

Extreme Programming and Goal Oriented User Interface Design in Practice

Antti J. Hättinen

Espoo 1.11.2002

Research Seminar on Software Engineering

UNIVERSITY OF HELSINKI

Department of Computer Science

<http://www.pharazon.org/publications/GO-XP.pdf>

Abstract

Extreme Programming and Goal Oriented User Interface Design in Practice

Antti J. Häätinen

Research Seminar on Software Engineering

Department of Computer Science

University of Helsinki

1.11.2002, 18 pages

The fundamental question of software engineering (SWE) is how to provide software that actually meets the end user needs. The economical problem is there after how to produce the SW that meets the set quality level in predefined time and budget. Interaction Design and Sari A. Laakso's GUIDe are visionary methodologies for user interface design where the primary emphasis is focused on finding the user's real goals and optimising the design to meet these goals with minimum cognitive and physical effort. Kent Beck's Extreme Programming (XP) is first of the new wave of lightweight SWE methodologies introducing a set of old, proven practices in a new and balanced way. There exists a fundamental difference between these two approaches; GUIDe emphasizes research and planning, while XP adapts to changing customer requirements flexibly. The GUIDe states that users can't usually argument their true needs, thus there exists a chance to extend XP further by introducing GUIDe to the planning level and keeping the current practical SWE practices. Combining these two methodologies, however, provide opportunity to solve the economical and quality problems of SWE. Thus, we introduce a new combined methodology called Goal Oriented Extreme Programming (GO-XP).

The author practiced the proposed new methodology on the SWE Project - course in University of Helsinki Department of Computer Science during

summer 2002. The goal of Keltsi-project was to produce proof of concept prototype of Internet yellow pages using semantic web -technology.

In this article, we first review the main points of the two original methodologies and then the actual experiences of the usage of the combined methodologies during the Keltsi-project. After this we synthesize the proposed new methodology. Finally we suggest the needed focus areas for the development of the new methodology.

Classification (Computing Reviews 1998):

D.2.1, D.2.2, D.2.9, H.1.2, H.5.2

Keywords:

User Interfaces, Extreme Programming, goal, Interaction Design, GUIDe, Keltsi, GO-XP.

Index

Abstract	i
Index	iii
Glossary	iv
1 Introduction	1
2 Extreme Programming.....	1
3 Goal Oriented User Interface Design	3
4 Critique of Methods.....	6
5 Experiences and Suggestions	7
5.1 Experiences of Keltsi Project	7
5.2 Suggestions for User Interface Design.....	9
5.3 Suggestions for Project Management.....	10
6 Goal Oriented Extreme Programming.....	11
7 Conclusion.....	15
8 Summary	16
References	17

Glossary

Term	Definition
Acceptance Test	An automated test written in code that tests that the user story and UI design is implemented as specified. Written by the design team in GO-XP.
Coach	Educates, motivates and leads the XP team to make decision by them selves.
Customer	{In XP} the representative of the projects client, who tells the requirements of the project in form of user stories, and between iterations evaluates and steers the project.
Extreme Programming	A new lightweight SWE methodology
Goal	Real end user's unconscious or conscious goal upon she performs action. Doesn't imply technological solution how the goal can be met. Can be found by asking <i>why</i> a person performs action. For example, calling to mother is not a goal, but wanting to congratulate her on her birthday is. Goals are not likely to change. Cooperian goals are very abstract, such as " <i>not wanting to look stupid</i> " [Cooper95].
GO-XP	Goal Oriented Extreme Programming. Combination of Extreme Programming and Interaction Design.
GUIDe	Goals - User Interface Design - implEmentation, an Interaction Design methodology by Sari A. Laakso.
Interaction Design	Design of requirements and UI by researching the <i>goals</i> of the real end users and communicating the requirements to implementers.
Interaction Designer	Not just a user interface designer, but requirements designer, who studies the user goals and communicates the goal optimised design to programmers.
Iteration	A one to three week period during several user stories are implemented and after a working program is ready.

Persona	A description of an example user. May be a combination made of several users [Cooper99] or a description of a single real user [Hackos98]. Accompanied with goals.
Planning Game	XP's design meeting, where the user stories are prioritised and the changes are designed and split into daily tasks.
Release	Typically for example a 3 months period, during several iterations is done.
Step	Atomic user interface action, such as a click or moving a hand.
Sub-goal	A design usable goal. Not as abstract as goal. Includes a realistic description of the situation.
SWE	Software Engineering
Task	Set of steps to perform a specific action. Several tasks lead to a <i>goal</i> . Task and step paths are optimised in Goal Oriented User Interface Design.
UI	User Interface
User Story	A story written by the XP customer or GO-XP design team that specifies the requirements of the system but doesn't imply design.
Walkthrough	The UI design (paper) prototype is reviewed by walking through the goals with real user.
Want	Marketing generated task, not a goal.
XP	See Extreme Programming

1 Introduction

The holy grail of software engineering (SWE) is how to make products that meet the customer needs within the specified time and budget. *Extreme Programming* (XP) [Beck00a] is a new lightweight SWE methodology addressing the problems of quality and schedule problems with combining old proven SWE practices to a new balanced combination. *Interaction Design* (and *GUIDe*) [Laakso00, Interacta02] approaches the problem by researching user's needs, culture and context and thus trying to find the requirements better than the customer and user can articulate by themselves.

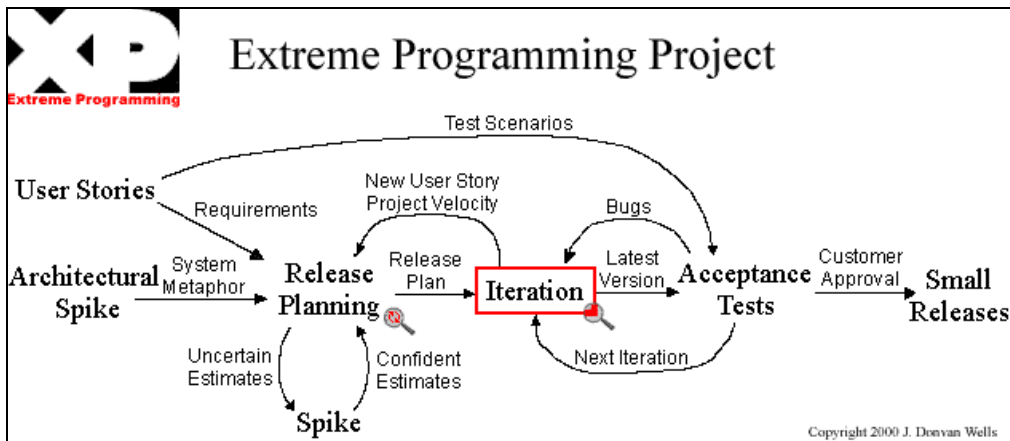
A combination of XP and *GUIDe*-methodologies was used on Keltsi – software engineering project in University of Helsinki Department of Computer Science during summer 2002 [Keltsi02, Hyvönen02]. The subject of the project was to make proof of concept prototype of Internet's yellow pages using semantic web technology.

In this article we first introduce the Extreme Programming and Goal Oriented User Interface Design –software engineering methodologies that were used in the Keltsi project. After the introduction, we review the critique presented in the literature and the experiences gained from the Keltsi-project. Finally we introduce a new combined methodology called Goal Oriented Extreme Programming (GO-XP).

2 Extreme Programming

Extreme Programming (XP) is first in the series of new **lightweight SWE** methodologies that are a counter movement against traditional, heavy and document oriented methodologies. XP's practices have actually nothing new. The new approach is to **combine old, proven SWE practices** to a new balanced palette, where the strengths and weaknesses are combined in a harmonious way.

Major XP practices include *fast iterations* (1 to 3 weeks), after which software is always ready to be submitted to the customer, *automated testing* that allows *simplistic coding* and flexible *refactoring* [Fowler99] of the design [Picture 1]. The *customer* writes *user stories* about the features he would like to have in the system. The customer priorities the user stories in a *planning game* meeting by the business value it would bring to the system.



Picture 1 - Extreme Programming Process Model [Wells02]

The XP's approach is also attractive from the employee's point of view. In the IT industry there is lack of experienced and skilful employees, and thus the companies must compete of the skill. The **XP offers humane and motivating working methods** that should persuade the top-level employees to the XP companies. Methods, such as *Focus on quality* and *Fast iterations*, allow people to see the results of their high quality work fast and thus rewarding. *Pair programming* brings in the social dimension of the work, and now everyone has a chance to feel an important part of a closely integrated team. Results [Nosek98] using two sided t-test show statistically significant evidence that **pair programming helps to complete tasks 40% faster than individual programmers**, while the people like the work more and the quality of work is better. The *40-hour week* focuses on the quality of the work and the work stress. The ideology is that the 8 hours in work should be of maximum efficiency and after this the rest of the day should be spent relaxing.

The benefits of these and many other practical methods are that the project can **focus on** internal and external **quality instead of scope** as in traditional methods. The argument is also that with these methods we can keep the *cost of change* limited [Beck00a] and not growing exponentially like in waterfall-methodology [Boehm81]. Fast iterations of XP are also good tools to **decrease the risk of failure**, since the customer can have feedback of the development in few week intervals and has the ability to steer the project [Beck00b].

3 Goal Oriented User Interface Design

In the user interface design community exists many different design methodologies that have different backgrounds in relevant fields of science. For example, from the field of engineering comes the usability engineering [Faulkner00] where the focus is in usability evaluation and testing. However, most of these methodologies don't reason the basis of their theories and on what theory these evaluations and tests are based on (other than the practioners experience).

There exist also several definitions for usability. ISO 9241-11 defines usability as follows [ISO9241-11]:

Usability: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

The effectiveness means the accuracy and completeness the users achieve specified goals. Efficiency measures the required resources expended to achieve the goals. Satisfaction measures the comfort and acceptability of use.

Interaction Design [Cooper99] and GUIDe [Laakso00] approaches the problem of usability from the user's point of view. **All devices are seen as tools for users to meet their goals.** A usable device is one that achieves user's goals with minimum amount of steps, and thus effort. Users activities

can be modelled using task analysis to a set of goals, sub-goals, tasks and steps [Hackos98]. Goals are not likely to change even though the user's workflow changes. By researching the user's goals by observation and interviews we can see the user's goals better than she can articulate them by herself.

Other important things to research are user's culture and context [Beyer98]. These are important factors in the design phase, when we have to consider the language we communicate with the user. From the marketing point of view knowing the cultural context is important in sense of creating ways to differ from the competitors in the target segment's way of thinking. Industrial design seeks also ways to communicate visually with the user preferably with her own language and values.

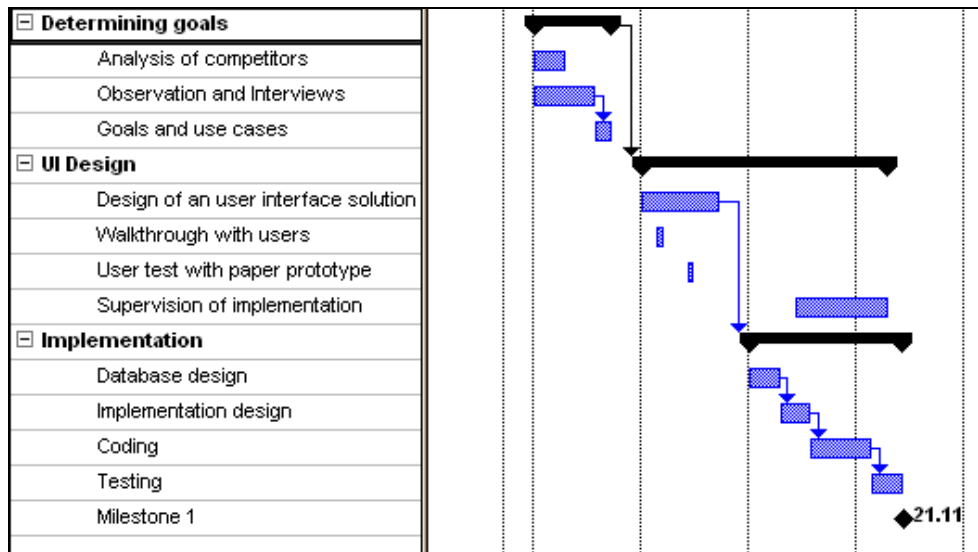
User's goals can be seen linking to Maslow's Need Hierarchy [Maslow54] in general level thus causing a basic *need* to transform into action. User's goals can be ultimate found by asking *why* he performs particular task. A goal doesn't ever relate to the technology or way to meet it, but has several optional task paths to implement it. For example a user walks a street in downtown and remembers that today is his mother's birthday. He wants to congratulate her mother and has several options how to do that. He can take his cellular phone from his pocket and call her, or he can go to the railway station and take a train to her hometown. *Why* he wants to congratulate his mother? He wants probably to maintain his good relationships with his family and to make her mother happy (goals). After formulating (consciously or unconsciously) his goals and sub-goals, he has several optional tasks he can perform with different level of physical and cognitive burden. Tasks can finally be split into atomical user interface steps, such as a click or movement of a hand. Interaction Design argues that user will choose from the optional tasks the one that he feels by his experience needs least effort. Thus, he would probably call his mother with cellular phone or we can design even more efficient ways to meet the goal.

We suggest that the **efficiency of usage** (the spent resources) can be **measured by simply counting steps needed to achieve a specified goal**, both the cognitive and physical burden. The best possible user interface is thus a device that requires zero atomic user interface actions to complete one's goal. An example of this is a sliding door, where the passer's goal of moving through the door is achieved without her needing to make any special actions (other than maybe slowing down a little). The house owner's goal of keeping the house warm is achieved also with the same device.

Sometimes the goals can be too general to be useful in UI design. For example it would be hard to draw conclusions Therefore, we would suggest the division between goals and sub-goals so, that the sub-goals are usable for design. This means that the sub-goal includes information about the realistic situation [Laakso00 – Personas' Goals]. For example persona called Markku might have a design usable sub-goal: *"I will go from home (Mäkeläinkatu 55) to a meeting (Tapiola, Espoo) tomorrow morning. The meeting should begin at 9 am."*

The process of GUIDe [Interacta02] [Picture 2] begins with the determining of goals by analysing the competing products and researching the actual end users by observation and interviews [Laakso00]. After the goals are found, a requirements specification is made including the *personas*¹ and goals describing the users. After the goals are found and prioritized, the user interface is designed by the goals to make the work flow as straightforward as possible. The usability is reviewed with usability *walkthroughs* and *user tests*. After this the detailed user interface specification is made and the implementation phase may begin. The designers spend time supervising the implementation of the specification and answering questions to details. The design and implementation can be further iterated.

¹ See Glossary - persona



Picture 2 - GUIDe Process Model [Laakso00]

The **benefits of** applying the **GUIDe** UI design process are that the customers are more interested to the final product than for the competitor's product thus **increasing the sales**, and the product will be **easy to sell** since the demonstration will show the power of the product in the most recurring use situations. The GUIDe designed product also **increases the efficiency of work** (in Tayloristic sense as in Ford's automobile factory) and saves money for the company as employees spend less time performing their tasks. The **need for user support and training** will be also **lower**.

4 Critique of Methods

One major concern in XP (and other SWE methods in general) is that how close it is to reality of common employee. **The basic assumption** in all methodologies is that the **team members are high-level professionals**. Often, especially during the latest IT boom, firms were hiring people almost straight from the street and school. It takes an extra effort to educate people new methodology and way of thinking to people who haven't got the experience. On the other hand, it might be as hard (or harder?) for the seasoned workers to unlearn the old SWE thinking when changing to XP. The conclusion is that a special **focus** must be put **on educating the people** with the theory and practice of XP when launching the project or bringing in

new people. Reports [Kärkkäinen02] show that XP can be a lot more inefficient than standard waterfall method when the team members aren't already familiar with the methodology. Making automatic tests was felt the most difficult part of the new process [Keltsi02, Kärkkäinen02]. An **experienced coach is required** for the team to teach and balance the methodology [Pelrine00] and *“to make get everybody else make good decisions”* [Beck00a, p.73]. *“Special skills and capabilities (won through experience) are required in order to live up to the requirements of a good coach”* [Pelrine00].

Making fancy user interfaces might not be justified when the product is a completely new concept and **there aren't any direct competitors**. In this situation the product differs already a lot by the technology or the new concept. However, even when there doesn't exist direct competition the product still competes versus other products that allow achievement of the same user goals.

Kent Beck criticizes Interaction Design of creating a bottleneck for the SWE performance by adding the interaction designers researching the users [Nelson02]. He argues also, that adding early requirement gathering and design phases to the SWE process is not as powerful as Interaction Design would be spinning inside XP practices.

5 Experiences and Suggestions

Next we review our general experiences of the Keltsi SWE project. After this we summarize the suggestions for the project manager and interaction designer in GO-XP project context.

5.1 Experiences of Keltsi Project

The first finding was that the **initial hypothesis of mapping the goals and user stories directly didn't work**. The problem was that goals were not concrete enough to imply the design. Representing the user goals directly to programmers as a design question raised the conflict between user interface

and software architecture design and we weren't able to speak in the same language or the same problem. Thus, it seems that the user interface design and sketches have to be completed well before the *planning game*. While we lacked the real customer for almost two months during the worst times, it would have been imperative to have the interaction designer communicating the requirements to the team.

The XP team is sensitive to the attitudes and abilities of the team members. While forming a XP team, a special **focus** should be put **on choosing suitable people** who are willing and interested of physically close team and pair work. For example in a 4 people team a single person's unwillingness to participate in pair work will render 50% of the team useless. A soloing and uncooperative team member can cause harm by not allowing the pairs other person to learn, by writing code not understood by others (since the communication offered by pair programming is lacking) and by after this having days off. The team has to decide whether to discard the changes and start from zero, or spend all day searching and learning the changes. Thus, I would suggest using interviews and psychological tests (such as Belbin's team role test [Belbin81]) to screen the most suitable team members. The key issue is the willingness for teamwork, though.

The close by teamwork will cause **XP** to be **inflexible for changes in working time and place**. A XP team should be working steadily in a same place from 9 AM to 5 PM all the time without outside interruptions. In our project we had the ability to work 20h/week in a dedicated working room for the whole summer, which was a good thing. We didn't notice this before the end of the project came closer and people started to have more and more days off. We didn't have other communication tools than the physical working environment and thus it was quite hard to work outside the agreed working hours. Often, in the final and hectic days of the project, most of the team didn't have an idea how the program actually worked or was structured. In this sense, XP is much more inflexible with working time than

many other methods. It's essential that people are coming to work strictly the same time. Pair programming is an important tool to handle the eventual leaves so that they don't cause harm to the proceeding of the project since the knowledge is now shared.

There should be **two separate people doing the project management and interaction design** (ie. the XP client). The experience was, that the daily project management routines, education and coaching consumed the hours so intensively that the user interface design activities suffered. The interaction designer should somehow have time to write also the acceptance tests for her design to make sure that the suggested user stories really exist in the system as indented. In our project, the programmers got the power edge in requirements specification since the interaction designer didn't have time to express the stories with the language understood by the programmers (code). Since, I would agree that also in the GO-XP depending of the interaction designer's skills and time, it would be a good idea to have a dedicated tester to write the *acceptance tests* [Beck00a] especially when the same person is also the project manager.

Fast prototypes are a good way to reduce risks and increase learning for inexperienced teams. Also it might be good to have faster iterations for inexperienced teams so that the prototypes could be applied fast. The more experienced teams already have the knowledge of most common implementing methods, and they can handle bigger pieces of the release.

5.2 Suggestions for User Interface Design

The **interaction designer should play the role of XP's customer in GO-XP**. Thus, the interaction designer is the person making the requirement and business decisions for the project and should have the corresponding point of view. Creditability of the designer in the eyes of the programmers depends heavily on her ability to represent real data of the end user needs (not assumptions). To do this, the designer has to do the hard fieldwork observing and interviewing the actual end users [Hätinen02]. After this the

user goals, culture and context must be represented to the programming team so that the design is based on facts and not on subjective assumptions. Otherwise there is danger of programmers running the project [Cooper99] not the client.

The basic assumption of **interaction designer** knowing better the customers needs **require also humble attitude towards the customer**. The designer has to listen to the customer and the user carefully to be able to draw the conclusions. The interaction designer is the flag carrier of both the designers and programmers, so she should build rapport in every direction and try to represent well her background groups.

5.3 Suggestions for Project Management

In GO-XP, the project manager is responsible for the efficiency of the production team. Thus, he **should act more like a coach of the project than a manager**. However, being a coach requires the project manager a special experience and skill set.

Unless you are using quick and dirty paper management tools, you have to have highly *usable* project environment. In our project, we had obsolete Pentium MMX computers, which slowed down our everyday work. But even greater problem were our meetings where we couldn't use otherwise magnificent WikiWikiWeb² –documentation tool since we didn't have laptops. With Wiki, it would have been quite easy to type the minutes straight into the management system, and thus we would have saved many hours of copying hand made notes.

Other problem was the lack of UML design documentation in the design sessions. It was hard to design the system while we most of the time didn't have the overview what we currently had. We decided to use UML drawing program, but since only one or two people actually knew how to use it, the

² <http://phpwiki.sourceforge.net>

drawings didn't take place until the last few weeks of the project. The **suggestion is to have courage to use just paper to draw the architecture** unless you have really usable UML tool. During the project, we tried many tools but didn't find any as good as paper. The documentation policy forced by the course's management made using the most efficient methods hard.

The project manager's main responsibility is to control that the methodology is strictly followed. Any loosening of process discipline is dangerous and should be monitored closely. The learning of completely new methodology is hard and under pressure people tend to quickly fall back to their older habits [Beck00a]. If this happens, lots of hours of education will go in vain. Good tools for controlling the performance are for example **setting clear performance targets for each phase of the project** (iteration, release, whole project, vision of the company) and communicate what is expected clearly and regularly.

6 Goal Oriented Extreme Programming

To solve the XP's lack of usability point of view and to map a pragmatic implementation phase to GUIDe, we would suggest a new combined methodology called Goal Oriented Extreme Programming (GO-XP). XP and GUIDe are two fundamentally different approaches to SWE [Nelson02]. XP assumes that the requirements are always shifting, since the customer changes her mind all the time, and therefore tries to adapt by being flexible to change. The basic assumption of GUIDe (ie. *Interaction Design* by Alan Cooper) is that the customer and user can't articulate what they really need by them selves, and there must be an *interaction designer* who researches the user's goals and communicates them to programmers.

The key issue is how to do the transition from goals to user stories, ie. how to communicate the requirements perceived by the interaction designers to the language of the programmers. Kent Beck states: "*XP says that you have a conversation between the customer team and the*

engineering team" [Nelson02]. Thus, it would be possible to replace the actual customer from the place of the customer team communicating to engineering team, and replace it with interaction design team as suggested by Alan Cooper.

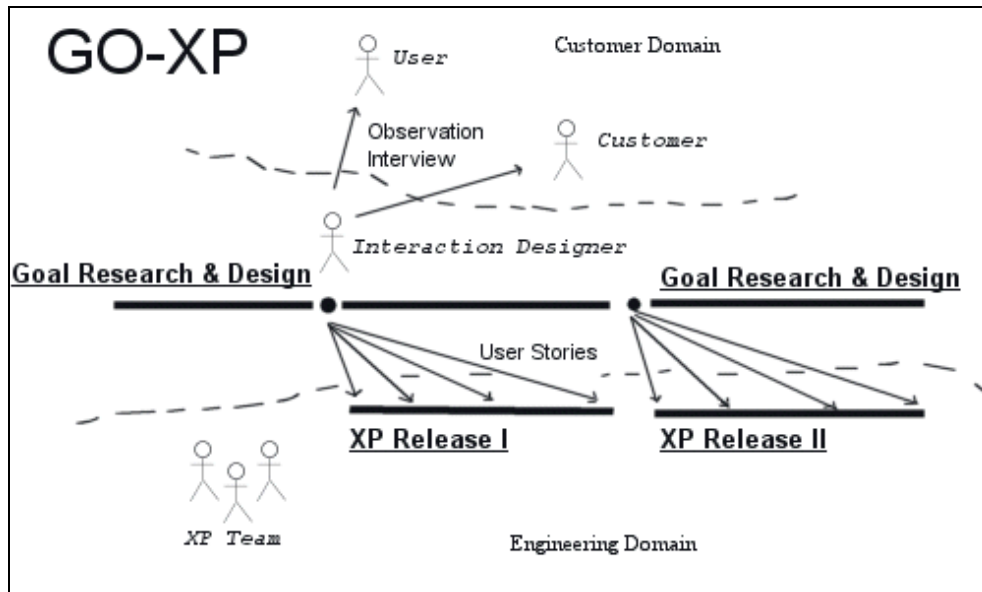
The communication between the XP-team doing implementation and the interaction team works in form of *user stories*. **The customer team then writes acceptance tests for the user stories.** Thus, if the requirements are written into tests (in code), they should be explicit and well understood by the programmers. Interaction designers should be specialists understanding both the world of programming and the world of the customer. Thus this would be a good way communicating with the programmers with their own language without a need to learn any new way of communication. This might also be more efficient way of communication to the programmers than talking back and forth, as suggested by Alan Cooper in his original Interaction Design [Nelson02, p.10].

The GO-XP process model [Picture 3] **adds a goal research and design phase preceding the normal XP release.** During the research and design phase the interaction design team observes and interviews the customers and real users acquiring the personas and goals. The interaction designers then prioritise the goals by their business value with the customer. The prioritised list of goals is then the basis for the user interface design of the release. The designers write the user stories, make walkthroughs and user tests with prototypes, and finally write the user stories into acceptance tests. The user stories, the user interface specification and the acceptance tests are then presented for the XP team who start to implement them by the normal XP methodology. The only difference is that the **interaction designer is playing the role of customer** for the XP team.

The preceding design phase raises the question how well does iterative design apply to user interface design. If the user interface is changing very

often, the users can't learn it and get frustrated. On the other hand, if long periods are spent just designing the UI, the implementation is postponed. In today's fast time to market requirements this is an imperative. It's also easier to get finance for smaller projects with lesser risk. **A balance must be found that how often the user interface can be iterated.** There exists also problem on the implementation phase of implementing the release level user interface specification in short iterations. It would seem that it is not possible to release the iteration products to the production system after every iteration if the UI design is not fully implemented and thus in balance. Also the varying performance of the implementation team causes problems for the user interface specification. The designers have to take account and design also the iterative implementation of the key features. How this is done in practice needs further study. Our suggestion with the experience from the Keltsi-project is to not to do more UI design than for each 3-month release. This however practically doubles the project length if the design phase is preceding the implementation. It should also be studied how much is it possible interleave the design and implementation phases. On the other side are the users who don't like the UI changing every week.

The advantages of GO-XP versus XP would be that the **requirements are better specified** than the real customers (representatives of the client) or users could possibly articulate thus reducing the need of shifting in XP project by three or four orders of magnitude [Alan Cooper in Nelson02, p.10]. Secondly **the quality of the requirements is** argued to be **better** and the **design** derived from them is supposed to be **of much higher level** with the Interaction Design phase of GO-XP. Also the **client organisation doesn't need to commit any full time resources** to the project as in XP. GO-XP also is suggested to **reduce the risk** of project failure, since the user requirements are specified with better quality.



Picture 3 - GO-XP Process Model

The personnel of GO-XP team should be divided to an interaction design team and a XP team. The optimal size of the design team would be 2 to 4 people. This way the pair programming principle could be accompanied also to research and design. The size also depends how the research and the acceptance test competence can be divided. The ideal duo for design would be persons who can do the field study, understand the business issues, are able to design by goals and write the acceptance tests in code. The good maximum number of people for the XP team would be 10 people. The optimal number of programmers is 4 to 8 people (even number). The team always requires the coach to educate and manage the team.

The weaknesses of the GO-XP might be the additional professionals needed for the project. Skilled interaction designers are rare [Nelson02]. Also the XP team's performance depends highly on the abilities of the coach [Pelrine00]. GO-XP is definitely not a SWE process model for inexperienced teams. XP is also inflexible in terms of working time since all team members have to be present strictly on the working hours and in the same physical space. As the team's experience grows, it would be reasonable to move towards longer (3 week) iterations and not-so-simplistic

design as XP suggests, since the project risk is reduced with Interaction Design.

GO-XP uses all other practices of both XP and GUIDe with the exception of the differences mentioned above. The interaction designers should demonstrate the products also to the real customer for example once per month to get feedback, commence walkthroughs and usability tests, and to generate marketing goodwill.

7 Conclusion

Goal Oriented Extreme Programming (GO-XP) is a new process model combined of Extreme Programming and Interaction Design (GUIDe) software engineering methodologies. The GO-XP seems to solve the problems of combining XP with usability point of view. This, however, would need practical proof by applying the process model with a professional team. Current experience tells that this is not a very efficient methodology at least for inexperienced teams.

The critical points of GO-XP are to find skilled interaction designers and an experienced coach for the XP team. Without these key persons being seasoned and motivated the whole process model crumbles. Running a normal waterfall –process model is safer with inexperienced people.

The next step would be to try GO-XP in practice preferably with a team of real professionals to gain more experience. An important step would be measuring the performance levels before and after the change of methodology. An important view would be also how long does it take for a GO-XP team to learn the same performance level as with their original methodology.

Other interesting focus areas would be how to run XP on bigger projects and larger organisations than 10 people. The open source practices could be

studied to help overcome the restraints of time and physical presence. An interesting view would also be the graphical and industrial design point of view for building the aesthetics of the product in cooperation with marketing practices.

8 Summary

Extreme Programming (XP) and Interaction Design are two fresh methodologies for software engineering. XP promises focus on quality and lowered risks by increased flexibility to steer the project in a more rewarding working environment. Interaction design promises to find out the user needs better than the users can articulate by them selves, and using this deeper information design the new product to be more efficient and usable than any solution before.

The literature [Nelson02] and experiences gained from the Keltsi –software engineering project in University of Helsinki summer 2002 suggest that a combination of these two methodologies could be even more efficient. Therefore a new Goal Oriented Extreme Programming (GO-XP) methodology was introduced. GO-XP integrates XP to the implementation phase of GUIDe (an Interaction Design methodology) by communicating the user goals and the user interface design by programming them in acceptance tests, language well understood by programmers.

However, the new methodology requires highly experienced professionals to work as interaction designers and XP team coaches. The evidence shows that GO-XP is not suitable for inexperienced teams. The supposed efficiency should be proven with a test with a professional team.

References

- Beck00a Beck K.,
Extreme Programming Explained: Embrace Change.
Addison-Wesley, USA, 2000.
- Beck00b Beck K., Fowler M.,
Planning Extreme Programming.
Addison-Wesley, USA, 2000.
- Beyer98 Beyer H., Holtzblatt K.,
Contextual Design. Defining Customer-Centered Systems.
Morgan Kaufmann, San Francisco, CA, 1998.
- Belbin81 Belbin R. M.,
Management Teams - Why They Succeed or Fail.
Butterworth Heinemann, London, 1981.
- Boehm81 Boehm B.W.,
Software Engineering Economics.
Prentice Hall, 1981.
- Cooper 95 Cooper A.,
About Face: The Essentials of User Interface Design.
IDG Books Worldwide, USA, 1995.
- Cooper99 Cooper A.,
The Inmates are running the Asylum: Why High Tech Products Dive Us Crazy and How to Restore the Sanity.
Sams, USA, 1999.
- Faulkner00 Faulkner X.,
Usability Engineering.
McMillan Press, New York, 2002.
- Fowler99 Fowler M.,
Refactoring.
Addison-Wesley, USA, 1999.
<http://www.refactoring.com/>
- Hackos98 Hackos J. T., Redish J. C.,
User and Task Analysis for Interface Design.
John Wiley & Sons, New York, 1998.
- Hyvönen02 Hyvönen E., Viljanen K., Hättinen A.J.,
Yellow Pages on the Semantic Web.
XML Finland 2002, Helsinki, 2002.
<http://www.cs.helsinki.fi/u/eahyvone/publications/yellowpages.pdf>
- Hättinen02 Hättinen A.J.,
Käyttäjien tavoitteiden selvittäminen kenttätutkimusmenetelmillä.
Bachelor Thesis, University of Helsinki, Department of Computer Science, Finland, 2002.
<http://www.pharazon.org/publications/Kayttajien%20tavoitteiden%20selvittaminen%20kenttatutkimusmenetelmilla.pdf>

- Interacta02 Interacta Design Oy
<http://www.interacta.fi/method.html>
- ISO 9241-11 ISO FDIS 9241 part 11, 1997.
- Jeffries00 Jeffries R., Anderson A., Hendrickson C.,
Extreme Programming Installed.
Addison-Wesley, USA, 2000.
- Keltsi02 Hättinen A.J., Aalto A., Lidgren P., Kajava T., Jalava M.,
Keltsi ohjelmistotuotantoprojekti loppuraportti.
University of Helsinki, Department of Computer Science, Finland, 2002.
<http://www.cs.helsinki.fi/group/keltsi/docs/loppuraportti-1.0.doc>
- Kärkkäinen02 Kärkkäinen V.,
Extreme Programming prosessimalli ja sen kokeilu ohjelmistotuotantoprojektissa.
Master Thesis series C2002-27, University of Helsinki, Department of Computer
Science, Helsinki, 2002.
- Laakso00 Laakso S.A.,
User Interfaces II – UI Design in Software Projects.
Course Material, University of Helsinki, Department of Computer Science, Helsinki,
2000.
- Maslow54 Maslow A. H.,
Motivation and Personality.
Harper, New York, 1954.
- Nelson02 Nelson E.,
Extreme Programming vs. Interaction Design.
FTP Online, 2002-01-15.
http://www.fawcette.com/interviews/beck_cooper/
- Nosek98 Nosek J.T.,
The Case for Collaborative Programming.
Communications of the ACM, March 1998, 105-108.
- Pelrine00 Pelrine J., Uhtes R., Noack C.,
*Why is it hard to learn eXtreme Programming? Philosophical and psychological
aspects of learning a new methodology.*
NOD 2000, 2000.
http://www.daedalos.com/DE/DOWNLOAD/eXtreme-Programming/whyIsItHardToLearnXP_NOD2000.pdf
- Wells02 Wells D.,
Extreme Programming: A gentle introduction.
<http://www.extremeprogramming.com>