

UPGRADE is the European Online Magazine for the Information Technology Professional, published bimonthly at  
http://www.upgrade-cepis.org/

**Publisher**

UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, http://www.cepis.org/) by Novática (http://www.ati.es/novatica/) and Informatik/Informatique (http://www.svifsi.ch/revue/)

**Chief Editors**

François Louis Nicolet, Zurich <nicolet@acm.org>  
Rafael Fernández Calvo, Madrid <rfcalvo@ati.es>

**Editorial Board**

Prof. Wolfried Stucky, CEPIS President  
Gloria Nistal Rosique and  
Rafael Fernández Calvo, ATI  
Prof. Carl August Zehnder and François Louis Nicolet, SVI/FSI

**English Editors:** Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson

**Cover page** designed by Antonio Crespo Foix, © ATI 2002

**Layout:** Pascale Schürmann

E-mail addresses for editorial correspondence:  
<nicolet@acm.org> and <rfcalvo@ati.es>

E-mail address for advertising correspondence:  
<novatica@ati.es>

**Copyright**

© Novática and Informatik/Informatique. All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, write to the editors.

The opinions expressed by the authors are their exclusive responsibility.

## eXtreme Programming

Guest Editor: Luis Fernández Sanz

### Joint issue with NOVÁTICA and INFORMATIK/INFORMATIQUE

#### 2 Presentation: eXtreme Programming

– *Luis Fernández Sanz, Guest Editor*

#### 4 A new method of Software Development: eXtreme Programming

– *César F. Acebal and Juan M. Cueva Lovelle*

*What is eXtreme Programming – also known as XP? The aim of this article is to answer that question, and to reveal the nature of this new method of software development to the uninitiated reader. We will try to be sufficiently informative so that you will all come away with some idea of the basic underlying principles, and for anyone who might want to delve deeper into the subject, we will provide suitable references.*

#### 9 Programming Extremism – *Michael McCormick*

*The author reviews antecedents and experiences of the "agile" methodology of software development called eXtreme Programming, comparing it to other methodologies and pointing to its advantages and disadvantages from a pragmatic standpoint, depending on the kind of project it applies to. He draws the conclusion that it is necessary to stay away from "religious" positions about existing methodologies.*

#### 11 The Need for Speed: Automating Acceptance Testing in an eXtreme Programming Environment

– *Lisa Crispin, Tip House and Carol Wade (Contributor)*

*How to focus acceptance testing for XP, how to design automated tests that are low-maintenance and self-verifying, how to apply the values of XP to test automation, and ways to gather metrics and provide useful reports.*

#### 18 Qualitative Studies of XP in a Medium Sized Business

– *Robert Gittins, Sian Hope and Ifor Williams*

*Qualitative Research Methods are used to discover the effects of applying eXtreme Programming in a software development business environment. Problems dominating staff development, productivity and efficiency are parts of a complex human dimension uncovered in this approach. The interpretation and development of XP's "Rules and Practices" are reported, as well as the interlaced communication and human issues affecting the implementation of XP in a medium sized business.*

#### 23 XP and Software Engineering: an opinion – *Luis Fernández Sanz*

*In this article, the author makes some reflections on certain specific aspects of eXtreme Programming as described in Kent Beck's book "eXtreme Programming explained. Embrace change". The analysis presented here is in relation to principles and techniques of software engineering.*

#### 27 XP in Complex Project Settings: Some Extensions

– *Martin Lippert, Stefan Roock, Henning Wolf and Heinz Züllighoven*

*XP has one weakness when it comes to complex application domains or difficult situations at the customer's organization: the customer role does not reflect the different interests, skills and forces with which we are confronted in development projects. We propose splitting the customer role into a user and a client role. The user role is concerned with domain knowledge; the client role defines the strategic or business goals of a development project and controls its financial resources. It is the developers' task to integrate users and clients into a project that builds a system according to the users' requirements, while at the same time attain the goals set by the client.*

Coming issue:  
"Information Retrieval"

## XP in Complex Project Settings: Some Extensions

*Martin Lippert, Stefan Rook, Henning Wolf and Heinz Züllighoven*

*XP has one weakness when it comes to complex application domains or difficult situations at the customer's organization: the customer role does not reflect the different interests, skills and forces with which we are confronted in development projects. We propose splitting the customer role into a user and a client role. The user role is concerned with domain knowledge; the client role defines the strategic or business goals of a development project and controls its financial resources. It is the developers' task to integrate users and clients into a project that builds a system according to the users' requirements, while at the same time attain the goals set by the client. We present document types from the Tools & Materials approach [Lilienthal/Züllighoven 97] which help developers to integrate users and clients into a software project. All document types have been used successfully in a number of industrial projects together with the well-known XP practices.*

**Keywords:** XP, Management, Participation, User, Client, Roles

### 1 Context and Motivation

It was reported that one of the major problems of the C3 project was the mismatch between the *goal donor* and the *gold owner* [Jeffries 00], [Fowler 00]. While the goal donor – the customer in the XP team – was satisfied with the project's results, the gold owner – the management of the customer's organization – was not. It is our thesis that XP, in its current form, fails to address the actual situation at the client's organization in a suitable way. The main stakeholder, i.e. the users

and their management, are merged into a single role: the customer. This one role cannot address the different forces in a development project. The users of the future system know their application domain in terms of tasks and concepts, but they rarely have an idea of what can be implemented using current technologies. Moreover, it is often misleading to view the users of the future system as the goal donor. They are unfamiliar with the strategic and business goals related to a project and, more important, they do not control the money.

Therefore we make a distinction between the role of the user and the role of the client. The users have all the domain knowledge and therefore are the primary source for the application

**Martin Lippert** is a research assistant at the University of Hamburg and a professional software architect and consultant at APCON Workplace Solutions. He has several years' experience with XP techniques and XP project coaching for various domains and has given a number of talks, tutorials and demonstrations (e.g. ICSE, XP, OOPSLA, ECOOP, HICSS, ICSTest and OOP). He is a member of the XP 2002 program committee. Among his publications are articles for "Extreme Programming Examined" and "Extreme Programming Perspectives" and he co-authored the book "Extreme Programming in Action", which is due to be published by Wiley in July 2002. <lippert@jwam.org>

**Stefan Rook** is software architect and consultant at APCON Workplace Solutions. He has solid project experiences with object-oriented technologies, architectures and frameworks as well as with XP. Among his current interests are evolution of frameworks and migration of applications, eXtreme Programming, large refactorings and suitable organizational structures for cooperating XP teams. Stefan Rook has given a number of talks, tutorials and demonstrations (e.g. XP Conference, ECOOP, OOP and ICSTest) and co-organizes the XP 2002 workshop on testing techniques. Among his publications are articles for "Extreme Programming Examined" and "Extreme Programming Perspectives" and he is co-author of the book "Extreme Programming in Action", which is due to be

published by Wiley in July 2002. He can be contacted at rook@jwam.org.

**Henning Wolf** is software architect at APCON Workplace Solutions in Hamburg. He is one of the original architects of the Java framework JWAM, supporting the tools & materials approach. His current interests are eXtreme Programming, architectures for multi-channelling applications and object-oriented technologies. Besides publications on various software engineering topics he is co-author of the book "Extreme Programming in Action", which is due to be published by Wiley in July 2002. He can be contacted at henning.wolf@itelligence.de.

**Heinz Züllighoven**, graduated in Mathematics and German Language and Literature, holds a PhD in Computer Science. He is professor at the Computer Science Department of the University of Hamburg and CEO of APCON Workplace Solutions Ltd. He is consulting industrial software development projects in the area of object-oriented design, among which are several major banks. Heinz Züllighoven is one of the leading authors of the object-oriented Tools & Materials Approach. A Tools & Materials construction handbook will be published by Morgan Kaufmann end of 2002. Among his current research interests are object-oriented development strategies and the architecture of large industrial interactive software systems. <zuellighoven@jwam.org>

requirements. The client sets the goals of the development project from a business point of view. The client will only pay for a development project if these goals are met to a certain degree.

We begin with a discussion of the roles in an XP project as defined by Kent Beck. We then split up the customer role into the user and the client role. These two roles change the situation of XP projects. While the user can be seen in a similar way to the XP customer, the client role requires more attention. We address the new project situation by using two document types geared to the client role: base lines and projects stages. We show when and how to use these document types and discuss their relation to story cards and the Unified Process (UP).

## 2 Roles in XP

XP defines the following roles for a software development process [Beck 99]:

- *Programmer*: The programmer writes source code for the software system under development. This role is at the technical heart of every XP project because it is responsible for the main outcome of the project: the application system.
- *Customer*: The customer writes user stories which tell the programmer what to program. “The programmer knows how to program. The customer knows what to program” ([Beck 99], pp. 142f).
- *Tester*: The tester is responsible for helping customers select and write functional tests. On the other side, the tester runs all the tests again and again in order to create an updated picture of the project state.
- *Tracker*: The tracker keeps track of all the numbers in a project. This role is familiar with the estimation reliability of the team. Whoever plays this role knows the facts and records of the project and should be able to tell the team if they will finish the next iteration as planned or not.
- *Coach*: The coach is responsible for the development process as a whole. The coach notices when the team is getting “off track” and puts it “back on track”. To do this, the coach must have a very profound knowledge and experience of XP.
- *Consultant*: Whenever the XP team needs additional special knowledge they “hire” a consultant in possession of this knowledge. The consultant transfers this knowledge to the team members, enabling the team to solve the problem on their own.
- *Big Boss*: The big boss is the manager of the XP project and provides the resources for it. The big boss needs to have the general picture of the project, be familiar with the current project state and know if any interventions are needed to ensure the project’s success.

While XP addresses management of the software development aspects with the Big Boss role, it neglects the equivalent of this role on the customer side. XP merges all customer roles into the customer role. We suggest splitting up the customer role into two roles: *user* and *client*.

## 3 The New User and Client Roles

The *user* is the domain expert which the XP team has to support with the software system under development. The user

is therefore the first source of information when it comes to functional requirements.

The *client* role is not concerned with detailed domain knowledge or functional requirements. The client focuses on business needs, like reducing the organizational overhead of a department by 100,000 USD a year. Given this strategic background, the client defines the goals of the software development project (“Reduce the organizational overhead of the loan department by 100,000 USD per year”) and supplies the money for the project. The client is thus the so-called *goal donor* and the *gold owner*.

It is often not easy to reconcile the needs of users and clients at the same time. What the users want may not be compatible with the goals of the client. What we need, then, are dedicated instruments to deal with both roles.

## 4 Story Cards and the Planning Game

We use story cards for the planning game, but we use them in a different way than in the “original” XP, and our planning game differs in some aspects, too. In our projects, users or clients rarely write story cards themselves. They do not normally have the skills or the required “process knowledge” to do so. Typically, we as developers write story cards based on interviews with users and observations of their actual work situation. These story cards are reviewed by the *users* and the *client*. The users must assess whether the implementation of the story cards will support them. They thus review the developers’ understanding of the application domain. The client decides which story cards to implement in the next development iteration, and with which priority. To avoid severe mismatches between the interests of the users and client both parties are involved in the planning game. This means that users can articulate their interests and discuss with the client the priorities of the story cards.

Our experience here is clear: users and client will normally reach a compromise on their mutual interests. But whatever the outcome of the planning game is, the decision about what is to be implemented next is made not by developers but by the client.

If a project is complex, there will be an abundance of story cards. In this case it is difficult for users, clients and developers to get the overall picture from the story cards. For this type of project, we use two additional document types: *project stages* and *base lines*. These are described in the next section.

Subgoal	Realization	When
Prototype with Web frontend is running	Presentation of prototype for users	31/3/00
Prototype supports both Web and GUI frontend.	Presentation of extended prototype for users and client	16/5/00
First running system installed	Pilot Web users use Web frontend.	30/8/00
...	...	...

Figure 1: Example project stages

Who	does what with whom/ what	What for	How to check
<u>Roock</u>	Preparation of interview guideline	Interviews	E-mail interview guideline to team
<u>Wolf, Lippert,</u>	Interview users at pilot customer	First understanding of application domain	Interview protocols on the project server
...	...	...	...
<u>Roock</u>	Implement GUI prototype	Get feedback on the general handling from the users	Prototype acceptance tests are OK; executable prototype is on project server

Figure 2: Examples of base lines

### 5 Project Stages and Base Lines

In projects with complex domains or large application systems, story cards may not be sufficient as a discussion basis for the planning game. In such cases, we need additional techniques to get the overall picture – especially for the contingencies between the story cards. If one story cannot be developed in the estimated period of time, it may be necessary to reschedule dependent stories. We may also need to divide the bulk of story cards in handy portions and make our planning more transparent to the users and the client. We have therefore enhanced the planning game by selected document types of the Tools & Material approach [Roock et al. 98]: *base lines* and *project stages*.

We use project stages and base lines for project management and scheduling. A project stage defines which consistent and comprehensive components of the system should be available at what time covering which subgoal of the overall project. Project stages are an important document type for communicating with users and clients. We use them to make development progress more transparent by discussing the development plan and rescheduling it to meet users’ and client’s needs. Figure 2 shows an example of three project stages (taken from the JWAM framework development project). We specify at what time we wish to reach which goal and what we have to do to attain this goal. Typically, the project stages are scheduled backwards from the estimated project end to its beginning, most important external events and deadlines (vacations, train-

ing programs, exhibitions, project reviews and marketing presentations) being fixed when projects are established.

Unlike the increments produced during an XP iteration, the result of a project stage is not necessarily an installed system. We always try to develop a system that can be installed and used as the result of every project stage, but we know that this is not always feasible. In large projects or complex application domains, developers need time to understand the application domain. During this period, developers may implement prototypes but rarely operative systems. We thus often have prototypes as the result of early project stages. Another example here is the stepwise replacement of legacy systems. It is often appropriate to integrate the new solution with the legacy system for reasons of risk management. Project stages then produce systems that can and will be used by users. But the project team may also decide not to integrate the new solution with the legacy system, perhaps because of the considerable effort required for legacy integration. In such cases, the project team will also produce installable increments, but it is clear that the increments will not be used in practice. Users are often reluctant to use new systems until they offer at least the functionality of the old system.

*Base lines* are used to plan one project stage in detail. They do not focus on dates but rather define what has to be done, who will do it and who will control the outcome in what way. Unlike project stages, base lines are scheduled from the beginning to the end of the stage.

In the base-lines table (for example, in Figure 2), we specify, who is responsible for what base line and what it is good for. The last column contains a remark on how to check the result of the base line. The base-lines table helps us to identify dependencies between different steps of the framework development (see “What-for” column). The last three columns are the most important ones for us. The first column is not that important because everybody can, in principle, do everything (as with story cards). However, it is important for us to know how to check the results in order to get a good impression of the project’s progress. The second and third columns contain indicators for potential re-scheduling between the base lines and also helps us to sort the story cards that are on a finer-grained level.

The rows of the base-line table are often similar to story cards, but base lines also include tasks to be done without story cards. Examples are: organize a meeting, interview a user, etc.

The way project stages and base lines are actually used depends on the type of development project in hand. For small to medium-size projects, we often use project stages, but no explicit base lines. In these cases, we simply use the story cards of the current project stage, complementing them by additional task cards. If the project is more complex (more developers, developers at different sites, etc.), we use explicit base lines in addition to story cards. If the project is long-term we do not define base lines for all project stages up front, but rather identify base lines for the

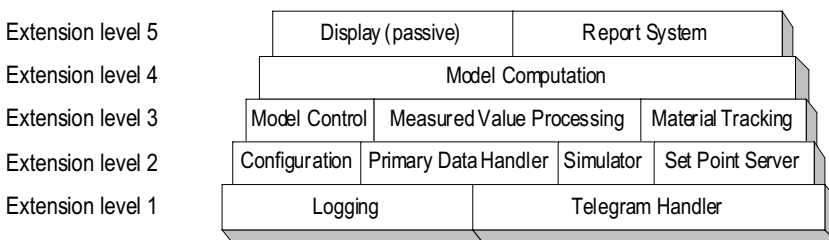
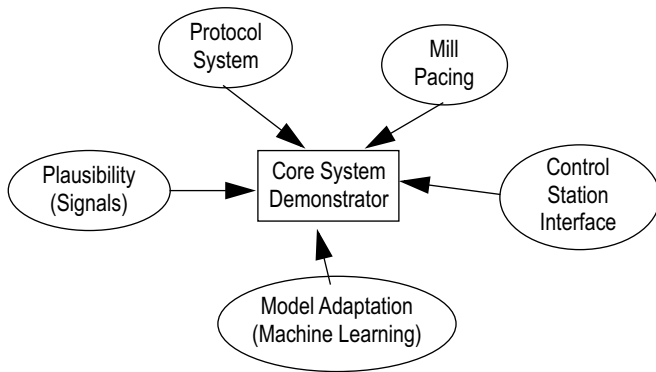


Figure 3: Example core system with extension levels



**Figure 4:** Example core system with specialized systems

current and the next project stage. Since a project stage should not be longer than three months, we work on a detailed planning horizon of from three to six months.

It is often a good idea to sketch the entire system as guideline for the project stages. We describe the concept of core system and specialized systems in the next section in order to provide an application-oriented view of the system architecture.

## 6 System Architecture

In line with the project stages, we divide the software system into a *core system* with *extension levels* [Krabbel et al. 96]. The core system is an operative part of the overall software system which addresses important domain-related needs. It is developed first and put into operation. Since the core system is usually still quite complex, it is subdivided into extension levels which are built successively. An example of a core system with extension levels is shown in Figure 3 (taken from the domain of hot rolling mills). The upper extension levels use the functionality of the lower extension levels. This way, we get an application-oriented structure that is useful for planning and scheduling. It is obvious that the lowest extension level must be created first, followed by the next-higher one, and so on.

Specialized systems are separated from the core system. They add well-defined functionality. An example of a core system with specialized systems is shown in Figure 4 (again taken from the domain of hot rolling mills). The specialized systems are drawn as circles.

Since specialized systems only depend on the core and not vice versa, we can deliver an operative and useful core system very early on and get feedback from the users. In parallel, different software teams can build specialized systems. Adhering to the one-way dependency of specialized systems, we achieve a maximum of independence among the special systems. They can be created in any order or even in parallel. Obviously, the core system has to provide the basic functionality for the whole system because it is the only way for the specialized systems to exchange information. The core system will usually provide a set of basic communication mechanisms allowing information transfer between different parts of the overall system.

The concept of core system and specialized systems can easily be used in the planning game. Users and client get an

impression of the whole system and can negotiate on the different values and priorities (users' needs, client's goals, technical constraints) in order to reach a compromise on the project's development schedule.

In addition, project stages are used to control the project's progress and timelines relating to the overall plan.

## 7 Conclusion

We have discussed the roles in a XP project as defined by Kent Beck. Based on our experience, we split the XP customer role into two roles: *user* and *client*. The user is the source of application knowledge, while the client defines the project goals and supplies the money for the project. Both parties must be integrated into the development project. We have shown how this can be done with the help of modified story cards, projects stages, base lines and an adapted planning game.

We do not suggest using all the presented new instruments for every project. They should be used as part of an inventory or toolbox, together with the familiar techniques defined by XP. We then use the instruments required for the project in hand. If the project situation is not complex, we will not burden the project with the additional roles and document types. But if the application domain or the project is highly complex, the sketched extensions to XP will be worth while.

Selection of the proper instruments from the toolbox may be difficult for the project team because we are not yet able to provide detailed guidelines. Evaluating project experience to provide such guidelines for tool selection will be one of our future tasks.

## References

- [Beck 99] Kent Beck: eXtreme Programming Explained – Embrace Change. Addison-Wesley. 1999.
- [Fowler 00] Martin Fowler: The XP2000 conference. <<http://www.martinfowler.com/articles/xp2000.html>>. 2000.
- [Jeffries 00] Ron Jeffries: Extreme Programming – An Open Approach to Enterprise Development. <<http://www.xprogramming.com/xpmag/>>. 2000.
- [JWAM] The JWAM framework. <<http://www.jwam.org/>>
- [Krabbel et al. 96] A. Krabbel, S. Ratuski, I. Wetzel: Requirements Analysis of Joint Tasks in Hospitals, Information systems Research seminar. In Scandinavia: IRIS 19; proceedings, Lökeberg, Sweden, 10–13 August, 1996. Bo Dahlbom et al. (eds.). – Gothenburg: Studies in Informatics, Report 8, 1996. S. 733–750, 1996
- [Lilienthal/Züllighoven 97] C. Lilienthal and H. Züllighoven: Application-Oriented Usage Quality, The Tools and Materials Approach, Interactions Magazine, CACM, October 1997
- [Roock et al. 98] Roock, S., Wolf, H., Züllighoven, H., Frameworking, In: Niels Jakob Buch, Jan Damsgaard, Lars Bo Eriksen, Jakob H. Iversen, Peter Axel Nielsen (Eds.): IRIS 21 “Information Systems Research in Collaboration with Industry”, Proceedings of the 21st Information Systems Research Seminar in Scandinavia, 8–11 August 1998 at Saeby Soebad, Denmark, pp. 743–758, 1998