Observations on Balancing Discipline and Agility

Barry Boehm
University of Southern California
boehm@sunset.usc

Richard Turner
George Washington University
rich.turner.ctr@osd.mil

Abstract

Agile development methodologies promise higher customer satisfaction, lower defect rates, faster development times and a solution to rapidly changing requirements. Plan-driven approaches promise predictability, stability, and high assurance. However, both approaches have shortcomings that, if left unaddressed, can lead to project failure. The challenge is to balance the two approaches to take advantage of their strengths and compensate for their weaknesses. We believe this can be accomplished using a risk-based approach for structuring projects to incorporate both agile and disciplined approaches in proportion to a project's needs.

This paper presents six observations drawn from our efforts to develop such an approach. We follow those observations with some practical advice to organizations seeking to integrate agile and plan-driven methods in their development process. The material presented here is drawn from our book Balancing Agility and Discipline: A Guide to the Perplexed (Addison Wesley, 2003).

1. Introduction

Our analysis of agile and disciplined methods have led to the following six observations:

1. Neither agile nor plan-driven methods provide a silver bullet.
2. Agile and plan-driven methods have home grounds where one clearly dominates the other.
3. Future trends are toward application developments that need both agility and discipline.
4. Some balanced methods are emerging.
5. It is better to build your method up than to tailor it down.
6. Methods are important, but potential silver bullets are more likely to be found in areas dealing with people, values, communications, and expectations management.

We will discuss each of these in turn. Following that elaboration we discuss how your organization can assess your current agility-discipline balance and then improve it to better fit your organization's evolving goals. Be forewarned that we will make some pretty heroic generalizations across wide swaths of agile and plan-driven methods. While we feel that each agile and plan-driven method shares some aspect of our assessment approach, we are aware that there is considerable variability in degree of applicability.

2. Observation 1: No Agile or Plan-Driven Method Silver Bullet

Neither agile nor plan-driven methods provide a methodological silver bullet that slays Fred Brooks' software engineering werewolf. [1] The nature of the werewolf concerns what Brooks calls the essential software engineering difficulties of coping with software's *complexity*, *conformity*, *changeability*, and *invisibility*. Some techniques have achieved the level of "lead bullets" in that they can slay normal wolves—that is, they can adequately solve some part of the software engineering problem. Elements of both agile and plan-driven approaches may be characterized as lead bullets.

Agile methods handle *changeability* and *invisibility* by building a shared vision of the project's objectives and strategies into each team member's shared store of tacit knowledge. But agile methods founder on handling *complexity* and to some extent *conformity*. They do not scale up to large complex projects, nor do they enforce obedience to order.

Plan-driven methods handle *conformity* and *invisibility* by investing in extensive documentation. Unfortunately, they founder on *changeability* and the increasing *complexity* represented by systems of systems and enterprise integration.

We have found that over the years, the increasingly rapid pace of change has identified a fallacy in the following assumption: If a software technique lead bullet can slay a software wolf this year, it will be able to slay the wolf's evolving offspring next year. Examples of techniques where this fallacy is apparent include

- The sequential, requirements-first waterfall process model that could work quite well for 1960s or 1970s batch, sequential, non-interactive applications.
- Pre-WYSIWYG word processing systems organized around separate edit, format, and runoff modules.
- Pre-Web book sales management systems that could not keep up with amazon.com.

Proceedings of the Agile Development Conference (ADC'03)
0-7695-2013-8/03 $17.00 © 2003 IEEE

Other examples of lead-bullet techniques with dwindling (but still important) niches are fixed-contract software management models, heavyweight formal methods, and static domain and enterprise architectures.

3. Observation 2: Agile and Plan-driven Method Home Grounds

There are definite home grounds for pure agile and pure plan-driven methods, although the actual extremes are rarely populated. There is a relationship with a method's position between the home grounds and the type of project and environment where it will most likely succeed.

We have identified five critical decision factors associated with agile and plan-driven home grounds (Table 1) and have graphically summarized them in Figure 1. Of the five axes in the polar graph, *Size* and *Criticality* are similar to the factors used by Alistair Cockburn to distinguish between the lighter-weight Crystal methods (toward the center of the graph) and heavier-weight Crystal methods (toward the periphery) [2]. The *Culture* axis reflects the reality that agile methods will succeed better in a culture that "thrives on chaos" than one that "thrives on order," and vice versa.

The other two axes are asymmetrical in that both agile and plan-driven methods are likely to succeed at one end, and only one of them is likely to succeed at the other. For *Dynamism*, agile methods are at home with both high and low rates of change, but plan-driven methods prefer low rates of change.

The *Personnel* scale refers to the extended Cockburn method skill rating scale discussed in Table 2, and places it in a framework relative to the complexity of the application. This captures the situation where one might be Level 2 in an organization developing simple application but Level 1A in an organization developing highly-complex applications. Here the asymmetry is that while plan-driven methods can work well with both high and low skill levels, agile methods require a richer mix of higher-level skills.

Table 1. Agile and plan-driven method home grounds

Characteristics	Agile	Plan-driven
Application		
Primary Goals	Rapid value; responding to change	Predictability, stability, high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent; high change; project-focused	Stable; low-change; project/organization focused
Management		
Customer Relations	Dedicated on-site customers; focused on prioritized increments	As-needed customer interactions; focused on contract provisions
Planning and Control	Internalized plans; qualitative control	Documented plans, quantitative control
Communications	Tacit interpersonal knowledge	Explicit documented knowledge
Technical		
Requirements	Prioritized informal stories and test cases; undergoing unforseeable change	Formalized project, capability, interface, quality, forseeable evolution requirements
Development	Simple design; short increments; refactoring assumed inexpensive	Extensive design; longer increments; refactoring assumed expensive
Test	Executable test cases define requirements, testing	Documented test plans and procedures
Personnel		
Customers	Dedicated, collocated CRACK* performers	CRACK* performers, not always collocated
Developers	At least 30 percent full-time Cockburn Level 2 and 3 experts; no Level 1B or -1 personnel**	50 percent Cockburn Level 3s early; 10 percent throughout; 30 percent Level 1Bs workable; no Level -1s**
Culture	Comfort and empowerment via many degrees of freedom (thriving on chaos)	Comfort and empowerment via framework of policies and procedures (thriving on order)

* Collaborative, Representative, Authorized, Committed, Knowledgable
** See Table 2. These numbers will particularly vary with the complexity of the application

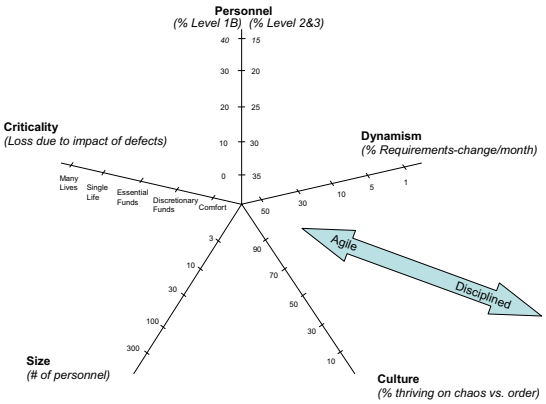

Figure 1. Dimensions affecting method selection

For example, a plan-driven project with 15 percent Level 2 and 3 people and 40 percent Level 1B people would initially use more than 15 percent Level 2 and 3 people to plan the project, but reduce the number thereafter. An agile project would have everybody working full-time, and the 15 percent Level 2 and 3s would be swamped trying to mentor the 40 percent Level 1Bs and the remaining Level 1As while trying to get their own work done as well.

By rating a project along each of the five axes, you can visually evaluate its home-ground relationships. If all the ratings are near the center, you are in agile method territory. If they are at the periphery, you will best succeed with a disciplined approach. If you are mostly in one or the other, you need to treat the exceptions as sources of risk and devise risk management approaches to address them.

4. Observation 3: Future Applications Will Need Both Agility and Discipline

In the past, there have been many small, noncritical, well-skilled, agile-culture, rapidly evolving projects occupying the agile home ground in the center of Figure 1. There have also been many people working on large, critical, mixed-skill, ordered-culture, stable projects occupying the plan-driven home ground at the periphery of the chart. However, things are changing.

Large projects can no longer count on low rates of change, and their extensive process and product plans will become expensive sources of rework and delay. As the use of agile methods progresses from individual early-adopter projects to enterprise-coupled mainstream applications, the Brooksian software development werewolves of complexity and conformity will be waiting for them. Thus, there will be a higher premium on having methods available that combine agility and discipline in situation-tailorable ways.

Table 2. Levels of software method understanding and use (after Cockburn)

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation
2	Able to tailor a method to fit a precedented new situation
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience can become Level 2.
1B	With training, able to perform procedural method steps (e.g. coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

Drawing on the three levels of understanding in Aikido (Shu-Ha-Ri), Alistair Cockburn has identified three levels of software method understanding that can help sort out what various levels of people can be expected to do within a given method framework [2]. We have taken the liberty of splitting his Level 1 to address some distinctions between agile and disciplined methods, and adding an additional level to address the problem of method-disrupters.

Level -1 people should be rapidly identified and found work to do other than performing on either agile or disciplined teams.

Level 1B people roughly correspond to the "1975-average" developer profile. They can function well in performing straightforward software development in a stable situation. But they are likely to slow down an agile team trying to cope with rapid change, particularly if they form a majority of the team. They can form a well-performing majority of a stable, well-structured disciplined team.

Level 1A people can function well on agile or disciplined teams if there are enough Level 2 people to guide them. When agilists refer to being able to succeed on agile teams with ratios of 5 Level 1 people per Level 2 person, they are generally referring to Level 1A people.

Level 2 people can function well in managing a small, precedented agile or disciplined project but need the guidance of Level 3 people on a large or unprecedented project. Some Level 2s have the capability to become Level 3s with experience. Some do not.

5. Observation 4:Some Balanced Methods Are Emerging

Some of the agile methods, such as Crystal Orange, DSDM, FDD, and Lean Development, have emerging approaches for achieving balance. The same is true of new, lighter versions of the Rational Unified Process. One interesting related approach called Code Science or AgilePlus has been used successfully on over a dozen projects up to 400 KSLOC in size. It uses most of the XP practices plus a componentized architecture, risk-based situation audits, business analyses, and on-demand automatic document generation. [3, 4]

The tailorable method defined in our book provides a risk-driven, spiral-type approach for balancing agile and plan-driven methods. It is not fully developed, but has worked well in situations where it has been applied. It is definitely not a cookbook approach; each project

will need some thought to apply it to the particular situation.

6. Observation 5: Build Your Method Up – Don't Tailor It Down

Plan-driven methods have had a tradition of developing all-inclusive approaches designed to be tailored down to fit a particular situation. Experts can do this, but non-experts tend to play it safe and use the whole thing, often at considerable unnecessary expense. Agilists offer a better approach of starting with relatively minimal sets of practices and only adding extras where they can be clearly justified by cost-benefit. In some cases, as with Crystal, they will have multiple core sets for different levels of size or criticality. Efforts are underway to develop similar approaches for building up plan-driven methods, as shown by the activities at Rational to provide more user-friendly tools to tailor the Rational Unified Process.

7. Observation 6: Focus Less On Methods – More On People, Values, Communications and Expectation Management

The agilists have it right in valuing individuals and interactions over process and tools. They are not the first to emphasize this. There is a long list of wake-up calls—Weinberg's 1971 *Psychology of Computer Programming* [5], the Scandinavian participatory design movement [6], DeMarco and Lister's 1987 *Peopleware* [7], and Curtis' studies of people factors [8] and development of the People Capability Maturity Model [9]. There is also a wealth of corroborative evidence that people factors dominate other software cost and quality drivers, such as the 1966 Grant-Sackman experiments showing 26:1 variations in people's performance [10] and the 1981 and 2000 COCOMO and COCOMO II cost model calibrations showing 10:1 effects of personnel capability, experience, and continuity [11]. But the agilists may finally provide a critical mass of voices amplifying this message.

7.1 People

Software engineering is done "of the people, by the people and for the people."

- *Of the People.* People organize themselves into teams to develop mutually satisfactory software systems.
- *By the People.* People identify what software capabilities they need, and people develop it for them.
- *For the People.* People pay the bills for software development and use the resulting products.

Unfortunately, software engineering is still struggling with a "separation of concerns" legacy that contends translating requirements into code is so hard that it must be accomplished in isolation from people concerns. A few quotes will illustrate the situation:

> *The notion of "user" cannot be precisely defined, and therefore it has no place in computer science or software engineering.* [12]
>
> *Analysis and allocation of the system requirements is not the responsibility of the software engineering group, but it is a prerequisite for their work.* [13]
>
> *Software engineering is not project management.*[14]

In today's and tomorrow's world, where software decisions increasingly drive system outcomes, this separation of concerns is increasingly harmful. Good agilist treatments of people and their ecosystems are provided in Jim Highsmith's *Agile Software Development Ecosystems* [15] and Alistair Cockburn's *Agile Software Development* [16]. Complementary plan-driven approaches are provided in Watts Humphrey's *Managing Technical People* [17] and his Personal Software Process [18] [19], as well as the People CMM developed by Bill Curtis, Bill Hefley and Sally Miller [20].

7.2 Values

Along with people come values - different values. One of the most significant and underemphasized challenges in software engineering is to reconcile different users', customers', developers', and other success-critical stakeholders' value propositions about a proposed software system into a mutually satisfactory win-win system definition and outcome. Unfortunately, software engineering is caught in a value-neutral time warp, where every requirement, use case, object, test case, and defect is considered to be equally important. Most process improvement initiatives and debates, including the silver bullet debate, are inwardly focused on improving software productivity rather than outwardly focused on delivering higher value per unit cost to stakeholders. Again, agile methods and their attention to prioritizing requirements and responding to changes in stakeholder value propositions are pushing us in more high-payoff directions. Other aspects of value-

based software engineering practices and payoffs are described in "Value-Based Software Engineering" [21].

7.3 Communications

Even with closely-knit, in-house development organizations, the "I can't express exactly what I need, but I'll know it when I see it" (IKIWISI) syndrome limits people's ability to communicate an advance set of requirements for a software system. If software definition and development occurs across organizational boundaries, even more communications work is needed to define and evolve a shared system vision and development strategy. The increasingly rapid pace of change exacerbates the problem and raises the stakes of inadequate communication. Except for the landmark people-oriented sources mentioned above and a few others, there are frustratingly few sources of guidance and insight on what kinds of communications work best in what situations. Cockburn's Agile Software Development is a particularly valuable recent source. It gets its priorities right by not discussing methods until the fourth chapter, and spending the first hundred or so pages discussing why we have problems communicating and what can be done about it. It nicely characterizes software development as a cooperative game of invention and communication, and provides numerous helpful communication concepts and techniques. Some examples are the skill levels discussed in Table 2, human success and failure modes, information radiators and convection currents, and the effects of distance on communication effectiveness. It's well worth reading whatever your location along the agility-discipline spectrum.

7.4 Expectations Management

Our bottom-line conclusion agrees with one of the major findings in a recent root-cause analysis of troubled DoD software projects [22]. It is that the differences between successful and troubled software projects is most often the difference between good and bad expectations management.

Most software people do not do well at expectations management. They have a strong desire to please and to avoid confrontation, and have little confidence in their ability to predict software project schedules and budgets. This makes them a pushover for aggressive customers and managers trying to get more software for less time and money.

The most significant common factor we've seen is that both agile and highly-disciplined plan-driven people demonstrate enough process mastery, preparation, and courage to be able to get their customers to agree to reducing functionality or increasing schedule in return for accommodating a new high-priority change. They are aware that setting up unrealistic expectations is not a win for the customers either, and they are able to convince the customers to scale back their expectations. Both agile short iterations and plan-driven productivity calibration are keys to successfully managing software expectations.

8. What Can You Do To About Balancing Agility And Discipline In Your Organization

In our era of increasingly rapid change, the most risky thing you can do is to continue with business as usual without doing a self-assessment of where your organization is, where its success-critical stakeholders want it to go, and how it will cope with future trends. Key stakeholders to consult include your users, customers, developers, suppliers, and strategic partners. Key future trends to consider include:

- The increased pace of change and need for agility;
- The increased concern with software dependability and need for discipline;
- Your ability to satisfy your stakeholders' evolving value propositions and to keep up with your toughest competitors;
- The increasing gap between supply and demand for Cockburn Level 2 and 3 people;
- Your ability to cope with existing and emerging technical challenges such as COTS integration, evolving Internet and Web capabilities, distributed and mobile operations, agent coordination, and multimode virtual collaboration.

A good context for performing such a self-assessment is provided in Jim Collins' recent book, *Good to Great* [23]. Although its primary focus is at the corporate level, its emphasis on balancing shared internal self-discipline and entrepreneurial agility can be applied at the software development organization level as well.

If you have already done such an assessment, then you are in excellent position to address the issue of how your organization should best balance agility and discipline. If not, you should at least take a first cut at a self-assessment so that you have some picture of where you are and where you want to go.

The steps below provide a simple recipe for balancing agility and discipline. Be sure, however, that

you perform them in consultation with your key stakeholders.

8.1 Step 1

Use Figure 1 to assess where your projects currently are with respect to the 5 key axes. If you have different organizations with different profiles (e.g. device-embedded software and business software), make separate assessments. Also, assess the likely changes in your organization's profile over the next 5 years.

8.2 Step 2

If your assessments show you comfortably in the agile or disciplined home ground now and in the future, your best strategy is to embark on a continuous improvement effort to become the best you can at agility or discipline. To start such an effort, the best next steps are:

a. Convene a representative working group of key stakeholders to assess alternative agile or disciplined improvement approaches and recommend an approach that best fits your situation.
b. Identify a reasonably tractable project, staffed with capable and enthusiastic people, to be trained in using the approach, to apply it, and to develop a plan for both dealing with problems encountered and for extending the approach across the organization.
c. Execute the plan for extending the approach, always including evaluation and feedback into continuous improvement during and after each project.

8.3 Step 3

If your Figure 1 assessments leave you mostly in the agile or disciplined home grounds, but with some anomalies, treat the anomalies as risk factors to be added to the charters of the groups performing steps 2a-c. Examples of potential anomalies are:

- Operating mostly in a disciplined home ground, but in an increasingly dynamic marketplace.
- Operating with agile fix-it-later developers with a growing, increasingly enterprise-integrated and dependability-oriented user base.
- Finding that your technical people are successfully adapting to dynamism, but that your contract management people are not.

The first two anomalies can be addressed via risk assessment and managerial techniques. The third would involve a more specialized approach to change management in the contracting organization, but done with their collaboration and the support of upper management.

If you have several organizations and several profiles, it is best to prioritize your approach to work on those you believe are most important and likely to achieve early successes. An exception is if there are projects in crisis that need, and are receptive to, significant help and redirection.

8.4 Step 4

If your Figure 1 assessments leave you with a highly mixed agility-discipline profile, you need to develop an incremental mixed strategy to take you from your current situation to the one you have chosen as a goal. For example, suppose that your organization primarily does 50-person, essential-funds critical projects with a mix of 20 percent Level 2 and 3 and 30 percent Level 1B personnel, with dynamism rapidly increasing from 5 percent/month to 10 percent/month, a culture only 30 percent oriented toward thriving on chaos, and a corporate steady-state goal to do all software internally. This profile is shown in Figure 2.

The current staffing profile and culture are not ideally suited to the anticipated future needs. One option for you would be to start on a long-term internal effort to upgrade your staff and change your culture. But a quicker and less risky approach would be to enter a strategic partnership with an agile methods company to serve as near-term trainers, co-developers, and mentors for your staff. This would expedite an initiative to bring as many of your Level 1A people up to Level 2 as possible, and to bring as many of your Level 1B people up to Level 1A, at least in some niche area. The agile methods company people could also serve as change agents in making your organizational culture more thrive-on-chaos oriented.

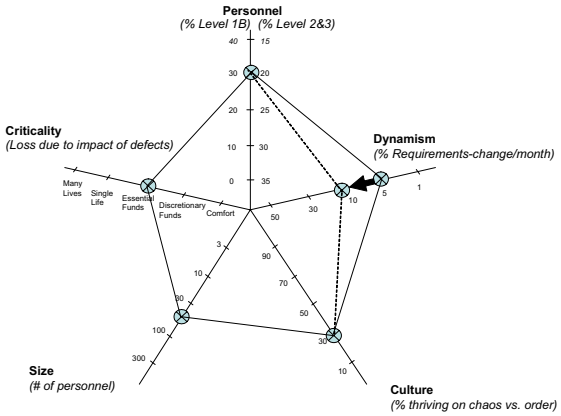

Figure 2. Sample highly-mixed profile

In other cases, you might be a growing pure-agile company with a need to add more discipline to accommodate larger and more critical products. You could employ a similar strategy with a disciplined services company to rapidly rebalance your operations, staff profile, and culture.

8.5 Step 5

Your organization should complement whatever agile/disciplined balancing options it pursues with sustained effort to improve your staff capabilities, value-oriented capabilities, and communication capabilities. It is also important to track your progress with respect to your plans and apply corrective action whenever new opportunities come up. A good checklist for staff capabilities is the People CMM. A good starting point for value-oriented capabilities is *Value-Based Software Engineering*. [24] A good mechanism for tracking multi-criteria, multi-initiative programs is the Balanced Scorecard technique. [25]

9. Conclusions

We are encouraged that the observations above show that agile and disciplined methods are two means toward the same end – satisfying customers with software that meets their needs within appropriate cost and schedule parameters. While each of the methods has a definite home ground, strategies are emerging for integrating them in such a way as to take advantage of their strengths while avoiding their weaknesses. We believe this is healthy for software development, and look forward to the innovative techniques that will grow and mature from these initial strategies.

[1] Brooks, F. "No Silver Bullet," *Information Processing 1986, Proceedings of the IFIP Tenth World Computing Conference*, ed. H.-J. Kugler (1986), pp. 1069-1076, Elsevier Science B.V., Amsterdam, The Netherlands.

[2] Cockburn, A. , *Agile Software Development*, Addison Wesley, Reading, MA, 2002.

[3] Manzo, J., "Odyssey and Other Code Science Success Stories," *CrossTalk*, October 2002, pp. 19-21, 30.

[4] Manzo, J., "Agile Development Methods, the Myths, and the Reality: A User Perspective," *Proceedings, USC-CSE Agile Methods Workshop*, March 2003 (http://sunset.usc.edu/events/past).

[5] Weinberg, G., *The Psychology of Computer Programming*, Van Nostrand-Reinhold, New York, 1971.

[6] Ehn, P. (Ed.), *Work-Oriented Design of Computer Artifacts*, Lawrence Earlbaum Associates, March 1990.

[7] DeMarco, T., T. Lister, *Peopleware: Productive Projects and Teams*, Dorset House, New York, 1999.

[8] Curtis, B., H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," Comm. ACM, 31 (11), November 1988, pp. 1268-1287.

[9] Curtis, B. et al, *People Capability Maturity Model*, Addison Wesley, Reading, MA, 2001.

[10] Grant, E. and H. Sackman, "An Exploratory Investigation of Programmer Performance Under On-Line and Off-Line Conditions," Report SP-2581, System Development Corp., September 1966.

[11] Boehm, *Software Engineering Economics*, Prentice Hall, Upper Saddle River, NJ, 1981; Boehm et al, *Software Cost Estimation with COCOMO II*, Prentice Hall, Upper Saddle River, NJ, 2000.

[12] Dijkstra, E., Panel discussion, Fourth International Conference on Software Engineering, 1979.

[13] Paulk, M. et al., *The Capability Maturity Model for Software: Guidelines for Improving the Software Process*, Addison Wesley, Reading, MA, 1994]

[14] Tucker, A., "On the Balance between Theory and Practice," *IEEE Softwar*e, Sept-Oct 2002.

[15] Highsmith, J., *Agile Software Development Ecosystems.* Boston, MA: Addison-Wesley, 2002.

[16] Cockburn, A., *Agile Software Development*, Addison Wesley, Boston, 2002.

[17] Humphrey, W., *Managing Technical People*, Addison Wesley, Boston, 1997.

[18] Humphrey, W., *A Discipline of Programming*, Addison Wesley, Boston, 1995.

[19] Humphrey, W., *Introduction to the Personal Software Process*, Addison Wesley, Boston, 1997.

[20] Curtis, B., Hefley, B., and Miller, S., *The People Capability Maturity Model*, Addison Wesley, 2001.

[21] B. Boehm, "Value-Based Software Engineering," *ACM Software Engineering Notes*, March, 2003.

[22] McGarry, J. and Charette, R., "Systemic Analysis of Assessment Results from DoD Software-intensive System Acquisitions," Tri-Service Assessment Initiative Report, Office of the Under Secretary of Defense (Acquisition, Technology, Logistics), 2003.

[23] Collins, J., *Good to Great*, HarperCollins, 2001.

[24] Boehm, B., "Value-based Software Engineering," ACM Software Engineering Notes, March, 2003.

[25] Kaplan R., D. Norton, *The Balanced Scorecard: Translating Strategy into Action,* Harvard Business School Press, Boston, MA, 1996