From Extreme Programming and Usability Engineering to Extreme Usability in Software Engineering Education (XP+UE→XU)

[1]Andreas Holzinger, [1]Maximilian Errath, [1]Gig Searle, Bettina Thurnher[2], Wolfgang Slany[3]
[1]*Institute for Medical Informatics, Statistics & Documentation, Medical University Graz*
[2] *Institute of Software Technology and Interactive Systems, Vienna University of Technology*
[3] *Institute for Software Technology, Graz University of Technology*
andreas.holzinger@meduni-graz.at maximilian.errath@meduni-graz.at
gig.searle@meduni-graz.at thurnher@qse.ifs.tuwien.ac.at wsi@ist.tugraz.at

Abstract

The success of Extreme Programming (XP) is based, among other things, on an optimal communication in teams of 6-12 persons, simplicity, frequent releases and a reaction to changing demands. Most of all, the customer is integrated into the development process, with constant feedback. This is very similar to Usability Engineering (UE) which follows a spiral four phase procedure model (analysis, draft, development, test) and a three step (paper mock-up, prototype, final product) production model. In comparison, these phases are extremely shortened in XP; also the ideal team size in UE User-Centered Development is 4-6 people, including the end-user. The two development approaches have different goals but, at the same time, employ similar methods to achieve them. It seems obvious that there must be synergy in combining them. The authors present ideas in how to combine them in an even more powerful development method called Extreme Usability (XU). The most important issue of this paper is that the authors have embedded their ideas into Software Engineering education.

1. Extreme Programming (XP)

XP is a software development process, which aims to solve frequent software development problems. The "extreme" in the name refers to the fact that well-known best practices in software development are brought to an extreme: if testing is good, we test *all* the time, including integration testing and functional testing; if code reviews are good, we review the code *all* the time; if short iterations are good, we make them really *very* short; if design is good, we make it everyone's daily business; etc. [1], [2]. The extremely shortened planning cycles (see figure 1) are characteristic, as is concentration on what is feasible and on what is most important *for the customer.*

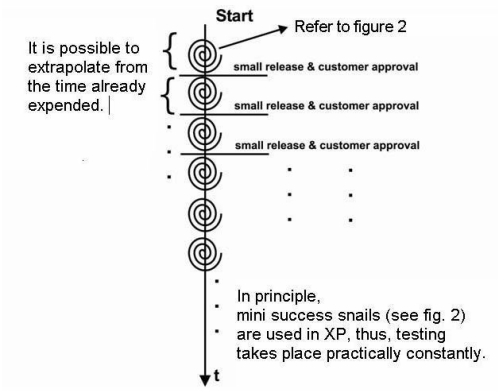

Figure 1: In principle XP uses mini success snails (compare with figure 2)

So-called **User Stories** (these are known as **scenarios** in Usability Engineering [3]), are used to drive the development process. Usability aspects are treated the same way as other features**.** In discussion with the developers, the customer puts the User Stories into the order of *absolute* importance for the economic success of the project, whereby the customer makes the final decision. Through discussion between customers and developers, it can be guaranteed that the creativity of the whole team can be of benefit to the project. Each User Story is assigned several engineering tasks. Engineering tasks which do not need the minimum realization of a User Story are not implemented: "Simplicity – the art of maximizing the amount of work *not* done – is essential". A User Story describes a feature which, from the point of view of the customer, cannot be broken down any further and which, in the case of implementation, will be of maximum use to the end user and includes usability aspects that are always connected to the rest of the functionality. For each story, a test case is developed, which determines the

Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05)
0730-3157/05 $20.00 © 2005 **IEEE**

functionality of the story. Hereby, the usability aspects are also tested, for example, empirically and as objectively as possible. The development of a story is completed when all test cases so far implemented for the achievement of all stories have been completed. This means that engineering tasks belonging to user stories which have not yet been implemented and are not absolutely necessary for those user stories which have been implemented, will be discarded, in full realization that this could later lead to large alterations.

2. Usability Engineering (UE)

In order to achieve good Usability [4], [5] the Usability Engineering Procedural Model (see figure 2) can be used together with Usability Engineering Methods (UEM) [6], [7]: The result of requirement analysis and the basic concept is produced in the form of sketches on paper (Paper Mock-ups).

With these paper models, usability tests can be **immediately** carried out with the end users. Mostly a specific task is presented and examined; *how long does it take the end user to complete the task*? (Time to perform a Task) and *which difficulties arise during the interaction*? (cognitive overload).
Investigations showed that the end users make more statements when working with such paper/pencil models, than when they are (immediately) confronted with a model on the computer or with a running system [8]. The results from these early usability tests, with which approximately 80% of all difficulties are discovered [9], [10], flow immediately into the development [11], [12].

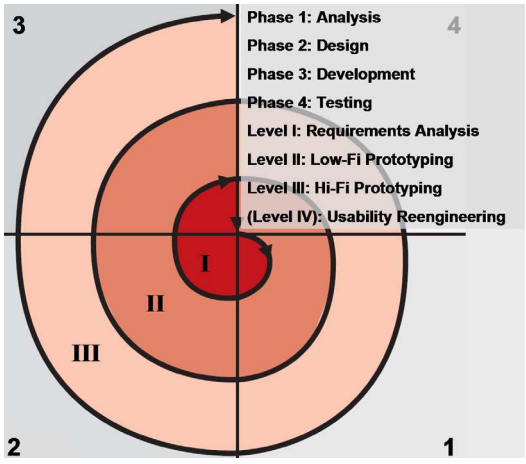

Figure 2: Typical UE spiral (success snail)

From the beginning, (analysis, design, rough conception, prototyping, usability testing), close communication is achieved between the software developer(s) and the end user(s).
The results, which are only present in the traditional, sequential software development model as documentation, are actually implemented and tested. This early testing leads to a high probability of early error recognition, which lowers the cost of subsequent error correction enormously. The cyclic, iterative procedure brings the software developers rapidly examinable results, more motivation and finally a substantial improvement in the quality of the software process by way of immediate end user feedback. The productivity of the team is thereby increased. Above all, the risk of a project failure is reduced, since experimentally confirmed results also offer a measure of security. However, the danger with this method, lays in the inadvertent dissipation of the end users' energies in extended analysis paralysis *(in German: Detailierungssucht)* – until they are unable to see the forest for the trees. Here, rapid prototyping [8] calls for the courage to build imperfect prototypes. On the other hand, there is a danger that certain details, which can only be implemented with great difficulty, are omitted because they appear trivial and subsequently increase development costs. Another disadvantage with this method is the difficulty in creating requirement specifications, since these would have to be adapted after each cycle, which can make fashioning a contract extremely difficult.

3. Extreme Usability (XU)

In XP, this danger of dissipating one's energies in details (developers are particularly susceptible) and the client's becomes *caught up in the detail* is consciously controlled by applying short iterations, frequent re-planning and focusing on simple design. This enables the client to get a realistic feeling of what can be achieved by the team, **if** the team implements only what he requested, and what needs to be pushed back to later versions in order to achieve the core functionality needed for the economic success of the project.
In particular, the well-known danger of *featuritis* is harnessed by the conscious decision to avoid thinking about what *could* happen later and *could* become meaningful, while being prepared to make extensive adjustments and changes at a later date. Extreme Usability (XU) could become such that all the *best practices* of UE are kept in the XP process during the planning games, with a restriction of the usability

aspects in the next iteration and the equal treatment of Usability and Functionality.
The advantage would be that, with the XP process, the adjustment and gradual improvement until the end of the project is explicitly built into the process, which is very helpful for UE. However, UE can improve the XP development method by focusing on the important aspects of the usability and employing the entire development team to make the customer continually aware of these aspects (by daily inquiry, discussion and testing); also the developers minds will be focused on the most important usability aspects, when at least one developer in the team possesses previous knowledge about UE and by implementing pair-programming, including the complete and frequent mixing of the pairs as well as passing on the *On-Site Customer* XP principle. Obviously, UE experience for all developers is an advantage in every project.
A practice of XP, which is often difficult to achieve in a realistic setting, is the customer-on-site because of the heavy time-restraints this poses on the customer. In XU, this difficulty could be transformed into an advantage by allowing *different* customers to take part in different iterations, if not releases, thus solving two problems at once:

- From the point of view of standard XP, the requirement that one and the same customer (the end-user in XU) has to be present at all times can be relaxed, thus possibly achieving a better over-all coverage of customer time in the team.
- From the point of view of standard UE, it is very attractive that the usability of the real system can at *all* times be tested on several different real end users, one at a time, but at any stage in great depth, with the possibility of redesigning the user interaction at any stage of the system, for a cost that can be accurately specified.

Practical Experiences with XU will be collected by the authors in several joint projects during spring 2005, the results and experiences will be presented at the time of the conference. In particular, a feasibility experiment with 225 students will be held, as follows:

All students will have been exposed, theoretically, to extreme programming as well as practically to Usability Engineering before taking part in the course. In the course, students will be divided into teams of approximately 10 people, each including one customer (end user), one manager, a coach/developer, and several normal developers. During the course, each team will have to solve a predefined task, that has a strong User Interface aspect but is slightly different for each team, in two releases; the first corresponding to two iterations, the second to only one (due to time limitations). They will work together in approximately 8 blocks of 8 hours each.
Customers will be represented by students that are particularly creative in defining new features and will never take part in developing code. Exactly at this point, further research is necessary: Usually the end users (customers) have neither a background in computer science nor are they experienced with these techniques. Consequently we will have to bring in real-life customers into the software engineering education. This would be a breakthrough, due to the fact that the students would get immense insight into problems involved with real-life customers.

Teams will observe all 12 XP practices, adapted to the short time available for the overall project, e.g., the 40h week will be relaxed to the Wednesday-8h-"week".
After each iteration a few developers will be switched randomly between teams in order to study the robustness of XP against personnel changes. Additionally, all customers will be switched randomly between teams so that the teams will be able to test their product on several customers, in particular w.r.t. Usability using a predefined set of criteria.

During planning games, usability aspects will be assessed by the thinking aloud method on paper mock-up simulations of features corresponding to user stories. Each team will have to deliver basic functionality for the first release of the subject that has been chosen by their first customer.
If this functionality cannot be demonstrated to the satisfaction of the second customer at the end of the first release, the corresponding team will be dissolved and the former team-members assigned to new or other teams. Members of teams that were not dissolved will get bonus points used for calculating their grades for the course. For the second release, again a new customer will be assigned at random to the teams, and the team will have to finish a product that covers both the already implemented feature as well as the one of the new customer.

At the final "trade show" (mini conference), the team-products will be assessed both from the point of view of the satisfaction of their two functionalities, quality in terms of unit-test coverage, readability of code, and experimental stress-testing, as well as their general usability. All students will be able to distribute points to different teams w.r.t. to these three measures (the number of members of a team will be subtracted from

that team's assessment numbers), and additional points will be distributed by impartial students assistants. A full 8 hours will be allocated for this final trade show and all students are requested to assess as many of the teams as possible.

4. XU in Software Engineering Education

In order to create awareness of the importance of integrating Usability issues into Software Engineering (SE) education much research is necessary. Most of all we must breach the gap between theory and practice by application of real-life projects together with industrial partners within the education of our students. However, we investigated the computer science curricula of other Universities and Universities of Applied Sciences to judge the degree of UE integration in the SE education track.

What we could learn was that UE is integrated **very little and far too late** in SE education. Usually, students of Computer Science have a two hour lecture for one semester in their master program. That means that, during a period of two years, the students learn "pure" development of systems using different development processes from the V-Model to RUP [13]. After these two years of programming it is very hard to sensitize the students for any sustainable software features such as quality aspects and especially usability. Software quality assurance includes much more than software testing [14].

To improve this situation we suggest a two step iteration of the computer science curricula in Austria:

1. Explain potential risks of a lack of usability. Risks can be seen from a technical perspective: e.g. necessary redesign if the customer refuses to accept the software. Risks can also be seen from a business perspective; including loss of sale, loss of brand reputation and market share [15];
2. Integrating UE in the SE education from scratch. One possible methodology to reach this is the integration of UE in the XP development process as suggested by the authors;

6. Conclusion

The combination of Extreme Programming (XP) and Usability Engineering (UE) which leads to a new method: Extreme Usability (XU), is very promising, especially for Software Engineering education. The authors are planning future in-depth research in real-life scenarios in order to collect more experience and are working towards developing a comprehensive guide for combining theory with practice: together today for a better software engineering of tomorrow!

7. References

[1] K. Beck, "Embracing Change with Extreme Programming", *IEEE COMPUTER*, vol. 32, 10, 1999, pp. 70-77.

[2] K. Beck, *Extreme Programming Explained: Embracing Change*, Addison Wesley, Boston (MA), 1999.

[3] J. M. Carroll and M. B. Rosson, "Getting around the task-artifact cycle: how to make claims and design by scenario", *ACM Transactions on Information Systems (TOIS)*, vol. 10, 2, 1992, pp. 181-212.

[4] B. Shackel, "The Concept of Usability" in *Visual Display Terminals: Usability Issues and Health Concerns*, J. Bennett, D. Case, S. J., and M. Smith, Eds., Prentice Hall, Englewood Cliffs (NJ), 1984.

[5] J. Nielsen, *Usability Engineering*, Morgan Kaufmann, San Francisco, 1993.

[6] A. Holzinger, "Usability Engineering for Software Developers", *Communications of the ACM*, vol. 48, 1, 2005, pp. 71-74.

[7] A. Holzinger, *Multimedia Basics, Volume 3: Design. Developmental Fundamentals of multimedial Information Systems*, Laxmi Publications, New Delhi, 2002. (www.basiswissen-multimedia.at)

[8] A. Holzinger, "Application of Rapid Prototyping to the User Interface Development for a Virtual Medical Campus", *IEEE Software*, vol. 21, 1, 2004, pp. 92-99.

[9] R. A. Virzi, "Refining the test phase of usability evaluation: how many subjects is enough? " *Human Factors*, vol. 34, 4, 1992, pp. 457-468.

[10] C. D. Allen, D. Ballman, V. Begg, H. H. Miller-Jacobs, J. Nielsen, J. Spool, and M. Muller, "User involvement in the design process: why, when & how?" presented at Conference on Human Factors and Computing Systems, Amsterdam (NL), 1993. pp. 251-254.

[11] A. Holzinger, "User-Centered Interface Design for disabled and elderly people: First experiences with designing a patient communication system (PACOSY)" in *Lecture Notes in Computer Science. Vol 2398*, K. Miesenberger, J. Klaus, and W. Zagler, Eds., Springer, Berlin et al., 2002. pp. 34-41.

[12] A. Holzinger, "Experiences with User Centered Development (UCD) for the Front End of the Virtual Medical Campus Graz" in *Human-Computer Interaction, Theory and Practice*, J. A. Jacko and C. Stephanidis, Eds., Lawrence Erlbaum, Mahwah (NJ), 2003. pp. 123-127.

[13] A. Holzinger, *Basiswissen IT/Informatik. Band 2: Informatik*, Vogel, Würzburg, 2003. (www.basiswissen-it.at)

[14] S. Feldman, "Quality Assurance: Much more than Testing", *ACM Queue*, vol. 3, 1, 2005, pp. 27-29.

[15] C. B. Fellenz, "Introducing usability into smaller organizations", *ACM interactions*, vol. 4, 5, 1997, pp. 29-33.