Nr./No. 2 – April / Avril 2002

INFORMATIK

INFORMATIQUE

Zeitschrift der schweizerischen Informatikorganisationen

Revue des organisations suisses d'informatique

eXtreme Programming

Herausgeber • ***Editeur***
SVI/FSI Schweizerischer Verband der Informatikorganisationen
Fédération suisse des organisations d'informatique
Postfach, 8037 Zürich
vertreten durch • ***représenté par***
Prof. Dr. Klaus R. Dittrich, Prof. Dr. Niklaus Ragaz, Prof. Dr. Carl August Zehnder

Redaktion • ***Rédaction***
François Louis Nicolet
Schwandenholzstr. 286, 8046 Zürich
Tel. 01 371 73 42, Ffax 01 371 23 00, E-Mail <nicolet@acm.org>

Beirat • ***Comité consultatif***
Prof Dr. Boi Faltings (EPFL Lausanne), Prof. Dr. Stefan Klein (Universität Koblenz-Landau), Dr. Ernst Lebsanft (Synlogic AG, Binningen), Prof. Dr. Rudolf Marty (Itopia, Zürich), Prof. Dr. Moira C. Norrie (ETH Zürich), Prof. Dr. Hans-Jörg Schek (ETH Zürich), Dr. Hans Stienen (Synspace, Binningen), Helmut Thoma (IBM Global Services Basel), Dr. Reinhold Thurner (Herrliberg), Rudolf Weber (Mummert + Partner AG Zürich)

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

Layout: Pascale Schürmann

Druck, Versand • ***Impression, expédition***
Stulz AG, 8942 Oberrieden

Inseratenverwaltung • ***Régie des annonces***
Jean Frey Fachmedien, Postf., 8021 Zürich
Telefon 01 448 86 34, Telefax 01 448 89 38
E-Mail <jiri.touzimsky@jean-frey.ch>

Jahrgang • ***Année de parution:*** 9. Jahr • 9ème année

Erscheinungsweise: Jährlich 6 mal
Parution: 6 fois par an

Auflage • ***Tirage:*** 5'000 Exempl.

Bezugsmöglichkeiten, Abonnements
Possibilités d'obtention, abonnements
Die Zeitschrift INFORMATIK/INFORMATIQUE ist offizielles Organ der folgenden Informatikorganisationen
La revue INFORMATIK/INFORMATIQUE est l'organe officiel des organisations d'informatique suivantes

SI	Schweizer Informatiker Gesellschaft Société Suisse des Informaticiens Schwandenholzstr. 286, 8046 Zürich admin@s-i.ch, Tel. 01 371 73 42, Fax 01 371 23 00 http://www.s-i.ch/
SwissICT	Schweizerischer Verband der Informations- und Kommunikationstechnologie (früher SVD und WIF) info@swissict.ch, Tel. 056 222 65 00 http://www.swissict.ch/
ch/open	Swiss Open Systems User Group Postfach 2322, 8033 Zürich info@ch-open.ch, http://www.ch-open.ch/

INFORMATIK/INFORMATIQUE kann von den Mitgliedern folgender Informatikverbände über ihren Verband bezogen werden
Les membres des associations suivantes peuvent obtenir la revue INFORMATIK/INFORMATIQUE auprès de leur association

GRI	Groupement Romand pour l'Informatique, Case postale 90, 1000 Lausanne 21
GST	Gesellschaft zur Förderung der Software-Technologie, Kreuzfeldweg 9, 4562 Biberist
SIK	Schweizer Informatik-Konferenz Postfach 645, 4003 Basel

Jahresabonnement / Abonnement annuel	**CHF 80.–**
Einzelnummer / Numéro isolé	**CHF 15.–**

SVI/FSI, Postfach, 8037 Zürich

ISSN 1420-6579

☞ **Visit our site http://www.svifsi.ch/revue**

eXtreme Programming
Guest Editor: Luis Fernández

Joint issue with NOVÁTICA and UPGRADE

2 Editorial: INFORMATIK / INFORMATIQUE 1994 – 2002 – *Carl August Zehnder*
4 Presentation – *Luis Fernández, guest editor*
5 A new method of Software Development: eXtreme Programming – *César F. Acebal and Juan M. Cueva Lovelle*
10 The Need for Speed: Automating Acceptance Testing in an XP Environment – *Lisa Crispin, Tip House and Darol Wade*
17 Qualitative Studies of XP in a Medium Sized Business – *Robert Gittins, Sian Hope and Ifor Williams*
22 XP and Software Engineering: An Opinion – *Luis Fernández Sanz*
25 XP in Complex Project Settings: Some Extensions – *Martin Lippert, Stefan Roock, Henning Wolf and Heinz Züllighoven*
4 Useful references – *Collected by Luis Fernández Sanz*

Mosaic

29 Informatik und Recht: Forensics – *Ursula Sury*
30 Europäische Harmonisierung der Studienabschlüsse: Neuer Bachelor und Master in Wirtschaftsinformatik an der Universität Fribourg – *Andreas Meier*
31 Architekten der Informationsgesellschaft – *Interwiew mit Andreas Meier*
32 IT-Security darf nichts kosten! – *Björn Kanebog*
35 IFIP NEWS
35 The European IST Prize
37 Turing Day – First Call for Participation
36 Einladung für das Symposium e-government

Schweizer Informatiker Gesellschaft • Société Suisse des Informaticiens

38 Informatik/Informatique geht – Informatik Spektrum kommt!
38 Informatik/Informatique disparaît – Informatik Spektrum apparaît!
39 Drei Fragen an Hermann Engesser, Chefredaktor, Informatik-Spektrum
39 Trois questions posées à Hermann Engesser, Rédacteur en chef, Informatik-Spektrum
43 DBTA Databases Theory and Application • 43 JUGS Java News • 43 SAUG Swiss APL User Group • 43 Security Veranstaltungen

Swiss ICT

42 Portrait Markus Fischer Mitglied des Vorstandes und des Strategie-Ausschusses SwissICT
45 Veranstaltungskalender
43 SwissICT-Salärumfrage 2002 – *Hans-Peter Uehli*
44 Managing Speed and Complexity – Symposium
44 Einladung zur Generalversammlung
45 Rückblick und Ausblick auf die Verbandstätigkeiten von SwissICT – *Hans-Peter Uehli*
46 Alter Wein in neuen Schläuchen? Was steckt hinter der Metapher E-Marketplace oder die neuen Brückenbauer – *Thomas C. Flatt*
47 Assessing Readiness for (Software) Process Improvement – *Hans Sassenburg*
50 Software Process Improvement: the new 'silver bullet'? – *Hans Sassenburg*

Editorial

INFORMATIK / INFORMATIQUE 1994 – 2002

ein Leuchtpunkt zwischen Chancen und Zwängen

Die Idee, für die schweizerischen Informatikfachleute eine eigenständige Fachzeitschrift zu gründen und diese den (allzu) vielen bereits existierenden Informatikverbänden als Verbandsorgan anzubieten, wurde im Dachverband SVI/FSI seit den Achtziger Jahren diskutiert. Auch hatten uns bereits Verlage zu diesem Thema angesprochen und finanziell günstige Lösungen offeriert, allerdings nur für inseratelastige “Computerheftli”, wo sogar die Textbeiträge teilweise von Public Relations Agenturen stammen. Genau dies wollten aber die Fachverbände für ihre Mitglieder nicht; sie suchten eine gehaltvolle Fachzeitschrift, die nicht kurzlebiges Produktewissen, sondern das in der Informatik so grundlegende Konzeptwissen vermitteln sollte und dies in einem der Schweizer Leserschaft zuträglichen Sprachenmix (deutsch, französisch, englisch).

Eine solche Zeitschrift existierte nicht und musste somit erst geschaffen werden. Eine zufällige Häufung kritischer Entwicklungen Ende 1993 machte die dafür nötigen Synergien frei:

- In der SI (Schweizer Informatiker Gesellschaft) mit ihren etwa 2000 Mitgliedern stand das bisherige Mitteilungsblatt *SI-Information* vor dem Aus, da die ehrenamtlichen Kapazitäten erschöpft waren.
- Der WIF (Wirtschaftsinformatik-Fachverband – inzwischen im SwissICT aufgegangen) mit 1800 Mitgliedern trat damals aus dem SKV aus und verlor damit dessen wöchentliches, aber nicht informatikbezogenes Organ.
- Der Dachverband SVI/FSI suchte (nach einem 1992 misslungenen Verbandsfusionsprojekt) nach geeigneten Möglichkeiten zur Unterstützung seiner Mitgliedorganisationen: eine schweizerische Fachzeitschrift wäre dazu ideal.

Die damaligen Präsidenten (Helmut Thoma SI, Walter Wüst WIF, Carl August Zehnder SVI/FSI) erkannten und nutzen die Chance und schufen in wenigen Wochen die organisatorische und wirtschaftliche Basis für ein Unternehmen, das schon damals mit einem Jahresbudget von 180’000 Franken rechnete. Möglich war das allerdings nur, weil sich der frühere Redaktor der *SI-Information*, François Louis Nicolet, damals beruflich neu orientieren wollte und seine Fachkompetenz in Informatik und Journalismus kombiniert mit seinem Enthusiasmus für Anliegen der Informatikverbände gleich in das neue Projekt einbringen konnte. FLN war von Beginn weg Seele, Motor und Realisator der neuen Zeitschrift und er blieb es bis zum heutigen Tag.

un flambeau entre chances et contraintes

Depuis les années huitante on a discuté au sein de l’association faîtière SVI/FSI l’idée de créer une revue spécialisée autonome pour les informaticiens en Suisse et de l’offrir comme organe associatif aux (trop) nombreuses associations d’informatique. Des maisons d’éditions nous avait déjà offert des solutions financièrement avantageuses, mais il s’agissait de magazines dominées d’annonces et de contributions rédactionnelles provenant essentiellement d’agences de publicité. C’est précisément ce que nos associations professionnelles ne voulaient pas pour leurs membres ; elles cherchaient une publication spécialisée substantielle donnant non des connaissances éphémères de produits mais les connaissances conceptuelles si fondamentales en informatique, dans les langues des lecteurs suisses (allemand, français, anglais).

Une telle revue n’existait pas et il fallait donc la créer. Une accumulation fortuite de développements critiques à la fin de 1993 a libéré les synergies nécessaires :

- A la SI (Société Suisse des Informaticiens) avec ses quelques 2000 membres, le bulletin d’information *SI-Information* allait cesser parce que les capacités volontariat s’épuisaient.
- WIF (Wirtschaftsinformatik-Fachverband, intégré maintenant dans l’association SwissICT) avec 1800 membres quittait la Société suisse des employés de commerce et perdait ainsi son organe hebdomadaire, qui d’ailleurs ne touchait pas à l’informatique.
- L’association faîtière SVI/FSI cherchait (après un essai de fusion échoué en 1992) des occasions appropriées d’assister ses organisations membres : une revue spécialisée suisse serait idéale.

Les présidents d’alors (Helmut Thoma SI, Walter Wüst WIF, Carl August Zehnder SVI/FSI) ont profité de cette occasion. En quelques semaines la base d’organisation et économique pour une entreprise qui comptait déjà à l’époque un budget annuel de 180 000 francs fut créé. Mais ce n’était possible que grâce à l’ancien rédacteur de *SI-Information*, François Louis Nicolet qui cherchait une nouvelle orientation professionnelle et offrit au nouveau projet sa compétence en informatique et en journalisme ainsi que son enthousiasme pour les besoins des associations d’informatique. Dès le début FLN fut l’âme, le moteur et le réalisateur de la nouvelle revue et il l’est resté jusqu’à présent.

Die INFORMATIK hatte dank SI und WIF von Beginn weg eine Auflage von 4000, die innert acht Jahren allerdings nur auf gut 5000 gesteigert werden konnte, trotz teilweiser oder zeitweiser Übernahme durch andere Mitgliedorganisationen des SVI/FSI (namentlich GRI, SVD/SwissICT, GST, /ch/open). Einfach war der Start nicht, und auch die folgenden Jahre brachten immer wieder schwierige Momente. Erinnert sei an dieser Stelle etwa an das "hinkende" Erscheinen (1995–1999), da der WIF damals aus finanziellen Gründen nur vier der sechs jährlichen Ausgaben für seine Mitglieder zu übernehmen bereit war. Ständige Unsicherheit brachte auch der Inserateteil, da Inserate in einer Verbandszeitschrift nicht so einfach zu gewinnen sind – einige wenige Gönner haben hier am Anfang wesentlich geholfen, später war es eine im Hintergrund wirkende Gruppe von "Freunden der Schweizer Informatik", die immer wieder zur Inserateakquisition beigetragen haben.

Trotz dieser Probleme entwickelte sich die Zeitschrift aber sowohl inhaltlich, wie auch äusserlich ständig weiter. Wer eine alte Nummer mit einer von 2002 vergleicht, sieht das am roten Kleid, aber auch an vielen Einzelheiten im Innern. Zur Qualitätssicherung beim Inhalt konnte der Redaktor auf die Unterstützung des redaktionellen Beirats zählen, dem Professoren und Praktiker gleichermassen angehörten. Zuverlässige Unterstützung erhielt die INFORMATIK von der Druckerei (Stulz AG, Oberrieden) und vom Inserateverwalter (Jean Frey Fachmedien, Zürich) sowie von weiteren Helfern im Bereich Übersetzung, Korrektur, Layout. Hinter allem aber stand von Anfang bis Ende die kräftige, zuverlässige und qualifizierte Hand unseres Redaktors F.L. Nicolet. Ihm zuerst, aber auch allen anderen Beteiligten, sei in diesem Moment des Abschieds ganz herzlich gedankt. Denn die Weiterführung einer inhaltlich anspruchsvollen, aber allein auf die Schweiz bezogenen Informatikfachzeitschrift konnte aus personellen, wirtschaftlichen und strukturellen Gründen nicht mehr gesichert werden – die Zeit der INFORMATIK/INFORMATIQUE ist damit abgelaufen.

Carl August Zehnder
Präsident SVI/FSI

Siehe auch:
Mitteilungen der Schweizer Informatiker Gesellschaft, Seite 38
Mitteilungen der SwissICT, Seite 43

Avec la SI et la SVI/FSI INFORMATIK/INFORMATIQUE débuta avec un tirage de 4000. Il n'a pu être augmenté au cours de huit ans qu'à quelque 5000 malgré la participation partielle et temporaire d'autres organisations affiliées à la SVI/FSI (notamment GRI, SVD, SwissICT, GST, /ch/open). Le commencement n'a pas été facile et les années suivantes ont apporté des moments difficiles. Rappelons ici la parution « boiteuse » (1995–1999) parce que WIF, pour des raisons financières, ne prenait que quatre des six éditions annuelles. La section des annonces ne nous a pas non plus épargné des incertitudes : il est difficile d'acquérir des annonces pour une revue associative. Quelques rares donateurs ont aidé substantiellement au début, puis les « amis de l'informatique suisse », un groupe agissant à l'arrière-plan, ont contribué à l'acquisition d'annonces.

Malgré tous ces problèmes le contenu et la présentation de la revue n'ont cessé de se développer. En comparant les premières éditions à celles de 2002 on s'en rend compte à l'« habit rouge » et à bien des détails à l'intérieur. Pour le contrôle de qualité des articles, le rédacteur à pu compter sur l'aide du comité consultatif rédactionnel auquel appartiennent professeurs et praticiens. L'imprimerie Stulz AG à Oberrieden a pris soin de l'impression et de l'expédition, et c'est à l'agence Jean Frey AG de Zurich que nous devons l'acquisition des annonces. S'ajoutent l'assistance, en grande partie bénévole, de traducteurs, traductrices, correcteurs et correctrices et de la responsable de la conception graphique et la mise en page. Derrière tout cela il y avait, du début jusqu'à la fin, la main forte, sûre et qualifiée de notre rédacteur F.L. Nicolet. En ce moment d'adieu, nous remercions très cordialement tous ces collaborateurs. Car, pour des raisons de ressources humaines, économiques et structurelles, nous ne pouvons plus garantir la continuation d'une revue d'informatique spécifiquement Suisse répondant à de hautes exigences professionnelles. INFORMATIK/INFORMATIQUE a fini son temps.

Carl August Zehnder
Président SVI/FSI

Voir aussi :
Communications de la Société Suisse des Informaticiens, page 38
Communications de la SwissICT, page 43

Presentation

Extreme Programming

Luis Fernández, guest editor

This issue is focused on XP (eXtreme programming), one of the recent proposals in the software development field that has achieved a really important media impact among software practitioners. As a new way of improving the agility of software projects, XP relies on several principles (automated testing, pair development, etc.) that shorten the project life cycle between releases. But these principles are also devised to obtain a general improvement of software quality and user satisfaction, avoiding problems due to delays and exceeding budget. All these promises have raised a general interest in this development philosophy. Trying to satisfy the curiosity of our readers, we have decided to publish an interesting set of paper intended to contribute to a deeper understanding of XP, the pros and the cons.

"Extreme programming: a new software development method" was presented in the Sixth Software Quality and Innovation Spanish Conference (2001) organised by the Software Quality Group of ATI. In this paper, a brief description of the main characteristics and principles of XP is included. This contribution helps the reader to know the fundamentals of Extreme Programming.

"The Need for Speed: Automating Acceptance Testing in an Extreme Programming Environment" (L.Crispin, T.House and C.Wade) presents details of one of the more robust contributions of XP: automated testing. Instead of the usual unconcerning in testing that can be observed in the traditional software organisations, XP stresses the importance of a proper and efficient testing practice. As well as a brief review and discussion of XP testing principles (in a Q&A format), this paper presents details about a practical experience using JUnit and other "tools". This paper was presented in the XP 2001 conference (thanks to both the negotiation of Michele Marchesi, co-chairman of the Conference, and the collaboration of the authors).

"Qualitative Studies of XP in a Medium Sized Business" is another paper from XP2001 (thanks again to Marchesi and the authors). The paper examines the benefits of a flexible management approach to XP methodology. Presentation and discussion of results of an empirical study using qualitative research techniques (questionnaires, direct observation, etc.) are included as part of a good review of situations that emerge when people try to apply XP in a real organisation.

"XP and software engineering" presents my own analysis of XP. From the perspective of somebody who has to approach extreme programming from the "outer world", this paper is focused on several important improvement proposals. Of course, the doubts of the author about the success of some principles of XP related to classical software engineering practices (e.g. configuration management vs common property of code) are also included.

One of the main advantages of XP is the agility of the proposed software process and its direct application to small projects with a high rate of requirements volatility. But, is XP suitable for larger or more complex projects? To answer this question, M.Lippert, S. Roock, H. Wolf and H. Züllighoven present an extension of the roles of client representatives and the creation of new document types to address the subsequent project situation.

I hope this variety of contributions would satisfy the increasing demand of information about XP of software practitioners (in general sense) and our readers as the persons who we detove our work to.

Useful references

See also the references in the papers published in this issue.

Books

K. Beck, eXtremme Programming. Embrace change, Addison-Wesley, 2000.

K.Beck & M.Fowler, Planning eXtreme Programming, Addison-Wesley, 2000.

R. Jeffries, A. Anderson, C. Hendrickson, K. Beck, R.E. Jeffries, eXtreme Programming Installed, Addison-Wesley, 2000

G.Succi, M.Marchesi, eXtreme Programming Examined, Addison-Wesley, 2001.

W.C.Wake, eXtreme Programming Explored, Addison-Wesley, 2001.

R.Hightower and N.Lesiecki, Java Tools for eXtreme Programming: Mastering Open Source Tools Including Ant, JUnit, and Cactus, John Wiley & Sons, 2001.

K. Auer, R. Miller, eXtreme Programming Applied: Playing to Win, Addison-Wesley, 2001.

M. Fowler, K.Beck, J. Brant, W. Opdyke and D. Orberts, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.

Conferences

Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, May 26–29, 2002, Alghero, Sardinia, Italy. http://www.xp2002.org.

XP Agile Universe, August 4-7, 2002, Chicago, Illinois, USA. http://www.xpuniverse.com.

Web sites

Xprogramming, an Extreme Programming Resource. http://www.xprogramming.com/

eXtreme Programming: A gentle introduction. http://www.extremeprogramming.org/

PortlandPatternRepository and WikiWikiWeb: http://c2.com/cgi/wiki

XP Developer: http://www.xpdeveloper.com/

JUnit and other XUnit testing frameworks. http://www.junit.org/

eXtreme Programming. http://www.armaties.com/extreme.htm

Pair programming. http://pairprogramming.com/

XP123 – Exploring Extreme Programming. http://xp123.com/

Collected by ***Luis Fernández Sanz***

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. Naturally the length of any technical article does not permit more than a brief introduction to any new method or technique, but 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.

Keywords: eXtreme Programming, XP, Software Development.

1 Introduction

XP is a new software development discipline which, amid much fanfare, has recently joined the welter of methods, techniques and methodologies that already exist.To be more precise: this is a lightweight method, as opposed to heavyweight methods like Métrica. Before we go on, we'd like to make a clarification: in this article we refer to XP as a "method", contrary to the official IT tendency to apply the term "methodology" (science of methods) to what are no more than methods[1] or even mere graphic notations.

It could be said that XP was born "officially" five years ago in a project developed by *Kent Beck* at *Daimler Chrysler*, after he had worked for several years with *Ward Cunningham* in search of a new approach to the problem of software development which would make things simpler than the existing methods we were used to. For many people, XP is nothing more than common sense. Why then does it arouse such controversy, and why do some people adore it while others heap scorn on it? As Kent Beck suggests in his book [Beck 00], maybe it's because XP carries a series of common sense techniques and principles to extreme lengths. The most important of these techniques are:

- Code is constantly reviewed, by means of *pair programming* (two people per machine)
- Tests are made all the time, not only after every class (*unit tests*), but also by customers who need to check that the project is fulfilling their requirements (*functional tests*)
- Integration tests are always carried out before adding any new class to the project, or after modifying any existing one (*continuous integration*), making use of *testing frameworks*, such as *xUnit*
- We (re)design all the time (*refactoring*), always leaving code in the simplest possible state
- Iterations are radically shorter than is usual in other methods, so that we can benefit from feedback as often as possible.

By way of summary, and to wrap up this section and to get down to the nitty-gritty, I will leave you with this quote from Beck's aforementioned book.

> *"Everything in software changes. The requirements change. The design changes. The business changes. The technology changes. The team members change. The problem isn't change, per se, because change is going to happen; the problem, rather, is the inability to cope with change when it comes."*

2 The four variables

XP sets out four variables for any software project: *cost*, *time*, *quality* and *scope*.

César Fernández Acebal received a degree in informatics engineering from Oviedo University. He worked as a teacher in Java and Web programming for students of higher professional education. Afterwards, he worked as technical director in a web site development company. He has combined these positions with a continuous educational activity related to Java, XML, Web development, etc. He is currently an IT architect of B2B 2000, an e-business company. His research interests include object-oriented programming and software engineering and agile software processes. He is a member of ATI, IEEE, Computer Society, ACM, etc. <acebal@ieee.org>

Juan Manuel Cueva Lovelle is a mining engineer from Oviedo Mining Engineers Technical School in 1983 (Oviedo University). He has the Ph. D. from Technical University of Madrid in 1990. From 1985 he is Professor at the Languages and Computers Systems Area in Oviedo University. ACM and IEEE voting member. His research interests include Object-Oriented technology, Language Processors, Human-Computer Interface, Object-Oriented Databases, Web Engineering, Object-Oriented Languages Design, Object-Oriented Programming Methodology, XML, WAP, Modelling Software with UML and Geographical Information Systems. <cueva@lsi.uniovi.es>

1. Ricardo Devis Botella. C++. STL, *Plantillas, Excepciones, Roles y Objetos* (*Templates, Exceptions, Roles and Objects*). Paraninfo, 1997. ISBN 84-283-2362-3

It also specifies that of these four variables, only three of them can be established by parties outside the project (customers and project managers), while the value of the free variable will be established by the development team in accordance with the other three values. What is new about this? It's that normally customers and project managers considered it their job to pre-establish the value of *all* the variables: *"I want these requirements fulfilled by the first of next month, and you have this team to work with. Oh, and you know that quality is the number one priority!"*

Of course when this happens – and unfortunately it happens quite often – quality is the first thing to go out of the window. And this happens for a simple reason which is frequently ignored: no one is able to work well when they are put under a lot of pressure.

XP makes the four variables visible to everyone – programmers, customers and project managers –, so that the initial values can be juggled until the fourth value satisfies everybody (naturally, with the possibility of choosing different variables to control).

Also the four variables do not in fact bear such a close relation with one another as people often like to think. There is a well known saying that "nine women cannot make a baby in one month" which is applicable here. XP puts special stress on small development teams (ten or twelve people at most) which naturally can be increased if necessary, but not before, or the result will generally be the opposite of what was intended. However, a number of project managers seem to be unaware of this when they declare, puffed up with pride, that their project involves 150 people, as if it were a mark of prestige, something to add to their CV. It is good, however, to increase the cost of the project in matters such as faster machines, more specialists in certain areas or better offices for the development team.

With *quality* too, another strange phenomenon occurs: often, *increasing the quality means the project can be completed in less time*. The fact is that as soon as the development team gets used to doing intensive tests (and we will be coming to this point soon, as it's the corner stone of XP) and coding standards are being followed, gradually the project will start to progress much faster than it did before. The project's quality will still remain 100% assured – thanks to the tests – which in turn will instil greater confidence in the code and, therefore, greater ease in coping with change, without stress, and that will make people programme much faster… and so on.

The other face of the coin is the temptation to sacrifice the internal quality of the project – that which is perceived by the programmers – to reduce the delivery time of the project, trusting that the external quality – that which the customers perceive – will not be affected too greatly. However, this is a very short term bet, which tends to be an invitation for disaster, since it ignores the basic fact that *everyone works better when they are allowed to do a quality job*. Ignoring this will cause the team to get demoralised and, in the long term, the project will slow down, and much more *time* will be lost than ever could have been hoped to be saved by cutting down on quality.

With regard to the project's *scope*, it is a good idea to let this be the free variable, so that once the other three variables have been established, the development team should decide on the scope by means of:

- The estimation of the tasks to perform to satisfy the customer's requirements.
- The implementation of the most important requirements first, so that at any given time the project has as much functionality as possible.

3 The cost of change

Although we cannot go into any great depth on this subject here, we believe it is important to at least mention one of the most important and innovative suppositions that XP makes in contrast to most known methods. We are referring to the cost of change. It has always been considered a universal truth that the cost of change in the development of a project increased exponentially in time, as shown in figure 1.

XP claims that this curve is no longer valid, and that with a combination of good programming practices and technology it is possible to reverse the curve, as we show in figure 2.

Naturally, not everyone agrees with this supposition (and in Ron Jeffries web site [Jeffries] you can read several opinions to this effect). But in any event it is clear that if we decide to use XP as a software development process we should accept that this curve is valid.

The basic idea here is that instead of changing for change's sake, we will design as simply as possible, to do only what is absolutely necessary at any given moment, since the very simplicity of the code, together with our knowledge of refactoring [Fowler 99] and, above all, the testing and continuous integration, all mean that changes can be carried out as often as necessary.

4 Practices

But let's get down to brass tacks. What does XP really entail? What exactly are these practices we have been referring to, which are able to bring about this change of mentality when it comes to developing software? Trusting that you, the reader, will excuse the enforced brevity of our explanation we will now give a brief description of these practices.

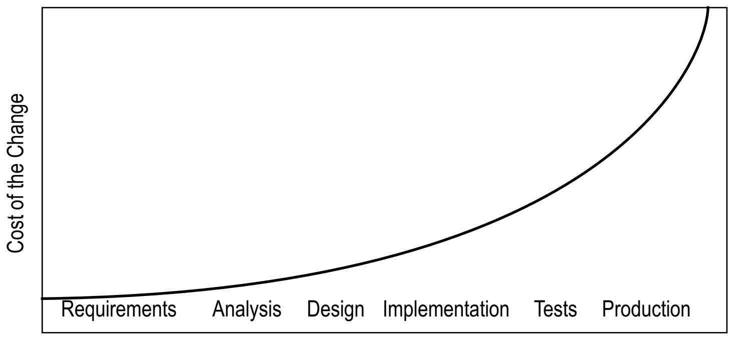


Fig. 1: Cost of change in "traditional" software engineering

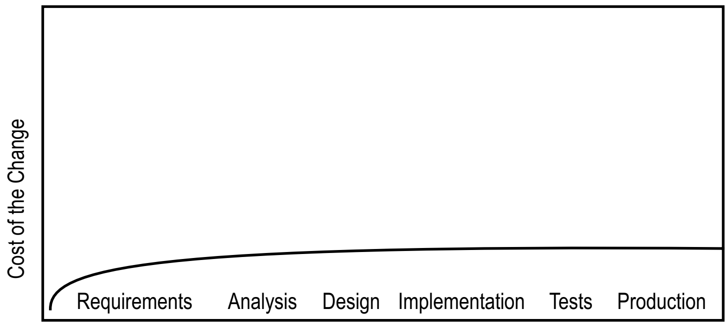


Figure 2: Cost of change in XP

Planning

XP sees planning as a permanent dialogue between the business and technical parties involved in the project, in which the former will decide the scope – what is really essential for the project –, the priority – what should be done first –, the composition of the releases – what should be included in each one – and the deadline for these releases.

The technical people, for their part, are responsible for estimating the time needed to implement the functionalities which the customer requires, for reporting on the consequences of decisions taken, for organising the work culture and, finally, for carrying out a detailed planning of each release.

Small releases

The system first gets into production just a few months at most before it is completely finished. Successive releases will be more frequent: at intervals of between a day and a month. The customer and the development team will benefit from the feedback produced by a working system and this will be reflected in successive releases.

Simple design

Instead of being hell bent on producing a design which requires the gift of clairvoyance to develop, what XP advocates is, at any given moment, that we should design for the needs of the present.

Testing

Any feature of a programme for which there is not an automated test simply does not exist. This is undoubtedly the cornerstone on which XP is built. Other principles are liable to be adapted to the characteristics of the project, the organisation, the development team… But on this one point there is no argument: if we aren't doing tests, we aren't doing XP. We should be using some automated testing framework to do this, such as JUnit [JUnit] or any of its versions for different languages.

Not only that, but we will write the tests even before we write the class which is to be tested. This is an aid to following the principle of *programming by intention*, that is, writing code as if the most expensive methods had already been written, and so we only had to send the corresponding message, in such a way that the code will be a true reflection of its intention and will document itself. On JUnit's web site, you can find interesting articles which explain how these tests should be written.

Refactoring

This responds to the principle of simplicity and basically consists of leaving the existing code in the simplest possible state, so that no functionality is lost nor gained, and all tests continue to be carried out correctly. This will make us feel more comfortable with the code already written and therefore less reluctant to modify it when some feature has to be added or changed. In the case of legacy systems, or projects taken over after they have already been started, we would be bound to need to devote several weeks just to refactoring the code – which tends to be a source of tension with the project managers involved when they are told that the project is going to be held up for several days "just" to modify existing code, which works, without adding any new functionality to it.

Pair programming

All code will be developed in pairs – two people sharing a single monitor and keyboard. The person writing the code should be thinking about the best way to implement a particular method, while his colleague will do the same, but from a more strategic viewpoint:

- Are we going about this in the right way?
- What could go wrong here? What should we be checking in the tests?
- Is there any way to simplify the system?

Of course, the roles are interchangeable, so that at any moment the person observing could take over the keyboard to demonstrate some idea or simply to relieve his colleague. Similarly the composition of the pairs could change whenever one of them were required by some other member of the team to lend a hand with their code.

Collective property of the code

Anyone can modify any part of the code, at any time. In fact, any one who spots an opportunity to simplify, by refactoring, any class or any method, regardless of whether they have written them or not, should not hesitate to do so. This is not a problem in XP, thanks to the use of coding standards and the assurance that testing gives us that everything is going to carry on working well after a modification.

Continuous integration

Every few hours – or at the very least at the end of a day's programming – the complete system is integrated. For this purpose there is what is known as an integration machine, which a pair of programmers will go to whenever they have a class which has passed a unit test. If after adding the new class together with its unit tests the complete system continues to function correctly, i.e. passes all the tests, the programmers will consider this task as completed. Otherwise they will be responsible for returning the system to a state in which all tests function at 100%. If after a certain time they are unable to

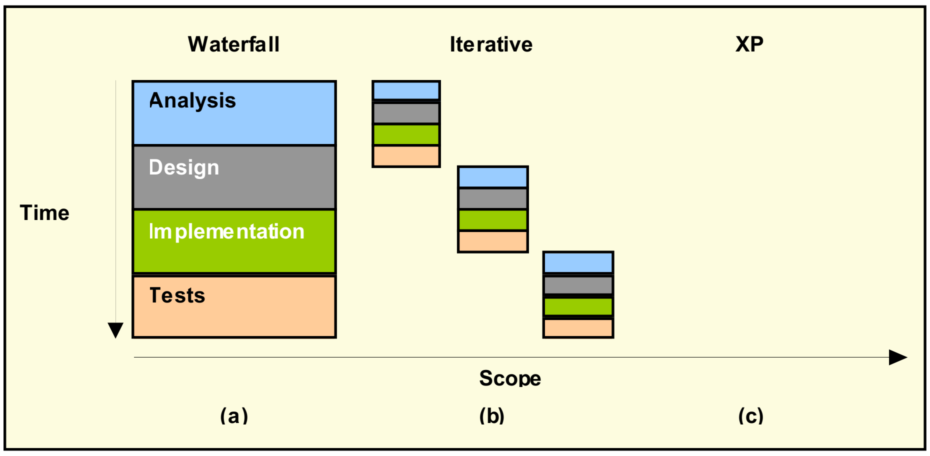


Figure 3: Comparison of long development cycles: (a) the waterfall model (b) shorter iterative cycles, for example, the spiral model and (c) the mix of all these activities which XP employs throughout the whole software development process

discover what it is wrong, they will bin the code and start over again.

40 hours a weeks

If we really want to offer quality, and not merely a system that works – which we all know, in IT, is trivial[2] – we will want each member of our team to get up each morning rested and to go home at 6 in the evening tired but with the satisfaction of a job well done, and that when Friday comes around he or she can look forward to two days' rest to devote to things which have nothing whatsoever to do with work. Naturally it doesn't have to be 40 hours, it could be anything between 35 and 45, but one thing is certain: nobody is capable of producing quality work 60 hours a week.

Customer on site

Another controversial XP rule: at least one real customer should be permanently available to the development team to answer any question the programmers may have for them, to establish priorities… If the customer argues that their time is too valuable, we should realise that the project we have been given is so trivial that they do not consider it worthy of their attention, and that they don't mind if it is based on suppositions made by programmers who know little or nothing of the customer's real business.

Coding standards

These are essential to the success of collective property of the code. This would be unthinkable without a coding based on standards which allow everyone to feel comfortable with code written by any other member of the team.

2. Ricardo Devis Botella. *Curso de Experto Universitario en Integración de Aplicaciones Internet mediante Java y XML.* (University Expert Course in the Integration of Internet Applications via Java and XML) University of Oviedo, 2000.

5 Planning

While XP is a code centred method it is not just that. It is also above all a software project management method, in spite of the criticism levelled by many people, perhaps after a too hasty reading of an article such as this one. But for anyone who has taken the trouble to read any of the books explaining the process, it will be clear that planning makes up a fundamental part of XP. The thing is that, given that software development, like almost everything in this life, is a chaotic process, XP does not attempt to find a non-existent determinism but rather provides the means necessary to cope with that complexity, and accepts it, without trying to force it into constraints of heavyweight or bureaucratic methods. We wholeheartedly recommended you to read Antonio Escohotado's gentle introduction to chaos theory [Escohotado 99], which we believe has a lot to do with the idea behind XP. In short, lightweight methods – and XP numbers among them – are adaptive rather than predictive [Fowler].

The life cycle

If, as has been demonstrated, long development cycles of traditional methods are unable to cope with change, perhaps what we should do is make development cycles shorter. This is another of XP's central ideas. Let's take a look at a chart which compares the waterfall model, the spiral model and XP:

6 Conclusions

As we said at the beginning article, XP, just one year after the publication of the first book on the subject, has caused a great furore among the software engineering community. The results of the survey below, commissioned by IBM, evidences the fact that opinions on the subject are divided [IBM 00]:

Pair programming comes in for some especially strong criticism, above all from project managers, though it is an opinion which is doubtless shared by many programmers with an overdeveloped sense of ownership regarding code ("I did this, and what's more I am so good at programming and I have such a command of the language's idioms that only I can understand it"), but a lot is also said about the myth of the 40 hour week, that "all this business about tests is all very well if you have plenty of time, but they are an unaffordable luxury under current market conditions"… and many other vigorous criticisms in a similar vein.

There are also people who say (and this criticism is perhaps more founded than the previous ones) that XP only works with good people, that is, people like Kent Beck, who are able to make a design which is good, simple and, at the same time – and maybe precisely for that reason – easily extendable, right from the outset.[3]

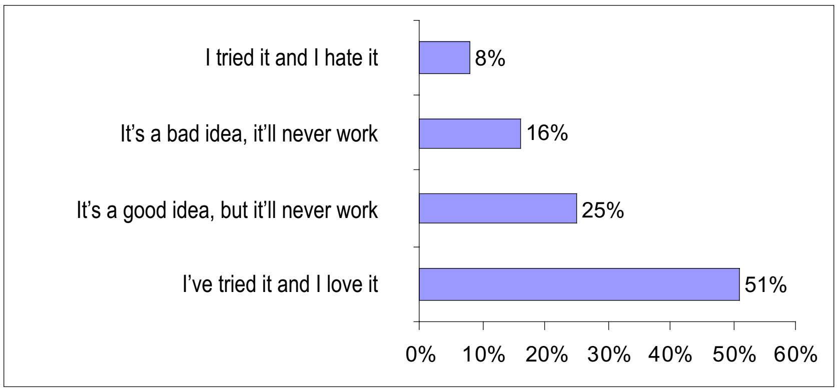


Figure 4: IBM survey (October 2000): What do you think about EXtreme Programming?

One of the things we are trying to say is that XP should not be misinterpreted due to the inevitable superficiality of articles such as the one you are reading. At the end of the day the creator of this method is not some upstart, but one of the pioneers in the use of software templates, creator of CRC files, author of the *HotDraw* drawing editor framework, and the *xUnit*.testing framework. Were it for no other reason, it would be worth at least taking a look at this new and exciting software development method.

However, none of the practices advocated by XP are an invention of the method; all of them existed before, and what XP has done is to put them all together and prove that they work.

In any event, Beck's first book is a breath of fresh air which should be compulsory reading for any *software engineer* or *software architect*, to use the term preferred by our friend Ricardo Devis, whatever conclusions you may finally draw about XP. At the very least, it's great fun to read.

3. Raúl Izquierdo Castanedo. *Comunicación privada*. (*Private communication*)

References

[Beck 99]
Kent Beck. Embracing Change with eXtreme Programming. Computer (magazine of the IEEE Computer Society).Vol. 32, No. 10. October 1999, pp. 70–77

[Beck 00]
Kent Beck. eXtreme Programming Explained: Embrace Change. Addison Wesley Longman, 2000. ISBN 201-61641-6

[Beck/Fowler]
Kent Beck, Martin Fowler. Planning eXtreme Programming. Addison-Wesley. ISBN 0201710919

[Escohotado 99]
Antonio Escohotado. Caos y orden (Chaos and order). Espasa Calpe, 1999. ISBN 84-239-9751-0. An interesting introduction to chaos theory, which in our view describes the attitude you need to approach software development from an XP point of view.

[Fowler]
Martín Fowler. The New Methodology. http://www.martinfowler.com/ articles/newMethodology.html

[Fowler 99]
Martin Fowler. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999. ISBN 0201485672

[Jeffries et al. 00]
Ronald E. Jeffries et al. EXtreme Programming Installed. Addison-Wesley, 2000. ISBN 0201708426

[IBM 00]
http://www-106.ibm.com/developerworks/java/library/java-poll-results/xp.html. Java poll results: What are your thoughts on EXtreme Programming? IBM survey, October 2000

[IEEE CS]
http://www.computer.org/seweb/Dynabook/Index.htm. eXtreme Programming. Pros and Cons. What questions remain? The first "dynabook" from the IEEE Computer Society was devoted to XP, with a series of related articles.

[JUnit]
http://www.junit.org. JUnit, an automated testing framework for Java, adapted from the framework of the same name for Smalltalk, and available in many other languages. These other versions, under the generic name xUnit, are available at the Ron Jeffries' web site [Jeffries], in the software section.

[Jeffries]
http://www.xprogramming.com. One of the most complete XP portals, by Ron Jeffries.

The Need for Speed: Automating Acceptance Testing in an XP Environment

Lisa Crispin, Tip House and Darol Wade

In "eXtreme Programming Explained", Kent Beck compares eXtreme Programming (XP) to driving a car: the driver needs to steer and make constant corrections to stay on the road. If the XP development team is steering the car, the XP tester is navigating. Someone needs to plot the course, establish the landmarks, keep track of the progress, and perhaps even ask for directions. Acceptance tests must go beyond functionality to determine whether the packages meet goals such as specified performance levels. Automating end-to-end testing from the customer point of view can seem as daunting as driving along the edge of a cliff with no guard rail. At Tensegrent, a software engineering firm in Denver organized around XP practices, the developers and the tester have worked together to design modularized, self-verifying tests that can be quickly developed and easily maintained. This is accomplished through a combination of in-house and vendor-supplied tools.

Keywords: Testing, Automated Testing, Acceptance Testing, Test Scripts, Tester, Test Tools, Web Testing, GUI Testing.

Introduction

The three XP books give detailed explanations of many aspects of the development side of XP. The test engineer coming from a traditional software development environment may not find enough direction on how to effectively automate acceptance tests while keeping up with the fast pace of an XP project. In an XP team, developers are also likely to find themselves automating acceptance tests – an area where they may have little experience. Automating acceptance testing in an XP project may feel like driving down a 12% grade in a VW bug with a speeding semi in the rear-view mirror. Don't worry – like all of XP, it requires courage, but it can – and should – be fun, not scary.

The XP practices we follow at Tensegrent include:

- pair programming
- test first, then code
- do the simplest thing that works (NOT the coolest thing that works!)
- 40-hour week
- refactoring
- coding standards
- small releases
- play the planning game

We apply these same practices to testing – including pair testing.

Lisa Crispin has more than10 years experience in testing and quality assurance, and is currently a Senior Consultant with BoldTech Systems (http://www.boldtech.com), working as a tester on Extreme Programming (XP) teams. Her article "Extreme Rules of the Road: How an XP Tester can Steer a Project Toward Success" appeared in the July 2000 issue of STQE Magazine. Her presentation "The Need for Speed: Automating Acceptance Tests in an Extreme Programming Environment" won Best Presentation at Quality Week Europe in 2000. Her papers "Testing in the Fast Lane: Acceptance Test Automation in an Extreme Programming Environment" and "Is Quality Negotiable?" will be published in a collection called Extreme Programming Perspectives from Addison-Wesley. She is co-writing a book Testing for Extreme Programming which will be published by Addison-Wesley in October 2002. Her presentations and seminars in 2001 included "XP Days" in Zurich, XP Universe in Raleigh, and STAR West. <lisa.crispin@att.net>

Tip House is Chief Systems Analyst at the OCLC Online Computer Library Centre Inc., a non-profit organization dedicated to furthering access to world's information, where he develops and supports test automation tools and document management systems for the Web. Although his main interest has always been software development, he also has a long-standing interest in software testing, software measurement, and quality assurance, having presented papers on these subject at development, measurement and testing conferences in the US and Europe. He has achieved Certified Quality Analyst, Certified Software Quality Engineer, and Lead Ticket Auditor certifications, and managed the independent test function at OCLC during their three year successful effort to become registered to the ISO9000 standards. <house@oclc.org>

Carol Wade, a technical writer for over twenty years, She has worked primarily in the field of computer software, writing end-user documentation. For nine years, she worked for Los Alamos National Laboratory, where she served as a writer and editor, and started a language translation service. Currently, she is the sole technical writer for Health Language, Inc., which has produced the first language engine for health care.

Do XP teams really need a dedicated tester? It's hard for a tester to answer this in an unbiased manner. In my experience, even senior developers don't have much testing experience, beyond unit and integration tests and perhaps load tests. They tend to write acceptance tests only for "happy paths" and don't think of the nasty evil steps that might break the system. At Tensegrent, we had one project wrapping up while another one was starting, so a decision was made to do the first two-week iteration of the new project with a developer serving as a part-time tester. By their own admission, without an experienced tester to push them, the developers got 90% of all the stories done by the end of the iteration. To the customer, this looked like nothing at all was done, and they were very unhappy. It took some work to win back the customer's trust.

How is Testing in XP Different?

How does acceptance testing in an XP environment deviate from traditional software testing? First of all, let's look at acceptance testing. Acceptance tests prove that the application works as the customer wishes. Acceptance tests give customers, managers and developers confidence that the whole product is progressing in the right direction. Acceptance tests check each increment in the XP cycle to verify that business value is present. Acceptance tests, the responsibility of the tester and the customer, are end-to-end tests from the *customer* perspective, not trying to test every possible path through the code (the unit tests take care of that), but demonstrating the business value of the application. Acceptance tests may also include load, stress and performance tests to demonstrate that the stability of the system meets customer requirements.

Should I strap on a helmet and arm the air bags?

Testing in an XP environment feels like a drive through twisting mountain roads at first. When I first read *eXtreme Programming Explained*, the very idea of testing without any formal written specifications seemed a bit *too* extreme. It's been difficult learning all the different ways I can contribute to the team's success. My roles can be confusing and conflicting – I'm part of the development team, but I need a more objective viewpoint. I'm a customer advocate, making sure the customer gets what she pays for. At the same time, I need to protect the developers from a customer who wants *more* than they paid for.

While XP is definitely a new way to drive, the road isn't as unfamiliar as some might think. For example, many people new to XP think that XP projects produce very little documentation. This hasn't been our experience. For one thing, the acceptance tests themselves become the main documentation of the customer requirements. They can be quite detailed and extensive. As an XP project progresses, many other documents may be produced: installation instructions, UML documents, Javadocs, developer setup documents, the list goes on. The difference between these and the documents in many traditional projects is, the XP project documents are up to date and accurate

Question: How do you write acceptance test cases without documents?

Answer: You don't need documents, because you have a customer there to tell you what she is looking for. Not that this is always easy. In my experience, it is fairly easy to get a customer to come up with tests for the intended functionality of the system. What is more difficult, and requires a tester's skill, is to make sure the customer thinks about areas such as security, error handling, stability, and performance under load.

Other differences between traditional and XP development are more subtle. It's really a matter of degree. XP projects move fast even when compared with the pace at the Web startup where I used to work. It's the fast lane on the Autobahn. A new iteration of the software, implementing new customer "stories", is released every one to three weeks. My goal is always to get acceptance test cases defined within the first day or two of an iteration, as these are the only written "specifications" available. For our projects, the acceptance test definitions have been a joint effort of the team.

From a tester's point of view, the developer to tester ratio in XP looks about as comfortable as driving through the desert in an un-air-conditioned Jeep. According to Kent Beck, there should be one tester for each eight-developer team. At Tensegrent, the ratio gets even higher.

Eeek! Are you sure protective gear isn't required?

Fear not! XP builds in checks and balances that enable a small percentage of test specialists to do an adequate job of controlling quality.

- Because the developers write so many unit tests, which they must write before they begin coding – the tester doesn't need to verify every possible path through the code.
- The developers are responsible for integration testing and must run every unit test each time they check in code. Integration problems are manifested before acceptance tests are run.
- The customer gives input to the acceptance tests and provides test data.
- The entire development team, not just the tester, is responsible for automating acceptance tests. Developers also help the tester produce reports of test results so that everyone feels confident about the way the project is progressing.

A caveat – if developers aren't diligent in writing and running unit tests and integrating often, you're going to have to hire more testers. A couple of iterations into our first project at Tensegrent, I told my boss I thought we'd have to hire more testers, there was no way I could keep up! The problem was simply that the developers hadn't gotten the hang of "test before code" yet. Once they did a thorough job of unit and integration testing, my job became much more manageable.

The roles of the players on an XP team are quite blurred compared with those in a traditional software development process. Thus our Tensegrent XP ("XP") philosophy is "*specialization is for insects*". Here are some of the tasks I perform as a tester:

- Help the customer write stories
- Help break stories into tasks and estimate time needed to complete them
- Help clarify issues for design
- Team with the customer to write acceptance tests

- Pair with the developers to develop test tools, automated test scripts, and/or test data.

Question: The whole concept of pair programming sounds weird enough. How can a tester pair with a programmer?

Answer: I'm not a Java programmer and our developers don't know the WebART scripting language, but we still pair program. The partner who is not doing the actual typing contributes by thinking strategically, spotting typos and bad habits, and even serving as a sounding board for the coder. This is a fabulous way for developers and testers to understand and work together better. It also gives the tester *much* more insight into the system being coded.

I was reluctant to pair test at first. If the developers wrote the test scripts, would I be able to understand them and maintain them? The developers weren't anxious to pair with me for testing, either. They felt too busy to spare time for acceptance testing. Then we had a project where I needed very complicated test data loaded into a Poet database for testing a security model. By pairing with a developer, I finished in at least half the time it would have taken to do it alone, and did a better job. Now developers take turns on "test support" to produce test scripts and data needed for automation, sometimes also to help define test cases if I'm having trouble understanding a story.

Once you've mustered the courage to switch to the XP fast lane, it feels fun and safe.

How do I Educate Myself About XP?

Just as you wouldn't attempt to drive a Formula One car without preparing yourself with training and practice, the XP team needs good training to start off on the right road and stay on it.

Start by reading the XP books. The first written about XP is *Extreme Programming Explained*, by Kent Beck. The other two are also essential: *Extreme Programming Installed*, by Ron Jeffries, Ann Anderson, and Chet Hendrickson; and *Planning Extreme Programming*, by Kent Beck and Martin Fowler.

You can get an overview and extra insight into XP and similar lightweight disciplines from the many XP-related websites, including:

http://www.xprogramming.com
http://www.extremeprogramming.org
http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap
http://www.martinfowler.com

When we at Tensegrent had assembled our first team of eight developers and a tester, we got together and went through *Extreme Programming Explained* and *Extreme Programming Installed* as a group, discussing each XP principle, recording our questions (many of them on testing) and deciding how we thought we would implement each principle. This took several hours but put us all on common ground and made us feel more secure in our understanding of the concepts.

Once your team has read and discussed the XP literature, it's time to get professional training. We hired Bob Martin of ObjectMentor, a consulting company with much XP expertise, for two days of intense training (see www.objectmentor.com for more information). After Bob answered all our questions, we felt much more confident about areas that had previously been difficult for us to understand, such as the planning game, automated unit testing and acceptance testing.

Don't stop there. Talk to XP experts. Look at the Wiki pages and sign up for the egroups. If no XP user group has been formed in your city, start one.

Automating Acceptance Tests

What can you automate?

According to Ron Jeffries, author of *XP Installed*, successful acceptance tests are, among other things, customer-owned and automatic. However, customer-owned does not necessarily mean customer-written. In fact, as Kent Beck points out in *Extreme Programming Explained*, customers typically can't write functional tests by themselves, which is why an XP team has a dedicated tester: to translate the customers ideas into automatic tests.

Even with a dedicated tester, though, the "automatic" criterion has given us some trouble. We automate whenever it makes sense, but like most things, it is a trade-off. When you have to climb a steep dirt road every day, a four-wheel drive vehicle is a necessity, but it's overkill if you're just cruising around the block.

For example, we haven't found a cost-effective way to automate Javascript testing (so, we just avoid using Javascript). And we're also struggling with how to automate non-Web GUI testing in an acceptable timeframe.

It costs time and money to automate tests and to maintain them once you've got 'em. Recently we had a contract for three two-week iterations with four developers and myself to develop some components of a system for a customer. While the system involved a user interface, the design of the UI itself was to be done later, outside of our project. We developed a very basic interface to be able to test the system. The system involved multiple servers, interfaces, monitors and a database. Full test automation would have been a big effort. It didn't make sense to spend the customer's tight resources on scripts that had a short life span. Still, I automated the more tedious parts of the testing so I could get the tests done in time. In addition, I needed scripts for load testing. About 40% of the testing ended up automated. For a longer project, I would prefer to automate more.

Principles of XP Functional Test Automation

To get more automation, you have to make automation pay off in the short term, and this means spending less time developing and maintaining the automated tests. Here are the principles we are using to accomplish this:

- *Drive the test automation design with a "Smoke Test"*, a broad but shallow verification of all the critical functionality.
- *Design the tests like software*, so that the automated tests do not contain any duplicate code and have the fewest possible modules.
- *Separate the test data from the test code*, so that you can deepen test coverage by just adding additional test data.

- *Make the test modules self-verifying* to tell you if they passed or failed of course, but also to incorporate the unit tests for the module.
- *Verify only the function of concern for a particular test*, not every function that may have to be performed to set up the test.
- *Verify the minimum criteria for success*. "Minimum" doesn't mean "insufficient". If it weren't good enough, it wouldn't be the minimum. Demonstrate the business value end-to-end, but don't do more than the customer needs to determine success.
- *Continually refactor the automated tests*, by combining, splitting, or adding modules, or changing module interfaces or behaviour whenever it is necessary to avoid duplication, or to make it easier to add new test cases
- *Pair program the tests*, with another tester or a programmer.
- *Design the software for testability*, such as building hooks into the application to help automate acceptance tests. Push as much functionality as possible to the backend, because it is much easier to automate tests against a backend than through a user interface. I sit in on the developers' iteration planning and quick whiteboard design sessions. If I perceive business logic getting into the front end, for example in Javascript, I challenge the wisdom of such a move.

An XP Automated Test Design

Appendix A gives an example of a lightweight test design illustrating the application of the principles we have been using successfully at Tensegrent. I'm using WebART (see the Tools section below) to create and run the scripts. However, this design approach should work with any method of automation that permits modularization of scripts. The appendix gives details on downloading both the sample scripts and WebART.

Who automates the acceptance tests?

Some sports appear to be individual, when in actuality, they involve a team. Winners of the Tour de France get all the glory, but their victory represents a team effort. Similarly, the XP team may have only one tester, but the entire team contributes to automating acceptance tests. If tools are needed to help with acceptance testing in an XP project, write stories for those tools and include them in the planning game with all the other stories. You'll probably need to budget at least a couple of weeks for creating test tools for a moderately size project.

In the early days of Tensegrent, we initiated a project for the specific purpose of developing automated test tools. This had several advantages, in addition actually producing the tools:

- *Practice with XP* writing stories, playing the planning game, estimating. This gave us confidence in our XP skills that served us future projects.
- *Practice with development technologies*. Developers could experiment with different approaches and get experience with new tools. For example, the developers investigated in advance the advantages of using a dom versus a sax parser on the XML files containing customer test data. Doing this in advance gave us more time to experiment and research technologies than we might have had later with a client project.
- *Mutual understanding*. The team tasked with producing an acceptance test driver consisted of only four members and me, so I was called on to pair program. This exercise gave me insight into how tough it is to write unit tests, write code and refactor the code. The developers gave a lot of thought to acceptance testing and we had long discussions about what the best practices would be. This is a great foundation for any XP team.

Tools

To keep the XP car humming, XP testers need a good toolbox: one containing tools designed specifically for speed, flexibility and low overhead.

I've asked several XP gurus, including Kent Beck, Ward Cunningham and Bob Martin, the following question: "What commercial tools do you use to automate acceptance testing?" Their answers were uniform: "Grow your own". Our team extensively researched this area. Our experience has been that we are able to use a third-party tool for Web application test automation, but we need homegrown tools for other purposes.

For *unit testing*, we use a framework called jUnit, which is available free from http://www.junit.org. It does an outstanding job with unit tests. Even though I am not a Java programmer, I can run the tests with jUnit's TestRunner and can even understand the test code well enough to add tests of my own. It's possible to do some functional tests with jUnit. Some XP teams use this tool for automating acceptance tests, but it can't test the user interface. We didn't find it to be a good choice for end-to-end acceptance testing.

Tools for Creating Acceptance Tests

Some XP pros such as Ward Cunningham advocate the use of spreadsheets for driving acceptance tests. We want to make it easy for the customer to write the tests, and most are comfortable with entering data in a spreadsheet. Spreadsheets can be exported to text format, so that you and/or your development team can write scripts or programs to read the spreadsheet data and feed it into the objects in the application. In the case of financial applications, the calculations and formulas your customer puts into the spreadsheet communicate to the developers how the code they produce should work.

At Tensegrent, we provide a couple of ways for documenting acceptance test cases. Usually we use a simple spreadsheet format, separating the test case data itself from the description of the test case steps, actions and expected results. We've also experimented with entering test cases in XML format which is used by an in-house test driver. We're continuing to experiment with the XML idea, but the spreadsheet format has worked well. See Appendix B for a sample acceptance test spreadsheet template.

Appendix C shows a *partial* excerpt of a sample XML file used for acceptance test cases. The test case consists of a description of the test, data and expected output, steps with actions to be performed and expected results.

Automated Testing for Web Applications

Test automation is relatively straightforward for Web applications. The challenge is creating the automated scripts quickly enough to keep pace with the rapid iterations in an XP project. This is always toughest in the early iterations. There are times that I feel like the slow old car blocking the fast lane. For that extra burst of speed, I use WebART (http://www.oclc.org/webart), an inexpensive HTTP-based tool with a powerful scripting language. WebART enables me to create modularized test scripts, creating many reusable parts in a short enough timeframe to keep up with the pace of development. Javascript testing presents a bigger obstacle. We test it manually and carefully control our Javascript libraries to minimize changes and thus the required retesting. Meanwhile, we continue to research ways of automating Javascript testing.

Our developers wrote a tool to convert test data provided by the customers in spreadsheet or XML format into a format that can be read by WebART test scripts so that we can automate Web application testing. Even small efforts like this can help you gain that competitive edge in the speedy XP environment.

Automated Testing for GUI Applications

Test automation for non-HTTP GUI applications has been more of an uphill climb. You can travel faster in a helicopter than a mountain bike, but it takes a long time to learn to fly a helicopter; they cost a lot more than a bicycle and you may not find a place to land. Similarly, the commercial GUI automated test tools we've seen require a lot of resources to learn and implement. They're budget breakers for a small shop such as ours. We searched far and wide but could not come up with a WebART equivalent in the GUI test world. JDK 1.3 comes with a robot that lets you automate testing of GUI events with Java, but it's based on the actual position of components on the screen. Scripts based on screen content and location are inflexible and expensive to maintain. We need tests that give the developers confidence to change the application, knowing that the tests will find any problems they introduce. Tests that need updating after each application change could cause us to lose the race.

We felt that the most important criteria for acceptance tests is that they be repeatable, because they have to be run for each integration. We decided to start by developing our own tool, "TestFactor-e", that will help customers and testers run manual tests consistently. It will also record the results. We plan to enhance this tool to feed the test data and actions directly into application backends in order to automate the tests. As we have only been developing web applications, this effort is on the back burner.

No matter what the system being tested, it takes time to get up to speed with automation. I plan to do manual testing in the first iteration. At the start of the second iteration, I can start automating, using the method described in Appendix A. There are times I run into a roadblock which sets me back a day or two. The solution to that is to find someone to pair with me. As the tester in an XP project, you may feel lonely at times, but remember, you aren't ever alone!

Reports

Getting feedback is one of the four XP values. Beck says that concrete feedback about the current state of the system is priceless. If you're on a long road trip, you check for road signs and landmarks that tell you how far along your route you've come. If you realize you're running behind, you skip the next stop for coffee or push the speed a bit. If you're ahead of schedule, you might detour to a more scenic road. The XP team needs a constant flow of information to steer the project, making corrections to stay in the lane. The team's continual small adjustments keep the project on course, on time and on budget. Unit tests give programmers minute-by-minute feedback. Acceptance test results provide feedback about the "Big Picture" for the customer and the development team.

Reports don't need to be fancy, just easy to read at a glance. A graph showing the number of acceptance tests written, the number currently running and the number currently succeeding should be prominently posted on the wall. You can find examples of these in the XP books. Our development team wrote tools to read result logs from both automated tests and manual tests run with "TestFactor-e". These tools produce easy-to-read detail and summary reports in HTML and chart format.

With all this feedback, you'll confidently deliver high-quality software in time to beat your competition. You'll meet the challenges of 21st century software development!

Appendix A: Lightweight Test Design

XP Automated Test Design

The sample scripts used to illustrate the test design are written with a test tool called WebART (http://www.oclc.org/webart/). Any test tool that permits modularization and parameterization of the scripts should support this design. To download a soft copy of the sample scripts, go to http://www.oclc.org/webart/samples/ and click on the "qwmain Sample Scripts" link.

The Sample Application

Our sample application is a telephone directory lookup website, http://www.qwestdex.com. This is certainly not intended as an endorsement of Qwest and we have no connection with them, it was just a handy public application with characteristics that allow us to illustrate the tests.

The Smoke Test

We will consider the critical functionality to be logging into the site and finding the businesses within a certain city and

Action	*Minimum Passing Criteria*
Go to login page	Page contains the login form
Login	Valid login name and password brings up profile page
Search for valid category in specified city	Valid search retrieves table of businesses
Logout	Page contains link to login page and home page

Table 1

category. Pretend that this is the most important story in the first iteration. Table 1 shows the basic scenario we want to test.

The Test Design

We know that there will be more functionality to test in subsequent iterations, but we will use the simplest design we can think of to accomplish these tests without duplication. Then we will refactor as necessary to accommodate the additional tests.

The modules will be Go to Login, Login, Go to Search, Search, and Logout. In figure 1 is a diagram showing how the modules are parameterized.

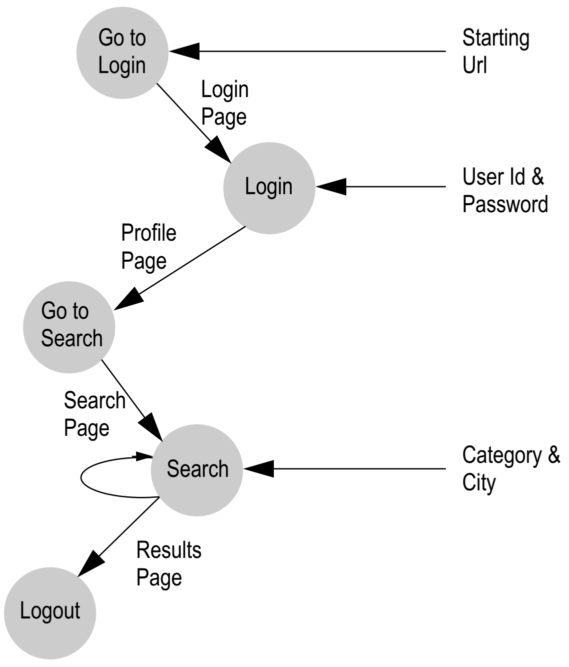


Fig. 1: The test design

Separating the test data from the code

The items on the right side of the diagram represent test data: the URL of the login page, the user id and password to use to login, and the category and city to search. The test data is segregated into a test case file, which is read in by the test when it executes. In figure 2 is sample content of that file to run a single test case.

Verification

The main modules use a set of primitive validation modules to check for the specific conditions required in a system response and determine a pass or fail condition. The validation modules in turn call *utility* modules to record the results.

This example uses the following three validation modules:

- *vtext* validates that a response contains specified text. for the text string.
- *vlink* validates that a page contains a specific link.
- *vform* validates that a page contains a specified HTML form.

Utility Modules

There are also two utility modules which are used by the main modules:

- *trace* – Displays execution tracing information in the WebART execution window, for debugging the tests.
- *log* – Records validation outcomes in a log file.

The “zslog” module in the sample scripts writes test results out in XML format. An in-house tool from Tensegrent called TestFactor-e builds an HTML page from this log file showing the results with colour-coding for pass, not run and fail. See Appendix B for an example.

Creating the Scripts

Creating the first set of scripts is the hard work. Once you have a working set of modules, you can reuse entire modules in some cases or turn them into templates in other cases. Here are the steps I use (preferably as part of a pair) to create test scripts:

1. Capture a session for the scenario I want to test. See “capqwest” in the sample scripts as an example.
2. Copy “qwmain”, “zsqwlogin” and the other supporting modules that I already have to new names. Strip out the code that was specific to that application.
3. Paste in the code specific to the scenario I want to test, copying from the captured script into the newly created “templates”. Use XP principles here: work in small increments, make sure your scripts work before you go on. For example, first see if you can get the login to work. Then add the search. Then add the logic for switching depending on the pass/fail outcome. Remember to do the simplest thing that works and add complexity only as you need it.

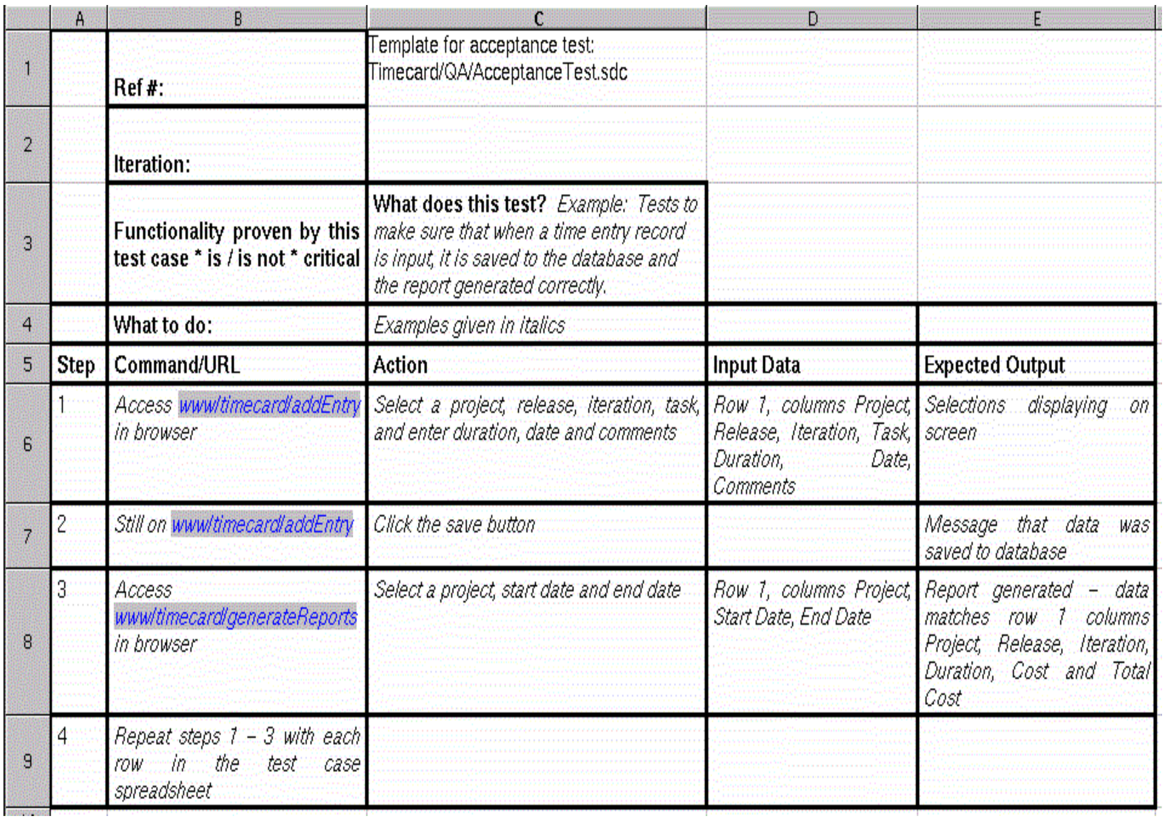
```
smoketest
   [
   :iter1:
   Url <url=http://qwestdex.com>
   UseridPassword <uid=bob&psw=bob>
   CatCity <cat=banks&city=dallas>
   ]
```

Fig. 2: Sample content of the file to run a test case

Appendix B: Partial Excerpt of XML Template for Acceptance Test Cases

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE at-test SYSTEM "at-test.dtd" [
  <!ELEMENT input ANY >
  <!ELEMENT loan-amount ANY >
  <!ELEMENT interest-rate ANY >
  <!ELEMENT term-of-loan ANY >
  <!ELEMENT output ANY >
  <!ELEMENT monthly-payment ANY >
]>
<at-test name="calc-monthly-payment" version="1.0" severity="CRITICAL">
  <at-project>mortgage-calc</at-project>
  <at-description>
    Enter loan amount, interest rate, term of loan (in months)
    to calculate monthly payment.
</at-description>
  <at-data-sets>
    <at-struct id="values">
        <input>
        <loan-amount>1000000000.00</loan-amount>
          <interest-rate>0.5</interest-rate>
          <term-of-loan>1200</term-of-loan>
        </input>
        <output>
          <monthly-payment>A big, fat wad of dough!</monthly-payment>
        </output>
          </at-struct>
    </at-data-sets>
    <at-plan>
      <at-step name="populate-loan-amount">
        <at-action>
          <at-text>Enter "{0}" in the "Loan Amount field".</at-text>
          <at-value dset="values" select="/input[2]/loan-amount"/>
        </at-action>
        <at-expect>
          <at-text>Cursor moved to "Interest Rate" field for input.</at-text>
        </at-expect>
      </at-step>
    </at-plan>
  </at-test>
```

Appendix C: Sample Acceptance Test Spreadsheet



	A	B	C	D	E
1		**Ref #:**	Template for acceptance test: Timecard/QA/AcceptanceTest.sdc		
2		**Iteration:**			
3		**Functionality proven by this test case * is / is not * critical**	**What does this test?** *Example: Tests to make sure that when a time entry record is input, it is saved to the database and the report generated correctly.*		
4		**What to do:**	*Examples given in italics*		
5	**Step**	**Command/URL**	**Action**	**Input Data**	**Expected Output**
6	1	*Access www/timecard/addEntry in browser*	*Select a project, release, iteration, task, and enter duration, date and comments*	*Row 1, columns Project, Release, Iteration, Task, Duration, Date, Comments*	*Selections displaying on screen*
7	2	*Still on www/timecard/addEntry*	*Click the save button*		*Message that data was saved to database*
8	3	*Access www/timecard/generateReports in browser*	*Select a project, start date and end date*	*Row 1, columns Project, Start Date, End Date*	*Report generated – data matches row 1 columns Project, Release, Iteration, Duration, Cost and Total Cost*
9	4	*Repeat steps 1 – 3 with each row in the test case spreadsheet*			

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 (XP) 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. The paper considers the difficulties of applying XP in a changing software requirements environment, and reports on early deployment successes, failures and discoveries, and describes how management and staff adapted during this period of change. The paper examines the benefits of a flexible management approach to XP methodology, and records the experiences of both management and staff, as initial practices matured and new practices emerged.

Keywords: Extreme Programming, Qualitative methods, Software Methodology.

1 Related Work

Previous qualitative research [Seaman 99], [Sharp et al. 99], [Cockburn/Williams 02], has concentrated on non-judgmental reporting, with the intent of provoking discussion within the culture being studied by providing observations and evidence, collaborators deciding for themselves whether any changes were required. This fieldwork study follows the format of [Gittins/Bass 02], whereby the researcher is immersed for a period in the software developer team; thereby the active researcher becomes instrumental in the development and improvement of XP. [Seaman 99] describes an empirical study that addresses the issue of communication among members of a software development organization. [Sharp et al. 99] use combined *ethnography* and *discourse analysis*, to discover implicit assumptions, values and beliefs in a software management system. [Cockburn/Williams 02] investigate *"The cost benefits of pair programming"*. [Sharp et al. 00] describe a "cross-pollination"' approach, to a deeper understanding of implicit values and beliefs.

XP developed recently from [Beck 00] and [Beck/Fowler 00], and more recently in [Jeffries et al. 00]. [Williams/Kessler 00] study lone and paired programmers, and [Williams et al. 00] the cost effectiveness of pairing.

2 The Study

Secure Trading, the focus of this paper, is a medium sized software company committed to implementing XP, and comprises a team of nine developers. Secure Trading decided to implement XP in a progressive manner, conscious of minimising disruption to the business process. Reference material from other companies, not specifically named in this paper, will only be used in general terms to highlight some typical problems facing established, and highly traditional companies,

Robert Gittins is final year PhD student at the Ada Lovelace Laboratory, University of Wales, BangorGwynedd. His research explores the interfaces between the disciplinary experts who collaborate to develop approaches to developing commercial software development solutions, especially for distributed systems. Communication between these disciplinary domains, and the cooperative solution of conflicting design problems, are the key areas of his investigation. His research goal is to contribute strategies, methods or algorithms for novel software tools to support the design process. <r.g.gittins@informatics.bangor.ac.uk>

Sian Hope is a senior lecturer at the School of Informatics, University of Wales, Bangor. Research interests are focused on contributing to enhancement of the discipline of Software Engineering by researching into practically applicable development methods which are firmly grounded on scientifically sound concepts. Fundamental approaches are sought, which provide a bridge between theory and practice and which are aimed at producing engineering methods, tools and metrics for all phases of software development. <sian@informatics.bangor.ac.uk>

Ifor Williams. After obtaining a degree in Computer Engineering from the University of Manchester Ifor went on to gain a PhD for research into computer architectures suitable for the efficient execution of object-oriented applications. This work resulted in the design of a machine (MUSHROOM) incorporating support for dynamic binding, object-based virtual memory, efficient garbage collection and targeted as a high-performance Smalltalk platform. Later he spent some time developing software for the medical diagnostics industry using prescribed conventional development processes before joining the rapidly changing.com world where XP proved to be invaluable. <ifor.williams@securetrading.com>

sensitive to their developer environment, and to the cost of disruption that change would incur on staff and production.

Secure Trading, had recently moved to larger offices. When research started, their involvement with XP consisted of some intermittent attempts at "pairing" developers. Their move presented opportunities for improving "pairing" proficiency, and the selective adoption of XP practices.

3 Qualitative Research Work

This research adopts some of the techniques historically developed in the Social Sciences [Gittins 02], *ethnography*, *qualitative interviews* and *discourse analyses*, an understanding of "grounded theory" was particularly important. Grounded theory can provide help in situations where little is known about a topic or problem area, or to generate new ideas in settings that have become static or stale. Developed by Barney Glaser and Anselm Strauss [Glaser/Strauss 67] in the 60s, grounded theory deals with the generation of *theory* from *data*. Researchers start with an area of interest, collect data, and allow relevant ideas to develop. Rigid pre-conceived ideas are seen to prevent the development of research. To capture relevant data, qualitative research techniques are employed [Gittins/Bass 02] that include the immersion of the researcher within the developer environment, qualitative data analyses, guided interviews, and questionnaires.

3.1 Qualitative Data

Qualitative evaluation allows the researcher to study selective issues in detail, without the pre-determined constraints of "categorised" analyses. The researcher is instrumental in the gathering of data from open-ended questions. Direct quotations are the basic source of raw materials, revealing the respondent's depth of concern. This contrasts with the statistical features of quantitative methods, recognised by their encumbrance of predetermined procedures.

3.2 Qualitative Interviews

[Patton] suggests three basic approaches to collecting qualitative data through interviews that are open-ended. The three approaches are distinguished by the extent to which the questions are standardised and predetermined, each approach having strengths and weaknesses, dependant upon the purpose of the interview:

1) *"Informal conversational"* interviews, are a spontaneous flow of questions where the subject may not realise that the questions are being monitored. 2) The *"General interview guide"* approach, adopted extensively for this study, predetermines a set of issues to be explored. 3) The *"Standardised open-ended interview"* pursues the subject through a set of fixed questions that may be used on a number of occasions, with different subjects.

In a series of interviews, data was collected using "Informal conversation" and verbatim transcripts taken from "General guided interviews".

3.3 Questionnaires

In an extensive questionnaire consideration was given to the "Rules and Practices" of XP. Questions targeted the software development process, XP practices, and both managerial and behavioural effectiveness. Behavioural questions were based upon Herzberg's "Hygiene and Motivation Factors" [Herzberg 74]. Ample provision was provided for open comments on each of the topics, and a developer floor plan provided for a respondent to suggest improvements to the work area. Repeating the questionnaire at three monthly intervals will help research and management by matching the maturing XP practices, as they progress, against developer responses.

4 Rules and Practices

4.1 Pair Programming

(See [Beck 00], [Beck/Fowler 00]) XP advances what has been reported for some time [Cockburn/Williams 02], [Williams/Kessler 00], [Williams et al. 00]; Two programmers working together generate an increased volume of superior code, compared with the same two programmers working separately. Secure Trading management, discussed the implementation of "Pairing" with the development team, who unanimously agreed to "buy-in" to the practice. The first questionnaire showed some of the team were unhappy with pairing. 28% of developers preferred to work independently, 57% didn't think they could work with everyone, and 57% stated that pair programmers should spend on average 50% of their time alone. XP practices recommend no more than 25% of a conditional 40-hour week be paired. Two developers summed up the team's early attitude to pair programming: *"I feel that pair programming can be very taxing at times, although I can see the benefits of doing it some of the time."*

"Not everyone makes an ideal pair. It only really works if the pair is reasonably evenly matched. If one person is quiet, and doesn't contribute, their presence is wasted. Also, if a person is really disorganised and doesn't work in a cooperative way, the frustration can (disturb) the other participant!"

Developers estimated that they spent approximately 30% of their time pairing, with partner changes occurring only upon task completion, changes being agreed and established *ad hoc*. Frequent partner swapping, and partner mixing, commands great merit in XP. Pairing practices matured with the introduction of a team "Coach" and later a "Tracker" [Beck 00]. Maintenance tasks were another problem which routinely disrupted pairing. Here control was reviewed and tasks better ordered to minimise this problem. In time, the impact of pairing activity upon developers will translate into evidence, returned in the periodic questionnaire reviews, and in the timeliness and quality of code produced.

4.2 Planning Games

(See [Jeffries et al. 00]) Planning games were introduced soon after pairing practices were established. The "customer" duly chooses between having more stories, requiring more time; against a shorter release, with less scope. Customers are not permitted to estimate story or task duration in XP and developers are not permitted to choose story and task priority. Where a story is too complex or uncertain to estimate, a "Spike" is created. Spike solutions provide answers to complex and risky stories. Secure Trading succeeded well in developing Planning games, utilising "Spike solutions" by logging a "spike" as a fully referenced story to quickly attack the problem, reducing a complex, inestimable story to a simple, and easily understood, group of stories. Results were very effective; "spike solutions" proved easy to develop and derived estimates for completion proved consistently accurate. It was common practice to have the essential elements of both *iteration* and *release* Planning games combined into one meeting. This practice worked for them in the context of the jobs they were planning.

4.3 Client On-site

(See [Beck 00]) Secure Trading rarely had this luxury. When required the "Client" role was undertaken by a client's representative, co-opted from the Customer services department by staff who had worked closely with the client and were able to accept that responsibility. Developer Manager: *"The inclusion of a representative from Customer services has proven to be hugely beneficial, providing immediate feedback of the system's successes and failures on a day-to-day basis."*

4.4 Communication

(See [Beck 00]) A great deal of attention is necessary in providing an XP environment in keeping with the practices to support XP. Key factors in communication are: the use of white boards, positioning and sharing of desk facilities to facilitate pair programmers, "stand-up" meetings, developers "buying-in" to the concepts of the *"rules and practices"* of XP, and "collective code ownership". Interviews and questionnaires revealed many areas of concern among developers. For example, 86% of developers disagreed that meetings were well organized; *"Agreements at meetings are not set in concrete"* and, *"Confidence is lost with meeting procedures, when agreed action or tasks are later allowed to be interpreted freely by different parties."* Management were quick to address these concerns by concentrating on the development of XP story card practices. Developers were encouraged to agree, and finalise with the client, the task description and duration estimates at timely Planning Game meetings. Story cards were fully referenced and signed by the accepting developer, thereby becoming the responsibility of the initiating developer until completion. Only the responsible creator of a Card was authorized to amend it.

The use and placement of White boards is said to be an essential supporting means of good communication in XP practices [Beck 00]. Mobile whiteboards were introduced by Secure Trading soon after pair programming practices gained momentum and used to record the story details agreed at Planning Game meetings. At one point, story cards were physically stuck to the boards in prioritised order with adjacent notes written on the board. This proved unpopular and developed into cards being retained but not stuck on the white board. Stories were written on the boards. Referenced stories contained ownership, estimation, as well as iteration and priority, which were displayed in columned format. On completion, the owner added the actual task duration. The information served to improve personal proficiency in estimation and in providing feedback towards establishing project "velocity" data, for future Planning Game meetings.

Stand-up meetings promote communication throughout the team. Secure Trading introduced this practice from day one. At ten o'clock every morning, a meeting allowed everyone to briefly state (standing promotes brevity) their work for the day, and discuss problems arising from the previous days activity. Anyone was free to comment, offer advice or volunteer co-operation. The benefits of adopting stand-up meetings were far-reaching and seen by developers and management as an effective way to broadcast activities, share knowledge and encourage collaboration amongst and between team members and management. Secure Trading meetings tended to degrade when reports migrated to topics of yesterday's activity, rather than those planned for the day. This activity persists and may remain or need to be resolved and modified as their particular brand of XP develops.

4.5 Simple Design

Beck [Beck 00] summarises simple design in *"Say everything once and only once."* However a comment by one developer interviewed revealed a common concern, *"Sometimes, it is a bit too simplistic, and issues seem to be avoided"*. XP states that it is important to produce a simple system quickly, and that "Small Releases" are necessary to gain feedback from the client. Secure Trading didn't see themselves in a position to implement this practice so early in their XP programme. XP allows companies to cherry-pick those practices they regard suitable for implementation, in the order they see fit.

4.6 Tests

Unit tests are written in XP before main code and give an early and clear understanding of what the program must do. This provides a more realistic scenario, as opposed to "after-the-code testing," that could, for many reasons, neatly match completed code. Time is saved both at the start of coding, and again at the end of development. Latent resistance to early unit testing became manifest, when the perceived closeness of a deadline loomed. This activity is perhaps the hardest to implement and requires commitment from developers. An early questionnaire revealed that 71% of Secure Trading developers regarded unit-testing practices in general to be "very poor". Developer Manager on early introduction of unit testing: *"If you already have a large complex system, it is difficult to determine to what extent testing infrastructure is to be retrospectively applied. This is the most difficult aspect in our experience. Starting from scratch it is much easier to make stories and code testable."*

4.7 Refactoring

(See [Fowler 99]). *"The process of improving the code's structure while preserving its function."* The use and reuse of old code is deemed costly, often because developers are afraid they will break the software. XP indicates that refactoring throughout the project life cycle saves time and improves quality. Refactoring reinforces simplicity by its action in keeping code clean and reducing complexity. Secure Trading had not developed refactoring activities in line with XP at that time. Many developers expressed concern with refactoring, more commonly reported by traditional companies: *"… with more people, we could spend more time refactoring and improving the quality of our existing code base."* The questionnaire revealed that 45% of developers considered refactoring sporadic or very poor.

4.8 Collective Code Ownership

(See [Beck 00], [Beck/Fowler 00]). This concept states *"Every programmer improves any code anywhere in the system at any time if they see the opportunity."* Collective code ownership has many merits: It prevents complex code entering the system, developed from the practice that anyone can look at code and simplify it. It may sound contentious, but XP Test procedures should prevent poor code entering the system. Collective Code Ownership also spreads knowledge of the system around the team. Secure Trading experienced growing pains in developing this principle, revealed by the comments of two developers: *"I have conflicting interests in collective code ownership. I think it is very good when it works, but there are times when some code I have written seems to just get worse when others have been working on it."*

"I like the idea of collective code ownership, but in practice I feel that I own, am responsible for, some bits of code." From the traditional perspective of individual ownership, it will be important to record how attitudes change, as XP practices mature.

4.9 Metaphor

A metaphor in XP is a simple shared story to encompass and explain what the application is "like", communicating a mental image, so that everyone involved can grasp the essence of the project in a term universally understood. This may seem to be a relatively easy, or lightweight, activity to adopt. However, the value of this practice was not immediately evident to developers, early difficulties developing and applying suitable metaphors were experienced and this practice was reluctantly abandoned for future consideration.

5 Companies Starting from Scratch

Long established and traditional companies, considering adopting XP, have, unlike Secure Trading, many more difficulties to overcome. They mostly comprise traditional teams of developers, who are comfortably established, working in small offices, in prohibitively cloistered environments. Management is often aware that legacy software in circulation is in the "ownership" of one or two heroic developers, at the cutting edge of their business. Some teams were reported as badly under-performing and in some circumstances management had resorted to consultants to resolve their problems with no significant success reported. Often with great reluctance, management allowed the research team to visit developer offices. Tension was evidently high. In these companies, "Risks" [Beck 00] are high, quality is compromised, communication difficult, and control largely ineffective. There are other considerations when starting from scratch; The Secure Trading developer manager reflecting upon attempts at implementing XP in his early projects stated: *"One of the key "discoveries" has been the relative ease to which XP has been employed on an all-new project, and the difficulty in applying XP retrospectively on an established system."*

6 Conclusions

A combination of qualitative and quantitative methods has helped identify uncertainties in applying XP practices in a medium sized software development company. How particularly one company interpreted and developed their *brand* of XP, moulded from their successes and failures. Successes in such areas as the use and development of "spike solutions", and Customer role-play within "Planning Game" activity, and from failures, as in developer reluctance to "buying-in" to "collective code ownership", and the difficulties of implementing the practice of "simple design", and in the use of "metaphors". Partial success was seen in "Pair programming", that having posed early problems, showed improvement in maturity. Future work will monitor the complex factors in the development of XP within small and growing companies at various levels of maturity. By acknowledging the characteristic unsharp boundaries of qualitative data sets, future work will investigate the use of fuzzy logic for data analyses.

Acknowledgement

This paper acknowledges the funding and support of the EPSRC (Award No. 99300131).

References

[Beck 00]
K. Beck: "Extreme Programming Explained: Embrace change". Addison Wesley. 2000

[Beck/Fowler 00]
K. Beck and M. Fowler: "Planning Extreme Programming". Addison Wesley 2000.

[Cockburn/Williams 02]
A. Cockburn and L. Williams: "The cost benefits of pairprogramming".
http://members.aol.com/humansandt/papers/pairprogrammingcostbene/pairprogrammingcostbene.htm.

[Fowler 99]
M. Fowler: "Refactoring: Improving the design of existing code", Addison Wesley. July 1999.

[Gittins 02]
R. G. Gittins: "Qualitative Research: An investigation into methods and concepts in qualitative research". Technical Paper: via http://www.sesi.informatics.bangor.ac.uk/english/home/research/technical-reports/sesi-020.htm

[Gittins/Bass 02]
R. G. Gittins and M. J. Bass: "Qualitative Research Fieldwork: An empirical study of software development in a small company, using guided interview techniques", Technical Paper: via http://www.sesi.informatics.bangor.ac.uk/english/home/research/technical-reports/sesi-021.htm

[Glaser/Strauss 67]
B. G. Glaser and A. L. Strauss: “The discovery of grounded theory: strategies of qualitative research” Chicago: Aldine Publications. 1967

[Herzberg 74]
F. Herzberg: “Work and the Nature of Man”, Granada Publications Ltd.1974

[Jeffries et al. 00]
R. Jeffries, A. Anderson and C. Hendrickson: “Extreme Programming Installed”. Addison Wesley 2000.

[Patton]
M. Q. Patton: “Qualitative Evaluation and Research Methods” (2nd Edit.). SAGE Publications

[Seaman 99]
C. B. Seaman: “Qualitative methods in empirical studies of software engineering”, IEEE Transactions on Software Engineering, Vol.25 (4):557–572 Jul/Aug 99.

[Sharp et al. 99]
H. Sharp, M. Woodman, F. Hovenden and H. Robinson: “The role of ‘culture’ in successful software process improvement.” EUROMICRO:1999 Vol.2, p17.

[Sharp et al. 00]
IEEE Computer Society. H. Sharp, H. Robinson and M. Woodman: “Software Engineering: Community and Culture”. IEEE Software, Vol. 17, No.1, Jan /Feb2000

[Williams/Kessler 00]
L. A. Williams and R. R. Kessler: “All I Really Wanted to Know About Pair Programming I Learned in Kindergarten”. Communications of the ACM. May 2000 Vol.43, No5.

[Williams et al. 00]
L. A. Williams, R. R. Kessler, W. Cunningham and R. Jeffries: “Strengthening the Case for PairProgramming”. IEEE Software, Vol. 17, No. 4: July/August,2000,pp19–25.

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

Luis Fernández Sanz received a degree in informatics engineering from Technical University of Madrid 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 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>

1 First contact

The first time I came across the term EXtreme Programming, 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 sports[2] 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 being narrow 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 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 was[3], 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 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 experi-

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

ments 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 field[4]. 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 possible[5]) 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, concentrating 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 software[6]. 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. 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

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".
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.

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.

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. 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 requirements. The client sets the goals of the development project from a business point of view. The client will only pay

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@acm.org>

Stefan Roock 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 Roock 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. <roock@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. <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. <zuelligh@informatik.uni-hamburg.de>

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
Roock	Preparation of interview guideline	Interviews	E-mail interview guideline to team
Wolf, Lippert,	Interview users at pilot customer	First understanding of application domain	Interview protocols on the project server
...	...	...	...
Roock	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, training 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

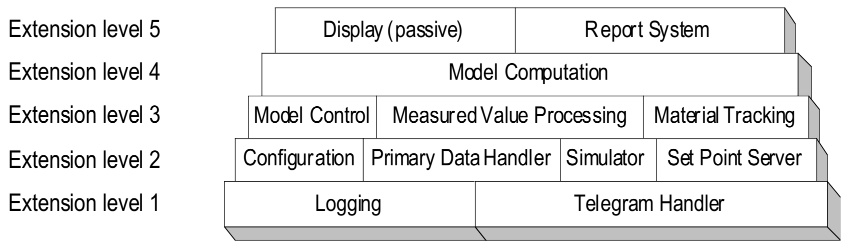


Figure 3: Example core system with extension levels

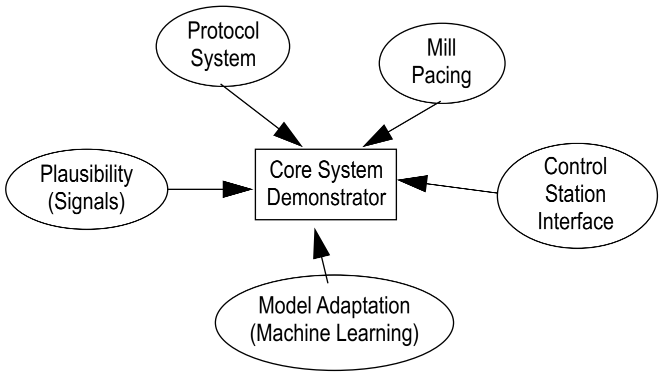


Figure 4: Example core system with specialized systems

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

Informatik und Recht

Forensics

Ursula Sury

Begriffliches

Ein neues Schlagwort ist aufgetaucht und immer mehr in aller Munde: Forensics

Was ist damit gemeint?

Forensics heisst "gerichtlich" oder "gerichtliches" und kommt vom lateinischen Wort Forum, wahrscheinlich von Forum romanum. Das Forum romanum war nebst Marktplatz auch der Platz, an dem Gericht gehalten wurde. Heute gilt Forum als eine (veraltete) Bezeichnung für Gerichtshof.

Das Wort "forensisch" wird vor allem in Bezug auf spezifisches Expertenwissen anderer Disziplinen, deren Kenntnisse für die Urteilsfindung relevant sind, verwendet. So spricht man von forensischer Medizin, der Gerichtsmedizin, die Fragen klärt, welches die Todesursache ist, wann der Tod eingetreten ist, ob die Leiche nach Todeseintritt noch bewegt wurde, etc. Im Bereich der forensischen Psychiatrie geht es um ähnliche Fragestellungen, wie, ob der Täter zur Zeit der Tat zurechnungsfähig war, der Täter therapiefähig ist, usw.

Im Bereich der IT sind mit Forensics die Fragen verbunden, wie man Sachverhalte, die sich vorwiegend mittels elektronischer Hilfsmittel abspielen, nachvollziehen und vor Gericht beweisen kann. Auch hier sind, wie bei forensischer Medizin oder forensischer Psychiatrie, wieder nebst der Rechtswissenschaft andere Disziplinen gefragt, wie insbesondere die Technik oder die Betriebswirtschaftslehre, speziell Personalmanagement, Führung und Organisation. In diesem Sinne kann man Forensics durchaus auch als einen Teilbereich der Security verstehen.

Forensics im Strafrecht

Im Strafrecht gibt es verschiedene Artikel, welche Daten oder die Leistung, welche eine Datenverarbeitungsanlage erbringt, zum schützenswerten Rechtsgut erklären und deren Verletzung mit Strafe ahnden. Dazu gehören beispielsweise das unbefugte Eindringen in ein Datenverarbeitungssystem (StGB Art. 143bis), die Datenbeschädigung (StGB Art. 144bis), der betrügerische Missbrauch einer Datenverarbeitungsanlage (StGB Art. 147), das Erschleichen einer Leistung (auch einer Leistung, die eine Datenverarbeitungslage erbringt, StGB Art. 150), Herstellung und in Verkehr bringen von Materialien zur unbefugten Entschlüsselung codierter Angebote (StGB Art. 150b).

Bei diesen Straftatbeständen geschieht die Straftat im elektronischen Bereich, in Programmen, Datenbanken usw. Ob überhaupt eine Straftat begangen wurde und ob diese, falls sie begangen wurde, dem Täter nachgewiesen werden kann, sind Fragen, mit welchen sich Forensics beschäftigt. Dies ist eine grosse Herausforderung für die Strafuntersuchungsbehörde, denn selbstverständlich ist es viel einfacher, festzustellen und auch nachzuweisen, dass in ein Bijouteriegeschäft eingebrochen wurde, als dass eine Datenbank gehackt wurde. An dieser Stelle muss erwähnt werden, dass gewisse der oben genannten Straftatbestände nur dann erfüllt sein können, wenn die Daten gegen unbefugten Zugriff besonders gesichert sind, so in StGB Art. 143, 143bis und 147.

Selbstverständlich gibt es auch viele andere Straftatbestände, die mit elektronischen Hilfsmitteln begangen werden können. Dazu gehören beispielsweise Pornographie, Rassismus, Nötigung, Erpressung, Urkundenfälschung etc. Hier ist das Schutzobjekt nicht ein IT-Vermögenswert wie Daten, sondern es handelt sich um traditionelle Straftatbestände, die aber häufig mittels dieser neuen Medien begangen werden. Was die Urkundenfälschung anbelangt, so muss noch erwähnt werden, dass nach Strafgesetzbuch auch elektronische Urkunden gefälscht werden können, sagt doch StGB Art. 110, 5. dass die Aufzeichnung auf Bild- und Datenträgern der Schriftform gleichgestellt ist, sofern sie demselben Zweck dient.

Forensics im Zivilrecht

Analoge Fragen wie im Strafrecht stellen sich auch im Zivilrecht. So hat hier zwar nicht der Staat, aber doch der Geschädigte dem Schädiger nachzuweisen, dass er einen Schaden zugefügt hat. Im ausservertraglichen Bereich, beispielsweise, wenn ein ehemaliger Mitarbeiter oder die Konkurrenz mittels Hacking Daten zerstört. Auch bei Vertragsverletzungen hat der Geschädigte dem ihn schädigenden Vertragspartner zu beweisen, dass er eine Vertragsverletzung begangen hat und er genau wegen dieser Vertragsverletzung zu Schaden gekommen ist. Mit der Digitalisierung unserer Gesellschaft, dem Abbilden und Unterstützen unternehmerischer Prozesse mit IT-Hilfsmitteln und dem Aufkommen von B2B und B2C müssen immer mehr Sachverhalte im elektronischen Bereich nachgewiesen werden, was verschiedene Vorkehrungen technischer und organisatorischer Art bedingt.

Fälschung oder Verfälschung?

Werden Sachverhalte mittels elektronischer Beweismittel, seien dies Urkunden oder

Ursula Sury ist selbständige Rechtsanwältin in Luzern und leitet den Fachhochschul-Lehrgang Wirtschaftsinformatik an der Hochschule für Wirtschaft HSW Luzern der Fachhochschule Zentralschweiz und ist Dozentin für Informatikrecht an verschiedenen Nachdiplomstudien, welche am Institut für Wirtschaftsinformatik der Hochschule durchgeführt werden. Die Autorin ist hauptsächlich im Bereich Informatikrecht, Datenschutz und Franchising tätig.

Augenscheinobjekte, nachgewiesen, stellt sich immer sofort die Frage der Echtheit. Im elektronischen Bereich sind Fälschungen oder Verfälschungen mit relativ geringem Aufwand und ohne leichten Nachweis vorzunehmen. Es ist schwieriger, einen auf Papier vorhandenen von Hand geschriebenen Vertrag zu fälschen oder zu verfälschen, als ein nur elektronisch vorhandenes Dokument. Der Angeklagte im Strafrecht oder der Beklagte im Zivilrecht wird denn auch sofort vorbringen, dass Fälschungen vorliegen. Wer einen Sachverhalt, sei dies im Strafprozess oder im Zivilprozess, liquid beweisen will, muss also nebst dem Beweis auch noch darlegen, dass dieser echt ist, also dass er nicht ge- oder verfälscht wurde. Denn nur aufgrund so validierter und somit valider Beweismitteln kann ein Richter ein Urteil aussprechen. Mit der Einführung der digitalen Signatur und der hoffentlich zukünftigen Standardisierung von Verschlüsselungstechniken sollte der Nachweis der Nichtverfälschbarkeit auch einfacher werden.

Der Nachweis der Echtheit wird aber nur mit technischen und organisatorischen Hilfsmitteln möglich sein, also mit forensischer Technik und forensischen Managementfragen. Im organisatorischen Bereich gehört dazu beispielsweise die sofortige Hinterlegung des elektronischen Beweismittels bei einem Notar, damit bewiesen werden kann, dass zwischenzeitlich, d.h. seit der Hinterlegung, keine Manipulationen vorgenommen wurden resp. werden konnten. Für den Alltag gehört sicher auch ein Backup-Verfahren mit Auslagerung bei Dritten dazu, das hilft, die Chronologie von Abläufen zu beweisen. Im technischen Bereich gehören die verschiedenen Verfahren der Protokollierung, beispielsweise von Logfiles, der Sichtbarmachung spezieller Speicherbereiche etc. dazu.

Illegale Sicherheiten

Unternehmungen, welche ein Maximum an Vorkehrungen treffen, um die Nachvollziehbarkeit und Beweisbarkeit von Sachverhalten in elektronischer Form sicherzustellen, laufen Gefahr, die gesetzlich geschützten Freiheiten ihrer Mitarbeiter, Kunden, Konkurrenten etc. zu verletzen. Speziell geht es um den Datenschutz, d.h. das Recht auf informationelle Selbstbestimmung, welches mit Protokollierungen von sämtlichen Bewegungen, Scanning von Adressen, der Aufbewahrung von Daten etc. häufig verletzt wird. Ein perfektes Einhalten der Ansprüche von Forensics verletzt somit den Datenschutz. Es gilt hier, einen Mittelweg zu finden. Nach wie vor ist es besser, mittels Prophylaxe im technischen Bereich und Averness möglichst viele Rechtsverletzungen zu verhindern, damit es gar nie zu forensischen, also gerichtlichen Fragen kommen kann.

Fazit

Forensics

- bedeutet gerichtliche Fragen der IT
- ist im Strafrecht und im Zivilrecht relevant
- beschäftigt sich v.a. mit der Nachvollziehbarkeit und Beweisbarkeit von Sachverhalten

Vergleichen Sie dazu auch in der Zeitschrift der schweizerischen Informatikorganisationen "Aufbewahrung von Unterlagen in elektronischer Form?" U. Sury, 1/2001 sowie "Vertragsabschluss und digitale Signatur" U. Sury, 2/2000.

Europäische Harmonisierung der Studienabschlüsse

Neuer Bachelor und Master in Wirtschaftsinformatik an der Universität Fribourg

Die Wirtschafts- und Sozialwissenschaftliche Fakultät der Universität Fribourg stellt ihre bisherigen Studiengänge auf das Bachelor- und Mastersystem um. Gleichzeitig werden ab Herbst 2002 zwei neue Abschlüsse angeboten, nämlich ein "Bachelor of Arts in Information Systems" und ein "Master of Arts in Information Management".

Mit der Unterzeichnung der Bologna-Deklaration im Juni 1999 hat der Bund die politische Absicht bekundet, die Ziele der europäischen Reform zur Einführung von Bachelor- und Masterstudiengängen mitzutragen und in der Schweiz umzusetzen. Die wichtigsten Anliegen dieser Erklärung sind die Harmonisierung der Studienstrukturen in Europa, die Verbesserung der Studienanerkennung, die Intensivierung der Mobilität und die Förderung der europäischen Zusammenarbeit in Fragen der Qualitätssicherung. Nicht zuletzt soll damit auch die Wettbewerbsfähigkeit der europäischen Universitäten im internationalen Umfeld erhalten bleiben resp. verbessert werden.

An der Universität Fribourg stellen die Mathematisch-Naturwissenschaftliche, die Rechtswissenschaftliche und die Wirtschafts- und Sozialwissenschaftliche Fakultät einige Studiengänge auf das Bachelor- und Mastersystem um. Nach einem dreijährigen Studium wird neu der Bachelorabschluss angeboten, bevor ein ein- bis zweijähriges Masterstudium in Angriff genommen werden kann. Im Gegensatz zum Bachelorstudiengang ist der Master mehr auf die aktuelle Forschung ausgerichtet und soll unter anderem den Einstieg in ein Doktoratsstudium ermöglichen.

An der Universität Fribourg sind zwei neuartige Studiengänge in Wirtschaftsinformatik konzipiert worden, ein "Bachelor of Arts in Information Systems" und ein "Master of Arts in Information Management":

Der Bachelorstudiengang vermittelt den Studierenden die Grundlagen in Wirtschaftsinformatik und Betriebswirtschaft, neben quantitativen Fächern und Methodenwissen. Themenschwerpunkte in Wirtschaftsinformatik sind auf dieser Stufe: Programmiersprachen und Softwareentwicklung, Projektmanagement, Informationssysteme im Web, Datenbanksysteme und Entscheidungsunterstützung, Sicherheit, Informations- und Telekommunikationsmanagement. Ein Novum bildet die teambasierte Sprachausbildung in Deutsch resp. Französisch sowie in Fachenglisch. Damit soll das Sprachvokabular und die Sprachkompetenz für die Wirtschaftswissenschaften vermittelt und die wissenschaftliche Diskussion gefördert werden.

Das Masterstudium in Wirtschaftsinformatik ist eine anspruchsvolle Vertiefungsrichtung. Da an der Universität Fribourg die Themen Informations- und Telekommunikationsmanagement, Supply Chain Management und Electronic Business einen hohen Stellenwert geniessen, bilden sie den Schwerpunkt im obligatorischen Teil. Der fakultative Teil kann aus dem breiten Angebot der Wirtschaftsinformatik, der Betriebs- und Volkswirtschaft und der Informatik ausgewählt werden. Neben der Vertiefung der wissenschaftlichen Methoden müssen die Studierenden auch ein dreimonatiges Praktikum absolvieren. Dieses wird als Projekt konzipiert und von einer Forschungseinheit begleitet.

Der Bachelor in Information Systems eröffnet den Absolventen anspruchsvolle Tätigkeiten bei der Nutzung von Informations- und

Kommunikationssystemen. Mit dem anschliessenden Master in Information Management wird die Informatikkompetenz mit einer Führungsqualifikation ergänzt und das wissenschaftliche Arbeiten im Team erprobt. Die Seminar- und Studienarbeiten sowie das obligatorische Praktikum werden in enger Zusammenarbeit mit Firmen und Organisationen vergeben.

Die Wirtschaftsinformatik ist ein junges und dynamisches Fachgebiet, das sich mit dem raschen Wandel in der Informationstechnologie und in den Wirtschaftswissenschaften weiterentwickelt. Acht Professoren der Wirtschaftsinformatik resp. der Informatik und über dreissig Assistierende helfen mit, die Studiengänge attraktiv und das Betreuungsverhältnis persönlich zu gestalten. Dank der Zweisprachigkeit der Region Fribourg ist es möglich, Fächer sowohl in Deutsch wie in Französisch zu belegen und damit einen zweisprachigen Bachelor oder Master zu erlangen. Die Universität Fribourg ist die einzige in Westeuropa, die einen solchen Doppelabschluss anbieten kann.

Andreas Meier, Fribourg, 14.1.02

Architekten der Informationsgesellschaft

Interview mit ***Andreas Meier***, Professor für Wirtschaftsinformatik und Vizedekan der Wirtschafts- und Sozialwissenschaftlichen Fakultät der Universität Fribourg.

Meier zeigt auf, weshalb ein Studium in Wirtschafts-informatik an der Universität Fribourg attraktiv ist. Dabei erwähnt er, dass der Übungsteil während des Studiums praxisbezogen ist und dass das Masterstudium ein obligatorisches Praktikum beinhaltet. So finden die Studierenden schon früh Kontakt zu einem möglichen Arbeitgeber.

Frage: Weshalb empfehlen Sie jungen Leuten ein Studium in Wirtschaftsinformatik? Genügt es nicht, wenn man den PC gebrauchen kann?

Andreas Meier: Nein, im heutigen Wirtschaftsleben genügt das längst nicht mehr! Es braucht qualifizierte Leute, die über den PC hinaus verstehen, was die Anforderungen an ein Informationssystem sind und wie ein solches System für bessere Entscheidungen genutzt werden kann. Insbesondere kennen die Wirtschaftsinformatiker und Wirtschaftsinformatikerinnen das Potenzial des Internet, sei es für firmeninterne Abläufe und für firmenübergreifende Geschäftsprozesse.

Frage: Wie sieht das Wirtschaftsinformatikstudium an der Universität in Fribourg aus? Und welche Kompetenzen kann man sich während des Bachelor- und Masterstudiums aneignen?

Andreas Meier: Im Bachelorstudium werden die Grundlagen der Betriebswirtschaft vermittelt, neben einer fundierten Einführung in die Informatik. Das Masterstudium hat die Schwerpunkte Informations- und Kommunikationsmanagement, Supply Chain Management sowie Electronic Business. Neben Fachkompetenz in Wirtschaftsinformatik vermitteln wir den Studierenden Methoden- und Sozialkompetenz.

Frage: Können Sie an einem Beispiel illustrieren, wie Sie die Studierenden über das Fachwissen hinaus fördern?

Andreas Meier: Zukunftsgerichtete Informatiklösungen können nur im Team entwickelt werden. Deshalb organisieren wir den Übungsteil vorwiegend in Gruppen, mit praxisbezogenen Fallstudien. Zudem müssen die Studierenden beim Masterstudium ein Praktikum auf dem Gebiet ihrer Vertiefungsrichtung absolvieren. Im Grundlagenteil für Wirtschaftsinformatik beispielsweise realisieren drei bis fünf Studenten einen Webauftritt für ein Fribourger Reisebüro. Diese mehrwöchige Arbeit umfasst die Recherche im Internet, den Entwurf einer Website, die Implementierung und das Einbinden von internationalen Datenbanken aus der Reisebranche.

Frage: Welchen Nutzen ziehen die Studierenden aus diesen teamorientierten Arbeiten?

Andreas Meier: Neben der fachlichen Herausforderung lernen sie frühzeitig, welches Potenzial in der Teamarbeit steckt. Natürlich müssen dabei Konflikte in der Zusammenarbeit überwunden werden. Damit das Projektziel nicht gefährdet wird, geben wir Unterstützung bezüglich der Planung und Organisation von Gruppenarbeit. Projektmanagement begreift man erst, wenn man im Team eine herausfordernde Aufgabe angeht.

Frage: Finden diplomierte Wirtschaftsinformatiker attraktive Jobs in der Praxis?

Andreas Meier: Bei den Studien-, Bachelor- und Masterprojekten arbeiten wir mit namhaften Firmen und Organisationen zusammen. Solche Arbeiten bilden ein Sprungbrett in die Praxis. Zudem sind rechnergestützte Informationssysteme heute im Wirtschaftsleben nicht mehr wegzudenken. Ein Absolvent kann deshalb unter verschiedenen Angeboten, teils verbunden mit einem Auslandaufenthalt, aussuchen.

Frage: Weshalb ist Fribourg für ein Studium in Wirtschaftsinformatik geeignet?

Andreas Meier: Das Departement für Informatik wurde bereits in den fünfziger Jahren gegründet und verfügt über eine grosse Tradition. Acht Professoren mit ihren Forschungsgruppen arbeiten auf dem Gebiet Sicherheit und Kryptographie, interaktive Systeme, Datenbanken, Internet und rechnergestütztes Lernen, Electronic Business und entscheidungs-unterstützende Systeme. Als fakultätsübergreifendes Departement pflegen wir innerhalb und ausserhalb der Universität Kontakte. Und nicht zu vergessen: We speak German, French and Java!

Über das Wirtschaftsinformatikstudium an der Universität Fribourg erhalten Sie nähere Auskünfte beim Departement für Informatik, Rue Faucigny 2, 1700 Fribourg, Tel. 026 300 83 22, e-Mail iiuf-secr-rm@unifr.ch, Website http://www.unifr.ch/economics.

IT-Security darf nichts kosten!

Björn Kanebog

In vielen Firmen in welchen die IT nicht zum Kerngeschäft gehört wird das IT-Budget vielfach sehr stark wenn nicht gar ganz gekürzt. Der Bereich IT-Security darf dann schon gar nichts kosten. Oft installieren Firmen ein Tool, z.B ein Antivirenprogramm und denken dass dann schon genug für Security gemacht wurde. An regelmässige Updates der Signaturdateien oder Schulung der Mitarbeiter oder gar an ein Antivirenkonzept, welches das Thema Viren und die korrekte Handhabung dieser Thematik beschreibt wird in den seltensten Fällen gedacht. Eben, IT-Security darf nichts kosten.

Auch wenn genügend Budget für IT-Security vorhanden ist; so wird dieser Betrag dann meistens nur für irgendwelche Security-Tools ausgegeben. In vielen Köpfen hält sich hartnäckig der Glaube dass ein Tool (Antivirenprogramm, Firewall, etc.) alleine des Rätsels Lösung ist. Die ganze Organisation, der ganze Bereich IT-Security Management, die personellen und organisatorischen Aspekte werden oft gar nicht berücksichtigt. Nur, ohne die Mitarbeit der Menschen in einer Firma gibt es keine Security

IT-Security wird in vielen Firmen so gehandhabt als würden bei ihrem Firmengebäude sämtliche Türen und Fenster 24 Stunden während 365 Tagen offen gelassen und es würde keinen interessieren ob und wer in das Firmengebäude hinein oder herausgeht und was dort eigentlich gemacht wird.

Wie sieht die allgemeine Lage aus? Durch die Anbindung an das Internet ergeben sich für den Kunden und für die jeweilige Firma diverse neue Möglichkeiten wie z.B. Online-Shopping, Electronic Banking, Firmenportrait und Produkteangebot im Internet, Produktesupport im Internet etc. Vieles wird dank dem "Internet" schneller und komfortabler.

Jede Medaille hat aber auch eine Kehrseite: der Einsatz des "Internets" bringt nicht nur Vorteile und neue Möglichkeiten sondern bringt auch neue (elektronisch und automatisch durchführbare) Gefahren wie Denial of Service (DoS), Daten-Diebstahl, -Modifikation und Zerstörung mit sich.

Aber Firmenwerte müssen geschützt werden. Grundsätzlich muss jede Firma ihre Firmenwerte schützen; denn Informationen sind in allen Firmen ein sehr wertvolles Gut. Darunter fallen z.B. Firmenstrategien, Geschäftszahlen, Erfindungen, Kundeninformationen etc.

Auch das Environment einer Firma, sprich das Gebäude und auch alle sich darin befindenden Personen- und Sachen müssen geschützt werden.

Ebenso darf nicht vergessen werden, dass jede Firma sehr daran interessiert sein muss, ihr Image zu erhöhen um damit u.a. den Bekanntheitsgrad ihres Firmennamens ihrer Produkte und Dienstleistungen zu erhöhen; was schlussendlich in einem höheren Umsatz für die Firmen resultieren soll. Einen "guten Namen" zu erreichen dauert sehr lange und kostet viel Aufwand. Ihn zu zerstören geht jedoch sehr schnell und kann mit wenig Aufwand erreicht werden.

Bei den obgenannten zu schützenden Firmenwerten spielt es also grundsätzlich keine Rolle ob eine Firma IT einsetzt oder nicht. Vor Jahrzehnten oder gar vor Jahrhunderten schon wurden die obgenannten Werte angegriffen. Der Einsatz von IT und die Anbindung einer Firma ans Internet erhöhen die Möglichkeiten eines Angreifers erheblich. Ebenso werden die Angriffsversuche jetzt aus grosser Distanz (über Kontinente) und automatisierbar möglich. Musste früher jemand um eine Bank um ihr Geld zu erleichtern bewaffnet in das jeweilige Bankgebäude eindringen; so könnte heutzutage ein "Bankräuber" eine nicht IT-mässig gesicherte Bank bequem von zu Hause um ihr Geld erleichtern. Beispiele von Angriffen auf die zu schützenden Firmenwerte gibt es zuhauf:

Angriffe auf Informationen (ECHELON)

"In dem vorläufigen Bericht des nichtständigen Ausschusses des Europäischen Parlaments wird bestätigt, dass es das globale Lauschsystem Echelon gibt, das von den Geheimdiensten der USA, Kanadas, Grossbritanniens, Australiens und Neuseelands betrieben wird, aber wahrscheinlich nicht mehr unter diesem Namen läuft. Gesagt wird allerdings auch, dass dessen Kapazitäten überschätzt wurden, vor allem was das Abhören der Internetkommunikation angeht, und dass man keinen Beweis für Wirtschaftsspionage gefunden habe. Duncan Campbell, dessen Berichte die Diskussion über Echelon ausgelöst haben, weist allerdings in den jetzt exklusiv von Telepolis veröffentlichten Dokumenten darauf hin, dass es dafür unübersehbare Hinweise gibt".

In den letzten Jahren wurden jedoch Hinweise auf diverse Patentanmeldungen der NSA veröffentlicht, die darauf schliessen lassen, dass hier erheblich mehr getan wird, als allgemein befürchtet.

Angriffe auf das Environment (Computervirus Code Red)

"Das US-Verteidigungsministerium hat den öffentlichen Zugang zu ihren Websites am Montag (23.07.2001) zur Abwehr des Computerviruses namens Code Red vorübergehend geschlossen. Oberstleutnant Catherine Abbott erklärt, dass die Web-Sites erst wieder zugänglich sein werden, wenn keine Gefahr mehr durch den Virus besteht".

Angriffe auf das Image (Hacker knacken Daten der Teilnehmer des WEF in Davos)

"Hacker haben vertrauliche Daten (Kreditkartennummern, Passwörter, private Telefon- und Handy-Nummern von Teilnehmern des Weltwirtschaftsforums (WEF) in Davos geknackt. Die Hacker hatten einen WEF-Server geknackt, auf dem die vertraulichen Daten der Teilnehmer des Treffens gespeichert waren".

Situation der Firmen bezüglich Security

Geeignete und wirkungsvolle Schutzmassnahmen erfordern einen sehr hohen Aufwand sowohl technisch, personell sowie auch organisatorisch. Dazu wird auch sehr viel Know How benötigt. Eine Firma muss um wirkungsvolle Schutzmassnahmen zu implementieren, zu unterhalten und auf dem neuesten Stand zu halten hochqualifizierte Security-Spezialisten beschäftigen. Security kann nicht mit einmaligem Installieren eines Tools als erledigt betrachtet werden. Es braucht vielmehr auch einen Paradigmawechsel in einer Firma welcher bei der Firmenleitung anfangen muss und sich dann durch die ganze Firma hindurchzieht. Das IT-Sicherheits-Management ist in diesem Zusammenhang mit einem nicht zu unterschätzenden Aufwand verbunden.

In der Presse konnte man vom Vorhaben "Project Anlehnet lesen. Dieses Projekt wurde aufgesetzt um u.a. die Anzahl der Zugriffsversuche, Attacken und "exploits" auf dieses Netzwerk mit seinen diversen Computersystemen festzustellen. In diesem Netz-

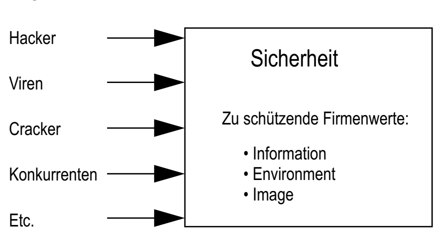


Bild 1

werk waren weder wichtige Produktionssysteme, noch wurden Hacker dazu ermuntert dieses Netzwerk anzugreifen. Die Anzahl und die Art aller dieser unerlaubten Angriffe auf dieses Netzwerk zeigt, dass wenn wir im Internet präsent sind, sehr vielen Angriffs-Gefahren ausgesetzt sind. Speziell gilt dies natürlich für Firmen welche wichtige Produktionssysteme und Informationen in ihrem Netzwerk haben. Die neuesten Aktivitäten wie z.B. der Wurm "Code Red" belegen das wieder einmal sehr deutlich.

Outsourcing-Kompetenz

Ein Outsourcing-Unternehmen kann den obgenannten technischen, personellen und organisatorischen Aufwand im Bereich Security auf mehrere Kunden verteilen. Für die Einrichtung und Pflege werden nicht einzelne Mitarbeiter, sondern Teams eingesetzt. Mit den Hardware- und Software- Lieferanten werden üblicherweise Wartungsverträge geschlossen. So kann den Kunden professionelle Sicherheit zu einem vergleichsweise günstigen Preis angeboten werden. Der Sicherheitsstandard beim Outsourcing ist in der Regel deutlich höher, als einzelne Firmen es sich sonst leisten könnten oder würden.

IT-Sicherheitsmassnahmen im Outsourcing

Für den Kunden wird individuell vom Outsourcing-Provider ein Lösungsvorschlag ausgearbeitet. Der Outsourcing-Provider sollte mit dem Kunden bei den folgenden IT-Sicherheitsmassnahmen definieren ob diese vom Kunden oder vom Outsourcing-Provider wahrgenommen werden sollen. Natürlich besteht auch die Möglichkeit sämtliche IT-Sicherheitsmassnahmen aus einer Hand zu beziehen.

Der Outsourcing-Provider sollte die folgenden IT-Sicherheitsmassnahmen in seinem Angebot haben und diese dem Kunden auch anbieten resp. den Kunden diesbezüglich beraten können:

IT-Sicherheits-Management
Benennen eines Sicherheitsbeauftragten für das IT-Sicherheits-Management.
Übergabe der aktuellen IT-Sicherheitsrichtlinien, -standards und -prozeduren sowie allfälliger Prüfunterlagen an den Outsourcing-Provider
Weiterführung der implementierten Aktivitäten soweit in der vom Kunden übergebenen Dokumentation betreffend IT-Sicherheitsrichtlinien, -standards und -prozeduren beschrieben. Dies gilt bis zu einer allfällig vereinbarten Abänderung und deren Implementierung zum vereinbarten Zeitpunkt.
Überprüfen der erhaltenen IT-Sicherheitsrichtlinien, -standards und -prozeduren und Empfehlen von Veränderungen.
Informieren der Kundenmitarbeiter über die anzuwendenden IT-Sicherheitsprozeduren, wie z.B. Anmeldeprozeduren, Prozeduren für die Verwendung von Passwörtern und den Einsatz von Anti-Virus-Programmen sowie Prozeduren zur Gewährleistung der Daten- und Gerätesicherheit.
Überprüfen der Änderungsempfehlungen des Outsourcers zu den IT-Sicherheitsrichtlinien, -standards und -prozeduren des Kunden. Der Kunde entscheidet über die Empfehlungen und vereinbart mit dem Outsourcing-Provider das weitere Vorgehen.
Erarbeiten eines Kapitels "IT-Sicherheitsmassnahmen" im Betriebshandbuch.
Pflegen des Kapitels "IT-Sicherheitsmassnahmen" im Betriebshandbuch.
Der Kunde stellt sicher, dass zu einem vom Sicherheitsbeauftragten des Outsourcing-Providers festzulegenden Zeitpunkt die von dem Outsourcing-Provider übernommenen Mitarbeiter über die erforderlichen *Benutzer Berechtigungen* verfügen. Der Outsourcing-Provider ist über alle Änderungen zu informieren.
Informieren des Outsourcing-Providers über geplante Änderungen der IT-Sicherheitsrichtlinien, -standards und -prozeduren des Kunden vor deren Implementierung.
Überprüfung, ob die tatsächlich implementierten IT-Sicherheitsmassnahmen mit den in diesem Dokument und dem Betriebshandbuch festgelegten IT-Sicherheitsmassnahmen übereinstimmen.

Physische Sicherheit
Bereitstellen physischer IT-Sicherheitsmassnahmen für die Einrichtungen des Kunden.
Bereitstellen physischer IT-Sicherheitsmassnahmen für Einrichtungen des Outsourcing-Providers.
Physische Beschränkung des Zugangs zu den unter dem Outsourcing-Provider IT-*Sicherheitskontrolle* stehenden *Datenverarbeitungsbereichen* auf autorisierte Personen. • in den Einrichtungen des Kunden • in den Einrichtungen des Outsourcing-Providers
Regelmässige Überprüfung der physischen Zutrittskontrollmassnahmen für die unter dem Outsourcing-Provider IT-*Sicherheitskontrolle* stehenden *Datenverarbeitungsbereiche* sowie die Überprüfung von Zugangsprotokollen auf ungewöhnliche Ereignisse • in den Einrichtungen des Kunden • in den Einrichtungen des Outsourcing-Providers
Physische Beschränkung des Zugangs auf autorisierte, im Betriebshandbuch namentlich festgehaltene Personen zu unter dem Outsourcing-Provider IT-*Sicherheitsverantwortung* stehenden LAN-Servern und IT-Infrastruktur in den Einrichtungen des Kunden.
Physische Beschränkung des Zugangs auf autorisierte Personen zu unter dem Outsourcing-Provider IT-*Sicherheitsverantwortung* stehenden LAN-Servern und IT-Infrastruktur in den Einrichtungen des Outsourcing-Providers.
Implementieren von Massnahmen zum Schutz der von dem Outsourcing-Provider kontrollierten Druckausgaben gegen unbefugten Zugriff.
Bereitstellen von gesicherten Bereichen für *Portable Storage Media* unter der Outsourcing-Provider IT-*Sicherheitskontrolle*.
Vor der Übernahme der IT-*Sicherheitsverantwortung* durch den Outsourcing-Provider, Durchführung einer Übernahmeinventur für alle unter dem Outsourcing-Provider IT-*Sicherheitskontrolle* stehenden *Portable Storage Media*.
Durchführen einer jährlichen Bestandsinventur und -abstimmung für die vorhandenen *Portable Storage Media* unter der Outsourcing-Provider IT-*Sicherheitskontrolle*. Informieren des zuständigen Managements beim Kunden und beim Outsourcing-Provider, wenn dabei Fehler festgestellt wurden.
Bereinigung der bei der jährlichen Bestandsinventur der *Portable Storage Media* unter der Outsourcing-Provider IT-*Sicherheitskontrolle* festgestellten Diskrepanzen und Information des Kunden.
Implementieren von Massnahmen zur wirkungsvollen Beseitigung von Restdaten auf *Portable Storage Media* vor deren Entsorgung oder deren Wiederverwendung, ausgenommen der Wiederverwendung beim Kunden.

Logische Zugriffskontrolle
Installieren und Pflegen der *Zugriffskontrollsoftware*, sofern der Outsourcing-Provider dies für notwendig erachtet. • Vorhandene Software • Neue Software (inkl. Erweiterungen)
Implementieren der Funktionen und Merkmale der *Zugriffskontrollsoftware*, in Übereinstimmung mit den im Betriebshandbuch definierten IT-Standards und –prozeduren.
Implementieren der Funktionen und Merkmale des unterstützten Betriebssystems, in Übereinstimmung mit den im Betriebshandbuch definierten IT-Standards und –prozeduren.
Festlegung der Sicherheitseinstellungen für *Betriebssystemressourcen* (OSRs) im Betriebshandbuch.
Implementieren der definierten Sicherheitseinstellungen für die *Betriebssystemressourcen* (OSRs) über die *Zugriffskontrollsoftware*.
Detaillierte Bezeichnung und Übergabe folgender Kundeninformation an den Outsourcing-Provider: • Kriterien des Kunden für die Klassifizierung und Kontrolle von Daten • *Datensicherheitsanforderungen* und deren Handhabung • Datenverschlüsselungsanforderungen
Sicherheitsanforderungen für den Zugriff auf die Applikationsressourcen über das Zugriffskontrollsystem. • Definieren • Implementieren
Sicherheitsanforderungen für die *Endbenutzerdaten* auf Servern über das *Zugriffskontrollsystem*. • Definieren • Implementieren
Bereitstellen und Unterstützen von einzusetzenden Verschlüsselungsprodukten gemäss Outsourcing-Vertrag (Hardware und/oder Software).
Verwaltung der kryptographischen Schlüssel und deren Verteilung.
Vor der Übernahme der IT-*Sicherheitsverantwortung* durch den Outsourcing-Provider, Durchführung einer Übernahmeinventur der *Benutzer Berechtigungen* für die folgenden *Ressourcen*. • *OS*, *DB* und *SAP*-Basis • Applikation • LAN
Definition von *Benutzer Identifikationen* sowie Benutzer Berechtigungen für Mitarbeiter des Kunden. • *OS*, *DB* und *SAP-Basis* • Applikation • LAN

Einrichten, Ändern, Inaktivieren und Entfernen von *Benutzeridentifikationen* sowie der zugehörigen *Benutzerberechigungen* für Mitarbeiter des Kunden.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Definition von *Benutzeridentifikationen* sowie zugehörigen *Benutzerberechtigungen* für die Mitarbeiter des Outsourcing-Providers.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Einrichten, Ändern, Inaktivieren und Entfernen von *Benutzeridentifikationen* sowie der zugehörigen *Benutzerberechtigungen* für die Mitarbeiter des Outsourcing-Providers.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Übergeben einer Liste der vorhandenen *Benutzeridentifikationen* der Kundenmitarbeiter zur jährlichen Überprüfung.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Jährliche Überprüfung aller *Benutzeridentifikationen* des Kunden. Auftrag an den Outsourcing-Provider welche *Benutzeridentifikationen* zu löschen sind.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Durchführung des Löschauftrags.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Jährliche Überprüfung der *Benutzerberechtigungen* aller Mitarbeiter des Outsourcing-Providers zugewiesenen *Benutzeridentifikationen*, die zur Serviceunterstützung benötigt werden und Entfernen der nicht mehr benötigten *Benutzeridentifikationen*.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Definieren von Prozessen und Kriterien für das Administrieren der Passwörter.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Setzen und Zurücksetzen von Passwörtern für die den Mitarbeitern des Outsourcing-Providers zugewiesenen *Benutzeridentifikationen* und deren Übergabe an autorisierte Mitarbieter des Outsourcing-Providers.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Setzen und Zurücksetzen von Passwörtern für die den Kundenmitarbeitern zugewiesenen *Benutzeridentifikationen* und deren Übergabe an autorisierte Kundenmitarbeiter.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Prüfen und Genehmigen von Anträgen für *hochautorisierte Benutzeridentifikationen*.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Vierteljährliches Überprüfen der *hochautorisierten Benutzeridentifikationen* und Entfernen der nicht mehr bewilligten *Benutzerberechtigungen*.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Überprüfen der *hochautorisierten Benutzeridentifikationen* und Übernahme der Kontrolle dieser *hochautorisierten Benutzeridentifikationen*.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Implementieren und Verwalten von IT-Sicherheitsmassnahmen für die Subsysteme und Applikationen, die nicht durch das *Zugriffskontrollsystem* geschützt sind (z.B. *DB*, Transaktionsserver, Domain Names Server, etc.).
- *OS*, *DB* und *SAP-Basis*
- Applikation

Sicherheits-/Integritätsfehlerkorrekturen (*Fixes* auf den im Rahmen der vertraglichen Vereinbarung unterstützten Systemen) werden über einen Änderungskontrollprozess (Change Management) behandelt.

Regelmässiges Durchführen der Systemsicherheitsüberprüfung (Security Health Checking).
- Zugriffskontrolleinstellungen
- *Hochautorisierte Benutzeridentifikationen* mit ihren *Benutzerberechtigungen*
- Schutz der *Betriebssystemressourcen (OSRs)*
- Betriebsbereitschaft der Antivirusprogrammen auf den jeweiligen Serverplattformen

Erfassen der Aufzeichnungen (*Logs*). Aufbewahren dieser Aufzeichnungen während einer vertraglich vereinbarten Frist. Bereitstellen von Berichten gemäss separatem Auftrag und gegen Entschädigung.

Informieren des Kunden über die vom Outsourcing-Provider festgestellten Sicherheitsschwachstellen und allfällige Risiken. Soweit möglich Unterbreiten von Empfehlungen zur Fehlerbehebung.

Der Kunde bestätigt innert drei Arbeitstagen durch schriftliche Gegenzeichnung den Empfang der vom Outsourcing-Provider berichteten Sicherheitsschwachstellen, Risiken und Empfehlungen.

Besprechen, Vorschlagen und soweit möglich Durchführen geeigneter Fehlerbehebungsmassnahmen zur Beseitigung der vom Outsourcing-Provider erkannten *Sicherheitsverletzungen*.
- *OS*, *DB* und *SAP-Basis*
- Applikation
- LAN

Kunden-Netzinfrastruktur (zusätzlich zur logischen Sicherheit)

Kontrollieren und Unterhalt der hochautorisierten *Benutzeridentifikationen* des *Netzbetriebssystems* (Server, *Router*, etc.).
- unter der *IT-Sicherheitsverantwortung* vom Outsourcing-Provider steht
- nicht unter der *IT-Sicherheitsverantwortung* vom Outsourcing steht

Ressourcen (Hardware und Software) die sich auf Endbenutzermaschinen befinden.
- Identifizieren und Definieren der IT-Sicherheitsmassnahmen
- Implementieren der festgelegten IT-Sicherheitsmassnahmen

Kontinuierliches Aktualisieren des Virenschutzes auf Endbenutzermaschinen.

Durchführung der Virenschutz-Selbstprüfungen für alle Endbenutzermaschinen und *Portable Storage Media*.

Reagieren auf Virenangriffe und Einleiten von Fehlerbehebungsmassnahmen zur Eliminierung festgestellter Viren auf Endbenutzermaschinen.

Datennetz (WAN)

Festlegung und Implementierung der IT-Sicherheitsmassnahmen für *Firewalls* und *Gateways* für Verbindungen zu anderen Netzen, ausgehend von:
- einem vom Outsourcing-Provider betriebenen Netz
- einem vom Kunden betriebenen Netz

Verwalten und Warten aller *Firewalls* und *Gateways* für Verbindungen zu anderen Netzen, ausgehend von:
- einem vom Outsourcing-Provider betriebenen Netz
- einem vom Kunden betriebenen Netz

Identifizieren und Überprüfen aller *Einwahlservices* und *Gateways* zu anderen Unternehmen.
- unterder *IT-Sicherheitsverantwortung* des Outsourcing-Providers
- unter der *IT-Sicherheitsverantwortung des* Kunden

Verwalten und Warten der IT-Sicherheitsmassnahmen für die Verbindungen zu anderen Unternehmen via *Gateway*- und *Einwahlservices*.
- unter der *IT-Sicherheitsverantwortung des* Outsourcing-Providers
- unter der *IT-Sicherheitsverantwortung des* Kunden

Hinzufügen, Ändern und Löschen der *Benutzerberechtigungen* auf die *Einwahlservices*.
- unter der *IT-Sicherheitsverantwortung* des Outsourcing-Providers
- unter der *IT-Sicherheitsverantwortung des* Kunden

Zugriffe von und zu SAP-Support-System wie z.B. auf/von SAP-Site Waldorff (OSS).
- Verwalten und Aufrechterhalten der IT-Sicherheitsmassnahmen
- Hinzufügen, Ändern und Löschen der *Benutzerberechtigungen*

Swiss Federation of Information Processing Societies

IFIP NEWS

Our sincere congratulations to Prof ***Alfred Strohmeier***, current delegate of SVI/FSI to the IFIP Technical Committee 2 (Software – Theory and Practice), who received recently the ACM ***Outstanding Ada Community Contribution Award*** for his important contributions to Ada technology and utilisation and especially his efforts for the promotion of object oriented technology and his contributions as expert towards the revision of the Ada language and his leadership in the definition of the Ada Semantic Interface Specification (ASIS).

Recognition

Two Swiss colleagues have received recently international awards and deserve our recognition.

Our sincere congratulations to:

- Prof. Kurt Bauknecht, IFIP Past-president, who has been elected IFIP Honorary Member by the IFIP General Assembly in recognition to his restless efforts and successes as IFIP President, Chairman of the IFIP Technical Assembly and within the GA.
- Prof Alfred Strohmeier, current delegate of SVI/FSI to the IFIP Technical Committee 2 (Software – Theory and Practice), who received recently the ACM "Outstanding Ada Community Contribution Award" for his important contributions to Ada technology and utilisation and especially his efforts for the promotion of object oriented technology and his contributions as expert towards the revision of the Ada language and his leadership in the definition of the Ada Semantic Interface Specification (ASIS).

Technical Committee Membership

The names of the Swiss TC representatives have been published in INFORMATIK, No 1, February 2002 and are always listet in http://www.svifsi.ch/ifip-2.html.

Please note the following additions to this list:

- Dr. Esther Myriam Gelle, Head of the Information Management Research Group at ABB Corporatee Research, has been appointed new Swiss Delegate to the IFIP TC 5 (Computer Applications in Technology).

We congratulate Mrs Gelle for this appointment and wish her full success in this new activity.

Invitation for Nominations for TC 11:

One position is currently to be filled: TC 11 Security and Protection in IP Information Processing Systems to replace Beat Lehmann who has served successfully for many years in this position.

Applications for nominations are invited by E-mail or mail *before April 15, 2002*, with a short cv, interests in the field, and other pertinent information to SARIT, c/o: P-A.Bobillier who can provide a more detailed description of TC representatives activities.

(bobilliep@freesurf.ch), 128 route de Soral, 1233 Bernex.

IFIP TC 11 has the following aims: to increase the reliability and general confidence in information processing, as well as to act as a forum for security managers and other professionally active in the field of information processing security. Its scope includes the

establishment of a frame of reference for security common to organisations, professionals and the public and the promotion of security and protection as essential parts of information processing systems.

TC 11 has seven Working Groups in which several Swiss specialists are active. Their names can be found at http://www.i-s.ch.

For more information on IFIP, its Technical Committees and Working Groups, see: http://www.ifip.org

SATW Schweizerische Akademie der Technischen Wissenschaften
Academie suisse des sciences techniques
Accademia svizzera delle scienze tecniche
Swiss Academy of Engineering Sciences

The European IST Prize

The European IST Prize is organised by the European Council of Applied Sciences and Engineering (Euro-CASE), with the sponsorship and support of the Information Society Technologies Programme of the European Commission.

The European Information Society Technologies Prize (EISTP) is open to any organisation (enterprise, laboratory, university or other institution) based in the European Union or in the Associated States or in Switzerland. Associated States include Bulgaria, Cyprus, the Czech Republic, Estonia, Hungary, Iceland, Israel, Latvia, Liechtenstein, Lithuania, Norway, Poland, Romania, Slovakia and Slovenia.

Awards will be given to outstanding contributions under the theme "novel products with high information technology content and evident market potential". Excellence is sought in converting innovative ideas and R&D results into marketable products. Products should be at least at demonstrable prototype stage and if already marketed should not have been introduced in the market more than 18 months prior to the opening of the yearly competition (normally on 1st January).

The selection criteria include technical excellence, innovative content, strategic business planning, potential for improving competitiveness, potential market value, capacity to generate employment by opening new markets or starting up new enterprises, contribution towards extending the acceptance and the understanding of information technology by society, and anticipated social benefits.

Application for the 2003 European IST Prize is only acceptable via the application form that can be obtained from the download page www.it-prize.org. Details are given in the leaflet on the European IST Prize and on this European IST Prize Web site. The application form, with enclosures, should be arrive by surface mail to Euro-CASE at the latest on 15 May 2002.

All applications received by Euro-CASE will be assessed by independent experts nominated by EuroCASE. The evaluations are confidential. Up to 20 Winners will be selected. The Winners will be individually contacted in advance by the organisers in order to prepare their presentations to the European IST Prize Exhibition and the Awards Ceremony.

The result of the selection of the up to 20 Winners will be communicated by letter to all applicants by the end of September 2002.

The Executive Jury, composed of European executives from Industry and University, nominated by Euro-CASE, will then select up to three Grand Prize Winners.

Each of the Winners will receive a Prize of € 5,000 and the European IST Prize Winner Certificate. Each Grand Prize Winner will receive a Grand Prize of € 200,000 and the European IST Prize Trophy. The Grand Prize Winners will be announced at the European IST Prize Awards Ceremony, concurrently with the 2002 European IST Event in Copenhagen on 4-6 November 2002.

Visibility of individuals and teams behind contributions selected as Winners will be ensured through personal invitation to the European IST Prize Awards Ceremony and the European IST Prize Winners Exhibition during the IST 2002 Event. The Winner companies and their products will be widely promoted, and the Winners will be invited to join the “European IST Prize Winners Club”.

All information submitted is treated as confidential. The evaluation is confidential. The organisers will not disclose the judgements and opinions given by the individual independent experts or the members of the Executive Jury.

The decisions of the Executive Jury are final. If not a sufficient number of applications submitted reach a standard which is considered as adequate, the organisers reserve the right to reduce the number of Winners or not to award Grand Prizes.

e-Government Symposium

Einladung für das Symposium e-government

13. Juni 2002, Kongresshaus Biel

Das Symposium wird von verschiedenen Partnern aus Bund, Kanton, Gemeinde und Privatwirtschaft unterstützt und finanziell getragen; Die Bundes-Seite vertreten Referenten des Bundesamtes für Kommunikation BAKOM, des Informatikstrategieorgans Bund ISB, der Stadt Biel, des Kantons Neuenburg und die privatwirtschaftliche Seite, Redner von Hewlett-Packard, Kudelsky, Wisekey, SAP, der Ruf Gruppe, Mummert &Partner und diversen kleineren Unternehmen aus dem IT-und /oder Consulting-Bereich. Als Key note Speaker referiert Prof. Dr. Beat Schmid, Medien- und Kommunikations-Institut der Uni St. Gallen. Wir erwarten 100-200 qualifizierte Teilnehmer aus den erwähnten Bereichen und Branchen.

In einer ersten Vorinformation und Einladung werden wir in den nächsten Tagen landesweit die Kantone, Städte und Gemeinden in einem Mailing anschrieben. Eine weitere Ziel- und Interessensgruppe sind natürlich auch die Schweizerischen Informatikverbände und Organisationen, die wir ebenfalls gezielt einladen möchten.

Interforum ist ein Verein, der sich auf die Organisation hochstehender Veranstaltungen spezialisiert hat. Auf unserer Website <http://www.interforum-bern.ch/> finden Sie andere von uns erfolgreich durchgeführte Events.

Auskunft und Anmeldung: pr@bischof.ch

First Call for Participation

Turing Day

Computing science 90 years from the birth of Alan M. Turing.
Friday, 28 June 2002, Swiss Federal Institute of Technology Lausanne (EPFL)
http://lslwww.epfl.ch/turingday

Purpose:

Alan Mathison Turing, born on June 23, 1912, is considered one of the most creative thinkers of the 20th century. His interests, from computing to the mind through information science and biology, span many of the emerging themes of the 21st century.

On June 28, 2002, we commemorate the 90th anniversary of Alan Mathison Turings birthday in the form of a one-day workshop held at the Swiss Federal Institute of Technology in Lausanne. The goal of this special day is to remember Alan Turing and to revisit his contributions to computing science. The workshop will consist of a series of invited talks given by internationally-renowned experts in the field.

Invited Speakers:

B. Jack Copeland
"Artificial Intelligence and the Turing Test"

Martin Davis
"The Church-Turing Thesis: Has it been Superseded?"

Andrew Hodges
"What would Alan Turing have done after 1954?"

Douglas R. Hofstadter
"The Strange Loop – from Epimenides to Cantor to Russell to Richard to Goedel to Turing to Tarski to von Neumann to Crick and Watson"

Tony Sale
"What did Turing do at Bletchley Park?"

Jonathan Swinton
"Watching the Daisies Grow: Turing and Fibonacci Phyllotaxis"

Gianluca Tempesti
"The Turing Machine Redux: A Bio-Inspired Implementation"

Christof Teuscher
"Connectionism, Turing, and the Brain"

Special Events:

- Display and demonstration of an original Enigma machine
- Exhibition of historical computers (Bolos Computer Museum)
- Demonstration of Turings neural networks on the BioWall
- Demonstration of a self-replicating universal Turing machine

Fees:

The registration fee includes participation in the workshop, lunch, and coffee breaks.

Students EPFL	free (without lunch)
Staff EPFL (incl. PhD students)	CHF 80.- / $ 45 / € 55
Students (incl. PhD students)	CHF 80.- / $ 45 / € 55
Others	CHF 200.- / $ 120 / € 135

Registration:

Please use the fax registration form on the Turing Day web-site:
http://lslwww.epfl.ch/turingday

Contact:

Christof Teuscher
Direct line: +41 21 693 67 14
Fax: +41 21 693 37 05
e-Mail: christof@teuscher.ch

Sponsors:

- Bolos Computer Museum, http://www.bolo.ch
- The Cogito Foundation, http://www.cogitofoundation.ch
- Elsevier Science, http://www.elsevier.nl
- Swiss Federal Institute of Technology Lausanne (EPFL), http://www.epfl.ch
- Logic Systems Laboratory (LSL), Swiss Federal Institute of Technology Lausanne (EPFL), http://lslwww.epfl.ch
- Dr. Charles Maillefer, Buchillon, Switzerland
- Digital Brainstorming, Pour-Cent Culturel Migros, http://www.digitalbrainstorming.ch
- School of Computer and Communication Sciences, Swiss Federal Institute of Technology Lausanne (EPFL), http://icwww.epfl.ch

We are looking forward to seeing you in beautiful Lausanne!

Schweizer Informatiker Gesellschaft
Société Suisse des Informaticiens
Società Svizzera degli Informatici
Swiss Informaticians Society

Präsident • Président: Prof. Klaus Dittrich
Sekretariat • Secrétariat: Tel. 01 371 73 42
Schwandenholzstr. 286 Fax 01 371 23 00
8046 Zürich E-mail: admin@s-i.ch
http://www.s-i.ch/

Editorial

Informatik/Informatique geht – *Informatik Spektrum* kommt!

Liebe Mitglieder der SI,

Es freut mich sehr, dass ich Ihnen rechtzeitig zum Abschied von unserer INFORMATIK/INFORMATIQUE die hochkarätige Nachfolge ankündigen kann. Ab Heft 3/2002 werden wir nämlich das in Deutschland seit 25 Jahren höchst erfolgreiche *Informatik Spektrum* gemeinsam mit der GI, unserer deutschen Schwestergesellschaft, herausgeben und Ihnen wie gewohnt zweimonatlich zustellen. Selbstverständlich sollen dort auch Fachbeiträge aller Art aus der Schweiz erscheinen – es liegt an Ihnen, solche zu schreiben. Weiterhin stehen uns einige Seiten je Heft für die Nachrichten der SI und ihrer Fachgruppen zur Verfügung, und der Herausgeberbeirat wird sukzessive um einige Mitglieder unseres Landes erweitert werden. Auch das *Informatik Spektrum* hat seine WWW-Seite, auf der alles "elektronisch" vorhanden ist.

Für unsere ausschliesslich französisch sprechenden Mitglieder bestehen Pläne, Zusammenfassungen bzw. ganze Artikel zu übersetzen und im WWW zur Verfügung zu stellen. Auf "unseren" Seiten in der Druckausgabe sind wie bisher alle gängigen Sprachen möglich.

Bitte bilden Sie sich ab Juni Ihr eigenes Bild unserer neuen Zeitschrift; ich bin sicher, Sie werden zufrieden sein.

Prof. Klaus Dittrich
Präsident SI

In Informatik/Informatique 6/2001 brachten wir die schlechte Nachricht: Informatik/Informatique wird eingestellt. Heute bringen wir die gute Nachricht: die Mitglieder der SI erhalten weiterhin eine Fachzeitschrift von hohem Niveau. Deren Redaktor stelle ich Ihnen hier vor:

Hermann Engesser, Redaktor von Informatik Spektrum, ist seit sieben Jahren beim Springer-Verlag und leitet dort den Programmbereich Informatik/Computerfachbuch. Damit ist er Chefredakteur des Informatik-Spektrums. Zwei weitere Zeitschriften, Informatik – Forschung und Entwicklung (IFE) und die gerade gegründete Software and System Modeling (SoSyM) werden von ihm betreut. Im Springer Buchprogramm ist er für die Informatiklehrbücher und für die Computerfachbücher verantwortlich. Vor zwei Jahren hat er die beiden Reihen Xpert.press für professionelle IT-Entwickler und -Anwender und X.Media.press für Mediengestalter gestartet.

François Louis Nicolet
bisher Redaktor Informatik/Informatique

Informatik/Informatique disparaît – *Informatik Spektrum* apparaît!

Chers membres de la SI,

Juste à temps pour l'adieu, j'ai le plaisir de vous annoncer la succession prestigieuse de notre INFORMATIK/INFORMATIQUE. A partir de l'édition 3/2002 nous sortirons la fameuse revue *Informatik Spektrum*, ensemble avec la GI, notre société soeur allemande et vous enverrons comme d'habitude tous les deux mois. Vous y trouverez aussi des contributions techniques de tout genre de la Suisse – il ne tient qu'à vous de les écrire. De plus, nous aurons quelques pages à disposition pour les nouvelles de la SI et de ses groupes, et le conseil éditorial sera augmenté de quelques membres de notre pays. *Informatik Spektrum* a aussi ses pages WWW sur lesquelles tout est disponible sous forme « électronique ».

Pour nos membres francophones, nous avons l'intention de traduire des résumés et même des articles entiers et de les mettre à disposition sur le Web. Sur « nos » pages de l'édition imprimée nos langues nationales sont possibles comme jusqu'à présent.

Formez-vous votre propre opinion de la nouvelle revue; je suis sûr que vous serez satisfait.

Prof. Klaus Dittrich
Président SI

Dans le numéro 6/2001 d'Informatik/Informatique, nous avions communiqué la mauvaise nouvelle de la cessation de la publication de cette brochure. Aujourd'hui, nous communiquons la bonne : les membres de la SI continueront de recevoir une revue de haut niveau, Informatik-Spektrum. Voici le profil de son rédacteur :

Hermann Engesser, rédacteur d'Informatik Spektrum, travaille depuis sept ans pour la maison d'édition Springer où il exerce les fonctions de directeur du programme « ouvrages sur l'informatique et les ordinateurs ». Deux autres revues, « Informatik – Forschung und Entwicklung » (IFE) et « Software and System Modeling » (SoSyM), la deuxième venant d'être créée, relèvent de sa compétence. Dans le programme de Springer concernant les livres, il est responsable des manuels d'informatique et des ouvrages spécialisés sur la pratique de l'ordinateur. Il y a deux ans, il a lancé les deux séries « Xpert.press » pour développeurs et utilisateurs professionnels de logiciels et « X.Media.press »pour concepteurs de média.

François Louis Nicolet
jusqu'ici rédacteur d'Informatik/Informatique

Drei Fragen an Hermann Engesser, Chefredaktor, Informatik-Spektrum

Was ist die Aufgabe des Redakteurs für das Informatik-Spektrum?

70 Seiten. Der Redakteur hat die Aufgabe, die 70 Seiten des nächsten Heftes zu füllen. Diese Aufgabe stellt sich sechsmal im Jahr. Der Redakteur löst diese Aufgabe nicht allein sondern in Zusammenarbeit mit den Herausgeberinnen und Herausgebern, mit Autoren, mit der Geschäftsstelle der *Gesellschaft für Informatik*, Verlagsmitarbeiterinnen und -mitarbeitern und last but not least zusammen mit den Leserinnen und Lesern des Heftes.

Beim *Informatik-Spektrum*, einer Fachzeitschrift für die *Gesellschaft für Informatik* mit einer Auflage von über 20'000 Exemplaren stehen aktuelle, praktisch verwertbare Informationen über einschlägige technische und wissenschaftliche Fortschritte in Theorie und Anwendung im Vordergrund. Seit dem Start der Zeitschrift vor 25 Jahren mit einer Auflage von knapp 2000 Exemplaren hat sich der Charakter der Zeitschrift zwar verändert, die genannte Aufgabenstellung ist geblieben. Gerade Informatik und IT sind Berufsfelder mit den unterschiedlichsten Zugängen und Ausprägungen.

Neben dem Informatik-Studium gibt es zahlreiche andere Ausbildungswege für den Einstieg oder den Umstieg zu diesem Beruf, der ja in Wissenschaft und Industrie wiederum verschiedene Berufsbilder kennt, die – als sei dies nicht genug – einer hohen Veränderungsrate ausgesetzt sind. Unsere Konzeption besteht darin, die Leserwünsche durch entsprechende Rubriken im Heft zu berücksichtigen. Den Rahmen bilden die Hauptbeiträge, fundierte Darstellungen zu wissenschaftlichen und technischen Entwicklungen, für die fachliche Orientierung, sowie fachgesellschaftliche Mitteilungen und eine Veranstaltungsübersicht, für das *Society Organizing*. Die gesellschaftlichen Folgen technischer und wissenschaftlicher Entwicklungen sind häufig umstritten. Unterschiedliche Positionen kommen in der Rubrik *Zur Diskussion gestellt* zu Wort. Ausführliche Diskussionsbeiträge gab es hier in den letzten Heften etwa zu den Themen *Open Source* und *Software-Patentierung*. Im *Aktuellen Schlagwort* wird jeweils ein aktuelles Thema oder eine neue Begriffsbilung kompakt definiert. In der *Student's Corner* schreiben Studierende über Ausbildung, Praktika und Projekte. In den *Historischen Notizen* liest man Interessantes und Überraschendes aus der Historie dieser jungen Wissenschaft, deren Geschichtsbildung gerade erst begonnen zu haben scheint. Gunter Duecks Kolumne ist zu einem festen Bestandteil des *Forums* geworden, in dem man ferner Berichtenswertes im Umfeld der Informatik findet. Wenn Sie das alles zusammenbinden, von der Planung des Heftes, von der Akquisition der Artikel über die Organisation des Begutachtungsprozesses in Zusammenarbeit mit dem Hauptherausgeber und dem Herausgebergremium, über die Redaktion der Beiträge bis zur Produktion des Hefts, dann haben Sie ein gutes Bild von der Aufgabe des Redakteurs. Wie gesagt, 70 Seiten.

Als Leser von Informatik-Spektrum schätze ich, dass die Beiträge von hohem fachlichen Niveau, verständlich und lesenswert sind. Was ist Ihr Geheimrezept?

Das Ziel einer guten Zeitschrift, einer guten Zeitung oder eines guten Buches ist, dass der Leser mit dem Autor anhand des Lesestoffs in einen Diskurs tritt. Bei einer Fachzeitschrift ist die Profession der Leser Voraussetzung. Damit können Autoren bei der Konzeption Ihrer Artikel ihre fachliche Expertise ausreizen. Verständlich zu schreiben, scheint mir am schwierigsten. Die französische Mathematiker-Gruppe N. Bourbaki versuchte früher bei ihren Seminaren, Verständlichkeit dadurch zu erreichen, dass jeder Teilnehmer nicht über sein eigenes Forschungsthema vortragen durfte, sondern nur über das eines ande-

Trois questions posées à Hermann Engesser, Rédacteur en chef, Informatik-Spektrum

Quelles sont les tâches du rédacteur d'Informatik-Spektrum ?

70 pages. Le rédacteur a pour tâche de remplir les 70 pages de la brochure à paraître et ceci six fois dans l'année. Le rédacteur n'est pas seul à s'acquitter de cette tâche ; il le fait en collaboration avec les éditeurs, des auteurs et le bureau de la *Gesellschaft für Informatik*, des employés de la maison d'édition et, ce qui n'est pas de la moindre importance, les lecteurs et lectrices de la revue.

Dans *Informatik-Spektrum*, revue spécialisée de la *Gesellschaft für Informatik* tirée à plus de 20 000 exemplaires, l'on trouve avant tout des informations d'actualité, d'utilité pratique, concernant des progrès techniques et scientifiques réalisés dans ce domaine dans la théorie et son application. Depuis sa création il y a 25 ans avec un tirage d'à peine 2000 exemplaires, les tâches que s'est données cette revue sont restées identiques, même si le caractère de la revue a changé. Précisément l'informatique et les STI sont des domaines professionnels auxquels on peut accéder par les moyens les plus divers et dans lesquels on peut se distinguer de différentes façons.

A part les études en informatique, il existe de nombreux autres moyens de formation pour accéder à cette profession ou y arriver après en avoir exercé une autre. Cette profession, qui englobe différentes formes d'activité dans le domaine de la science et de l'industrie, est livrée à un taux élevé de changements. Notre concept est de tenir compte des souhaits des lecteurs en insérant dans notre brochure les rubriques appropriées. L'encadrement en est constitué par les articles principaux, de solides exposés sur des développements scientifiques et techniques pour la partie informative, ainsi que par des communications de la société d'informatique elle-même et un aperçu des manifestations prévues, pour le côté associatif *Society Organizing*. Les conséquences pour la société des développements scientifiques et techniques sont souvent contestées. Les différentes positions peuvent s'exprimer dans la rubrique *Zur Diskussion gestellt* (Matière à discussion). Dans les derniers cahiers, il y a bien eu des articles détaillés donnant lieu à discussion sur les thèmes *Open Source* et *logiciel et brevet*. Dans la rubrique *Aktuelles Schlagwort* (A l'ordre du jour), sont définis de manière compacte, soit un thème d'actualité, soit une nouvelle interprétation. Dans la rubrique *Student's Corner*, écrivent des étudiants sur la formation, des stages et des programmes. Dans les *Notes historiques*, l'on peut lire des choses intéressantes et surprenantes relevées dans les étapes de cette science nouvelle dont l'histoire semble n'en être qu'à ses débuts. La colonne de Gunter Dueck est devenue partie intégrante du *Forum* du fait qu'il continuera d'y rapporter des sujets importants dans le domaine de l'informatique. Si vous rassemblez tous ces éléments, de la planification de la revue, de l'acquisition des articles, en passant par l'organisation du processus d'examen en collaboration avec l'éditeur principal et le comité des éditeurs et par la rédaction des articles proposés, jusqu'à la production de la brochure, vous aurez alors une image parfaite des tâches d'un rédacteur. Comme dit, 70 pages.

En tant que lecteur d'Informatik-Spektrum, j'apprécie le fait que ses articles témoignent d'une haute qualification, sont compréhensibles et méritent d'être lus. Quel est votre secret ?

Le but d'une bonne revue, d'un bon quotidien ou d'un bon livre est d'établir, par le biais de son contenu, un dialogue avec le lecteur. Lorsqu'il s'agit d'une revue spécialisée, la profession du lecteur est une condition primordiale. Les auteurs peuvent ainsi, par la manière de concevoir leurs articles, aller jusqu'au bout de leur compétence. Ce qui me semble le plus difficile, c'est d'écrire d'une manière compréhensiren. Durch den Wechsel der Perspektive wird Verständlichkeit auf hohem Niveau erreicht.

Bei einem Informatik-Fachartikel scheint mir dies deshalb noch schwieriger zu sein, weil es mehr Parameter der Verständlichkeit gibt. Neben der theoretischen Fundierung des Dargestellten und der Klarheit der Folgerungen kommen hier etwa Praxisrelevanz oder Einfluss auf andere, oft außerhalb der Informatik liegende Gebiete zum tragen. Verständlichkeit für den Experten wie für den Nicht-Experten auf dem jeweiligen Fachgebiet versuchen wir dadurch zu erreichen, dass wir jeweils einen Vertreter dieser beiden Gruppen um ein Gutachten bitten. Es ist für mich erstaunlich, wieviele konstruktive Anregungen die Peers haben und die Chance, diese Anregungen im druckfertigen Artikel zu realisieren, wird von vielen Autoren gerne genutzt, die diesen Perspektivwechsel ja gelegentlich auch in Ihrem Artikel notieren.

Bei den Meeren haben die Wellen an der Oberfläche eher geringen Einfluss auf das Klima. Trotzdem erfreuen Sie den Beobachter. Doch es sind die tieferen Strömungen, denen das Wettergeschehen langfristig folgt. Ähnlich ist es mit einer zweimonatlich erscheinenden Fachzeitschrift. Lesenswerte Artikel mit Tagesaktualität werden Sie hier vergeblich suchen. Es geht um die tieferen Strömungen. Doch gerade in der Informatik sind diese Entwicklungen und Ihre Folgen für Mensch, Technik, Philosophie und Gesellschaft von einer solchen Brisanz und Dynamik, dass viele Leser diese Themen lesenswert finden, weil Sie mehr darüber wissen wollen, wissen müssen.

Wenn dann der Leser bei der Lektüre eines Artikels auch noch ein intellektuelles Vergnügen empfindet, dann ist das mehr als genug… und der Redakteur ist schon wieder weiter. Er liest vielleicht gerade die Erstversion eines Artikels für das übernächste Heft.

Informatik-Spektrum war bisher die Zeitschrift der Mitglieder der GI. Nun kommen auch die Mitglieder der SI dazu. Was ist für die neue Leserschaft vorgesehen?

Ich würde mich besonders freuen, wenn sich die SI-Mitglieder von Anfang an im *Informatik-Spektrum* wiederfinden. Es versteht sich von selbst, dass die Aktivitäten der SI – wie bisher schon in der *Informatik/Informatique* – unter den für das Informatik-Spektrum neuen *SI-Mitteilungen* abgedruckt werden.

Auch Hauptbeiträge und Beiträge zu den Rubriken sind hochwillkommen. Hier haben ja auch in der Vergangenheit schon SI-Autoren im *Informatik-Spektrum* publiziert.

Generell sehe ich in dem dann für beide Organisationen verlegten *Informatik-Spektrum* eine Win-Win-Situation mit mehr Informationen für alle Leser und mit einer stärkeren Außenwirkung. "Jedem Anfang wohnt ein Zauber inne…", sagte Hermann Hesse. Das ist für mich das Motto für Heft 3/2002, das erste gemeinsame *Informatik-Spektrum* für GI und SI.

ble. Le groupe de mathématiciens français N. Bourbaki a essayé dans le passé, lors de ses séminaires, de parvenir à la compréhension en demandant à chaque participant de ne pas parler de son propre thème de recherche, mais de celui d'un autre. Le changement de perspective apporte une compréhension de haut niveau.

S'agissant d'un article spécialisé concernant l'informatique, ceci me paraît encore plus difficile puisqu'il existe dans ce domaine davantage de paramètres de compréhension. Le sujet exposé ne doit pas seulement avoir de solides fondements en théorie et les déductions en être claires ; l'importance de la pratique ou l'influence sur d'autres domaines souvent extérieurs à l'informatique comptent également pour beaucoup. Nous essayons de parvenir à faire comprendre un texte par un expert autant que par une personne qui ne l'est pas dans cette spécialité en demandant, selon le cas, son avis à un représentant de ces deux groupes de personnes. Je trouve surprenantes les suggestions constructives des *Peers* et de nombreux auteurs saisissent la chance de réaliser celles-ci dans un article prêt à l'impression. Il arrive aussi qu'à l'occasion, ces auteurs relèvent dans leur article ce changement de perspective.

Celui qui observe la mer se réjouira de regarder les vagues qui se meuvent à la surface bien que celles-ci n'aient que peu d'influence sur le climat. Ce sont, en fait, les courants en profondeur qui, à long terme, déterminent le temps. Il en va de même pour une revue spécialisée paraissant tous les deux mois. Vous y chercherez en vain des articles valables d'une actualité limitée à la seule journée. Ce qui compte, ce sont les courants en profondeur. Cependant, précisément en informatique, ces développements et leurs conséquences pour l'homme, la technique, la philosophie et la société sont empreints d'une force et d'une dynamique telles que bien des lecteurs trouvent ces thèmes dignes d'être lus, parce qu'ils veulent en savoir davantage, doivent en savoir davantage.

Alors, lorsque le lecteur éprouve, en outre, un plaisir intellectuel à la lecture d'un article, c'est plus qu'il n'en faut et le rédacteur a déjà continué son chemin. Il est peut-être déjà en train de lire la première version d'un article destiné à la brochure suivante.

Informatik-Spektrum était, jusqu'à présent, la revue des membres de la GI. Viennent s'ajouter maintenant les membres du SI. Qu'est-il prévu pour ces nouveaux lecteurs ?

Je serais particulièrement content si, dès le début, les membres de la SI se retrouvaient dans *Informatik-Spektrum*. Il va de soi que les activités de la SI – comme c'était le cas déjà jusqu'à présent dans *Informatik/Informatique* – paraîtront dans la nouvelle rubrique *Communications de la SI* d'*Informatik-Spektrum*. Tous les articles et toutes formes de participation aux rubriques existantes seront les bienvenus. Mais, dans le passé déjà, des auteurs de la SI avaient publié des articles dans *Informatik-Spektrum*.

Sur un plan général, je vois plutôt dans *Informatik-Spektrum*, édité pour les deux organisations, une situation *win-win* qui apportera davantage d'informations à tous les lecteurs et aura une plus grande influence extérieure. « Tout commencement porte en lui un charme », disait Hermann Hesse. C'est ma devise pour la l'édition 3/2002, premier numéro d'*Informatik-Spektrum* en commun pour la GI et la SI.

DBTA

Datenbanken, Theorie und Anwendung
Bases de données, théorie et application
Chairman: Paul Eisner
Correspondent: Moira Norrie
http://www.unizh.ch/groups/dbtg/DBTA/

DBTA plans a major event to celebrate its 15 years anniversary. We intend to have a two day seminar in late summer 2002 or in spring 2003, at a well-known and agreeable place. We think about analysing social implications of IT, and comparing them with technological advances achieved, with a special focus on the database area. Renown key-note speakers will be proposed.

We invite DBTA members to share in our planning and to contribute their ideas. Please write to Paul.Eisner@swisslife.ch.

JUGS

Java User Group Switzerland
Chairman: Arthur Neudeck
http://www.jugs.ch/

Java News

Welcome to the last issue of *Java News* published through *Informatik/Informatique*. Sad to see that publication disappear, life is tough sometimes!

Do you find this type of information interesting? Do you think I should keep on publishing that column online? If so, please send a quick e-mail to silvano.maffeis@softwired-inc.com and let me know!

Visit http://www.jugs.ch for a list of upcoming Java events hosted by JUGS. Also, you will find this issue of JavaNews online on the JUGS Web: It's easier to click on URLs rather than typing them in!

- Finalists announced for the JavaWorld editor choice awards. These awards recognize innovative companies, organizations, and individuals for their commitment to developing new Java tools and technologies.
  http://www.javaworld.com/javaworld/jw-02-2002/jw-0211-finalists.html
- Java Web Services Developer Pack released. All you need (and more) to start developing Web services in Java: SOAP, JAXP, JAXM, a UDDI registry server, Tomcat Servlet engine, etc.
  http://java.sun.com/features/2002/01/wspack.html
- JavaOne is coming! For those that still have the luxury of a travel budget, JavaOne is *the* get-together for Java-enthusiast and hunters of useless-but-really-cool gadgets (25–29 March, San Francisco):
  http://servlet.java.sun.com/javaone/home/0-sf2002.jsp
- Vendors embrace J2EE 1.3. Eighteen companies including BEA, IBM, Borland, Pramati, and Silverstream have passed the J2EE 1.3 compliance tests since its debut five months ago. From InfoWorld:
  http://www.infoworld.com/articles/hn/xml/02/01/29/020129hnj2eeday.xml
- And the winner is… Well, being the fastest not always means being the winner. But still, a special honourable mention goes to Pramati Server 3.0; first application server to pass the Sun Microsystems J2EE v1.3 compatibility test suite
  http://www.pramati.com/corporate/press/press_cts.htm
- Divide and conquer. Aspect-oriented programming (AOP) is an extension of object-oriented programming, it allows developers to cleanly separate "concerns" in their software systems. The inventors of AOP claim that this leads to systems, which are easier to maintain, to extend, and to understand. An intro-article from JavaWorld:
  http://www.javaworld.com/javaworld/jw-01-2002/jw-0118-aspect.html
- Mac OS X and Java. The boldest announcement Jobs made in his "Macworld" keynote is that Mac OS X will be the default OS in all new Macs by the end of this month. Mac OS X, built on top of Unix, contains the Java 1.3.1 runtime, as well as command line tools such as javac, java, jar, rmi, and rmid.
  http://www.javaworld.com/javaworld/jw-01-2002/jw-0118-macworld.html?
- JBuilder 6 Enterprise features UML support and refactoring tools.
  http://www.javaworld.com/javaworld/jw-01-2002/jw-0118-iw-jbuilder.html

Silvano Maffeis, JUGS,
mailto:silvano.maffeis@softwired-inc.com

SAUG

Swiss APL User Group
Chairman: Hans Steffen
hans.steffen@bfs.admin.ch

Im Dezember des vergangenen Jahres starb Robin Trpisovsky. Für Robin nahm die Sprache APL eine wichtige Stelle ein, obwohl er sie in seiner Arbeit nur am Rande einsetzen konnte. Er war fasziniert von den einzigartigen Möglichkeiten von APL, schätzte aber die Akzeptanz, mit welcher die Kunden der Sprache begegneten, realistisch ein. Für seinen vielseitigen und kritischen Geist war die Arbeit mit APL nicht einfach irgendwelches Programmieren, sondern ein ausgesprochenes intellektuelles Vergnügen. Um dies etwas verständlicher zu machen, sei der heikle Versuch gewagt, die Besonderheit des Programmierens mit APL, etwa im Vergleich mit Java, herauszuarbeiten. Während die Entwickler von Java sich offensichtlich vom Bestreben leiten lassen, einen Baukasten ständig zu vergrössern, welcher schliesslich für jeden erdenklichen (wenn auch noch so trivialen) Vorgang ein passendes Element enthält, beruht APL auf einem kühn konzipierten Satz von mathematischen Operationen, von denen man in anderen Sprachen nur träumen kann. Ist der Programmierer imstande, in seinen Projekten die mathematischen Strukturen zu erkennen, kann er häufig diese direkt in APL abbilden – auf atemraubend effektive Art.

Robin Trpisovsky war die mathematische und intellektuelle Faszination, die APL auf ihn ausübte, immer anzumerken. Er trug sie in die regelmässigen Workshops der Swiss APL User Group (SAUG) hinein. Sein kritischer, durchdringender Geist spielte in diesen Workshops eine wichtige Rolle, ebenso wie sein breites fachliches Wissen. Robin, der ursprünglich Architekt war, hatte auch philosophische Interessen. Eine Zeitlang beschäftigte er sich mit logischen Problemen des Suchens und Fragens, mit der so genannten erotetischen Logik. Seine grundlegende Art des Denkens führte ihn dazu, dass er dem heutigen Programmierbetrieb, der unter dem Druck von Zeitlimiten stattfindet, eine gewisse Skepsis entgegenbrachte. In schnell zusammengeschusterten Programmen sah er nicht besondere Leistungen, sondern spätere Problemquellen. Die SAUG wird Robin Trpisovsky vermissen, seine fachliche Kompetenz ebenso wie seine kritischen Fragen.

Urs Oswald

Security

Chairman: Rolph Haefelfinger
Correspondent: Thomas Kohler
http://www.fgsec.ch/

Wir bitten Sie, sich die folgenden Veranstaltungen vorzumerken:

23. April 2002 (später Nachmittag)
Das neue Recht in der Informatik
Raum Zürich – www.fgsec.ch

20. Juni 2002 (später Nachmittag)
Wireless Security
Raum Zürich – www.fgsec.ch

26. September 2002 (später Nachmittag)
Forensics mit anschliessender Generalversammlung 2001
Raum Zürich – www.fgsec.ch

9. November 2002 Nachmittag
5. Berner Tagung für Informationssicherheit
Bern – www.fgsec.ch

Schweizerischer Verband der Informations- und Kommunikationstechnologie

Präsidium:	Maya Lalive d'Epinay
Redaktion:	info@swissict.ch
Verbandsdirektion	Hans-Peter Uehli, hans-peter.uehli@swissict.ch
	Telefon 056 222 65 00, Telefax 056 222 65 67
	Postfach 1345, 5401 Baden
Web:	http://www.swissict.ch/

Portrait Markus Fischer
Mitglied des Vorstandes und des Strategie-Ausschusses SwissICT

Als Nicht-Informatiker bin ich während den 26 Jahren, die ich für die SBG/UBS tätig war, mit zahlreichen Auswirkungen dieses faszinierenden Branchen-Clusters in Berührung gekommen, und schliesslich haben mich die Informations- und Kommunikations-Technologien und ihr enormes Potential für Wirtschaft und Gesellschaft derart in ihren Bann gezogen, dass ich mich im Jahr 2000 vom Banking verabschiedet habe.

Begonnen haben meine Begegnungen mit der damaligen "EDV" unter einem etwas zwiespältigen Eindruck. In die ersten Jahre meiner Bankkarriere fiel nämlich der Fall UBISCO der damaligen SBG. Quasi als Zuschauer und im Rahmen der dazu erhältlichen Informationen hat uns diese "EDV-Entwicklung" – und die Summen, um die es dabei ging – nachhaltig beeindruckt. Leider war damit auch ein zeitlicher Rückschlag für uns "Benutzer" verbunden, so dass die "dummen" Terminals samt den ersten Anwendungen und ihren eintönigen Zahlen- und Zeichenmenus erst gegen Ende der Siebziger Jahre auf uns zukamen. Die damit zu erzielenden Fortschritte waren indes gewaltig, zumindest gemessen an den damaligen Möglichkeiten.

So richtig den Ärmel reingenommen hat es mir Mitte der Achtziger Jahre mit der Einführung der PC's. Damals waren spezifische Anwendungen für Bankfachleute noch rar, weshalb wir uns selber an die Konzeption, Gestaltung und Entwicklung von kleinen Applikationen heranwagten. Unser erstes lauffähiges Programm zur Verwaltung, Überwachung und Berichterstattung von risikobehafteten Kreditpositionen wurde 1988 ausgebreitet und hatte auf einer einzigen 5 _ Zoll Diskette Platz!

Die Neunziger Jahre waren für mich von fünf massgeblichen Entwicklungen geprägt:

1. Einführung und Ausbreitung der letzten Host-Applikationen
2. Strategische Neuausrichtung des Banking, verbunden mit der Neugestaltung der Geschäftsprozesse und der internen Organisation
3. Flächendeckende Einführung vernetzter PC-Lösungen und Client Server Technologien
4. "Entdeckung" der Internet-Technologien, verbunden mit der Konzeption, Entwicklung und Einführung von Intranet- und Web-Lösungen
5. Entwicklung von Web Banking Strategien und Services für das Privatkunden-, Firmenkunden- und Anlagekunden-Geschäft

Während diesen Phasen habe ich meine Sporen als Projektleiter abverdient und überaus wertvolle Erfahrungen in zum Teil sehr komplexen, interdisziplinären und auch multikulturellen Projekten sammeln dürfen. Gleichzeitig konnte ich mein Fachwissen und die gesammelten Erfahrungen als Ausbilder und Prüfungsexperte der schweizerischen Bankfachprüfungen sowie als Referent an Nachdiplomkursen und Fachtagungen weitergeben.

Der bisherige Höhepunkt bildete jedoch ohne Zweifel das Venture "plenaxx", welches nebst der UBS von vier weiteren strategischen Partners (Mobiliar Versicherungen, Schweizerischer Gewerbeverband, Swisscom, Valora) mitgetragen wurde und zum Ziel hatte, den Schweizer KMU, Verbänden, Schulen und Institutionen innert kürzest möglicher Zeit eine e-Business & e-Collaboration Plattform samt entsprechender e-Services anzubieten und nach professionellen Kriterien zu betreiben. Obwohl wir innerhalb von nur acht Monaten das Unternehmen, die Marke und die Plattform aufbauten und in Betrieb nahmen, musste dieser Start-up aufgrund des dramatischen Einbruchs in der e-Szene bereits im ersten Betriebsjahr "gegroundet" werden, weil sich die ursprünglich geplanten Ziele punkto Cash und Zeit unter den neuen Gegebenheiten als nicht erreichbar herausstellten.

Trotz dieses Misserfolges hat mir die Führung dieses Ventures zusammen mit den Kollegen der Geschäftsleitung enorm viel gebracht und auch grosse Freude bereitet. Nachdem uns schon der forcierte Aufbau voll gefordert hatte, hatten wir beim anschliessenden ebenfalls forcierten Abbau und der Einstellung des Unternehmens und der Plattform innert weniger Monate nochmals Gelegenheit, unsere Fähigkeiten unter Beweis zu stellen. Diese zusätzliche Dimension an Erfahrung kommt mir bei meiner Tätigkeit als Berater und Dozent sehr vonstatten.

Ungeachtet der Abkühlung und stellenweise auch Frustration in der e-Szene bin ich überzeugt, dass wir das gigantische Anwendungs- und Nutzen-Potential der IC-Technologien und insbesondere von Lösungen in den Bereichen Web, Extranet, Intranet sowie von Multi (inklusive Mobile) Device fähigen Services erst zu geringen Teilen erschlossen haben. Die nächsten Jahre bleiben also ungemein spannend, und wir werden noch staunen, was in Anwendungsgebieten wie e-business, e-government, e-learning, mobile commerce, location based services und pervasive computing noch alles auf uns zu kommt. Nicht die Zukunft voraussagen, sondern sie schaffen und gestalten, ist dabei meine Devise: let's do it!

Markus Fischer ist 49jährig, verheiratet, wohnhaft in Lugnorre, und verbringt seine Freizeit am liebsten mit analoger und digitaler Fotografie, Computing und Multimedia, Aviatik und Reisen zu Vulkanen und anderen abenteuerlichen Destinationen.

Abschied von der Zeitschrift INFORMATIK

Sie halten die letzte Ausgabe der Zeitschrift INFORMATIK/INFORMATIQUE in der Hand. Der Redaktor, Herr François Louis Nicolet, tritt in den wohl verdienten Ruhestand. Leider war es den an der Zeitschrift beteiligten Verbänden nicht gelungen, einen geeigneten Nachfolger für ihn zu finden.

Im Jahre 1993 wurde eine "Projektgruppe Zeitschrift" ins Leben gerufen, welcher Prof. C. A. Zehnder (Präsident SVI/FSI), Helmut Thoma (Präsident SI) und Walter Wüst (Präsident WIF) angehörten. Am 1. Februar 1994 erschien die erste Nummer.

Die INFORMATIK/INFORMATIQUE diente bekanntlich als offizielles Mitgliederorgan von vier Verbänden: Schweizer Informatiker Gesellschaft SI, Schweizerischer Verband der Informations- und Kommunikationstechnologie SwissICT, Groupe Romand de l'Informatique GRI und Swiss Open Systems User Group ch/Open.

Die INFORMATIK war auf qualifizierte Informatik-Fachleute ausgerichtet. Oft haben die Übersichtsartikel und die einführenden Beiträge zu aktuellen Themen der Informationstechnologie die Leserschaft herausgefordert, dies nicht zuletzt wegen ihres hohen Abstraktionsniveaus. Für eher praxisorientierte Leser dürfte die Lektüre der INFORMATIK deshalb nicht immer leichte Kost gewesen sein. Erfreulicherweise hat die Mitgliederumfrage vom Herbst 2001 ergeben, dass etwa die Hälfte unserer Mitglieder zwischen 15 und 60 Minuten mit dem Studium der Zeitschrift INFORMATIK verbracht hat.

Die letzte Nummer der INFORMATIK markiert jedenfalls das Ende einer Ära in unserer Verbandsgeschichte – gleichzeitig aber auch einen Neubeginn: SwissICT ist selbstverständlich weiterhin auf ein offizielles Publikationsorgan als Kommunikationskanal zu seinen Firmen- und Einzelmitgliedern angewiesen. So wird der Vorstand an der Generalversammlung 2002 den Antrag stellen, mit dem Schweizerischen Verband der Telekommunikationsbenützer *asut* zusammenzuarbeiten, um zukünftig gemeinsam eine Zeitschrift herauszugeben, die gleichzeitig als unser offizielles Publikationsorgan dient.

Die neue Verbandszeitschrift wird dementsprechend verbandsinterne Berichte und Nachrichten enthalten und daneben auch mehrere Seiten mit Fachbeiträgen zu aktuellen Themen der Informations- und Kommunikationstechnologie.

In den letzten 10 Monaten, in denen ich für die Redaktion der Beiträge von SwissICT zuständig gewesen bin, habe ich die freundliche Unterstützung von Herrn Nicolet zu schätzen gewusst. Für seine Redaktion und Produktion der INFORMATIK habe ich einen hohen Respekt gewonnen. Ich bedanke mich an dieser Stelle bei Herrn Nicolet ganz herzlich für die angenehme und erfolgreiche zusammenarbeit. Ich wünsche ihm eine gute Gesundheit und viele fröhliche Stunden im neuen Lebensabschnitt. Seine Zeitschrift mit dem inhaltlich hochstehenden IT-Fachbeiträgen und dem markanten, knallroten Titelblatt wird mir in guter Erinnerung bleiben.

Im Namen des Vorstandes und der Mitglieder von SwissICT überbringe ich Herrn Prof. C. A. Zehnder die Zeichen unserer Dankbarkeit und unsere Anerkennung für seine bewundernswerte Initiative und seinen unermüdlichen, selbstlosen Einsatz zu Gunsten der Verbandszeitschrift INFORMATIK, auf die wir alle so stolz sein können.

Hans-Peter Uehli
Verbandsdirektor

Es war gewiss eine Herausforderung, den hohen Ansprüchen der in SwissICT und SI vereinigten Informatik-Fachleute gerecht zu werden. Ich hatte das Glück, in diesen Gesellschaften anerkannte Experten der verschiendenen Teilgebiete der Informatik zu finden, die als Gastredaktoren meine Arbeit unterstützten. Viele Autoren der Fachbeiträge sind Mitglieder der an der Zeitschrift beteiligten Gesellschaften, was einmal mehr ihre fachliche Kompetenz bezeugt. Eine Mitgliederzeitschrift dient auch dazu, über das Leben des Verbands zu berichten. Hier hat Herr Hans-Peter Uehli in jeder Ausgabe für vielseitige Beiträge gesorgt. Für seine Arbeit und die sehr angenehme, freundschaftliche Zusammenarbeit danke ich ihm ganz herzlich.

François Louis Nicolet

SwissICT-Salärumfrage 2002

Seit 1981 führt der Informatikverband SwissICT jährlich die massgebende Salärumfrage für Informatikberufe in der Schweiz durch. Die Anzahl Nennungen und damit auch der Wert der Erhebung steigt von Jahr zu Jahr: 2001 wurden 19'278 Salärnennungen von 236 Unternehmen ausgewertet! Einmal mehr hat diese erfreuliche Beteiligung das grosse Bedürfnis und den hohen Stellenwert der SwissICT-Salärumfrage für unsere Wirtschaft dokumentiert.

Wir würden uns freuen, wenn sich Ihr Unternehmen an der diesjährigen Salärumfrage beteiligen würde. Ende September 2002 erhalten die Teilnehmer kostenlos die aktuellste und repräsentativste Salärstatistik der Informatik- und Telekom-Spezialisten in der Schweiz.

Die Liste der Funktionen sowie ein Beispiel aus der Standardauswertung finden Sie auf unseren Webseiten unter Publikationen / Bestellformulare http://www.swissict.ch/publi/bestellformular.php

Stichtag für die Salärumfrage ist jeweils der 1. Mai. Die Erfassungsunterlagen werden deshalb im April an die teilnehmenden Unternehmen verschickt. Die ausgewerteten Daten liegen bis Ende September in Buchform vor.

Unternehmen, die bereits einmal teilgenommen haben, erhalten die Erfassungsunterlagen unaufgefordert zugestellt. Falls Sie zum ersten Mal an unserer Salärumfrage teilnehmen möchten, bitten wir Sie, die detaillierten Erfassungsunterlagen anzufordern. Zögern Sie nicht, uns anzurufen, falls Sie noch weitere Fragen haben.

SwissICT
Badstrasse 7
5401 Baden
Tel: 056 / 222 65 00
eMail: info@swissict.ch

Hans-Peter Uehli
Verbandsdirektor

Symposium

Managing Speed and Complexity

Verbandsdirektion und Programmkommission von SwissICT freuen sich, Sie zum diesjährigen Symposium INFORMATIK 2002 einzuladen

Managing Speed and Complexity

Montag, 27. Mai 2002 bis Mittwoch, 29. Mai 2002

im Seehotel Waldstätterhof, Brunnen SZ

Die Geschäftsprozesse werden immer schneller und komplexer. Von dieser Entwicklung ist die Informations- und Kommunikationstechnologie gleich zweifach betroffen: als massgebliche Antreiberin des Geschehens, aber zugleich auch als Getriebene vom eigenen Erfolg.

Das Symposium INFORMATIK 2002 nimmt sich dieser aktuellen Thematik an. An drei Tagen werden hochkarätige Fachleute aus Forschung und Praxis, darunter Prof. Frederic Vester (Autor von "Die Kunst vernetzt zu denken"), Prof. Walter Brenner (Universität St. Gallen), Jens Alder (CEO Swisscom) und Prof. Helmut Merkel (Vorstandmitglied Karstadt Quelle) interessante Antworten liefern und mit teilweise unkonventionellen Einsichten aufwarten.

Jeder Symposiumstag hat ein eigenes Schwerpunktthema. Am Montag wird die "IT-Revolution" aus der Sicht von Politik, Wirtschaft und Kultur beleuchtet. Am Dienstag wird erörtert, inwieweit die neuen Technologien den Fortschritt beschleunigen oder bremsen. Der dritte Tag steht unter dem Motto "Die Veränderung führen" und behandelt die Frage, wie sich die IT laufend auf die sich verändernden Geschäftsprozesse abstimmen lässt, ohne dabei das Tagesgeschäft aus den Augen zu verlieren.

Das alle zwei Jahre stattfindende SwissICT Symposium richtet sich an Führungskräfte von Informatik-, Organisations- und Fachabteilungen. Als Teilnehmender finden Sie hier eine Plattform für die Auseinandersetzung mit einem wichtigen Thema. Sie pflegen dabei gleichzeitig den Gedankenaustausch mit Kollegen und können auftanken, abseits vom Tagesgeschäft.

Hans-Peter Uehli
Verbandsdirektor

Einladung zur Generalversammlung

Donnerstag, 25. April 2002 um 16:15 Uhr im Marriott Hotel in Zürich

Programm

15:45 Uhr	Eintreffen der Teilnehmenden, Kaffee
16:15 Uhr	Begrüssung der Teilnehmer *Maya Lalive d'Epinay, Präsidentin*
16:20 Uhr	Gastreferat Wird die Liberalisierung der Telekommunikation zu Ende geführt? *Marc Furrer*
17:00 Uhr	Kaffeepause
17:30 Uhr	**Generalversammlung**

1 Protokoll der Generalversammlung vom 18. Mai 2000 gemäss Homepage von SwissICT www.swissict.ch
Maya Lalive d'Epinay

2 Jahresberichte 2001
Maya Lalive d'Epinay / Guido Auchli
- Jahresrückblick
- Kurzbericht der Arbeits- und Fachgruppen
- Kontakte mit anderen Verbänden
- Veranstaltungsprogramm 2002/03

3 Jahresrechnung 2001
Hans-Peter Uehli / Othmar Flück
- Erfolgsrechnung und Bilanz
- Bericht der Revisoren
- Genehmigung der Rechnung / Erteilung der Décharge

4 Vorschau 2002
Beat Schmid / Hans-Peter Uehli
- Verbandsstrategie, Grobzielsetzungen
- neue Verbandszeitschrift
- Veranstaltungen

5 Budget 2002
Hans-Peter Uehli

6 Erneuerungswahlen
Maya Lalive d'Epinay

7 Diverses
Maya Lalive d'Epinay
Beantwortung von Fragen

8 Abschluss der Generalversammlung
Maya Lalive d'Epinay

ab 18:30 Uhr Gemeinsamer Imbiss

Anmeldung

bis spätestens *15. April 2002* per Fax 056 222 65 67 oder via eMail info@swissict.ch.

Anfragen

Anfragen an die Generalversammlung sind spätestens 20 Tage vor der Versammlung schriftlich an die Geschäftsstelle einzureichen.

Rückblick und Ausblick auf die Verbandstätigkeiten von SwissICT

Im Laufe des Jahrs 2001 hat SwissICT 16 Ausbildungsveranstaltungen durchgeführt. Dabei konnten die Tagungen "Die Zukunft der IT ist web-based" und "eBusiness" sowie das "2. eGovernment Symposium" über 100 Teilnehmer verzeichnen.

Gegenüber dem Vorjahr haben an der Salärumfrage 2001 4% mehr Firmen teilgenommen. Dabei ist die Anzahl der ausgewerteten Saläre auf 19'278 (+8.2%) gestiegen.

Seit dem 1. Juli 2001 werden die Mitglieder von SwissICT durch die Mitarbeiter der neuen Geschäftstelle in Baden betreut.

Die Mitgliederbefragung hat - mit einem fantastischen Rücklauf von 38% der Firmenmitglieder und 32% der Einzelmitglieder - eine hohe Zufriedenheit der Mitglieder mit den Dienstleistungen von SwissICT attestiert. Trotzdem wir haben manch kritische Bemerkung als Anlass genommen, unsere Dienstleistungen zu überprüfen und zu verbessern.

In einer einmaligen Aktion konnten über 150 vergünstigte Abonnemente bekannter IT-Zeitschriften abgegeben werden.

SwissICT hat im Rahmen der Vernehmlassungen zum Bundesgesetz über die elektronische Signatur eine positive Stellungnahme publiziert.

Im Januar 2002 sind die neuen Internetseiten aufgeschaltet worden, die im laufenden Jahr schrittweise ergänzt und verbessert werden.

Der Expertenausschuss ist eine attraktive Plattform für den fachlichen Erfahrungsaustausch. Er hat bereits vier neue Arbeits- und Fachgruppen gegründet und es werden weitere folgen.

Trotz fusionsbedingtem Rückgang der Anzahl Mitglieder und einer hohen Nachforderung der Mehrwertsteuerverwaltung kann im Jahr 2001 eine ausgeglichene Erfolgsrechnung präsentiert werden.

Es bleibt doch noch viel zu tun um ein führender Verband der Informations- und Kommunikationstechnologie zu werden: In diesem Sinne hat der Strategieausschuss eine umfassende Verbandsstrategie entworfen, welche der Generalversammlung vom 25. April 2002 vorgelegt wird. Dazu lauten die Stichworte:

1. Gegenwert erbringen
2. Mitgliederpotenzial ausschöpfen
3. Interessen der Mitglieder vertreten
4. Konsolidierung der schweizerischen Verbandslandschaft unterstützen

Im Verbandsjahr 2002 sind 24 Ausbildungsveranstaltungen unter anderem zu den Themen EAI - Brücke zwischen Anwendungsinseln, Applikationsentwicklung mit XML, Requirements Engineering, Serverkonsolidierung unter Windows, Management von eProjekten, Methoden und Werkzeuge des Softwaretestings, eGovernment, Metriken und Benchmarking, Modellverträge im Informatikbereich, Risikomanagement in IT-Projekten, J2E versus .net, Project Management und Technologie-Update vorgesehen.

Das traditionelle dreitägige INFORMATIK Symposium 2002 dreht sich um das aktuelle Thema "Managing Speed and Complexity" und wird in der gediegenen Ambiance des Hotels Waldstätterhof in Brunnen durchgeführt.

Die Aktion für vergünstigte Abonnemente von IT-Zeitschriften wird wiederholt.

Wir rechnen damit, dass - dank der Erfassung variabler Lohnbestandteile - die Teilnehmer an der Salärumfrage 2002 ein weiteres Mal gesteigert werden können.

Für das Jahr 2002 hat sich SwissICT die folgenden Ziele gesetzt:

1 Ausbauen der Dienstleistungen zur Verbesserung des Mehrwerts für die Mitglieder
2 Ablösen der Zeitschrift INFORMATIK durch eine attraktive Verbandszeitschrift
3 Durchführen von gezielten Werbeaktionen zur Verbreiterung der Mitgliederbasis
4 Intensivieren der Medienarbeit und verankern von Verbandsname und Verbandszielen im breiten Publikum
5 Beobachten von aktuellen Themen als Grundlage für die Einflussnahme zur Verbesserung der gesetzlichen Rahmenbedingungen für die ICT-Branche
6 Übernehmen der Initiative bei Verhandlungen zur Konsolidierung der ICT-Verbandslandschaft, etwa über die Schaffung eines Dachverbandes.

Die Verbandsdirektion dankt all den Mitgliedern, die sich in den Gremien von SwissICT persönlich engagiert haben und mit ihrem Einsatz die Erfolgsgeschichte von SwissICT erst möglich gemacht haben.

SwissICT befindet sich auf dem richtigen Weg, "A Leading Swiss Association in Information & Communication Technology" zu werden.

Hans-Peter Uehli
Verbandsdirektion

Veranstaltungskalender

25.04.2002
Generalversammlung
Versammlung

7./8. Mai 2002
Leadership
Seminar

27.- 29. Mai 2002
INFORMATIK 2002 in Brunnen
Symposium

6./7. Juni 2002
Requirements Engineering
Seminar

11. Juni 2002
J2E versus .net*
Abendveranstaltung

13. Juni 2002
Serverkonsolidierung unter Windows
Abendveranstaltung

20./21. Juni 2002
Management von eProjekten
Seminar

27. Juni 2002
Softwaretesting: Methoden und Werkzeuge
Tagung

9./10. Juli 2002
Verbessern von Software-Entwicklungsprozessen
Seminar

22./23.August 2002
Entscheidungsfindung
Workshop

22. August 2002
3. eGovernment Symposium
Symposium

* = Arbeitstitel

SwissICT-Mitglieder erhalten eine ausführliche Einladung. Anmeldungen nehmen wir gerne per Fax 056 222 65 67 oder via info@swissict.ch entgegen.

Alter Wein in neuen Schläuchen?
Was steckt hinter der Metapher E-Marketplace oder die neuen Brückenbauer

Der elektronische Handel zwischen Businesspartnern ist in der Schweiz keine Vision mehr, sondern Realität. Technologische Schranken werden laufend aufgehoben und neue virtuelle Brücken für den Handel entstehen.

Im 18. Jahrhundert erlangte das Londoner Kaffeehaus Lloyds grosse Berühmtheit. Wer immer auch für seine Schiffe Fracht suchte oder aufgeben wollte, Waren anbieten und kaufen oder auch nur die aktuellsten Neuigkeiten aus Übersee erfahren wollte, fand sich dort ein. Bald waren die Händler nicht mehr nur unter sich. Schiffsladungen mussten vorfinanziert werden, Investitionen waren notwendig und so durften auch die Banken und Financiers nicht fehlen. Das Klumpenrisiko, mit einem Schiff die Weltmeere zu befahren, konnten einzelne häufig nicht mehr alleine tragen und so entstanden Gemeinschaften, welche das Risiko gemeinsam tragen – die ersten Versicherungen entstanden. All dies fand in einem kleinen Kaffeehaus – aber einem riesigen Marktplatz – seinen Anfang.

Der Gedanke des Marktplatzes - einem Ort, an dem Käufer, Verkäufer, Händler und Dienstleister zusammenkommen - legte die Grundlage für den Begriff des E-Marketplace. Eine Analogie, welche auf sehr abstrakter Ebene durchaus ihre Gültigkeit hat. Auch auf elektronischen Marktplätzen treffen sich im metaphorischen Sinne die Teilnehmer des Handels. Geht es aber um eine konkretere Vorstellung über die Rolle der elektronischen Marktplätze, so finden wir sehr unterschiedliche Interpretationen. Nicht zuletzt deshalb wurde fast jedes zweite Internet Venture der letzten Jahre im einen oder anderen Zusammenhang als Marktplatz bezeichnet. Phil Evans von der Boston Consulting Group ging 1998 sogar soweit, dass er das Internet selbst als die Agora – also den Marktplatz überhaupt – des 21. Jahrhunderts bezeichnet hat. Als Vision durchaus richtig, im Alltag des täglichen Geschäftes aber nicht geeignet, um die Rolle und Funktionalität von elektronischen Marktplätzen einzuordnen. Was Phil Evans aber richtig beschrieben hat, ist die Tatsache, dass die mit Hilfe der elektronischen Kommunikation und dem Internet eine Trennung von physischen Gütern und Information stattfindet. Marktplätze im klassischen Sinne hatten ihren Ursprung darin, dass Information immer an ein physisches Produkt oder Transportmedium gekoppelt war und deshalb Wissen niemals jedermann gleichzeitig - unabhängig von dessen Aufenthaltsort - zugänglich gemacht werden konnte. Um Informationskongruenz zu erreichen, mussten sich Menschen also gleichzeitig am selben Ort aufhalten – eben auf Marktplätzen, Börsen oder in Kaffeehäusern.

Dem ist heute nicht mehr so. Information steht heute praktisch ohne Zeitverzögerung allen Interessierten zur Verfügung. Auch müssen Güter nicht mehr physisch vor Ort inspiziert werden oder Zahlungsmittel in Form von Geldnoten oder Münzen ausgetauscht werden. Alles Elemente, welche wir mit dem Begriff Marktplatz assoziieren. Hinter der Metapher E-Marketplace und speziell den B2B Marktplätzen muss sich also etwas anderes verbergen, als das Bild der Marktschreier auf einer griechischen Agora.

Beschaffen oder Einkaufen?

Im Handel zwischen Unternehmen erleben wir zwei, in ihrer Natur sehr unterschiedliche Prozesse: Die strategische Beschaffung und der operative Einkauf.

Bei der strategischen Beschaffung werden Anbieter und deren Leistung verglichen. Diese Leistungen gehen weit über das reine Produktangebot und dessen Preis hinaus. Das Wissen über den Beschaffungsmarkt und eine saubere Spezifikation der Bedürfnisse sind entscheidend. Elektronische Marktplätze können dabei zum Teil im Bereich der Informationsbeschaffung unterstützen. Das aber wohl wichtigste Element ist die Durchführung von elektronischen Ausschreibungen und “reverse Auctions”. Das heisst, dass in einer elektronischen Auktion derjenige Anbieter den Zuschlag erhält, welcher das beste Angebot (was bei weitem nicht immer nur preisgetrieben ist) unterbreitet. Dies geschieht online und real-time – ohne Austausch von Papier oder physischer Präsenz der Beteiligten in vielen zähen Verhandlungsrunden. Die für eine elektronische Auktion notwendigen Applikationen werden von spezialisierten Unternehmen betrieben – zum Beispiel “Marktplätzen”. Bieter und Käufer nutzen diese Dienstleistung und brauchen selbst keine aufwendige Infrastruktur, sondern nur einen PC mit Internet-Zugang.

Eine wesentlich gewichtigere Rolle nimmt der Marktplatz aber im operativen Einkauf bzw. in der Auftragsabwicklung ein. Dieser ist dadurch charakterisiert, dass alle strategischen Beschaffungsentscheide gefällt sind, und nur die richtigen Produkte, zur richtigen Zeit beim richtigen Lieferanten bestellt, von diesem geliefert und schliesslich bezahlt werden. Diese Abläufe sollen einfach, schnell, kostengünstig und absolut transparent sein. Um diese Effizienz zu erreichen werden in vielen Unternehmen IT Systeme eingesetzt, die von der Auftragserfassung über die Produktionsplanung bis hin zur Auslieferung und Rechnungsstellung sämtliche Prozesse unternehmensweit abbilden. Diese Systeme sind auch in der Lage, selbst wiederum Bestellungen bei Zulieferern auszulösen, wenn Lagerbestände sinken, ein Produktionszyklus geplant ist oder Bestelleingänge zunehmen.

Werden Bestellungen nicht automatisch durch Planungssysteme erzeugt, so erlauben E-Procurement Systeme eine dezentrale elektronische Beschaffung durch die Mitarbeiter eines Unternehmens. Diese flexibilisieren und vereinfachen die Bestellprozesse, stellen aber gleichzeitig sicher, dass trotzdem alle notwendigen Informationen in den Finanz- und Planungssystemen zur Verfügung stehen. Ausserdem stellen E-Procurement Applikationen das Interface zum Menschen her und binden diesen in die Bestellprozesse ein. Workflows können definiert und Produktinformationen werden nicht nur als rohe Artikelstämme, sondern in benutzerfreundlichen Katalogstrukturen präsentiert.

Die Krux in der Geschichte liegt aber an den Unternehmensgrenzen und damit an den Grenzen des Einflusses der eigenen Informatiksysteme.

Evergreen statt Internet

Mögen unternehmensinterne Prozesse schon hoch optimiert und von Informatiksystemen unterstützt sein, so finden wir an der Unternehmensgrenze ein ganz anderes Bild. Per Telefon, Brief, Fax oder bestenfalls e-mail wird bestellt. Die Lieferanten müssen die eingehenden Daten in ihrer Auftragsverwaltung erfassen und stellen Rechnung – wiederum auf Papier. Dass diese Rechnungen häufig am falschen Ort im Unternehmen ankommen, erneut manuell erfasst (häufig nicht nur einmal) und verarbeitet werden müssen, kennen wir alle aus eigener Erfahrung.

Das Problem ist nicht neu. Bereits in den 70er Jahren wurde erkannt, dass ein elektronischer Austausch von Informationen nicht an Unternehmensgrenzen halt machen sollte.

EDI (Electronic Data Interchange) war das Schlagwort. Als Transportmechanismen wurden X.25 und X.400 eingesetzt. Im EDIFACT Standard wurden die wichtigsten Geschäftsvorfälle und die dazugehörigen Datenstrukturen definiert. Insbesondere in der produzierenden Industrie und im Detailhandel wurden damit Lieferketten optimiert. Trotz vermeintlicher Standards sind viele dieser Systeme sehr proprietär und ermöglichen einzig eine Punkt zu Punkt Verbindung zwischen zwei Unternehmen. Die Implementations- und Integrationskosten sind relativ hoch und fallen für jede zusätzliche Handelsbeziehung erneut an. Ausserdem hat sich gezeigt, dass trotz aller Bemühungen um Standards, die Bedürfnisse von Industrien und einzelnen Unternehmen zu heterogen sind, als dass eine Lösung für alle passen würde.

Nun stellt sich die Frage ob nicht eben dank des Internets all diese Schwierigkeiten überwunden sind und wir problemlos mit jedem unserer Handelspartner kommunizieren können. Die Antwort ist ja und nein. Ja deshalb, weil wir alle mit dem Internet verbunden sind und dieses uns erlaubt, Dokumente und Daten standardisiert auszutauschen.

Nein deshalb, weil eben nur der Datentransport standardisiert ist. Nicht aber die Struktur der Daten. Dass jedes Unternehmen für sich wiederum spezielle Bedürfnisse hat, und dass Prozesse zwischen Unternehmen noch viel weniger standardisiert sind leuchtet ein. Genau hier kommt der B2B e-marketplace ins Spiel.

Der Marktpatz als "Hub"

Seine Aufgabe ist es, zwischen Anbietern und Käufern zu vermitteln. Er stellt sicher, dass jeder Handelspartner Informationen (also strukturierte Daten) in der Form erhält, welche seine IT Systeme verstehen. Er stellt sicher, dass Daten wirklich end-to-end ausgetauscht werden und nicht irgendwo im Internet verloren gehen. Auch erlaubt er auf Sicherheitsanforderungen einzelner Unternehmen einzugehen, ohne dass davon die anderen Handelspartner unmittelbar betroffen sind. Dies reicht von speziellen Konfigurationen der Firewalls, über die Verschlüsselung von Daten bis hin zum Aufbau eines virtual private networks. Der e-marketplace ist dafür besorgt, dass nicht jeder Verkäufer mit jedem Käufer langwierige Integrationsprojekte durchführen muss, im Laufe derer man sich auf gemeinsame Prozesse, Datenformate und IT Systeme einigen muss.

Als zentraler "Hub" reduziert ein elektronischer Marktplatz also die Komplexität bisheriger Handelsnetze entscheidend. Jeder Teilnehmer stellt nur noch eine Verbindung, nämlich diejenige zum Marktplatz her. Für die Weiterleitung sind die Architekten oder Betreiber der Marktplätze verantwortlich.

Der elektronische Marktplatz bereitet Daten und Informationen für jeden Teilnehmer in der von ihm gewünschten Form auf. Dies gilt sowohl für einmalige Ereignisse, wie das Aufbauen und Strukturieren von Katalogen, als auch für transaktionsorientierten Datenaustausch wie etwa Bestellabwicklungen und Zahlungen. Aus One-to-One-werden folglich so many (Kunde/Anbieter)-to-one (Marktplatz)-to-many (Anbieter/Kunde)-Beziehungen, welche die Kosten für alle Beteiligten reduzieren.

Fazit

Der Begriff e-marketplace ist insofern passend, als tatsächlich Anbieter, Käufer und deren Partner wie Finanzdienstleister oder Logistikunternehmen zusammengebracht werden. Die Aufgabe, die der E-Marketplace übernimmt ist aber viel weniger farbig und schillernd, als die Metapher vielleicht vermuten lässt. Es sind nämlich vielmehr IT-Systeme, die zusammengebracht, Datenstrukturen die übersetzt und Prozesse die gesteuert und überwacht werden. Der Traum des völlig transparenten Informationsaustausches ist trotz des Internets noch nicht wahr geworden, EDIFACT ist nicht tot und XML ist nicht das Esperanto der strukturierten Datenkommunikation. Nicht zuletzt deshalb positioniert sich zum Beispiel Conextrade heute mehr als Brückenbauer zwischen Unternehmen denn als Marktplatz. Die darunter liegenden Konzepte, Technologien und Architekturen sind für den Anwender letztendlich nicht entscheidend – dafür ist der Marktplatz verantwortlich. Entscheidend ist das eigentliche Ergebnis: Inseln, welche bisher nur schwierig zu erreichen waren (die IT Systeme und Prozesse der Unternehmen) finden nun Zugang zum Strassennetz des elektronischen Handels. Wir alle haben dann vielleicht wieder etwas mehr Zeit für das Kaffeehaus.

Dieser Artikel wurde erstmalig in der Sonderbeilage der NZZ vom 05.2.2002 veröffentlicht.

Dr. Thomas C. Flatt
CEO der Swisscom-Tochter Conextrade
und Vorstandsmitglied SwissICT

Assessing Readiness for (Software) Process Improvement

Hans Sassenburg

There has been an increasing interest in the application of models and standards to support quality assurance in the software industry. The growing familiarity with the Capability Maturity Model has led in recent years to large scale Software Process Improvement programs. Positive results are being achieved, but the majority of the improvement programs unfortunately die a silent death. Large investments are lost and the motivation of those involved is severely tested. Case studies have revealed a number of critical factors, which determine the success or failure of (Software) Process Improvement programs. Instead of evaluating these critical success factors only once at the start of (Software) Process Improvement programs, it is recommended to assess them periodically throughout the entire lead-time of (Software) Process Improvement programs. By regularly determining where weak points exist or may be imminent and by paying attention to these weak points in time, the probability of (S)PI programs succeeding can be substantially increased. Experiences so far in applying this method may be described as encouraging.

SPI Experiences

Working as external consultants, we have acquired a great deal of experience in various European organisations in recent years. In this respect we have encountered many badly planned implementations of SPI programs, but also some good ones. Three cases from everyday practice are discussed, each containing a description of the following points:

- the establishment of the SPI program,;
- the most important factors for the success or failure of the SPI program;
- the extent to which the SPI program will ultimately lead to structural improvements.

Case A: "Solving Today's Problems First"

The first case relates to an internationally operating industrial company. The R&D department has a matrix structure within which multidisciplinary projects are carried out. Over 250 software engineers work in the software department. At the end of 1999 our firm was called in to carry out a CMM assessment. The study was commissioned by the head of the software department but it was found that most of the problems encountered were much more general in nature: unclear and unstable system specifications, no structured project approach and little attention to the quality of the process and the product. Our findings and recommendations were presented to the Board and the management team. The findings were accepted, but we were asked for proof that the recommendations made would rapidly (read: today) lead to success. The report disappeared into a filing cabinet – unused. The failure factors here were:

- Senior management did not realise the strategic importance and the added value of software in the product. They were only surprised that software always creates problems and arrives too late.
- The organisation deliberately avoids pursuing a long-term business strategy because people believe it is impossible to forecast how the market will develop in the coming years.
- There is no willingness to invest in structural improvements of business processes: "formalisation will lead to bureaucracy and this hampers the necessary creativity".

The lack of success in starting an improvement program can in this case be attributed to senior management failing in its role.

Case B: "Quick Results"

The second case study concerns a branch of an internationally operating company. Over 200 software engineers work in the R&D department, allocated to projects in which software is by far the predominant discipline. A CMM assessment was carried out in 2000, after which we were asked to offer support in specific improvement areas in the form of consulting and workshops. The success and failure factors were:

- Management appeared to be aware of the importance of software in the various products and released both capacity and money to achieve improvements in accordance with the CMM.
- A separate group – the Software Engineering Process Group – was appointed to co-ordinate the SPI program. Unfortunately, in practice, there was only one person working full-time for the whole R&D department.
- People were not sufficiently involved in determining the bottlenecks and in thinking about improvements. Proposals for improvements were mainly written by external consultants and after a short discussion with a number of key figures were made compulsory for the entire organisation.

Although the organisation is making substantial progress, there may be some doubt about the long-term return on the capital and effort invested in the project. The primary aim is to eliminate all the findings of the assessment as quickly as possible at the lowest possible cost in order to score within the company. The organisation is also seizing every other opportunity to distinguish itself positively in the market and within the company as a whole, so that many people are becoming snowed under with extra activities. Over-working people in this way may prove counter-productive in the long term.

Case C: "Increasing Maturity as Objective"

The third case study concerns an organisation operating independently within a larger company. Various product programs are developed in different groups. In total, there are almost 100 software engineers. Stimulated by strategic statements at the highest level in the company, the organisation started a Software Process Improvement program in co-operation with our consulting. Senior management recognises the added value of software in the various products and creates the preconditions in which structural improvements are possible. In addition to the crucial role of management, there are other success factors:

- A steering committee, consisting of senior management, line management and SPI co-ordinators, has been appointed within which progress is discussed every quarter. The SPI co-ordinators play a facilitatory role and line management reports on progress.
- Formal assessments on a two-yearly basis alternate with more frequent self-assessments, as a result of which the organisation is taught to make strength/weakness analyses itself and to detect and eliminate bottlenecks.
- A number of people are released on a full-time basis to co-ordinate the SPI activities and everyone involved is regularly sent on training courses to acquire the necessary new knowledge.

The active role of management, the organisation around the SPI program, the release and training of people and the extensive provision of information are very clear factors for success. Nevertheless, the continuity of the program is in danger. As yet, the organisation has not succeeded in quantifying goals and results and bringing them into line with the overall business objectives. As a result, people get bogged down in aiming at wrong goals such as "achieving CMM level 2", so that the high investments cannot be justified.

Derived Critical Success Factors

Now, what can be learned from these case studies? It may be concluded that the success of the improvement programs is not guaranteed in any of the three situations discussed. In fact, sustainment is very doubtful. And we believe that the practical situations outlined may be said to be representative of the average situation in many other organisations. It is further evident that the critical success factors may be regarded as important in every improvement program to be started. Conversely, failure factors must be eliminated. So what exactly are the critical factors for success? A more detail analysis of the case studies discussed, as well as other practical experiences, results in the following overview.

Role of Management

The role of management is crucial in improvement programs. An understanding of the need for improvement must be translated into a clear business strategy from which concrete goals for improvements must be derived. Management undertakes to create the necessary room for investments and plays an active role in the introduction and further implementation of the improvement program.

- Awareness
  Senior and line management are aware of the need for improvement. Strength/weakness analyses have shown what the organisation's position is in the external market as well as the efficiency of the internal management of the business.
- Business Strategy
  The organisation has formulated a clear long-term strategy indicating where it aims to be within a number of years. Concrete, quantified and measurable goals for im-

Score	Evaluation Dimensions		
	Approach	Deployment	Results
0 - Poor	No awareness	None	Ineffective
1 - Weak	Recognition	Limited	Spotty results
2 - Fair	Some commitment	Inconsistent	Intuitive results
3 - Marginal	Support	Deployed	Measurable results
4 - Qualified	Total commitment	Consistent	Positive results
5 - Outstanding	Leadership	Pervasive	Expectations exceeded

Table 1: Score matrix

provement have been derived from this long-term strategy.

- Commitment
  Management plays a leading, active role in the further development of the business strategy. The necessary space for drawing up, introducing and implementing an improvement program is created so that it is clearly visible for the whole organisation. The management continues to play a leading and active role throughout the implementation.

Project Organisation

Improvement programs do not become successful simply by getting off to a good start. A project team with members drawn from all levels of the organisation must be formed to monitor progress, revise priorities if necessary and perform interim measurements (or have these performed).

- Steering Committee
  The project team, or steering committee, consists of representatives of senior management, line management and the appropriate co-ordinator(s). Where possible, representatives of other parts of the organisation are also involved as listeners or advisers.
- Progress Reviews
  Progress discussions are arranged at regular intervals – e.g. every quarter – by the co-ordinator(s). During these discussions, which are chaired by the senior management, every line manager reports on the progress achieved with respect to the plan, the expected progress in the coming period and problems and risks, which have been identified. Senior management ensures that the quantified goals are actually achieved and adopts a pro-active attitude in solving identified problems and eliminating risks.
- Assessments
  Assessments are made at various points in the improvement process. Formal assessments, carried out by external experts, are used to establish independently how far an organisation has progressed and where the priorities for improvement lie. Self-assessments are made between times so that the organisation learns to perform strength/weakness analyses itself and to identify bottlenecks and take remedial initiatives.

Resource Management

Improvement means change and will encounter resistance. A crucial role is then reserved for the co-ordinator(s) of the improvement program. The 'champions' will be released or will have to be recruited. Another way of thinking and working often results in the need to train people or provide support by means of tools. Resistance can be removed by involving people as actively as possible in the improvement program.

- Assignments
  The knowledge, experience and skills required by co-ordinators are carefully mapped out. On the basis of these profiles internal staff are released and/or external consultants are recruited. They are regarded as 'champions' and facilitators who support the organisation in deciding on and implementing improvements: they are therefore not responsible for these. They win respect on the basis of their achievements in the past.
- Training and Tools
  The co-ordinators of improvement programs are confronted with resistance at all levels of the organisation. If necessary, they are trained in change management, together with any other persons involved. The changes in the organisation may possibly lead to changes in the content of the work or to work being distributed differently. Possible training requirements are identified in good time and space is created for following the necessary training courses. The extent to which certain tools can be used to support changes is also looked at.
- Deployment
  All the parties concerned at the various levels of the organisation are actively involved in the improvement program. Everyone participates as far as possible in thinking about possible initiatives, investigating bottlenecks and formulating improved working procedures. The responsibility for these activities is placed at the lowest possible level of the organisation.

Information Sharing

Improvement programs require investments on the part of the management and the motivation of everyone involved. Progress must be regularly reported to all concerned. All other information which contributes to better understanding and motivation should be communicated by means of the available resources. The successes achieved both internally and externally should be brought to everyone's attention.

- Progress Reporting
  The improvement program is launched during a specially convened meeting, attended by everyone involved. Senior management shows its commitment by explaining the need for improvement. These interactive meetings are repeated periodically to report on progress and both senior management and line management are actively present.
- Communication
  Communication media such as notice/bulletin boards and newsletters are available and are known to everyone in order to support the required exchange of information. Their use is encouraged, but care is taken to ensure that the organisation is not swamped with information. Communication is not

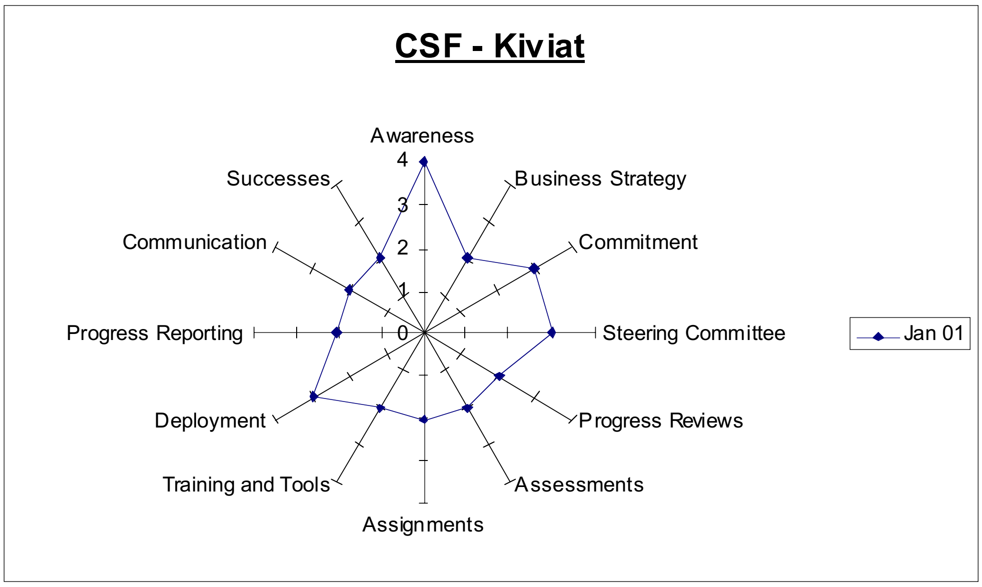


Figure 1: Example evaluation results

restricted to one's own organisation: external publicity is deliberately sought.
- Successes
  Demonstrable successes, resulting directly from the improvement program and in line with the overarching business strategy, are regularly achieved. These successes are brought clearly to the organisation's attention. Wherever it appears useful, external organisations are invited to come along and tell their success stories so that the organisation itself can learn from these and people are further motivated.

Method to Periodically Assess the Critical Success Factors

The theorist might possibly conclude that every improvement program to be started should incorporate these success factors as preconditions. The pragmatist, however, will realize that this ideal image can never be achieved and will look to see how this list of success factors can be used as an instrument in successfully starting and sustaining an improvement program. A number of organisations in which we are working has been asked to evaluate this list of success factors every quarter during the progress reviews. By way of preparation for these periodic reviews, all concerned (i.e. members of the steering committee) evaluate the success factors independently of each other by assigning a score. The score is determined by evaluating three separate dimensions (based on the CMM self-assessment method as used in Motorola[1]):

1. See: "Achieving higher SEI levels", Michael K. Daskalantonalis, Motorola, IEEE Software, July 1994

- Approach
  Criteria are the organisation's commitment to and management's support for the success factor, as well as the organisation's ability to implement the success factor.
- Deployment
  Criteria are the breadth and consistency of success factor implementation across the organisation.
- Results
  Criteria are the breadth and consistency of positive results over time and across the organisation.

Guidelines for evaluation are given in Table 1.

The score for each success factor is the average of the three separate scores for each dimension. The scores are collected by the SPI co-ordinator and presented during the Progress Reviews. The scores are discussed and compared with the required status as determined earlier. If the score of a success factor is below this actions are defined for improving the score in the next quarter. Figure 1 gives an example of a practical situation. The Kiviat-plot presents the evaluation results for January 1996.

Preliminary Results and Experiences

Five organisations were asked to try out the method. Three organisations ultimately agreed to co-operate in doing this. In the other two organisations senior management refused to co-operate: they were not convinced of the possible benefits. The three trials started on January 1995. The provisional results may be described as encouraging:

- Senior management initially appears to be achieving much more positive scores than line management and the people on the shop floor, which is leading to extensive discussions. The result is that everyone has a better understanding of each other's situation.
- In all cases, bottlenecks in the current improvement processes are revealed more quickly. This has led to extra actions, a development which is felt to be both positive and motivating by everyone involved.
- The method has proved to be easy to adapt to the specific wishes of an organisation and is more generally applicable in any randomly selected improvement processes. Success factors can be added, adjusted or omitted as required.

On the basis of the above results all the participating organisations have reported that the probability of the improvement processes ultimately succeeding has increased. This has given us confidence to start developing the method further. It is expected that an evaluation in the near future will result in a definitive set of success factors and a more finely tuned definition.

Author

Ir. J. A. Sassenburg is managing director of SE-CURE AG. In this position he advises and supports companies in setting up, introducing and guiding the implementation of extensive improvement programs. He is also guest-lecturer at the Berne University and chairman of the Swiss SPI network SPItze.

Software Process Improvement: the new 'silver bullet'?

Hans Sassenburg

Introduction

From the end of the eighties onwards it became clear that the use of information technology tools in our society would expand enormously. This trend is characterised, among other things, by revolutionary developments in the software industry. But the problems experienced in the management of projects are becoming progressively greater. Budgets are substantially exceeded, delivery times are not met, the ultimate product does not satisfy expectations and, in addition, it contains many defects. In the past, numerous 'silver bullet' solutions have been put forward to cope with the problems outlined. The use of advanced methods/techniques and the acquisition of certificates have often cost a great deal of time and money, but has seldom led to the desired result. Managers are often completely at a loss and ask themselves how these problems can be solved. In recent years an increasing interest in improving the software development process has been observable: Software Process Improvement. This article deals with the question of the extent to which this attention to the process alone is sufficient, together with an explanation based on a practical example.

The software crisis

Tumultuous developments are taking place in the software industry. Turnover in 'embedded software' – software incorporated into products such as television, audio, telephone, medical and telecommunications equipment – will increase sharply in the coming years. A growth from 10 to 40 billion dollars in Europe, say the forecasts. Observable trends:
- A substantial growth in the use and complexity of software in products and as stand-alone products.
- More demand for open systems with standardised interfaces, so that link-ups can be made to other (standard) products.
- Greater importance of hierarchical and robust architectures, aimed at future adaptations and expansions.

- Ever-larger and more highly qualified development teams, working on a geographically scattered basis.
- The availability of progressively newer techniques.
- An increasingly higher investment level for the development of new products.
- A strong growth in subcontracting projects to specialised companies (both within the Western Europe and also to low wage countries).

Running counter to these trends, within both companies and government organisations there is unrelenting pressure aimed at:

- Shortening project lead times, so that products can be introduced to the market faster.
- Increasing productivity by operating more efficiently.
- Improving customer-satisfaction by fulfilling prior agreements in which functionality, quality, costs and delivery times are laid down as precisely as possible.
- Anchoring activities in the organisation in order to search continually for and implement improvements: the self-teaching organisation (TQM).

The areas of conflicting interests, which have thus arisen, are not new. Numerous attempts have already been made in the past to find an appropriate answer. Among other things, this was sought in the use of advanced methods and techniques. The results were disappointing, however. At the end of the eighties excessive attention suddenly emerged for the IS-9001 standard, together with the appertaining ISO/9000-3 directive. Many organisations have attempted to obtain certification as quickly as possible. This has often become an end in itself and that certificate is mainly used as a commercial visiting card. As such, these certificates have made scarcely any contribution to achieving a structural improvement in the management of projects. So what is the answer?

Process approach

Even in undisciplined organisations one occasionally comes across projects which have worked extremely successfully. The success of such projects, however, is generally attributable to the heroic efforts made by a project team instead of following a disciplined procedure laid down for the organisation, and therefore applying to every project. Because of the absence of a clearly established process, future results depend entirely on the availability of these same people for a subsequent project. If success depends on the availability of people, this cannot constitute the basis for long-term continuity and improvement. That can only be achieved by establishing and continually improving those processes, which play a part in software development for the whole organisation. This idea is beginning to be more and more widely accepted and is known as: Software Process Improvement.

In 1986 the Software Engineering Institute (Pittsburg/USA) – at that time headed by Watts S. Humphrey – started developing a 'process maturity framework' to help organisations improve their software development processes. This was prompted by a request from the American government (Ministry of Defense) to supply a method for assessing suppliers with regard to their ability to implement software projects effectively. After years of experience with this framework, based on investigations carried out in many companies, this led in 1991 to the release of the Capability Maturity Model. The CMM distinguishes between five levels of maturity which an organisation may have reached. This stratified structure is based on ideas of such quality gurus as Deming, Juran and, particularly, Crosby ('Maturity Grid').

SPI in Europe

In our consulting work we increasingly come across companies which embrace the SPI concept and use the CMM as a reference framework for improving their software capability. At the start of the nineties, this was still confined to larger, internationally operating companies which were active in extensive embedded software projects. Many millions of guilders were invested in extensive improvement programmes. Slowly but surely attractive results are now beginning to become visible. Don't forget: here it is a question of invest first, earn later. Following the example of these organisations, the SPI concept gradually also found application in smaller organisations, but still in the area of 'embedded software'. Since 1995, however, we have seen a change taking place. The entire banking and insurance business is paying great attention to SPI and major improvement programmes will start up have started up since. It is, of course, interesting to see that the problems in this business are identical to those in the technical field. Our experience in numerous organisations shows that the software industry in Europe is in no way inferior to that in other parts of the world. It turns out that over 90% of the organisations known to us are in the bottom regions of the CMM. We only come across more mature organisations occasionally, and these are generally small development departments in small organisations.

Other possible solutions

The Software Process Improvement concept and the Capability Maturity Model as a reference framework are rapidly gaining in popularity. Numerous organisations are taking this process approach on board and regard it as the latest 'silver bullet' solution. Is this euphoria justified? A comparison with a completely different discipline may possibly prove instructive: the construction world.

- Control parameters

  A number of control parameters can be distinguished in the construction world which may be regarded as important for the successful completion of a building project: the quality of the construction plan, including working agreements, the availability of the necessary tradesmen and the availability of the right tools. A comparison with the software industry is given below.

Construction world	Software industry
Construction plan, incl. Working agreements	Development processes
Tradesmen	Training, experience, skills
Tools	Technology

  All the control parameters are important, both in the construction world and in the software industry. Placing too much emphasis on only one of the parameters is too one-sided and will result in sub-optimisation. After all, what good are excellent tradesmen if the construction plan displays too many defects? What is the point of the most modern tools if there are no well-trained specialists available to use them?

  Proposition 1: *"All the control parameters must be in balance with each other."*

- Requirements

  Secondly, one may ask oneself which requirements must be met by the various control parameters. Does it make sense to impose the same requirements on the construction plan, the tradesmen and the tools for building a garage as for building a new tunnel? Of course not.

  Proposition 2: *"The ultimate requirements at the level of the control parameters are determined by the characteristics of the product to be realised."*

  By analogy, in the software industry it may be said that aiming uncritically to achieve the highest CMM level is not useful for every organisation.

- Priorities

  Now let's assume that the level of all the various control parameters is not in accordance with the characteristics of the product to be realised. Which control parameter should then take priority? In the software industry at present it is said that attention to the process should take priority. But is this really true? Would you give preference to a good carpenter with a bad hammer or a poor carpenter with a good hammer?

Proposition 3: *"People constitute the most important control parameter in every discipline"*.

Opting for good tradesmen will prove to be the most justified choice in every case. When it comes to making a choice in the software industry, it is therefore probably advisable to invest in recruiting and maintaining qualified staff in addition to improving the development process or buying the latest technology.

Assessments

Over the last ten years we have increasingly come into contact with organisations which want to use SPI and CMM. The questions most frequently put to our consultants are:

- How much does a CMM assessment cost?
- How do we reach the next CMM level as quickly as possible?

On the basis of the propositions above, caution is advisable. In every organisation that asks us these questions we try to show that:

- In addition to attention for the dimension 'Process', attention to other control parameters is desirable, namely 'People' and 'Technology'
- It will be necessary to identify the characteristics of the present and future products ('Product') in order to be able to decide what requirements must be met by the various control parameters.

Only then can a sensible strength/weakness analysis of an organisation be made. In every case we have found that organisations appreciate this expansion of scope and that, in particular, the management is delighted with this. That is particularly because an attempt is also made to find out what over-arching business objectives are eventually aimed at. Recommendations resulting from an assessment should be brought into line with each other as far as possible here.

An assessment generally proceeds as follows:

- Preparation (lead time 2-4 weeks)
  In this phase the organisation to be investigated is studied on the basis of an organisation description, the available manuals and some intake interviews. Next, the scope of the research is determined in consultation with the client by selecting a number of representative projects. The persons to be interviewed during the implementation phase are then selected.
- Implementation (lead time 2-4 days)
  The interviews are held during this phase. Before they start, a kick-off presentation is given to all those directly involved. A crucial role is reserved for the client in convincing all those concerned of the need for the investigation by pointing out the main business objectives. Next, the interviews are held. These are strictly confidential and open in nature. For this reason it is preferable not to allow the client to be present at the interviews unless otherwise explicitly agreed. On average an interview lasts for 1.5 hours.
- Analysis (lead time 2-4 days)
  The information collected during the implementation phase is analysed in this phase. Firstly, on the basis of the characteristics of the product it is decided what requirements would have to be satisfied by the control parameters 'People', 'Technology' and 'Process' both in the present situation and within three to five years. After that, the maturity level of each control parameter is determined on a scale of 1 to 5. The CMM is used as a reference here for 'Process'. The risks, conclusions and recommendations are then drawn up. All this information is recorded in a report. The recommendations are generally long-term in nature, varying from one to two years.
- Reporting (lead time 1 day)
  In this phase the report is discussed with the client. First, there is a discussion with the client himself and later a presentation is given to all the persons interviewed, as well as others concerned and interested parties. Once again, a crucial role is reserved here for the client in promising everyone involved that the recommendations made will actually be followed up in practice.
- Follow-up (lead time 1 year)
  The last phase of an assessment is also the most important phase. On the basis of the recommendations made, the client will have to decide how an improvement process can be initiated and what resources will have to be made available for this purpose.

Practical example

In 1999 we carried out an assessment in a company which specialises in automation applications in the petrochemical processing industry: depot monitoring, data acquisition, process control and trend analyses. First we were asked to analyse a project which had gone off the rails and to help it to get back on course as quickly as possible. By mutual agreement, however, it was decided to make an assessment covering several projects in order to try to analyse the 'capability' of the entire software development department. The results are presented in figure 1. The characteristics of the type of product mean that the control parameters must at least be at level 3. It was found, however, that the 'People' dimension and the 'Process' dimension (CMM level 1) fell short in this respect. It proved that the so-called software developers had little or no training in information technology. Their backgrounds varied from biology to electrical engineering. The development process we encountered had not been laid down, which resulted in unrealistic planning, unclear working agreements and an ad-hoc approach to work. On the basis of the recommendations made it was then decided to start three improvement processes:

- Recruiting some experienced information technologists;
- Setting up a training program for the present development engineers;
- Improving the software development process.

Despite the high investment level required for this the company is now in a much better position to plan new projects in terms of budgeting and lead times and the quality of the end products is increasing. The number of complaints reported by customers has decreased by a factor of 4, despite a doubling of the installed base.

Conclusion

This article has dealt with the increasing attention being paid to improving the software development process. In Europe, we see an explosive growth in this interest, particularly in the banking and insurance business. By making an extrapolation to the construction world, however, an attempt has been made to show that attention to other control parameters is also important. Recruiting qualified and experienced staff and keeping them motivated probably deserves even higher priority and is at present an extremely difficult issue, given the tightness of the present labour market. In addition, intelligent use should be made of the available technology as a support in the development process. The challenge will lie in managing to find the correct balance between these different control parameters in relation to the type of product being developed. In that sense, SPI therefore cannot and must not be described as the 'silver bullet' solution. That does not exist.

Author:

Hans Sassenburg is Managing Director of SE-CURE AG (www.se-cure.ch). SE-CURE AG is a training and consulting firm in the field of Software Process Improvement and Software Architecture. He is a guest lecturer at the University of Berne (hsassenburg@se-cure.ch).