

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher

UPGRADE is published on behalf of CEPIs (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by NOVÁTICA <http://www.ati.es/novatica/>, journal of the Spanish CEPIs society ATI (Asociación de Técnicos de Informática <http://www.ati.es/>).

UPGRADE is also published in Spanish (full issue printed, some articles online) by NOVÁTICA, and in Italian (abstracts and some articles online) by the Italian CEPIs society ALSI <http://www.alsi.it> and the Italian IT portal Tecnoteca <http://www.tecnoteca.it/>.

UPGRADE was created in October 2000 by CEPIs and was first published by NOVÁTICA and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

Editorial Team

Chief Editor: Rafael Fernández Calvo, Spain rfoalvo@ati.es
Associate Editors:

- François Louis Nicolet, Switzerland, nicolet@acm.org
- Roberto Carniel, Italy, carniel@dgt.uniud.it

Editorial Board

Prof. Wolfried Stucky, CEPIs President
Fernando Piera Gómez and
Rafael Fernández Calvo, ATI (Spain)
François Louis Nicolet, SI (Switzerland)
Roberto Carniel, ALSI – Tecnoteca (Italy)

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

Cover page designed by Antonio Crespo Foix, © ATI 2003

Layout: Pascale Schürmann

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

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

Upgrade Newsletter available at

<http://www.upgrade-cepis.org/pages/editinfo.html#newsletter>

Copyright

© NOVÁTICA 2003. All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, write to the editors.

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

Next issue (Oct. 2003):
“e-Learning – Borderless
Education”

Software Engineering – State of an Art

Guest Editor: Luis Fernández-Sanz

Joint issue with NOVÁTICA

- 2 Presentation: Software Engineering. A Dream Coming True?
– Luis Fernández-Sanz

The guest editor presents the issue, that focuses on a really broad field like Software Engineering (SE) which has been driving the evolution of software development since the late sixties of the past century. The papers cover different areas of interest related to the application of engineering principles to software development and maintenance. As usual, a list of useful references is also included for those interested in knowing more about this subject.

- 5 Software Project Management. Adding Stakeholder Metrics to Agile Projects
– Tom Gilb

In this article the author offers an analysis of the implications of the new agile methods in the field of software development.

- 10 Model-Driven Development and UML 2.0. The End of Programming as We Know It? – Morgan Björkander

This paper focuses on the idea of a truly model-driven software and analyses the influence that the new version of UML (Unified Modeling Language) is having on this process.

- 15 Component-Based Software Engineering – Alejandra Cechich and Mario Piattini-Velthuis

This paper studies the important role that components play in the field of Software Engineering.

- 21 An Overview of Software Quality – Margaret Ross

The author reviews important issues concerning quality in software development and also deals with the issues of users with disabilities and the influence of legislation regulating this aspect.

- 26 Lessons Learned in Software Process Improvement – José-Antonio Calvo-Manzano Villalón, Gonzalo Cuevas-Agustín, Tomás San Feliu-Gilabert, Antonio de Amescua-Seco, M^a Magdalena Arcilla-Cobián, and José-Antonio Cerrada-Somolinos

This paper describes the lessons learned by SOMEPRO, a Software Engineering R & D group in the Universidad Politécnica de Madrid, in more than ten software process improvement projects.

- 30 A New Method for Simultaneous Application of ISO/IEC 15504 and ISO 9001:2000 in Software SME's – Antònia Mas-Pichaco and Esperança Amengual-Alcover

The authors offer their view, based on practical experiences, of the always thorny problem of applying best software development practices to organizations where resources are especially limited.

- 37 Empirically-based Software Engineering – Martin Shepperd

This paper presents an overview of empirical Software Engineering and its implications for practitioners and researchers in four areas (object-orientation, inspections, formal specification and project failure factors.)

- 42 Software Engineering Professionalism – Luis Fernández-Sanz and María-José García-García

The aim of this paper is to provide a brief overview of what goes into making up our true perception of software engineers as specialised professionals within the field of Information Technologies.

- 47 Searching for the Holy Grail of Software Engineering – Robert L. Glass

In this article, the author defends eclecticism in development methods and the contribution that Software Engineering should make in this respect whenever the nature of a project demands flexible methods in order to be successful.

- 49 Free Software Engineering: A Field to Explore – Jesús M. González-Barahona and Gregorio Robles

This article analyses the existing points of contact between Software Engineering and the development of free software, and puts forward a few future lines of research in this respect.

Software Project Management. Adding Stakeholder Metrics to Agile Projects

Tom Gilb

Agile methods need to include stakeholder metrics in order to ensure that projects focus better on the critical requirements, and that projects are better able to measure their achievements, and to adapt to feedback. This paper presents a short, simple defined process for Evolutionary project management (Evo), and discusses its key features.

Keywords: agile methods, metrics, project management, project stakeholders.

1 Introduction

A British Computer Society Review 2001 paper indicated that only 13% of 1,027 surveyed IT projects were ‘successful’ [1]. Another report indicates that although there has been some recent improvement, 23% of their surveyed projects were considered total failures and only 28% totally successful (on time, within budget and all functionality) [2] (extracts from Extreme Chaos 2001, Standish Report). The US Department of Defence, a few years ago, estimated [3], that about half its software projects failed. This represents an improvement on the 75% reported for failed DoD projects when the Waterfall Method dominated [4], but it is of extreme concern. We must be doing something very wrong. What can senior management and IT project management do about this situation in practice?

Some people recommend complex development process standards such as CMM, CMMI, SPICE and their like. I am not convinced that these are ‘good medicine’ for even very large systems engineering projects, and certainly they are overly complex for most IT projects.

Other people recommend agile programming methods – these are closer to my heart – but maybe, for non-trivial projects - they are currently ‘too simple’?

I believe agile [1] methods would benefit, if they included ‘stakeholder metrics’. All projects, even agile projects, need to identify, and focus on, the ‘top few’ critical stakeholder requirements. These top requirements need to be quantified and measurable in practice. Quantified management is a necessary minimum to control all but the smallest upgrade efforts.

In addition to the benefits of focus on the critical requirements and measurement of progress, ‘feedback’ is a third feature of using stakeholder metrics. This is one that agile projects, especially, can utilize.

In this paper, I shall present a simple, updated ‘agile’, evolutionary project management process and explain the benefits of a more focused, quantified approach.

The evolutionary project management process, which I would recommend, is shown in Table 1.

This process and policy capture all the key features: you need read no more! However, in case any reader would like more detail – here it is! I will comment on the process and policy definition, statement by statement.

2 Project Process Description

1. Gather from all the stakeholders the top few (5 to 20) most critical goals that the project needs to deliver.

Projects need to learn to focus on *all stakeholders* that arguably can affect the success or failure. The *needs* of all these stakeholders must be *determined* – by any useful methods – and *converted* into project *requirements*. By contrast, the typical agile model focuses on a user/customer ‘in the next room’. Good enough if they *were* the only stakeholder, but disastrous for most real projects, where the critical stakeholders are more varied in type and number. Agile processes, due to this dangerously narrow requirements focus, risk outright failure, even if the ‘customer’ gets all *their* needs fulfilled.

Tom Gilb is a consultant, author and teacher. He has published nine books and numerous papers. Immediately forthcoming is his latest book “Competitive Engineering.” He primarily works at changing systems engineering cultures in major multinationals. His major technical interests are in the areas of requirements engineering, design and architecture, evolutionary project management and specification quality control (Inspection). His clients include McDonnell Douglas/Boeing, BAE Systems, Hewlett Packard, Nokia, Sony/Ericsson, Philips, CitiGroup, Intel, Microsoft, Canon, and United Defense. He does pro-bono work for US DoD, UK MoD, various charitable organizations, and in developing countries (for example, India, China, Korea).

He started working for IBM in 1958 for 5 years and has been an independent consultant since then. He was born in California and lives in Norway. <<http://www.gilb.com>>, <Tom@Gilb.com>

A Simple Evolutionary Project Management Method

Tag: Quantified Simple Evo Project. Version: July 8, 2003

(3). Owner: <Tom@Gilb.com>. Status: Draft.

Project Process Description

1. Gather from all the key stakeholders the top few (5 to 20) most critical performance (including qualities and savings) goals that the project needs to deliver. Give each goal a reference name (a tag).
2. For each goal, define a scale of measure and a ‘final’ goal level. For example: *Reliability: Scale: Mean Time Between Failure, Goal: >1 month.*
3. Define approximately 4 budgets for your most limited resources (for example, time, people, money, and equipment).
4. Write up these plans for the goals and budgets (*Try to ensure this is kept to only one page.*)
5. Negotiate with the key stakeholders to formally agree the goals and budgets.
6. Plan to deliver some benefit (that is, progress towards the goals) in *weekly* (or shorter) increments (Evo steps). See Figure 1.
7. Implement the project in Evo steps. Report to project sponsors after each Evo step (weekly, or shorter) with

your best available estimates or measures, for each performance goal and each resource budget.

- *On a single page*, summarize the *progress to date* towards achieving the goals and the costs incurred.
- Based on numeric feedback, and stakeholder feedback; *change whatever needs to be changed to reach goals.*

8. When all goals are reached: “Claim success and move on” [Gerstner, 2002]. Free the remaining resources for more profitable ventures

Project Policy

1. The project manager, and the project, will be judged exclusively on the relationship of progress towards achieving the goals versus the amounts of the budgets used. The project team will do anything legal and ethical to deliver the goal levels within the budgets.
2. The team will be paid and rewarded for ‘benefits delivered’ in relation to cost.
3. The team will find their own work process, and their own design.
4. As experience dictates, the team will be free to suggest to the project sponsors (stakeholders) adjustments to ‘more realistic levels’ of the goals and budgets.

Table 1: The evolutionary project management process.

2. For each goal, define a scale of measure and a ‘final’ goal level. For example: *Reliability: Scale: Mean Time Before Failure, Goal: >1 month.*

Using Evo, a project is initially defined in terms of clearly stated, quantified, critical objectives. Agile methods do not have any such quantification concept. The problem is that vague targets with no quantification and lacking in deadlines do not count as true goals: they are not measurable, and not testable ideas.

Note in Evo, during a project, requirements can be changed, and tuned, based on practical experience, insights gained, external pressures and feedback from each Evo step. They are not cast in concrete, even though they are extremely specific.

3. Define approximately 4 budgets for your most limited resources (for example, time, people, money, and equipment).

Conventional methods do set financial and staffing budgets, but usually at too macro a level. They do not seem to directly, and in detail, manage the array of limited resources we have. Admittedly there are some such mechanisms in place in agile methods, such as the incremental weekly (or so) development cycle (which handles time). However, the Evo method sets an explicit numeric budget for any useful set of limited resources.

But Evo does not stop there! Evo cycles estimate and then record, actual resource use, and analyse the deviation (between estimate and actual use), on every Evo cycle, in order to under-

stand and control the economics of the project – concurrently with measuring and monitoring the performance characteristics. This is an essential distinction between incremental and evolutionary development methods. Evolutionary methods measure and react to this feedback.

4. Write up these plans for the goals and budgets (*Ensure this is kept to only one page.*)

All the key quantified performance targets and resource budgets, are presented simultaneously on a single overview page. Additional detail about them can, of course, be captured off of this one ‘focus’ page.

5. Negotiate with the key stakeholders to formally agree the goals and budgets.

Once the requirements, the version derived from our developer’s understanding of stakeholder needs, are clearly articulated – we need to go back to the real stakeholders and check that they agree with our ‘clear’ (but potentially incorrect or outdated) interpretation.

It is also certainly a wise precaution to *check* back later, during the project evolution, with the *specific stakeholders* who will be impacted with a *particular* Evo step,

- as to how they feel about a particular choice of step content (design) –
(that impacts the performance and cost aspect estimates):

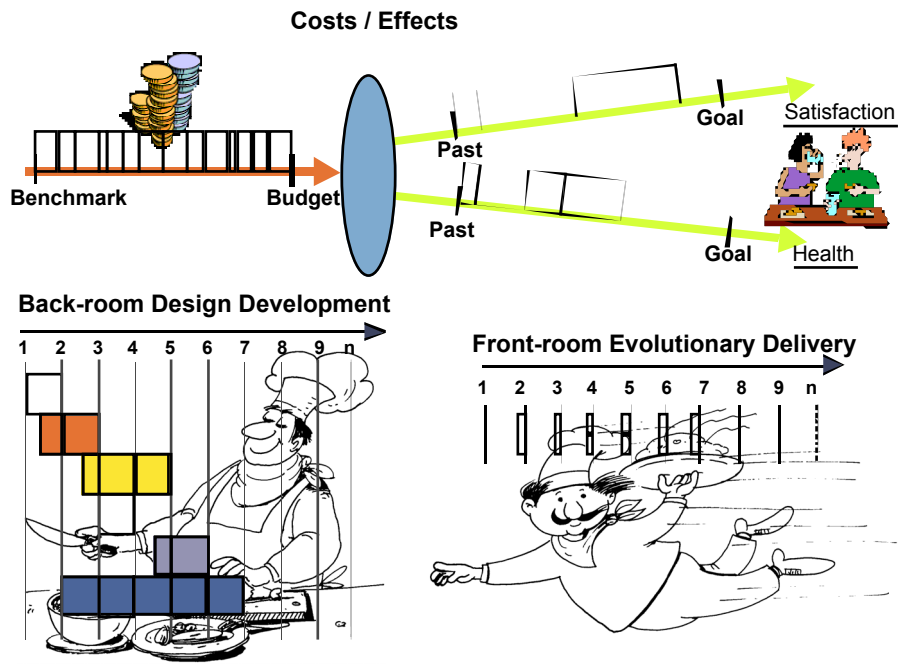


Figure 1: Evolutionary result delivery takes system components readied for integration in the ‘backroom’ using arbitrary acquisition duration (as in kitchens), and presents them to stakeholders in frequent short Evolutionary result delivery cycles (as in the dining room). (Illustration. by courtesy of Kai Gilb)

- whether estimates are realistic in the real current implementation environment?,
- and to check for any new insights regarding the long term requirements.

6. Plan to deliver some benefit (that is, ‘progress towards the goals’) in weekly (or shorter) increments (Evo steps).

A weekly delivery cycle is adopted by agile methods; this is good. However, the notion of measurement each cycle, on multiple performance and resource requirements, is absent.

Choice of the next Evo step is based on highest stakeholder value to cost ratios. It is not simply, “What shall we do next?” It is “What is most effective to do next – of highest value to the stakeholders with consideration of resources?”

The agile methods’ notion of *agreeing with a user*, about function to be built, during that weekly cycle is healthy, but the Evo method is focused on systematic, weekly cycle, measured delivery towards long-range higher-level objectives, within numeric, multiple, resource constraints. This means that the Evo method is more clearly focused on the wider stakeholder set values, and on total resource cost management.

The Evo method is NOT focused on simply writing code (‘we are programmers, therefore we write code’). The Evo method is focused on delivering useful results to an organically whole system. We reuse, buy or exploit existing code just as happily as writing our own code. We build databases, train and motivate users, improve hardware, telecommunications, create websites, improve working environment, and/or improve motivation. So we become more like systems engineers (‘any tech-

nology to deliver the results!’), than programmers (‘what can we code for you today?’).

7. Implement the project in Evo steps. Report to project sponsors after each Evo step (weekly, or shorter) with your best available estimates or measures, for each performance goal and each resource budget.

- On a single page, summarize the progress to date towards achieving the goals and the costs incurred. See Figure 2.
- Based on numeric feedback, and stakeholder feedback; change whatever needs to be changed to reach goals.

All agile methods agree that the development needs to be done in short, frequent, delivery cycles.

The Evo method, specifically insists that the closed loop control of each cycle is:

- done by numeric pre-cycle estimates,
- end-cycle measurements,
- analysis of deviation from estimates,
- and appropriate change to immediate planned cycles,
- to estimates,
- and to stakeholder expectation management (‘this is going to late, if we don’t do X’).

It is the use of stakeholder metrics that allows Evo to have such control.

The clear intention to react to the feedback from the metrics and to react to any changes in stakeholder requirements is a major feature of Evo. It helps ensure the project is kept ‘on track’ and it ensures relevance. See Figure 3.

		Estimate		Actual		Estimate		Actual	
		Step 12 Buttons.Rubber				Step 13 Buttons.Shape & Layout			
Goals		Impacts				Impacts			
1	USER-FRIENDLINESS.LEARN 30 by one year 5	-10	33%	-5	17%	-5	20%	5	-20%
2	RELIABILITY 99 by one year 200	-3	-3%	-1	-1%	20	20%	2	2%
Resources		Impacts				Impacts			
	PROJECT-BUDGET 2500 by one year 100000	2000	2%	2500	3%	1000	1%	1000	1%

Figure 2: The use of an Impact Estimation table [6][7] to plan and track critical performance and cost characteristics of a system (Illustration courtesy of Kai Gilb). The pair of numbers in the three left hand columns (30, 5 etc.) are defined benchmarks (30, 99, 2500) and Goal levels (5, 200, 100,000). The ‘%’ figures are the real scale impacts (like 20) converted to a % of the way from benchmark to the Goal levels (like 20% of the distance from benchmark to Goal). <[http:// www.gilb.com](http://www.gilb.com)>

8. When all Goals are reached: ‘Claim success and move on’ [8] (L. V. Gerstner, Jr. Retired Chairman and CEO, IBM). Free remaining resources for more profitable ventures.

A major advantage with numeric goal and budget levels, compared to a stream of ‘yellow stickies from users’ (an Agile method reference), is that it is quite clear when your objectives are reached within budgets. The set of numeric goal-and-budget levels define the success level, so ‘success’ is well defined formally in advance.

Projects need to be evaluated on ‘performance delivered’ in relation to ‘resources used’. This is a measure of project management ‘efficiency’. When targets are reached, we need to

avoid misusing resources to deliver more than is required. No additional effort should be expended to improve upon an objective, unless a new improved target level is set.

3 Project Policy

1. The project manager, and the project, will be judged exclusively on the relationship of progress towards achieving the goals versus the amounts of the budgets used. The project team will do anything legal and ethical to deliver the goal levels within the budgets.

Projects need to be judged primarily on their ability to meet critical performance characteristics, in a timely and profitable way. This cannot be expected if the project team is paid ‘by effort expended’.

2. The team will be paid and rewarded for benefits delivered in relation to cost.

Teams need to be paid by results delivered in relationship to costs. By their project efficiency. Even if this means that super efficient teams get terribly rich! And failure teams go ‘bankrupt’. Long live the capitalist free market mechanism!

When only 13% of 1027 IT projects are ‘successful’ [2], we clearly need to find better mechanisms for rewarding success, and for not rewarding failure. I suggest that sharp numeric definition of success levels (for example, Goal [China, End 2005]: 65%), and consequent rewards for reaching them, is minimum appropriate behaviour for any software project.

3. The team will find their own work process and their own design.

Agile methods believe we need to reduce unnecessarily cumbersome corporate mandated processes. I agree.

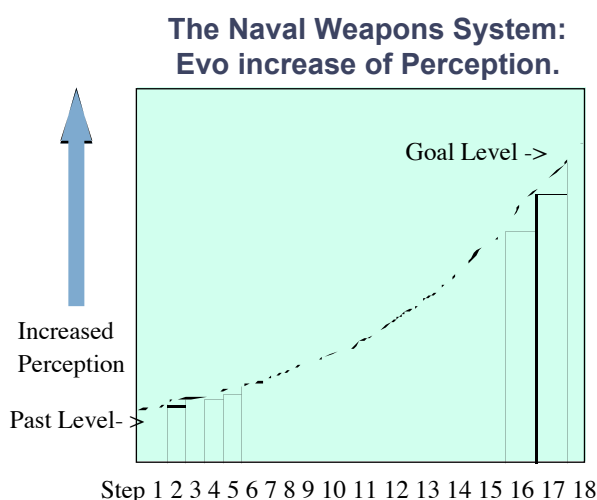


Figure 3: Results are cumulated numerically step by step until the Goal level is reached. In a UK Radar system (according to the author experience), the system was delivered by gradually building database info about planes and ships, tuning recognition logic, and tuning the radar hardware.

They also believe in empowering the project team to find the processes, designs and methods that really work for them locally. I heartily agree! But I believe that sharp numeric definition of objectives and budgets, coupled with frequent estimation and measurement of progress, is a clearly superior mechanism for enabling this empowerment. The price for this, a few estimates and measures weekly, seems a small price to pay for superior control over project efficiency.

4. As experience dictates, the team will be free to suggest to the project 'sponsors' (one type of stakeholder) adjustments to 'more realistic levels' of the goals and budgets.

No project team should be 'stuck' with trying to satisfy unrealistic or conflicting stakeholder dreams within constrained resources. The project team can only be charged with delivering inside the 'state of the art' performance levels at inside the 'state of the art' costs. Exceeding 'state of the art' performance is likely to incur 'exponential' costs.

4 Summary

A number of 'agile' methods have appeared, trying to simplify project management and systems implementation. They have all missed the central points, namely, quantification and feedback. Evolutionary project management (Evo) uses *quantified* feedback about critical goals and budgets. It also insists that early, frequent, small, high stakeholder value deliveries (Evo steps) are made to real users. This allows better focus, more measurement of progress, and more flexibility to change. It is time agile methods adopted quantified, critical stakeholder metrics.

References

- [1] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta. Agile Software Development Methods. Review and Analysis, VTT Publications, 2002, <<http://www.inf.vtt.fi/pdf/>>.
- [2] A. Taylor. IT projects sink or swim, BCS Review 2001, <<http://www.bcs.org.uk/review/2001/html/p061.htm>>.
- [3] J. Johnson, K. D. Boucher, K. Connors and J. Robinson. Collaborating on Project Success, Software Magazine, February 2001. <<http://www.softwaremag.com/L.cfm?Doc=archive/2001feb/CollaborativeMgt.html>>.
- [4] Personal Communication, Norm Brown, SPMN/Navy.
- [5] C. Larman and V. Basili. 'Iterative and Incremental Development: A Brief History', IEEE Computer, June, 2003, pp 2–11.
- [6] T. Gilb. Competitive Engineering. A Handbook for Systems & Software Engineering Management using Planguage, Addison-Wesley, 2004
- [7] T. Gilb. Principles of Software Engineering Management, Addison-Wesley, 1988.
- [8] L. V. Gerstner. Who Says Elephants Can't Dance? Inside IBM's Historic Turnaround, HarperCollins, 2002.

Other references

- C. Larman
Agile and Iterative Development, A Manager's Guide, Addison Wesley, 2003.
- J. Johnson
Turning Chaos into Success, Software Magazine, December 1999
<<http://www.softwaremag.com/L.cfm?Doc=archive/1999dec/Success.html>>.