

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 <rfoalvo@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 <rfoalvo@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 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.

Keywords: eXtreme Programming, XP, Software Engineering

1 First contact

The first time I came across the term EXtreme Programming (also known by the initials XP), my mind was immediately overrun with images of those well known extreme sports: people who love danger and who spend their time skiing down impossible mountainsides, making bungee jumps, etc.¹ Of course the expression was deliberately coined by Kent Beck to benefit from the fashion for this kind of sport in order to make a bigger splash on the IT scene. Perhaps the idea was also to suggest a world of "winners", people who flirted with danger, who were always "cool" (another in word) whatever they did and who turned their back on convention. Regrettably, however, I discovered it was something altogether simpler and less glamorous than the risk and adventure of extreme sports² but, fortunately, I did recognise the tremendous appeal of a new approach to software development.

Curiosity naturally led me to seek out references on XP where I could learn some more about it. Coincidentally, and at the same time, I began to hear opinions from experienced people; people who could never be accused of not being broad minded. While some of them were drawn to this idea (although somewhat sceptical about its potential for becoming a common practice), others pointed out in no uncertain terms the "madness" of expecting to achieve quality in software developed using XP practices. I have been in this field long enough to experience several passing fads which claimed to be the cure of all the ills besetting software development (of which there are plenty), and high hopes were placed in all of them (mainly by their mentors and supporters, sometimes with obvious commercial interests at heart) in terms of their scope and life span. There were many such trends and most passed away, although it's only fair to admit that some did make a contribution to current good practices. Which is why something told me that I could be looking at just another passing fad wrapped up

1. This "confession" may not seem so surprising when you read McCormick's article [McCormick 01] which you can find elsewhere in this edition.
2. The truth is that, in terms of physical appearance, there is seldom any possible comparison between IT people and the extreme sports community.

Luis Fernández Sanz received a degree in informatics engineering from Technical University of Madrid (Spain) in 1989 and a Ph. D. degree in informatics from University of the Basque Country in 1997 (as well as an extraordinary mention for his doctoral thesis). He is currently head of the department of programming and software engineering at Universidad Europea-CEES (Madrid). From 1992, he is the coordinator of the software engineering section of Novática. He is author or coauthor of several books about software engineering and software measurement, as well as different papers in international journals and conferences. He is member of the Software Quality Group of ATI and he has acted as chair of the VI Spanish Conference on Software Quality and Innovation organised by ATI. He is a member of ATI and the Computer Society of the IEEE.
<lufern@dpris.esi.uem.es>

in impressive sounding terms. Even when a colleague, the type who like to loudly proclaim any nonsense they have just learnt (preferably something trendy) and who need constant attention to compensate for their inferiority complex, piped up that he knew what XP was and waxed lyrical about how wonderful it all was³, subjectivity got the better of me.

But there is one thing I just cannot help doing: I can't help trying to find out for myself about the things which arouse my curiosity, and I will not take anyone else's opinion as gospel. In my quest for information about XP it was not hard for me to find various web sites (see the brief list at the end of this article) which contained a copious selection of documentation, links and resources on the subject of eXtreme Programming. Of course, I had Beck's classic book [Beck 00] in which he explains the essence of XP. I even discovered the existence of monographic international congresses on XP with several editions already held. However something was worrying me. I could see the same signs that I had seen before in some of the previous fads: a lot of words (albeit reasonable and attractive ones), a certain implicit assumption that XP is a dogma of faith (nobody doubted that its principles were properly justified) and, especially, a shortage of really reliable data and real life

3. It goes without saying that he planned to reap all the benefits of XP straight away as his idea was to implement it immediately (as I write, I have had no reports that he has done anything, just like on so many other occasions).

experiences. In fact, a simple search involving direct consultations with famous “extremes” in the international arena always resulted in the same response: at best some academic experiments somewhat divorced for any professional reality, some experience, but few projects (though there were some) with specific data and much less any formal experiments. However, my intention is not to put XP practitioners down in any way with this: it is sadly only too common a state of affairs in the software world in any field⁴. However Chrysler’s famous C3 project in which Beck applied XP practices for the first time is much talked about.

My main problem when it comes to expressing an opinion about XP is that I don’t have any first hand references: I have had no opportunity to apply XP in a project nor have I managed to find colleagues in Spain who are practising it in a professional environment. It’s true there are companies like IBM who have given support to specific aspects of XP (for example, with JUnit) and who have documentation on it. Other major companies like Sun give it some kind of exposure in their technical documents, almost certainly because their marketing people don’t want to miss out on the advertising draw it has (at least in terms of the name and the frequency with which it crops up in publications).

Anyway, after a careful reading of Beck’s book [Beck 00] and with the inspiration of various documents taken from web sites on XP and the occasional interesting article like McCormick’s [McCormick 01], I would like to share some of my reflections with the readers.

2 Reflections on XP from a Software Engineering viewpoint

One of the bases of XP is the technical promise which Beck makes at the beginning of his book [Beck 00]. The problem lies in the changes which have to be made in a software development. XP aims to minimise changes by concentrating only on those changes which simplify code (which should be as simple as possible⁵) or which will facilitate the rest of the code. Consequently, it is important to design without having to complicate the design in an attempt to cater for possible future expansions or increased functionality or performance. From my point of view, this concern about change is laudable, though I am not so sure that it is always a good idea to ignore future extensions for the sake of simplifying the design as much as possible. In this respect, the reasoning behind the approach to projects is based on three variables: the scope of the project, the quality and the cost. Beck’s rationale lays emphasis on the maximum reduction of the scope of the project (that is, concen-

trating only on functionality and the features which are strictly necessary) to gain advantages in the other two variables.

Once the ground rules have been established for action, a great deal of what the developers do should be aimed at simplifying the design (without making any extra effort with an eye to the future), performing automated tests and accumulating a lot of practice in the modification of designs so as to lose the fear of making changes. In fact, a phrase has been coined which uses an analogy to sum up XP’s philosophy: “Driving is not about getting the car going in the right direction. Driving is about constantly paying attention”. In this case, the driver is the customer with whom we should have a constant interaction. To achieve this the customers’ directors have to be convinced that, if they can’t free a person to attend to the developers constantly, maybe their bet on the system under construction is not worth making.

From my point of view, it is especially satisfying that software tests (always the most hated and neglected part of a development) are being given more importance and, it is also worth underlining that productivity in this area (and in others such as coding) depend on the introduction of automated environments. In my experience, generally speaking there tends to be practically no use of support tools in automated tests by organisations developing conventional software⁶. However, I am not sure if XP’s strategy concerning changes and design will always be appropriate. What I do like is the fact that emphasis is put on simplicity achieved through good design, since I believe that all too often developers opt for code which “more or less works” and take little time to consider what might be the simplest and most understandable code to implement a functionality.

XP also promotes certain interesting values of human team and project management (as well as some technical aspects): communication, simplicity, feedback with the customer and courage when tackling the problems and challenges thrown up by design. But, above all, what it proposes is a life cycle or process which is “lightweight” (as McCormick says [McCormick 01]) or agile. This fast process, which is a successor to the RAD concept and evolutionary prototyping, involves very fast iterations (releasing versions, builds or whatever we want to call them) run at very frequent intervals: one a day or even every few hours. This requires the tests to be very agile and highly organised, which is only possible if they can benefit from efficient and well organised automation.

Each XP iteration involves:

- A coding phase: code is the essential building block and the primordial aim of XP as a concise and precise communication vehicle, regardless of whether we work with visual environments or text editors or code generators.
- An indispensable automated test phase. Personally I find it very satisfying to see how in XP tests are not a torture but rather something which is more fun than programming.

6. In several surveys when giving training courses on software tests, a significant percentage of the attendees admitted a very poor level of automation and support environments, and also of test organisation and procedure.

4. At the end of the day, I am influenced by my work on software measurement which led up to my doctoral thesis (as well as an extraordinary mention for my doctoral thesis), and to the publishing of the first book in Spain on this subject, with the collaboration of the outstanding researcher and professor Javier Dolado (my thesis director), together with other eminent researchers from Spain and Great Britain, [Dolado/Fernández 00].

5. Although as Einstein once said, “it should be as simple as possible but no simpler”.

They also help to lengthen software's life span since they are an aid to efficient change management. The danger lies in lowering the stringency level of the tests without establishing a clear tolerable error rate. One direct consequence of this is the incorporation of test measurements: of code coverage and the like. This is a cause of some satisfaction for me given the current dearth of software development organisations which implement software measurement.

- A phase of active listening with the customer. As I said earlier, this requires the almost full time dedication of one of the customer's employees to manage and check customer requirements.
- A design phase to simplify and correct anything which isn't appropriate.

While this simple review does not aim to match the detailed descriptions to be found in other articles in this issue, perhaps it may serve to comment on certain features of XP in relation to software engineering philosophy. However, there are a number of general considerations concerning XP's applicability which I would like to include.

3 Some of my general reflections about XP

In Beck's book he advocates multi-disciplinarity in IT personal. He proposes that we should forget distinctions between analysts, programmers and test personnel since XP's application requires development personnel to play various roles and be equipped to carry them out efficiently: everyone should take part in analysis, design, programming and tests. From my point of view, this idea is very attractive to say the least. The rigid division of work often leads to a loss of effectiveness and efficiency in software development, regardless of any efforts made to improve teamwork.

However, there is currently a very serious lack of professionals in information technologies, and for software development in particular, in spite of the effect of the economic downturn on employment. In fact, there are studies by organisations such as Forrester Research [Forrester 01] which are already predicting a recovery of the IT business for 2003, based on forecasts of expenses made by corporate managers.

It would, however, be easier to get trained personnel to turn their hand to the agile processes of XP if a serious attempt were made to introduce the qualification or profession of software engineer [Dolado 00], instead of just having a qualification in computer science or, of course, just revamping other qualifications centred exclusively on the knowledge of programming languages without a solid grounding in analysis, design and software testing. By this I do not mean that those who don't have a solid training in software engineering cannot contribute their intelligence towards the correct application of the XP philosophy. Of course, it is possible to train and coach people in the use of XP but I find it hard to believe that this training could be successful with people who lack a solid grounding in software development.

Another of the fears I harbour regarding XP is that it will provide a great excuse for those who like to work in the development process in a state of pure chaos. XP, as its own exponents say, does not consist of relinquishing the notion of a

controlled process but rather it is merely the adoption of a strategy of simplification and agility in development work. In spite of the fact that, on occasions, work on models to improve processes have fallen into the error of establishing heavyweight and somewhat rigid processes, the intellectual contributions made by CMM [Paulk et al. 93], SPICE [ISO 98], etc. have been fundamental in bringing about a clear awareness that it is necessary to organise the way we work, and that chaos can only end in tears. It is vital that we do not lose what we have achieved: the awareness (although the we may sometimes fail to put it into practice) that a development process which is well designed and well suited to the problem is fundamental if we want to prevent our projects from failing.

Finally, although XP lays stress on effective communication, I can see a problem in the fact that it always insists on face to face conversation. While a constant and face to face exchange of opinions on a project is a rare commodity in most conventional projects, it is nonetheless true that documentation is also an important asset in the control of the project. In XP I understand that its orientation towards small projects with volatile requirements encourages agility in personal verbal communication. However, what happens in the case of personnel turnover? Or with problems of absences due to sickness or for other reasons? It is true that XP's idea of collective ownership of code (everyone can know and change any part of an application) means greater ease in handling personnel turnover but I believe that we should never pass up the chance to have good documentation so as to be able to understand and maintain an application. Later we can try to remember the content and format of the documentation we think will be suitable for a project or an application (or we can even use automated documentation tools). But, at the end of the day, documentation is necessary, as well as code (the real instrument of communication for exponents of XP).

In this article, I hope I have been able to convey, to the limits of my knowledge of it (and having never attempted to apply it in real projects), the idea that I have of XP. Naturally I am open to any comments on my opinions.

References

- [Beck 00]
K. Beck, *EXtreme Programming explained*. Embrace change, Addison-Wesley, 2000.
- [Dolado 00]
J. J. Dolado, "El cuerpo de conocimiento de la Ingeniería del software" (Body of software engineering knowledge), *Novática*, no. 148, November-December, 2000, pp56-59.
- [Dolado/Fernández 00]
J. J. Dolado and L. Fernández (eds.), "Medición para la gestión en la ingeniería del software" (Measurement for software engineering management), Ra-Ma, 2000.
- [Forrester 01]
Forrester Research, "End Predicted for IT Sector Slump", 2001.
- [ISO 98]
ISO, *ISO/IEC TR 15504-1998*. Information technology. Software process assessment. Parts 1-9, International Organization for Standardization, 1998.
- [McCormick 01]
M. McCormick, "Programming extremism", *Communications of the ACM*, Vol. 44, no. 6 June, 2001, pp199-201.

[Paulk et al. 93]

M. Paulk et al., Capability maturity model for software. Version 1.1. Technical Report CMU/SEI-93-TR024, Software Engineering Institute, February, 1993.

Some web resources on XP

<http://www.xprogramming.com/>

<http://www.extremeprogramming.org/>

<http://c2.com/cgi/wiki>

<http://www.xpdeveloper.com/>

<http://www.junit.org/>

<http://www.armaties.com/extreme.htm>

<http://www.xp2001.org/>:

the XP2002 conference is currently accepting submissions

<http://pairprogramming.com/>

<http://xp123.com/>