders@toxboe.net](mailto:anders@toxboe.net)

## Index

1	Abstract.....	2
2	Problem area .....	3
2.1	Problem formulation.....	3
2.2	Boundaries of the project.....	4
3	Method.....	4
3.1	Theoretic field.....	4
3.2	Operationalization of theoretic field.....	4
3.3	Extreme Programming.....	5
3.4	User centered design.....	6
4	Critique of methods.....	7
5	Harmonies and disharmonies between XP and UCD.....	8
5.1	Area of use .....	8
5.2	Documentation .....	9
5.3	Prototyping.....	10
5.4	Personas .....	10
5.5	Scenarios.....	11
5.6	Cultural probes.....	11
6	New Method.....	12
6.1	Adding an initial phase .....	12
6.2	The project inception phase .....	13
6.3	Implementing user-centered design with XP iterations.....	13
7	Method in use.....	14
7.1	Workshops .....	14
7.1.1	Scope of workshops.....	14
7.1.2	Testing for proof.....	15
7.1.3	Procedure and results of workshops .....	15
7.2	Future workshops beyond this project.....	19
7.3	Changes to the principles of XP after implementing UCD .....	19
7.4	Advantages and disadvantages of the new method.....	20
8	Conclusion .....	21
9	References.....	21
9.1	Websites.....	22

## 1 Abstract

In this paper, I introduce a combination of User Centered Design (UCD) and eXtreme Programming (XP) that seeks to bring the users closer to the center of attention than in XP. The new methodology is used to guide decision making in XP with use of personas, scenarios and cultural probes.

## 2 Problem area

When developing an entirely new software product for a target group of users, a series of User Centered Design (UCD) tools can assist in incepting ideas fit for the target group of users. In usual utilization, many of these UCD tools have, until now, been a part of a requirements gathering, when combined with software development methodologies. Most UCD tools have been separated in a process for themselves, which ends with a last step called “implementation”. The responsibility of turning the requirements into something useful is left for the programmers to do.

Especially in explorative development projects, there is an uncertainty about central aspects of the task. As the programming starts and the first working software appears in these types of projects, aspects that were not included in the requirements seem to scratch the surface during development.<sup>1</sup> To implement these new and changed requirements with traditional development methodologies, it is needed to go back one step and start over with the new requirements gathering. eXtreme Programming (XP) suggests among other things to discard phases and requirements gathering and instead do all the phases at the same time in iterations.

The two methods (XP and UCD) have great things, which could complement each other. XP is in lack of knowing their true users and UCD is in lack of a flexible and explorative development methodology that lasts throughout the entire project.

In this paper, I introduce eXtreme Programming and User-Centered Design and how they can help being fertile for new ideas. After the introduction of the two, I review the relevant critique from the literature, which leads on to how the two can complement each other. The two are put up against each other by showing harmonies and disharmonies between them. After this I suggest a new methodology that combines the two. The new methodology is put to test by a series of workshops that will test whether the two are compatible throughout the first few iterations.

### 2.1 Problem formulation

Extreme Programming (XP) has its focus on a company’s contract-development with a customer. What if we have no customer, and we don’t have a concrete idea of what we want to create, but still want the benefits of what the explorative method of XP has to offer? Is it possible to make the agile method of XP even more agile, even more explorative, and even more fertile for ideas by cutting the contract part of XP away, adding UCD tools, and redefining the role of the customer?

This leads to the problem formulation of this paper:

To combine the systems development method of XP with UCD in order to make a new method that focuses more on developing for users than on developing for a contract, by starting from scratch by incepting the idea about the product with the users in the center of attention and keeping them there throughout all phases of development.

---

<sup>1</sup> Christensen, 1991

The formulation of the problem can be divided into two parts: (1) incepting the idea and (2) developing the product. I will examine what UCD tools work best for each part.

## **2.2 Boundaries of the project**

Even though I try to combine XP and UCD, I will not go into detail about how the specific tools of each area are used for gaining the best output by themselves. The primary task for this paper is to connect the two areas by illustrating how they can complement each other. I will propose a transition between the inception of the idea and the development of the product as well as how UCD tools can help XP bring the user into design decisions.

For this reason, I will for instance not describe what personas<sup>2</sup> are and how they should and should not be used. I will merely show how they can be connected to XP. It is a premise for the user, who wishes to implement the combination of the techniques mentioned, that he either knows how to use these on beforehand or is able to read up on it by following the references of the paper.

## **3 Method**

### **3.1 Theoretic field**

The theoretic field for this project is UCD and XP. The purpose of this paper is to make XP more user-centered, why user-centered design is the obvious choice for XP to mate with.

There are many different definitions of exactly what UCD is. To avoid confusion, I will use the following definition:

*“To develop a system with a focus on the user requires that the designers are able to engage in the user and understand the users motivation for using the system” Nielsen 2004 (2)*

In the quote, I find that the word *designers* can be substituted with the word *programmers* as well as *decision-makers*. Design is closely related to decision-making and programming, which is why this paper seeks to keep the user in the center of two groups as well as with designing. All groups should be able to engage in- and understand the users motivation for using the system.

### **3.2 Operationalization of theoretic field**

As the purpose of this paper is to create a method of systems development, which combines XP and UCD, it is necessary to test the findings of the project. Doing workshops that simulate XP planning meetings with UCD tools mixed into them has carried out this test. As XP is an iterative development method, more than one workshop

---

<sup>2</sup> Nielsen, 2004 (1)

has been carried out. This has left room for reflection between each workshop. Each new workshop was a new opportunity to perfect the new method, why fine-tuning of the procedure of each workshop were made.

Next will follow a brief description of both fields to clear up definitions of concepts for the reader

### **3.3 Extreme Programming**

Extreme Programming (XP) is an explorative and agile development method<sup>3</sup> that seeks to satisfy the customer through early and continuous delivery of valuable software. XP welcome changing requirements through the whole development process by being flexible with its short iterations (2-4 weeks) that allow rapid feedback and tools like refactoring<sup>4</sup> and unit testing that allows simplistic coding<sup>5</sup>. Together, this helps steer the project in the right direction, which in turn decreases the risk of failure<sup>6</sup>. Collective code ownership is an important part of XP and is enforced by continuous testing and pair programming.

XP swears to the philosophy of the *Manifesto for Agile Software Development*<sup>7</sup>, which represent the thoughts behind the method. The Agile Manifesto values:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

There is a focus on quality rather than scope as in traditional methods. This is exemplified in various ways, for instance by pair programming and by setting a 40-hour workweek to relieve the programmers from stress.

Roles in XP are divided into two: The customer and the developers. The term “customer” is broad in XP. It covers both the person who makes the business decisions, users, business management, operations, etc. It is both the person who will really use the system, when it is in production as well as the one, who makes the decisions and more.<sup>8</sup>

The complex problem that the customer presents the development team is divided into smaller bits called user stories. Each user story contains functionality that can be implemented in the time of one or two iterations, and is only a one to two sentences long description of the functionality. At the beginning of each iteration, a number of user stories are selected by the customer to be implemented in the coming iteration. The

---

<sup>3</sup> <http://agilemanifesto.org/>

<sup>4</sup> Fowler, 1999

<sup>5</sup> Beck, 2000 (1)

<sup>6</sup> Beck, 2000 (2)

<sup>7</sup> <http://agilemanifesto.org/>

<sup>8</sup> Beck, 2000, p. 18 & Beck and Fowler, 2000, p. 6

process of dividing the problem into user stories is called storytelling. An example of a user story is:

*“The system should check the spelling of all words entered in the comments field.” Beck, 2000 (2), p. 14*

In the beginning of each iteration, a planning meeting is held. Here the user stories, which are to be implemented in the oncoming iteration, is chosen. The prioritization of user stories is based on what user stories bring most value to the project.

User stories are only used in the planning meetings to prioritize and estimate the feature in the user story. When development starts, the programmers go to the customer to get a more detailed view.

Functional acceptance tests with scenarios for each user story, is used to make sure that the stories has been correctly implemented. XP scenarios are mechanistic in nature, and can be tested without human involvement if clever enough scripts are put to work.

XP is based on old and proven practices combined in a new way to make up for weaknesses in the more traditional phase-divided methods<sup>9</sup>. XP techniques like pair programming have been suspected to be cost-inefficient, but research results<sup>10</sup> show that efficiency increases by 15%. So does the quality of work, collective ownership of code, and stress. These results reject the majority of the critique of XP.

### **3.4 User centered design**

User-centered design exists in many shapes and forms. From being an integrated part of system development methodologies like OOA&D<sup>11</sup> to being stand-alone tools with no plan for implementation.

UCD places the person, and not the system at the center, by trying to answer questions about users and their goals and tasks, in order to direct the development and design of a product. The product can be anything from physical products<sup>12</sup> to software products – or even a process: anything in which the users interact. One UCD technique is a plain evaluation of an existing system – another, writing personas, is a detailed way of describing the product’s users.

A number of criteria can be established for whether a UCD tool is appropriate to be combined with XP. In establishing these criteria, the principles behind the Manifesto for Agile Software Development should be respected as well as the short development cycles of XP. The criteria are as follows:

- The tool should produce a minimum of documentation thus making it possible to focus more on quality.

---

<sup>9</sup> By traditional methods are meant in particular the waterfall-model

<sup>10</sup> Cockburn, 2002

<sup>11</sup> Matthiassen, 2000

<sup>12</sup> Norman, 2004

- If the UCD tool is to be set up inside the development loops, it should have a time-span of the same or less than an XP iteration (1-2 weeks).
- The tool should embrace change rather than make a set recipe for the development.
  - By taking short time, the tool can be used as a means of feedback
  - By supporting understanding of the users, the tool can give a more detailed feedback than without the tool, thus helping decision-making.

The scope of this paper is limited, why I will only list a few UCD techniques that I see as appropriate for matching with XP with the above-mentioned criteria in mind. The list is not a prioritization of which techniques are better than other, but merely mentioned to give the reader an idea about what kind of tools I am thinking of. Harmonies and disharmonies between the mentioned techniques and XP is to come later in this paper.

UCD techniques that can be combined with XP include personas<sup>13</sup>, scenarios<sup>14</sup>, prototyping<sup>15</sup>, and cultural probes<sup>16</sup>.

Other findings of the UCD toolbox include constructing conceptual models, rich pictures, Hierarchical Task Analysis (HTA)<sup>17</sup>, and use cases as well as continuously evaluating the product through for instance heuristic and thinking aloud tests.<sup>18</sup>

## 4 Critique of methods

Even though a large part of the critique directed towards XP has been rejected by prior mentioned research, one thing still remains un-defended: too little user involvement in the design process. XP discards the phase-structure in the traditional waterfall-model<sup>19</sup> by letting the programming, requirements gathering, and planning iterate. Big initial requirements gathering and documentation are dropped in order to focus more on quality of the product itself inside the development loops.

When Kent Beck originally introduced XP's customer, a real end-users was in mind, as he believes in the teams's direct accountability to the end-users. Unfortunately, not all XP projects have followed this model. The broader picture with multiple end-users was cut away with the big initial requirements analysis and degraded to fit under the hat of the "customer", who makes business decisions as well as represent the end-users. Even though many advantages have been achieved by the new structure, a new problem has been created: end-users are not represented in their *diversity* and detail in XP.

---

<sup>13</sup> Nielsen, 2004 (1)

<sup>14</sup> Definition: Preece 2002, Nielsen 2004, Grudin 2002, Beck 2000

<sup>15</sup> Floyd, 1984

<sup>16</sup> Gaver, 2004

<sup>17</sup> Preece, 2002

<sup>18</sup> Preece, 2002

<sup>19</sup> Hughes, 2000

When integrating UCD and XP, the big question is where decisions are made. Many UCD tools suggest that decisions about the final product are made in the requirements document, which is created before any actual coding starts and is created on a basis of thorough analysis of the end-users. XP argues for the opposite. Considerable decision-making is happening in every iteration in XP instead of only once before implementation starts. The decision-making in XP is spread out over the life of the project as requirements gathering is.

In many cases, interaction design and UCD can be seen as the same thing as interaction design uses UCD tools in its processes. In a debate with Alan Cooper (Interaction Design), Kent Beck (XP) argues that the interaction designer becomes a bottleneck, as all decision-making are at one point. Instead of adding early requirement gathering and design phases to XP, Kent Beck argues that interaction design would be more powerful if integrated inside the loop of development according to the extreme programming practices.<sup>20</sup>

Lene Nielsen explains<sup>21</sup>, that a main obstacle with using personas is poor description of the step from personas to code. This paper seeks to do give a description.

Antti J. Häätinen has suggested mapping user goals directly to XPs user stories with his GO-XP<sup>22</sup> method, which tries to combine GUIDe<sup>23</sup> and XP. He later found that this solution did not work. The problem was, that by presenting the user goals directly to the programmers as a design issue, a conflict between user interface and software architecture rose. The programmers could not tell the interface and architecture problems apart, as they each represented separate problems. This can also be a reason to reject combining XP with the HTA tasks, that I earlier suggested. Häätinen suggests that the user interface design and sketches have to be completed well before the XP's planning game.

## **5 Harmonies and disharmonies between XP and UCD**

Even though you would not suspect the two to have many similarities, a few do actually exist. It is those similarities and harmonies that can tie XP and UCD together. In order to find out which UCD tools that fit best for use with XP, I'll examine harmonies and disharmonies between selected UCD tools and XP as well as other key areas of difference.

### **5.1 Area of use**

While UCD focuses on creating designs for the organizational informational system<sup>24</sup>, XP has its focus more on the developed program itself. A combination of the two methods can hopefully help XP in broadening its vision into considering other things than just what the customer think is right.

---

<sup>20</sup> Nelson, 2002

<sup>21</sup> Nielsen, 2004 (1)

<sup>22</sup> Häätinen, 2002

<sup>23</sup> GUIDe: Goal-Oriented User Interface Design. See Häätinen, 2002

<sup>24</sup> Nelson, 2002

Both XP and UCD are toolboxes for executing specific design tasks. Both have collections of tools that have set recipes for carrying out certain tasks. XP has more than that though, as it preaches a set of principles to follow<sup>25</sup>. UCD has, hence the name, its focus only on design, whereas XP concentrates both on design as well as planning tasks. UCD tools are therefore more appropriate for being implemented in XP than XP is in being implemented in UCD.

## **5.2 Documentation**

XP praises working software over comprehensive documentation. By that XP despises keeping a big pile of documentation updated when requirements change. UCD tools that work as part of the requirements gathering will produce documentation as an end result. The implementation of these tools as a preliminary phase to XP might be one of the biggest drawbacks for XP, as XP was created as a reaction to development in phases<sup>26</sup> and big requirements gathering.

In order to keep XP in the spirit of agile development, it is important that the preliminary tasks, which are carried out before the actual programming begins, take as short time as possible<sup>27</sup>. It would be mutiny to implement the old phase structure that XP once boldly escaped from. For this reason, it is important that the role of the UCD tools inside the XP loops is kept to guiding decision-making and not keeping a pile of documentation updated when requirements change. UCD tools to be combined with XP should be chosen on a basis of their ability to gather information about the user as well as how much documentation they produce and require to be frequently updated.

The fact that XP tries to keep the documentation to a minimum does not mean that documentation is abandoned or that the documentation does not exist in some other form. The documents are predominantly on the client side. Before the client gets in contact with the XP development team, they've made up their own mind about what the system is going to do and whether to spend for instance one or two million dollars on development. XP starts on the phase called implementation (see figure 1): some kind of preliminary investigation must have been made on the client side before the development team is contacted and the implementation starts.

On the contrary to other systems development methodologies, XP refuses to write the ideas of the client down on something other than note cards. Instead of creating big collections of paper documentation, the client is brought into the game and works as living documentation for the initial and later thoughts. In this case, it is the knowledge of the client that is the documentation, which is the privileged kind of documentation that can make decisions on its own.

---

<sup>25</sup> Agile manifesto

<sup>26</sup> Nelson, 2002

<sup>27</sup> Nelson, 2002

### 5.3 Prototyping

The backbone of XP is incremental delivery of useful software in short iterations. One of the most important strengths of this approach is rapid feedback that can effect future development. This is also the objective of prototyping: to encourage reflection in design<sup>28</sup>.

Floyd (1984) distinguishes between three broad classes of prototyping: prototyping for exploration, prototyping for experimentation, and prototyping for evolution. Characteristics of the latter is a breakdown of the linear ordering of development steps mapped onto successive development cycles in order to prioritize feedback, learning, and cope with changing requirements. In evolutionary prototyping, working software is produced, which is why some authors would rather call it developing in versions. Floyd argues that any type of prototyping should “be considered as one procedure within system development that should be combined with others”. (Floyd 1984, p. 14).

XP is a methodology for systems development, which uses evolutionary prototyping, but combines it with other procedures. Inside the development loop XP uses exploratory prototypes called *spikes*, which are used to explore solutions for tough technical- or design problems<sup>29</sup>.

The basic problem that XP tries to cope with is risk<sup>30</sup>. Using prototypes is a great way to manage risk as solutions can be explored before implemented or only part of a solution needs to be implemented before found possibly unfit for the system. As prototyping is in the spirit of XP development, I encourage using it where it is found appropriate and possible – whether it being inside or outside the development loops.

### 5.4 Personas

Personas can serve as an expansion of the user representation within the XP methodology, support arguments about features of the system<sup>31</sup> as well as helping prioritize between them<sup>32</sup>. As each XP user story describes a feature, the mating between personas and XP seems obvious.

Programmers and decision-makers can engage in the worlds of the persona by understanding the motivation for actions done by the persona. This is why personas can make up for the lack of focus on the user in XP and make XP more user-centered.

Personas require field study of the target user group that the personas should represent. If done thoroughly, this field study takes longer than the length of an XP iteration, and should therefore not happen inside an XP iteration loop, but before the loops begins.

---

<sup>28</sup> Preece, 2002, p. 241

<sup>29</sup> [extremeprogramming.org](http://extremeprogramming.org)

<sup>30</sup> Beck, 2000 (1)

<sup>31</sup> Nielsen, 2004 (1)

<sup>32</sup> Grudin, 2002

Personas are documentation of the results from initial field study of the users. Agile development despises documentation in large forms (documentation is not bad, but working software is better) as it forces the developers to use more time on updating and keeping updated on documentation than on delivering working software. Because of this, methods of letting the programmers engage in the personas, which do not require spending too much time might be needed. One way to do this could be to randomly send out fictitious mails from the personas to the programmers<sup>33</sup>.

There is a risk that using personas to represent the user will not make up for real-life feedback from an actual user. Using personas should, if used, be seen as a complimentary to- instead of a substitution of users.

## **5.5 Scenarios**

Scenarios, that are built on- and use the same narrative fundamentals as personas<sup>34</sup>, can help explore features that the personas feel are important. Features deducted by scenarios give a hint of whether the specific feature is important for the persona or not, thus helping the prioritization of features.

Scenarios can vary in type and be used for completely different things. One type have roots in already created personas and tell a personal story about how the persona interacts with the system, based on his or her motivation and goals for doing so.<sup>35</sup> This type is used to engage in the personas mind to design a better system or gather requirements for the system. Another type of scenarios, that is a part of XP, are used to test if already working software works as it was intended. The two types of scenarios are built up in entirely different ways. When combining the UCD and XP use of scenarios, this disharmony in concept definitions should be addressed.

## **5.6 Cultural probes**

When not having incepted the entire idea of the project from start, cultural probes can help search for inspiration. This method can act as part of a preliminary investigation to inspire for new ideas.

Collecting and analyzing cultural probes takes time, and is therefore separately not suited for working inside the XP iteration loops. Cultural probes are used to probe for inspiration in the target user group and can therefore effectively work as a part of the field study needed to construct the personas.

Alternatively, the cultural probes could also be used to gather and inspire new information about the users when development on a new release has started. As new information about the problem area of the system has been gathered through developing part of the system, it is possible to create questions in the cultural probes, that ask more directly to the problem than the initial probes did before development started. When used

---

<sup>33</sup> Grudin, 2002

<sup>34</sup> Nielsen, 2004 (1)

<sup>35</sup> Nielsen, 2004 (1)

in this way, it is important that the cultural probes do not become a phase, but a tool that can be used if found necessary.

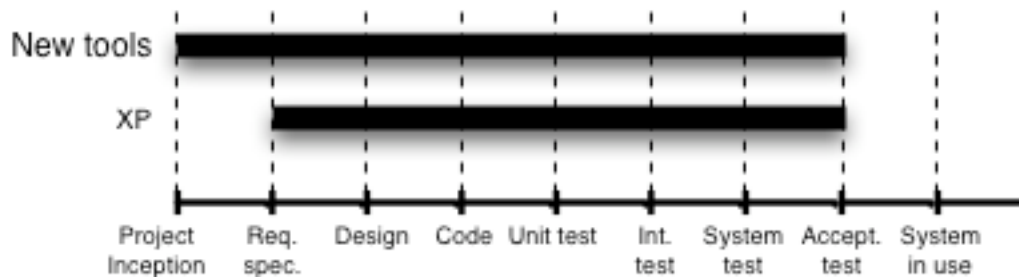
A third way of using the cultural probes is to send them out just before development begins. As the cultural probes do not all return at the same time, the results can be used as small bits of new inspiration at the time they come in.

## 6 New Method

To fill the hole that XP left blank after getting rid of the requirements analysis and phases in general, I propose a method that melts user-centered design and XP into one. As Kent Beck argues in debate with Allan Cooper, the best outcome will be to combine the tools of interaction design with the iterative processes of XP<sup>36</sup>.

In order for this to work, an initial phase is needed. Even though Kent Beck would argue that adding phases and hierarchical structures to XP is shooting yourself in the foot, I believe that combined in the right way, it is possible to get the benefits of them both.

### 6.1 Adding an initial phase



**Figure 1. Phases from Abrahamsson (2003). The new tools will create a project inception phase as well as assisting the planning tasks of XP.**

In traditional XP development, the customer has a basic idea about what the end-product is going to be like as the customer has a goal with the system. The developers themselves do not have any say in this: the project inception is not in the scope of XP, as the customer has already laid out the basic idea of from the start in XP. As we don't have a customer to lay out the basic idea, we turn our point of interest to the users.

To incept the ideas we need to take a look at the user group for whom we want to design the product for. It is in them that we can find inspiration to what the product specifically is going to do. The project inception phase is therefore used to examine the target user group in order to incept ideas.

---

<sup>36</sup> Nelson, 2002

## **6.2 The project inception phase**

The project inception phase is a stand-alone phase outside the scope of the XP iterations. Here any of the UCD tools can be used to incept the basic idea of the product. The freedom of choice between tools is rather large, as long as the end result is personas based on field study of the users.

I have chosen to focus on only 3 tools, which I believe make a good transition both to XP, but also between each other. Another combination of tools could have been selected, but as I am testing how they work as a transition, I have decided to only focus on a small range of tools rather than going all out. The selected tools are:

1. Cultural probes (Gaver 1999, Hemmings 2002, Mattelmäki 2002)
2. Personas (Nielsen 2004 (1))
3. Scenarios (Nielsen 2004 (1))

At first, the basic idea about what the system should do should be formulated in a few sentences. This will constitute the basic idea about the product, which the customer initially has in traditional XP development. The cultural probes use the basic idea to design its questions, and will result in a pile of ideas based on the user's needs. The cultural probes will inspire for new ideas more than give correct answers. These ideas will further be summed up and incorporated in the initial formulated idea. Personas are then created in order to figure out the difference in the ways different groups of people interact with the idea of the product and finally scenarios are created to support- and bring the personas to life.

The transition from the initial phase to the iterative XP development happens with the use of personas, which is why this output of the initial phase is a prerequisite for future development.

## **6.3 Implementing user-centered design with XP iterations**

As we don't have a customer, who makes the business decisions and prioritize between user stories, we need something or somebody who can facilitate this role. Needs, goals and personalities of the users as well as financial considerations should be used to evaluate which user stories (features) are more important than others.

A solution for prioritizing between user stories is to rate each story by how much value they bring to each persona as well as the financial backup. The rating should be friendly to the budget and consider that even though one user story may be more valuable to the biggest amount of users, there is only enough money to implement a user story that is half as valuable. Another consideration is that one type of user might bring in more value than the other types of users. All considerations should therefore have separate weight in the decision-making.

The user stories are short. This makes it possible to cope with the loads of user stories in order to make a decision on which to implement and how long it will take to implement them. If the user story represents a tough technical problem, it should be described in a

short separate document for the programmers as the customer would normally help out in person with this.

The users wishes are elaborated for the decision-makers at the planning meetings through personas. If we for instance have 4 different personas and a user story is only highly prioritized by 1 persona, we might choose to find another user story that has a broader prioritization among all 4 personas. In other terms, we can use the personas both as a tool to establish a common understanding of the user group, but also for the project coordinator to prioritize user stories.

When user stories are created in XP, the customer creates scenarios along with them. These scenarios are different from the scenarios that are built from personas, and are used for the acceptance test at the end of each iteration. These tests should contain small steps of interaction, that can determine whether a user story has been implemented the right way or not.

## **7 Method in use**

The method is put to test with a series of workshops. Each workshop will be a simulation of the altered XP iteration planning meetings. Test of the use of UCD tools in development is out of the scope of this product, but is planned to take place in a future workshop later this year.

To prove whether the new method works in all aspects of XP is unfortunately not possible within the scope of this project. This is a major source of error in determining if the new method of systems development is a success or not.

### **7.1 Workshops**

A series of workshops were carried out with a group of students, who were in the process of developing a system for letting users rent movies online. The development of the system was a part of a deliverable for the same course that this project was written for. As the development of their product was restricted in scope by time as this project was, the ambition was to develop a prototype of the system.

#### **7.1.1 Scope of workshops**

The students that participated in the workshop, and who developed the prototype, had no prior experience with working in an XP project. One of biggest problems Hätinen ran into when trying to test a combination of XP and GUIDe was that the test-subjects did not have any prior experience with programming in an XP setup. As I do not wish to make the same mistake and as the time-span of this project is limited, I decided *not* to let the workshops have anything to do with the actual development of the product, but only the planning aspects of XP. Testing of the programming itself has been postponed to future workshops with experienced programmers. The qualifications and knowledge that the workshop participants needed to have to be a part of the planning meetings could be taught within a short lecture on the basic principles to follow.

The development were carried out by the student group themselves without my intervention. This meant that fundamental tools of XP like pair programming, unit testing, and refactoring were not used in the process of developing the product. As the effect of these tools has been tested many times before<sup>37</sup>, I found it an accepted limitation of the testing.

It is the use of personas and scenarios for coming up with- and prioritizing user stories, that is to undergo testing, as this is the part of XP, that has undergone the most radical change in this paper.

In order not to let go of the principles of XP completely during development, I emphasized to the student group, who was to develop the prototype, that they should conduct the development having the following principles in mind:

- Keep prototype as simple as possible
- Only implement the user stories and neglect room for future features.
- Develop the prototype together as a group instead of alone as one person

### **7.1.2 Testing for proof**

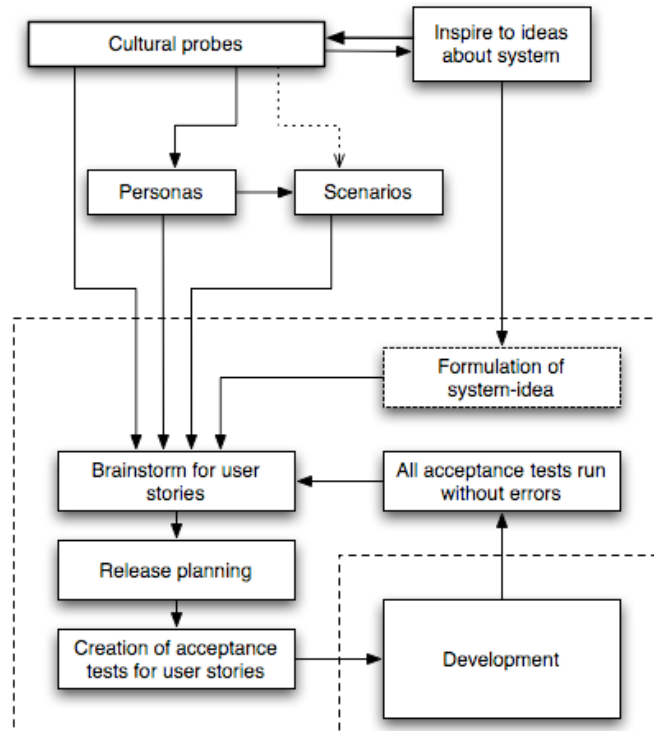
I am testing for proof, that the use of personas and scenarios can be used in prioritizing user stories, with the method I've proposed. The usability of the method is defined in the ability to produce a satisfactory development output from the decisions made at the planning meetings.

### **7.1.3 Procedure and results of workshops**

As more than one workshop was held with the student group, I used the time between each workshop for reflection on how it could have been done better. Each workshop is therefore a little bit different in its procedure of how each part is carried out.

---

<sup>37</sup> Cockburn, 2002



**Figure 2. Diagram of the procedures of the workshops. The dashed polygon illustrates the activities carried out in the workshops. “Formulation of system-idea” was only done in the first iteration, which is illustrated by a dotted border.**

Before the first workshop began, a field study of the users had been done with the use of cultural probes. The cultural probes led to personas and helped inspire the scenarios, which was created from the personas. The cultural probes also helped to inspire ideas about what the system should do. All these things were input to the first planning meeting and had been carried out beforehand by the student group themselves.

### Formulation of the basic idea

At the first planning meeting, where the student group started out by formulating the idea about what the system should do. This formulation of the basic idea was only carried out in the first iteration, which is illustrated by dotted lines around the box in the diagram.

The basic idea was formulated to be (translated from Danish):

A system that can stream video for rental purposes. The system should have functions including search, drawing between videos by lot, video top 10 lists, trailers, ordering of take-away food, reviews, as well as have a webshop with different movie-merchandise. The system should be a portal that can gather many things at one place and at the same time be easy and fun.

**Table 1. Formulation of the basic idea of what the system should do.**

### Brainstorm

After the formulation of the basic idea, the next step was the brainstorm, which used the inputs from the initial phase and the basic idea as a starting point. This resulted in around 70 different user stories, which were more or less detailed. Some user stories were as specific as “The system should be able to start and stop the video stream”. Other user

stories were more general in nature as “The user should be able to buy take-away food through the portal”. The less detailed user stories were meant to be elaborated at future planning meetings, but was written as a very general feature specification in order not to forget the idea in the future.

The user stories that are easy to come up with in a brainstorm are functional features. It is much more difficult to concretize visual design issues like “the site should support user experience” or “the site should be friendly to disabled visitors”. Statements like these came up on the second planning meeting. We decided that two developers from the student group were appointed to spend their efforts in the next iteration on concretizing the vaguely defined visual design statements.

At the third planning meeting, the concretized visual design statements were presented. Examples of user story titles for these statements were: “Mark form fields red if error”, “Generic film list design”, and “Navigate to top link on all pages”. These stories were all chosen for implementation in the fourth iteration while others were sent back for further concretizing.

### Release planning

Next step in the workshop was release planning. This is the step where user stories are estimated and grouped into iterations. This is also the step, where the changes to the XP method has the most influence. In order to find out which user stories to implement in what iteration, each user story was scored by each persona. Zero was given in score if the persona had no interest in the user story, 1 if the story had some value, and 2 if the user story would do something important for the persona. The scores were then added up for each persona, and soon the user stories with the highest priority was found.

	Persona 1: Stine Willum	Persona 2: Søren Nielsen	Sum
Show how long the movie is	2	2	4
Show link to trailers	1	2	3
Gift cards	1	1	2
Filter for children	0	2	2
Buy flowers through the portal	0	0	0
Etc.	-	-	-

**Table 2. Example of how user stories were prioritized at the first planning meeting.**

After evaluating the first workshop, it was found that there needed to be some voice of the financial backup of the project. Therefore it was decided to add another point-giver called “the customer”, that could also score each user story from 0 to 2. After reviewing the literature, it was found that Jonathan Grudin and John Pruitt (2002) had already done something similar before. Ideas from their priority matrix resulted in adding a -1 to the score if the persona or customer was confused, annoyed or harmed by the feature as well as adding weighted scores so that opinions from either one persona or the customer would weigh more or less than the others. The sum of the weight should be 100 so that if everybody fancies the feature and rates it 2, the sum would be 200. If the feature harms everybody, the sum would be -100.

The weight of each persona and the customer was determined by a discussion in the student group. The specific rating of a user story is based on arguments that might not have any connection to the rating of another user story. For this reason, it is important to underline, that the sum is not a definitive tool to find which user stories to implement in the next iteration, but merely a guide to help in making the decision. Adding comments to the difficult-made ratings can further help the final decision-making.

	Persona 1: Stine Willum	Persona 2: Søren Nielsen	Customer	Weighted sum
Weight:	20	30	50	
Show how long the movie is	2	2	0	100
Show link to trailers	2	2	2	200
Gift cards	1	1	2	150
Drinking games for films	1	-1	1	40
Free simple games	1	2	2	180
Etc.	-	-	-	-

**Table 3. Example of how user stories were prioritized from the second planning meeting and onwards.**

The top 20 rated user stories were chosen for further investigation. It was then decided which user stories were to be implemented in the next iteration among these regardless of their scored sum, but instead based on a human judgement of what seemed more important to implement first.

As the participants had no experience in estimating how much time user stories take to implement, we decided to skip this part. Instead the judgement of how many user stories it was possible to implement in the next few iterations were made, by dividing the top rated user stories into groups that seemed within the reach of being implementable in one iteration.

### Acceptance tests

For each user story that was selected for the on-coming iteration, one or more acceptance tests were created. An acceptance test is a basic step-by-step scenario that should be able to run, when the user story has been implemented. The step-by-step scenarios were created in a manner, so that they could have been automated by a script instead of run manually. Examples of the acceptance tests are:

#### **Search for actor**

- Step 1:**  
Fill in "Tom" in the search text field
- Step 2:**  
Click "Search" button
- Step 3:**  
Find and click "Tom Cruise" link in the list of search results
- Step 4:**  
Find the text "Tom Cruise" in the info page about Tom Cruise

#### **Send link to film-trailer**

- Step 1:**  
Find film
- Step 2:**  
Click "Send trailer to a friend"
- Step 3:**  
Fill in e-mail in e-mail text-field
- Step 4:**  
Click "Send link"

#### **Pay with credit card**

- Step 1:**  
Click "Pay" link
- Step 2:**  
Find the text "Order information" and title of the film selected on the page
- Step 3:**  
Check "I accept the conditions"
- Step 4:**  
Fill in credit card info in the form fields
- Step 5:**  
Click "Pay" button

**Table 4. Examples of acceptance tests on user stories.**

After creation of the acceptance tests, the student group parted to develop the prototype. At the second planning meeting, the acceptance tests were run. Normally this will take place well before the planning meeting, so that any errors can be corrected. As I had decided only to do workshops to test the method, time were used in the beginning of the next planning meeting to verify that the acceptance tests could run.

As all acceptance tests could run, it was time for brainstorming for new user stories as the uncertainty about what way development could go had been made smaller during development. The whole treadmill of the planning meeting had then started, as the next step was release planning, creation of acceptance tests, development, etc.

## **7.2 Future workshops beyond this project**

The next step is to take the new method further beyond the planning game and into actual development. The main challenge here is to communicate the personas to the developers by letting them engage in the personas. Writing personas and persona-based scenarios for the developers to read is one way – another is to let the personas come to life.

Grudin & Pruitt (2000) has used a variety of initiatives to let the personas come to life. Their idea is to routinely put small bits of persona information in front of the team. This can for instance be done by creating e-mail addresses for each persona in order to send out fictitious mails to the developers from each persona. They also maintained a website with persona information as well as creating posters, flyers, handouts, and giveaways that has persona information on them. Posters can include persona comparison information or just information about one persona. Grudin & Pruitt (2000) think of the persona effort as an on-going campaign and it should be.

Personas can be used for more than just prioritization user stories. A straightforward expansion of the use of personas is to create acceptance the tests with a base in the personas. As different personas have different ways of reaching their goal, they also have different ways of using the feature implemented in the user story. One or more acceptance tests could be created for each persona rather than for what the developers think is right.

In order to find defects or things that could be better, a task-group can be assigned in an iteration to go through the software product in order to see if things can be improved for one specific persona. The result of this examination would then be turned into user stories at the next planning meeting.

These are some of the tools, which are to be put to test in the future planned workshop with a company that uses XP as their preferred systems development methodology.

## **7.3 Changes to the principles of XP after implementing UCD**

After changing the processes of a systems development methodology like XP, there is a danger of tampering with the basic principles that the methodology was first built on. For XP, these principles are in the rough constituted in the agile manifesto, which was

mentioned earlier in this paper. There are especially two of the four principles of the agile manifesto that could use a closer look.

By introducing prioritization of user stories by personas, a new tool is added. This could be against the principles, which states that *individuals and interactions should be chosen over processes and tools*. This dilemma should be seen as a judgement on which tools to use that varies from project to project. This tool of persona-prioritization has been added to solve a problem that was found to exist in some projects – other projects might not have this project, in which case there is no need to implement the tool. Whether the tool should be used or not can be compared to the decision whether to use XP or some other methodology: different projects require different development methodologies.

The second principle in danger is that of *working software over comprehensive documentation*. As long as the added documentation helps more than it hurts, this principle should not be in danger. Documentation is not the forbidden fruit of XP, but merely something that is favored less as working software. I have chosen to use the extra documentation of personas and scenarios especially at the planning meetings, whereas development is another thing. Here I propose other methods of letting the developers learn about the personas, which does not include reading long documents on each persona.

#### **7.4 Advantages and disadvantages of the new method**

The new method has solved some of the problems that XP had when not using contract development as well as in general, including too little focus on the actual users of the system. For the method to be able to run perfectly smooth, some problems needs to be addressed: some that are present in traditional XP and some that has hit the surface with the introduction of this new method.

The number one problem in the workshops was process of concretizing visual design ideas into user stories. The participants requested some sort of tool, which could help concretize the visual design ideas in the time of one iteration. It was a frustration over how important a good user interface was rated in the theory, and how hard it was to use the theory in practice. The participants did not have the experience in designing user interfaces and could not engage in the personas enough to know exactly what concrete ideas the personas had about the design.

At the start of each planning meeting, the participants had fresh minds. The details of the discussions that went on at the prior planning meetings were put away in less visited places of the participant's brain. This meant that unless the details had been written down in details in the rating-comments, the rating of each user story had to somewhat start over. This resulted in the user stories being rated differently from meeting to meeting. Whether this was due to new knowledge is not clear. A user story that had top ratings could easily go from a 1<sup>st</sup> to 30<sup>th</sup> place in overall rating. With numbers for ratings, and only two personas to make a difference, it is not much that makes the difference. A solution to this problem would have been to introduce more personas in order to get a more detailed overall rating.

The ratings of user stories with importance for each persona and the customer follows a market point of view. As mentioned before, user interface stories are rated very important by the theory, but did not necessarily have a top score between other features. This can be explained by the fact that getting the system up and running is more important than the font-headers being colored blue.

## 8 Conclusion

The use of personas to prioritize between user stories has proven to be an effective combination of UCD and XP. The conflict in interests the XP customer has in traditional XP development; by both representing financial interests of the client as well as the interest of specific end-users, has come closer to a solution. The customer, who in XP speaks the voice of all end-users, has been elaborated to include a much more detailed picture of the different end-users.

The method is still agile, but is so in a more detailed sense. The different personas each represent different directions the project can go, thus the risk of missing out on certain end-user problems has been minimized.

With the extra UCD tools added to the initial phase as well as inside the development loops, the method is more explorative than XP or UCD techniques alone. Cultural probes explore and bear ideas and personas concretize them.

XP has been expanded to take more organizational problems into considerations with the personas. There has been a shift from XP's focus on the system to a more equal focus on both the system and the users.

## 9 References

- Abrahamsson P. et al., *New Directions on Agile Methods*, 25<sup>th</sup> International Conference on Software Engineering, Portland, Oregon, USA, 2003
- Beck K. (1), *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000
- Beck K. (2) & Martin Fowler, *Planning Extreme Programming*. Addison-Wesley, 2000
- Christensen S. & Kreiner K., *Projektledelse i løst koblede systemer*, Jurist- og Økonomforbundets Forlag, 1991
- Cockburn A. & Williams L., *The cost and Benefits of Pair Programming*, University of Utah Computer Science, 2002
- Cusomano M.A. & Selby R. W., *Microsoft Secrets*, Harper Collins Business, 1995
- Floyd C., *A systematic look at prototyping*, Intitut für Angewandte Informatik, 1984
- Fowler M., *Refactoring*, Addison-Wesley, 1999

Gaver B., Boucher A., et al., *Cultural Probes and the Value of Uncertainty*, Interactions, 2004

Grudin J. & Pruitt J., *Personas, Participatory Design and Product Development: An infrastructure for Engagement*, PDA 2002, Malmö, 2002

Hughes B. & Cotterell M., *Software Project Management*, Second edition, McGraw-Hill, 2000

Hätinen A., *Extreme Programming and Goal Oriented User Interface Design in Practice*, Research Seminar on Software Engineering, University of Helsinki, 2002.

Matthiasen L. et al., *Object Oriented Analysis and Design*, Marko Publishing Aps, 2000

Nielsen L. (1), *Engaging Personas and Narrative Scenarios*, Samfundslitteratur, 2004

Nielsen L. (2), *Design through Engagement – HCI working paper*, CBS-MIK 2004

Norman D., *Emotional Design*, Basic Books, 2004

Preece J. et. al., *Interaction Design beyond human-computer interaction*, John Wiley & Sons, 2002

## **9.1 Websites**

Manifesto for Agile Software Development  
- <http://agilemanifesto.org/>

Nelson E., *Extreme Programming vs. Interaction Design*, FTP Online, 2002-01-15,  
[http://fawcette.com/interviews/beck\\_cooper/](http://fawcette.com/interviews/beck_cooper/)

Extreme Programming: A Gentle Introduction  
<http://extremeprogramming.org>