

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. Wolffried 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"

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.

Keywords: eXtreme Programming, XP, methodologies, Software Engineering

The battle lines are drawn. Hostilities have broken out between armed camps of the software development community. This time the rallying cry is, “XP!”

At recent OOPSLA conferences advocates of Extreme Programming (XP) made themselves conspicuous with their red “I XP – Do You?” badges. Some well-known authors and consultants in the OO community reinvented themselves as preachers of XP; others muttered under their breath about the end of the world.

If you’ve been in a dark cave (or cubicle) for more than a year, and you somehow haven’t heard about XP, it’s a lightweight OO development process. Like all good religions, XP is built around a codifiable belief system and a collection of practical techniques. These techniques include small teams, pair programming, JAD with business stories, very short development iterations, automated testing, and discovered design. This

Extreme Programming, a lightweight OO development process, is the latest eruption between programmers and software engineers.

is not a general introduction to the EXP methodology. Those interested should refer to [Beck 99], [Internet].

What XP uncovered (again) is an ancient, sociological San Andreas fault that runs under the software community – programming versus software engineering (a.k.a. the scruffy hackers versus the tweedy computer scientists). XP is only the latest eruption between opposing continents.

Which brings us back to the OOPSLA conferences. Imagine you were an anthropologist moving invisibly among the social cliques at OOPSLA. If astute (and politically incorrect) enough to risk some broad stereotypes, you might discern two opposing belief systems (see the table).

If it weren’t for Microsoft bashing, what would ever bring these tribes together? They resemble Republicans and Democrats battling ideologies caught up in the divisive dualism of either-or positions on hot-button issues (while the rest of the country rolls its eyes and stays home from the polls).

But, just maybe, the software development world isn’t black-and-white. Maybe it’s a fuzzy grayscale. For example, there have been projects where “hack (er, prototype) until it works” RAD approaches – some more “extreme” than XP – were successful, even tactically appropriate. I’ve seen a few. However, in some cases I later saw those same RAD teams try applying their techniques to other kinds of efforts, only to fail disastrously and wonder why.

At the other extreme, I’ve also watched helplessly while a high ceremony heavyweight process brought an organization of talented, formerly productive software engineers to a dead stop. Crimes were committed in the name of SEI CMM and ISO 9001. Yet, eventually management had to let go their dreams of Malcolm Baldrige awards and let their people get some real work done.

On the other hand, I once had the privilege to observe an organization achieve CMM maturity Level 4¹ certification without the baggage of a productivity-killing, paperwork-clogged high ceremony methodology. Lean, mean ... yet mature.

1. On a scale of 1 to 5, as assessed by the SEI. Most (90% by some estimates) development organizations never even reach Level 2.

Michael McCormick has worked with information technology in the United States since 1977. He has worked with object-oriented design and development processes since 1991, including many techniques now incorporated in eXtreme Programming. From 1997 to 2000 he was Principal Methodologist in the Object Technology Centre of a Fortune 100 company. Currently he is senior system architect in a large American bank. He is also affiliated with the Software Engineering Institute at Carnegie-Mellon University and the Object Technology User group at Saint Thomas University. In the past year he’s been quoted in eWeek, Information Week, and Network World. His article “Programming Extremism” triggered much dialog in America, was cited in an academic paper, and debated at a conference (XP Universe). Michael hopes that constructive XP debate will continue in Europe, and welcomes reader e-mail at m.mccormick@acm.org.

© ACM. The original version of this article, “Programming Extremism”, has been published in “Communications of the ACM”, June 2001, Vol. 44, pp. 109–111. We republish it with the kind permission of the author and of ACM.

Observed Belief Patterns in Software Community	
Group "P" Beliefs	Group "S" Beliefs
"Code is easy to change."	"Code is expensive to change."
Likes verbal communication.	Likes written specification.
"The code is the design."	"Code is poor design artifact."
"Good designs emerge."	"Good design comes up front."
"Programmers collaborate."	"Programmers can't communicate."
Codes with peers.	Reviews code for defects.
Informal requirements suffice.	Formal specs and change control.
Loved RAD.	Smugly says, "I warned you!"

An XP evangelist² recently accused the Software Engineering Institute of setting back the practice of computer programming. Now certainly CMM has been abused, but this attitude betrays a misunderstanding (and mistrust) of software engineering's goals. The goals are worthy, and (surprise) they can even be implemented with lightweight methodologies where appropriate.

It would be enlightening to conduct a CMM assessment of a team successfully practicing XP. In theory, I see no reason why the XP team should not achieve a maturity level of 2 or better. CMM Level 2 is about managing project requirements and schedules effectively and repeatably. XP claims to do just that, using story cards and a planning game.

On the software engineering side of the fault line, there is an equal amount of misunderstanding and mistrust. Superficially, XP resembles RAD in some respects, and at least as many crimes have been committed in the name of rapid prototyping as in the name of SEI CMM. Yet, as with CMM, in many such cases RAD itself was less to blame for its failures than were the people who misused it. Besides, XP theoretically demands a level of discipline and rigor well above RAD.

It's time to stop the methodology crusades. A one-size-fits-all development process does not exist. Software projects vary wildly in technology, size, complexity, risk, critically, regulatory, and cultural constraints, and many other key variables. Alistair Cockburn³ has done insightful work mapping out the spectrum of software projects, and the parallel Methodology Space. He argues persuasively that there is a sweet spot where XP will flourish, mainly on smaller, less critical projects.

What's needed is not a single software methodology, but a rich toolkit of process patterns and "methodology components" (deliverables, techniques, process flows, and so forth) along with guidelines for how to plug them together to customize a methodology for any given project. My own work led me away from one-size-fits-all methods, and toward tailorable process frameworks based on proven best practices.

By recognizing each project's unique needs and circumstances, and giving them the flexibility they need to succeed (while

2. Ron Jeffries, speaking at the Current Object Practices and Experience conference (COPE '99), St. Thomas University, St. Paul MN, 11/16/99.
 3. Cockburn's writings on Methodology Space are available from the Humans and Technology Web site (members.aol.com/humansandt/). Especially recommended is his article "One Methodology Per Project."

balancing their parochial needs against strategic goals of the enterprise to improve reuse, quality, cost, and so forth), we found corporate development guidelines gain more acceptance from development teams. A process rejected by practitioners is doomed to fail. Methodologists, managers, and Software Engineering Process Group police must resist the temptation to blame such rejections on the so-called practitioners, which would be like the Coca-Cola Company blaming consumers for the failure of New Coke.

Even if XP is best suited only to certain projects, it ought to be one of the tools in our bag of tricks. How often (if ever) one actually uses XP (or any other process) becomes a matter of project circumstances, not religious beliefs.

The dream (now misguidedly advocated by some members of the Object Management Group) of a single, standard grand unified process is fool's gold. It would be much more useful to collaborate on a series of process frameworks, or a meta-methodology, based on abstractions of proven process patterns. Under certain project conditions, XP might be an instantiation of the framework. Under other conditions, something like RUP⁴ or OPEN⁵ might emerge. Robert Martin has even demonstrated that XP can be viewed as a sort of degenerate case of RUP.⁶ In the end, we're still left with the underlying social conflict – those scruffy programmers and tweedy software engineers pitted against each other. Even if the XP issue is defused and the current furor subsides, another conflict seems likely to erupt again.

If the opposing groups are like Democrats and Republicans, then maybe what the software development community needs is a Reform Party. Is there a disaffected, apathetic silent majority that is neither scruffy nor tweedy, that isn't violently emotional about methodology? You bet there is. Most of us want tools that work, not religious dogma.

Who speaks for the pragmatists? It's time for a third party in software development politics.

References

[Beck 99]
 Beck, K. The XP Bible Remains Extreme Programming Explained: Embrace Change. Addison-Wesley, Cambridge, Mass., 1999.
 [Internet]
www.extremeprogramming.org; www.XProgramming.com; or [/c2.com/cgi/wiki?Extreme-Programming](http://c2.com/cgi/wiki?Extreme-Programming).

4. Rational Unified Process, a relatively formal OO development process derived in large measure from Ivar Jacobson's Objectory, but now marketed by Rational Corp. under this some-what misleading name. Recently RUP has moved belatedly toward a flexible framework in an effort to embrace (or co-opt, depending on your point of view) XP (see also footnote 6); www.rational.com/products/rup/index.jsp for more on RUP.
 5. The OPEN group, a consortium of companies and individuals, propose an alternative process standard. It rivals RUP, but remedies some of its shortcomings. In its second version, OPEN was already evolving to more of a framework than a fixed process before the advent of RUP or XP; www.open.org.au/.
 6. Rational's latest Unified Process is rubbery enough it can be squeezed and stretched to look almost like XP. Robert Martin dubbed his "extreme RUP" dx (read it upside-down); www.objectmentor.com/publications/RUPvsXP.pdf.