PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING RESEARCH AND PRACTICE

Volume I

Editors:
Ban Al-Ani
H. R. Arabnia
Youngsong Mun

Associate Editors:
J. L. R. Becerra, L. Chung, H. A. X. Costa, S. Dascalu, E. Grant, C. Juiz, A. Korthaus, H. Reza, M. Schader, N. Subramanian

Las Vegas, Nevada, USA
June 23-26, 2003

This set of volumes contains papers presented at the International Conference on Software Engineering Research and Practice (SERP'03). Their inclusion in this publication does not necessarily constitute endorsements by the editors or by the publisher.

Volume I ISBN: 1-932415-19-X
Volume II ISBN: 1-932415-20-3
Set ISBN: 1-932415-21-1

Printed in the U.S.A.

Foreword

It gives us great pleasure to introduce this collection of papers to be presented at the 2003 International Conference on Software Engineering Research and Practice (SERP'03), June 23 through 26, 2003, at Monte Carlo Resort, Las Vegas, Nevada, USA. The SERP'03 conference is co-sponsored and organized by Computer Science Research, Education, and Applications Press (CSREA); International Technology Institute (ITI); Korean Society for Internet Information (KSII); The World Academy of Sciences for Information Technology (WAS); and a number of computer science book publishers, usersgroups and newsgroups.

The program committee would like to thank all those who submitted papers for consideration. About 40% of the submissions were from outside the United States. Each submission was evaluated by two referees (except for papers that were directly submitted to chairs of sessions who were responsible for the evaluation of these papers.) The overall paper acceptance rate was about 39%.

We are very grateful to the many colleagues who helped in organizing the conference. In particular, we would like to thank the members of the SERP'03 Program Committee who we hope will offer their help again in organizing the next year's conference (SERP'04). The SERP'03 Program Committee members are:

Prof. Martin Schader, University of Mannheim, Mannheim, Germany
Dr. Trevor Smedley, Dalhousie University, Halifax, Canada
Dr. Nary Subramanian, Anritsu Company, Richardson, Texas, USA
Professor K.-Y. Sung, Handong University, Korea
Ladan Tahvildari (Student Member), University of Waterloo, Canada
Dr. Gregory Vert, University of Nevada, Reno, USA
Prof. Houman Younessi, Rensselaer Polytechnic Institute, USA
Dr. Andrea Zisman, City University, London, UK
Dr. Raed Abu Zitar, Philadelphia University, Amman, Jordan
Members of MultiConference Task Force for Software Engineering
Members of World Academy of Sciences Task Force for Information Technology

We would also like to thank the following: Prof. H. J. Siegel for his continued support for the MultiConference, presenting a tutorial and the MultiConference Keynote; Dr. Tim Field and Mr. Chan Field of APC for managing the printing/publication of the proceedings; April Turner, Brooke Gray and Janice Whitehead-Rhodes of APC for meeting publication and printing deadlines; Prof. Iyad Ajwa for maintaining the conference web sites; The staff of Monte Carlo Resort in Las Vegas (conference division); The staff of Universal Conference Management Systems and Support (UCMSS, San Diego, California) for the professional service they offered us, in particular, thanks to Tom Scrivner and Bobbie Carter. Last but not least, we would like to thank the SERP's Associate Editors: Jorge L. R. Becerra, Lawrence Chung, Heitor A. X. Costa, Sergiu Dascalu, E. Grant, Carlos Juiz, Axel Korthaus, Hassan Reza, Martin Schader and Nary Subramanian.

We present the proceedings of SERP'03.

H. R. Arabnia, Ban Al-Ani and Youngsong Mun
Co-Chairs, SERP'03 Program Committee

Contents

Volume I

Session: Using the J2EE Platform for Efficient Component-Oriented Development of Enterprise Systems (JEFF-CODES)

Session: Software Performance Engineering

Workshop: Adaptable Systems/Software Architectures

Session: Software Engineering: Concepts, Issues and Applications

Session: Modeling Notations, Tools and Domains + Log Files ...

Session: Software Architectures

Session: Components Based Software Engineering

Sessions: Requirements Engineering

Contents

Volume II

Session: Software Specifications: Techniques, Tools and Case Studies

Session: Object Oriented Technology + OCL

Session: Software Quality + Testing + Correctness

Session: Implementation and Users Issues + Software Maintenance

Session: Software Engineering Methods and Environments

Session: Software Reuse + Reverse Engineering + Tools

Session: Semantics

Session: Late Papers and Post-Conference Papers

Software Development: Planning x Agility

Enrico de Sousa Visconti
Escola Politécnica da USP/Microsiga
enrico.visconti@poli.usp.br

Prof. Dr. Edison Spina
Escola Politécnica da USP
edison.spina@poli.usp.br

Abstract: *This paper discusses some of the most common doubts about how much to spend in planning comparing methods from CMM to XP. Best practices are presented and evaluated for each one in the light of the final product characteristics.*

Keywords: *software development, CMM, XP, productivity, reliability*

1. Introduction

Software is an abstract entity that doesn't have physical nor electrical properties, like weight, volume, color or voltage. Software Engineering is, then, different from other engineering areas, it can not use the foundation of natural laws, it has to use only human conceptions [1].

The creation of a suitable mathematical model is an impossible mission. The software development process gets to an uncountable number of different proposals and enhancement discussions targeting stakeholders' satisfaction: customer needs product quality, developers better work conditions, sponsors, profit maximization.

The methodology definition is not a simple matter; there are too many variables in this process, like enterprise differences (size or profile), product application, stakeholders' stile, customer/enterprise relationship, contracts, certifications, etc. The great diversity found takes to an impossible definition of a perfect method as an absolute true.

Too many efforts are being used in this quest. Nowadays, economic globalization, harder competition and customer greater requirements to the "software industries" are growing. The customer stands for higher quality products, lower costs and better support. Customer wants post selling up grades and corrections, companies needs to maximize their investment by increasing efficiency to their processes, cutting costs and getting better productivity.

In chapter 2 one will find some historical data about software development; chapter 3 discuss Software Reliability concepts; chapter 4 presents some important concepts of Software Productivity; chapters 5 and 6 discusses quickness x planning in CMM and XP and chapter 7 presents some conclusions on it.

2. Ascension and Crisis of Software Engineering

The first computational systems emerged by the forties, and were based upon hardware and the programmings were made directly at the machine panel's bottoms [1]. The computational business ware based on hardware and there were no differences between programmers and operators. What were based on hardware went to be based on software. It was, still, a strict workmanship. By the next decade, with compilers, linkers, assemblers and programming languages, what were based on hardware became based on software relying only on inspiration and

expertise. The software got some independence upon hardware [3].

At the sixties, computational systems got a great leap. Machines became faster each day. And cheaper. Programming languages and operational systems powerful and friendly. This revolution made the application and the complexity of the computational systems grow. The complexity brought an entirely new problem: the software bugs! The customer confidence was no more the same. The so-called human failure had come to the computers and this were named "software crisis"[2].

To fight this crisis, new technology had to permit reliable software design. In 1968, the name Software Engineering was coined [1]. New techniques like structured design, object oriented and formal description were in. Almost all new developments were focused on fight human failures in software process.

A new decade, the seventies, and new technology, the multi-programming systems, another one and the personal computers fantastic growth and the nineties with the vulgarization of the networking and internet, and the software engineering had to deal with all of it [3] [4]. The software systems are growing faster than software engineering, and another crisis, the software engineering crisis. Bigger teams working in bigger projects lead to greater planning complexity.

An utopian objective for software engineering rises to minimize the computing systems paradox: *the system that works for men is created by men's work.* Software made by computers? This has been a visionary future for years. Software engineering automation is, by now, slower than systems complexity rise. Human factor is, right now, the basis for reliability and productivity in software industry.

3. Software Reliability

Foreseeing software reliability is one of the most technological challenges. It is not function of lifetime as, in software, an abstract entity; there is no wear out. Failures are not caused by any external agents, they already were there, in latency, as errors made by programmers before the usage had begun. Software reliability is focused in two directions, at the project itself and at the validation tests phase (if it is considered independent of the project it self).

During the tests phase it is possible to foresee some of the reliability indexes but during the project phase an even more abstract problem may be evaluated: the human error. This is a qualitative analysis using process quality requirement, and there will be a bond to the human reliability!

A fascinating, but complex, problem is foreseeing the human reliability and what components will influence it.

Human error, before being considered a technological or psychological matter, was considered a religious problem by using freewill (the human freedom to be resembling the Creator). Socrates said: "if men knows clearly all the facts surrounding, certainly will take the right decision"[5]. Socrates took men of the focus, blaming surrounding circumstances that took off men vision from the facts, so the knowledge of the surrounding facts got the reliability flag. Human error got to be a knowledge matter.

To get knowledge, men had to get experience, by their own or by transmitted knowledge. The bigger the man experience, the bigger is his reliability. On the other hand, too many information can cause another obstacle to take right decisions, it is important to get only relevant information into account. Getting full knowledge, on all circumstances, men could simply choose what information is relevant to optimize the solution. Error free is such an intangible

situation for men but, error minimization could be good enough.

A prospective vision put men as a random error source in a reliability model [6]. A retrospective vision studies each form of error and takes actions that makes they never happen or have its effects minimized.

Human error taxonomy took place to a preventive situation. Some of the taxonomies:

a) *Ability, rule and formal knowledge* - Rasmussen [7] - depending upon the basis to decision making. Important for identifying the root cause and permitting a corrective action.

b) *Intuitive and analytical cognition* - Hammond [8] - Studies a cognition scale for decision make process that leads to the error. Offers a tool for evaluation of which process has to be better documented;

c) *Discrete action* - Swain e Guttman [9] - Highly behavioral, analyses only action taken categorized by timing, commission, sequence and oversight.

The knowledge of human error process is the key to reliability through process re-engineering.

4. Software Productivity

Software doesn't take any raw material; it takes only labor to be built. People costs not only by their labor, but also by the training and motivation to good work. All these costs have to be compensated in the product. Neither considering using cheaper people nor bad installations, the only alternative is productivity.

Larry Moore [2] stands that the constant volume of a cube (Fig. 1) may represent software production: the product of budget by time by productivity. A bad use of resources takes to the software higher prices. Not so trained people and/or not enough time leads to re-work, which is the worst problem of all. Maintenance is 70% of all costs [10] but it is a euphemism. A lot of project is made at this phase. Adaptative or evolutive maintenance are re-work, they implement functionalities that were undertaken during the project. Even at the validation test phase (40% of costs) is to be considered as a part of project test and re-work.

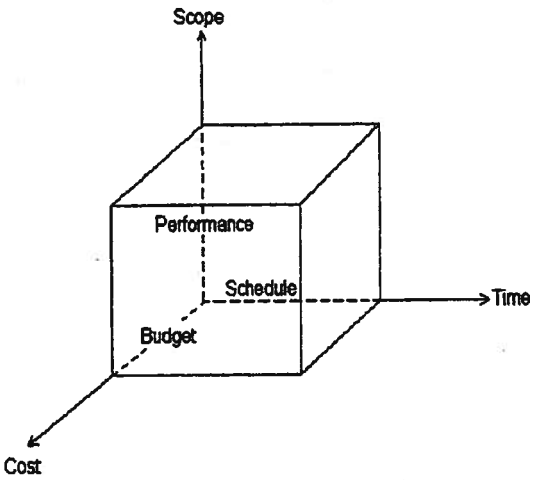


Fig.1 – Software Development Constraints

5. Planning x Quickness

The CMM (Capability Maturity Model) [11] is an example of the process managing methods from TQM (Total Quality Management). Here, the product quality is a function of the managing efforts in the process. The main CMM premise is focused in the process. Better knowledge of the processes takes to better products. DoD (US - Department of Defense) has reports great earnings from this method in software projects. From [12]: 73% lower development costs; 96% lower re-work costs; 37% lower delivery time; 80% less product defect; 21:1 ROI (Return Of Investment).

New studies, in more than 200 software projects, QSM Associates Michael Math verified that almost half of the projects had their initial requirements altered [13].

Client's wishes had more importance than formal requirements. This is a tendency that not may be ignored: rapid changes!

"We needed to know what to build rather than how we should build it" [14]. These ideas lead to the "Agile Methodology" as an alternative to the exigent and complicated consumer market. It is based on the principles [15]:

- Low information exchange cost and quick decision-making;
- Conversation (and sociability!) stead of documentation;
- The customer is a team part, what makes his desires, step by step, clearer;

It is notable that the key point of this "new" process is the human factor [15]. "Software development is a form of art" being directly affected by human capacity of developing new solutions. Based on these principles, Kent Beck has created the concept known as "Extreme Programming" (XP) in 1999. XP was motivated basically by the e-business development [16] and has become the most popular model of the "Agile Metodology".

6. Comparative Visions

These new ideas create some agitation among the software community: traditionalists against revolutionaries. Some responses to the "Agile Manifest":

"Companies focused in innovation instead of production excellence and increasing consistence, previsibility and reliability, that could be gotten without lack of efficiency" [17].

As Steven Rakitin stated, terms let at second importance are the essence, and that ones let to the left are mere excuses to support hackers irresponsible coding, without any discipline from Software Engineering [13].

Aside radicalisms, critics from revolutionary team must be listened, as problems stated are real. Care must be taken saying that XP is enough to the company success. A well-defined and restricted scope exists where these practices could be the only form of organization to the project. Some of these characteristics, that must exist, are listed below:

- *Small team* – even considering that exists some successful records on 250 people teams [15], the team size shall be between 15 and 20 people;
- *Small projects* – less than 250K lines;
- *A propitious ambient to challenges and changes*: these are things not imputable; there would be too many resistance;
- *Customer as part of the project team*: bringing the customer to be part of the solution;
- *Documentation lack*: if it were not important for your client, it would be for you, from were one could get the lessons learned? If there are nothing to be learned, one can let this aside;
- *Flexible time*: any project may have some time variations; the customer presence in the project may bring problems to other projects;
- *Individual talents*: there are no others Kent Becks available in the world to lead teams, too few people has his abilities to put people together and produce a better project than that they would do alone. [18]

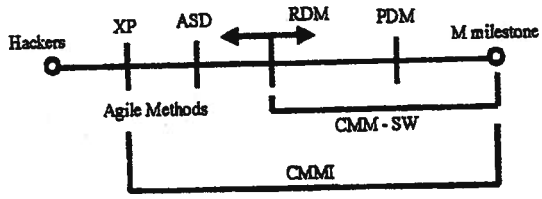
The kind of people addressed here is essential to the successes of the model presented. Without this there would exist only a group generating erratic code. With low-grade people, the problem would be even worst. A disaster! With high-grade professionals, in this model, the company would give no additional value to the project [15].

The CMM and XP models are, in some degree, complimentary models. CMM suggests good practices in key areas to get to excellence without saying how to do. A lot of practices suggested by XP may be used to get there [17]; mainly on requirements changes control as the customer would be part of the development team.

Two extremely defined sides of the spectrum are clear [18]:

- *Hackers*: no discipline, based only upon individual capacity
- *Micromilestone*: inflexible processes based on well-defined contracts, sometimes immutable.

Both sides have problems and virtues. The equilibrium on the project definition is the most important decision to make.



XP – Extreme Programming
ASD – Adaptative Software Development
RDM – Risk-Driven Models
PDM – Plan-Driven Models

Fig.2 – Software development methodologies spectrum

With this problem in mind, DoD-United States Department of Defense - decided to move efforts to CMMI - Capability Maturity Model Integration, publicized it in March 2002 integrating the various earlier CMM models. It is clear, and shown in the fig. 2 how "Agile Methods" were integrated in the new model. It made it more generalist and flexible in its application field.

7. Conclusions

As it is impossible to get a software development model immune to all human errors, these models became the most important investment in software reliability and productivity. A good interpretation of the plausible sources of human errors has to be used to lead the development in this scenario.

The most important critics to the XP model are focused in its name, as "extreme" takes to extreme measures, not always adequate and always restricted domain. The greatest value of the proposed model seems to be the strong inclusion of the customer in the project.

The discussion on "Agile Methods" x traditional models for software development methodology, that, some times, takes almost religious connotation, is an improvement factor to both of them. Criticism by the revolutionaries are the best chance to get some liveliness in the "old" methodology it self, mainly in some bureaucratic ambient like those in CMM.

CMMI contemplates this and help the way to a more productive result. Everybody has a wonderful opportunity to help finding a better way to the software development era.

8. Bibliography

[1] Pankaj Jalote - "An Integrated Approach to Software Engineering" - New York Ed. Springer-Verlag – 1991

[2] Larry Moore - "Managing Software Development to Cost and Schedule" – IEEE – 1993 – pp. 475-482

[3] Abraham Silberchatz et al. - "Operating Systems Concepts" - Addison-Wesley Publishing Company, 3 ed. – 1991

[4] Andrew S. Tanenbaum - "Redes de Computadores" - Rio de Janeiro - Editora Campus – 1994

[5] Jose M. Nieves e Andrew P. Sage - "Human and Organizational Error as a Basis for Process Reengineering: With Applications to Systems Integration Planning and Marketing" – IEEE Trans. Syst., Man, Cybern, vol. 28, no. 6, pp. 742-762, 1998

[6] W. Rouse e S. Rouse - "Analysis and Classification of Human Error" - IEEE Trans. Syst., Man, Cybern, vol. 13, no. 4, pp. 539-549, 1983

[7] J. Rasmussen, "Skill, rules, knowledge: Signal, signs and symbols; and others distinctions in human performance models" - IEEE Trans. Syst., Man, Cybern, vol. 13, no. 3, pp. 257-267, 1983

[8] K. R. Hammond, "Intuitive and Analytical Cognition: Information Models" in Concise Encyclopedia of Information Processing in Systems and Organizations, A. P. Sage, Ed. Oxford, U.K.: Pergamon, 1990, pp. 306-312

[9] A. Swain and H. Guttman, "Handbook of Human Reliability Analisys with Enphasis on Nuclear Power Plant Applications", (NUREG/CR-1278), Washington, DC: Nuclear Regulatory Commision, 1983

[10] Bertrand Meyer - "Object-Oriented Software Construction" – Ed. Prentice-Hall, 1988

[11] - Addison Wesley - "The Capability Maturity Model: guides for improving the software process" – Software Engineering Institute, 1997

[12] - DACS - "A Business Case for SPI Revised - Measuring ROI from Software Engineering and Management", Sept – 1999 http://www.dacs.dtic.mil/techs/reispi2

[13] – Jim Highsmith e Alistar Cockburn - "Agile Software Development: The Business of Innovation", Software Management - Computer, Sept. 2001, pp.120-122

[14] – Niloy Banerjee e Sukrit Bhattacharya - "Creating an Agile Software Development Organization: A key Factor for Survival in Today's Economy", IEEE, 2002, pp.230-233

[15] – Jim Highsmith e Alistar Cockburn - "Agile Software Development: The People Factor", Software Management - Computer, Nov. 2001, pp.131-133

[16] – Arthur English - "Extreme Programming: It's Worth a Look", IEEE – IT Pro, May|June 2002, pp.48-50

[17] – Mark. C. Paulk - "Extreme Programming from a CMM Perspective", IEEE – Software, Nov|Dec 2001, pp.1-8

[18] – Barry Bohem - "Get Ready for Agile Methods, with Care", IEEE – Software, Jan 2002, pp.64-69