

# Mastering rapid delivery and change with the SAIV process model

Barry Boehm, A. Winsor Brown

## Abstract

*Ensuring on time, within-budget delivery is increasingly difficult in the information technology (IT) field because of the increasingly rapid rate of requirements volatility of IT systems under development. This paper describes the Model-Based (System) Architecting and Software Engineering (MBASE)'s Schedule as Independent Variable (SAIV) approach to this problem, and illustrates the nature of the solution with examples.*

## 1. Introduction

Future software developers will be increasingly challenged to develop on time, within-budget, on-target IT systems because of several complicating trends. These especially include increasing complexity (systems of systems, networks of networks, agents of agents), rapid change (of technology, organizations, and market conditions), and decreased control of content (via unavoidable dependence on commercial-off-the shelf (COTS) solutions). Current software development processes have serious difficulties in coping with these trends, particularly for rapid-delivery systems undergoing significant in-process change.

We have encountered these challenges in developing and evolving our MBASE approach over several years of experience in applying it to an annual series of digital library projects [1, 2]. These projects are largely web-based services developed by 5-person MS-student teams, using the MBASE Guidelines [3] and the MBASE Electronic Process Guide [4].

The teams' main challenges are to develop a Life Cycle Architecture (LCA) package, described below, for a USC Libraries client's application in 12 weeks during the fall semester; and to develop and transition an Initial Operational Capability (IOC) in 12 weeks during the spring semester. These are extreme examples of schedule being the independent variable, since the USC semester schedule is fixed and the students disappear (to graduation or summer jobs) at the end of the spring semester.

E-commerce companies have had similar challenges in delivering to short fixed schedules in a climate of rapid change. We have also had the opportunity to refine MBASE in collaboration with some of our USC-CSE E-commerce Affiliates, particularly Rational, C-Bridge, and MediaConnex.

These experiences have led to the SAIV approach described below. We begin with a short summary of the MBASE process framework. We next describe the SAIV process strategy and process elements, in the context of USC digital library examples. We conclude by summarising our SAIV experience across 19 delivered applications, and by discussing the critical success factors involved in the approach.

## 2. The MBASE Process Framework

Software projects are guided by models that they adopt (knowingly or unknowingly) to help their participants make decisions affecting the project. These models include Product models such as object models, architectures, and traditional requirements models; Process models such as lifecycle and risk management models; Property models such as cost, schedule, and performance models; and Success models such as contractual agreements, correctness, business-case analysis, and stakeholder win-win. Many of the most serious difficulties encountered by software projects can be traced to clashes among the models they have adopted [2.5].

MBASE uses a process framework in which stakeholders express their initial desired success models, and proceed to adjust these and their associated product, process, and property models to achieve a consistent and feasible set of models to guide the project and its stakeholders. The actual process, as illustrated in Figure 1, generally takes several iterations, and requires some common intermediate checkpoints. MBASE also uses an extension of the original spiral model [6] to include stakeholder win-win model negotiation and a set of common anchor point milestones [7]: key life-cycle decision points at which a project verifies that it has feasible objectives (LCO); a feasible life-cycle architecture and plan (LCA); and a product ready for operational use (IOC).

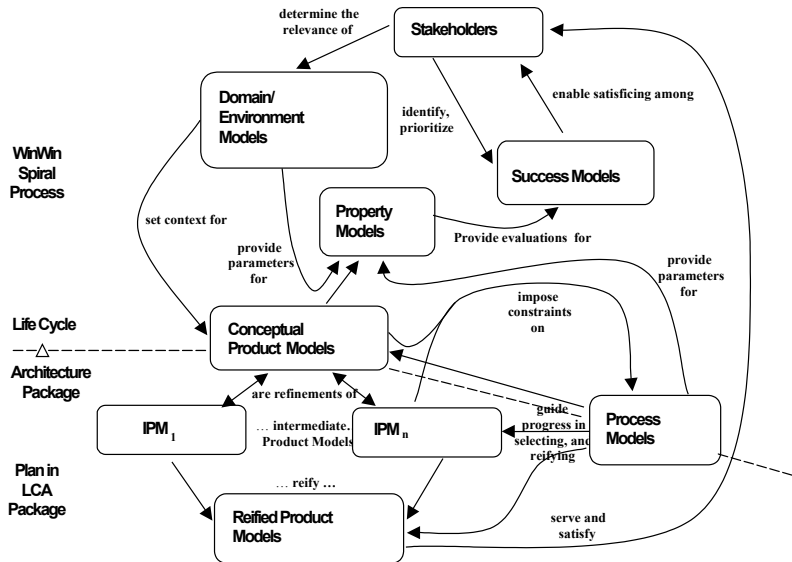


Figure 1: MBASE Process Framework

Thus, if the overriding top-priority success model is to “Demonstrate a competitive agent-based data mining system on the floor of COMDEX in 9 months,” this constrains the ambition level of other success models (provably correct code, fully documented). It also determines many aspects of the product model (architected to easily shed lower-priority features if necessary to meet schedule), the process model (SAIV), and various property models (only portable and reliable enough to achieve a successful demonstration).

The achievability of the success model needs to be verified with respect to the other models. In the 9-month demonstration example, a cost-schedule estimation model would use various product characteristics (sizing of components, reuse, product complexity), process characteristics (staff capabilities and experience, tool support, process maturity), and property characteristics (required reliability, cost constraints) to determine whether the product capabilities achievable in 9 months would be sufficiently competitive for the success models. Thus, as shown within Figure 1, a cost and schedule property model would be used for the evaluation and analysis of the consistency of the system’s product, process, and success models. If they are shown to be consistent, the project passes its LCA milestone and follows the process plan in the LCA package to refine the architecture into an operational product.

In other cases, the success model would make a process model or a product model the primary driver for model integration. An IKIWISI (I’ll know it when I see it) success model

for a small application would initially establish a prototyping and evolutionary development process model, with most of the product features and property levels left to be determined in the process. A success model focused on developing a product line of similar products would initially focus on product models (domain models, product line architectures), with process models and property models subsequently explored to perform a business-case analysis of the most appropriate breadth of the product line and the timing for introducing individual products.

MBASE differs from Model-Based Systems Engineering (MBSE) [8], in that MBSE concentrates almost exclusively on product models (and their associated property models). This is also the case for the Software Engineering Institute's Model-Based Software Engineering [9] and Honeywell's Model-Based Software Development [10] approaches.

MBASE is most compatible with the Rational Unified Process [11,12,13], which has adopted the MBASE anchor point milestones. MBASE has adopted Rational's Inception/Elaboration/Construction/Transition phase definitions for the activities between the milestones.

### **3. The SAIV Process Model**

The SAIV Process model provides a general version of the process described for the fixed-schedule e-commerce project above. It consists of six major steps:

1. Shared vision and expectations management
2. Feature prioritisation
3. Schedule range estimation
4. Architecture and core capabilities determination
5. Incremental development
6. Change and progress monitoring and control

#### **3.1. Shared Vision and Expectations Management**

As graphically described in Death March [14], many software projects lose the opportunity to assure a rapid, on-time delivery by inflating client expectations and overpromising on delivered capabilities. The first step in the SAIV process model is to avoid this by obtaining stakeholder agreement that meeting a fixed schedule for delivering the system's Initial Operational Capability (IOC) is the most critical objective, and that the other objectives such as the IOC feature content can be variable, subject to meeting acceptable levels of quality and post-IOC scalability.

Often, the librarians and computer science students have unrealistic expectations about what is easy or hard for each other to do. We have found that providing them with lists of developer and client "simplifiers and complicators" improves their ability to converge on a realistic set of expectations for the delivered system [15]. The resulting shared vision enables the stakeholders to rapidly renegotiate the requirements as they encounter changing conditions.

#### **3.2. Feature Prioritisation**

With MBASE at USC, stakeholders use the USC/GroupSystems.com EasyWinWin requirements negotiation tool to converge on a mutually satisfactory (win-win) set of project requirements. One step in this process involves the stakeholders prioritising the requirements by assessing their relative importance and difficulty, each on a scale of 0 to 10. This process is carried out in parallel with initial system prototyping, which helps ensure that the priority assessments are realistic.

### **3.3. Schedule Range Estimation**

The developers then use a mix of expert judgement and parametric cost modelling to determine how many of the top-priority features can be developed in 24 weeks under optimistic and pessimistic assumptions. For the parametric model, we use COCOMO II, which estimates 90% confidence limits on both cost and schedule [16]. Other models such as SLIM [17], SEER [18], and Knowledge PLAN [19] provide similar capabilities.

### **3.4. Architecture and Core Capability Determination**

The most serious mistake a project can make at this point is just to pick the topmost-priority features with 90% confidence of being developed in 24 weeks. This can cause two main problems: producing an IOC with an incoherent and incompatible set of features; and delivering these without an underlying architecture supporting easy scalability up to the full feature set and workload.

First, the core capability must be selected so that its features add up to a coherent and workable end-to-end operational capability. Second, the remainder of the lower-priority IOC requirements and subsequent evolution requirements must be used in determining a system architecture facilitating evolution to full operational capability. Still the best approach for achieving this is to encapsulate the foreseeable sources of change within modules [20].

### **3.5. Incremental Development**

Since the core capability has only a 90% assurance of being completed in 24 weeks, this means that about 10% of the time, the project will just be able to deliver the core capabilities in 24 weeks, perhaps with some extra effort or occasionally by further reducing the top-priority feature set. In the most likely case, however, the project will achieve its core capability with about 20-30% of the schedule remaining. This time can then be used to add the next-highest priority features into the IOC (again, assuming that the system has been architected to facilitate this).

An important step at this point is to provide the operational stakeholders (users, operators, maintainers) with a Core Capability Demonstration. Often, this is the first point at which the realities of actually taking delivery of and living with the new system hit home, and their priorities for the remaining capabilities may change.

Also, this is an excellent point for the stakeholders to reconfirm the likely final IOC content, and to synchronize plans for conversion, training, installation and cutover from current operations to the new IOC.

### **3.6. Change and Progress Monitoring and Control**

As progress is being monitored with respect to plans, there are three major sources of change, which may require reevaluation and modification of the project's plans:

1. Schedule slips. Traditionally, these can happen because of unforeseen technical difficulties, staffing difficulties, customer or supplier delays, etc.
2. Requirements changes. These may include changes in priorities, changes in current requirements, or needs for new high-priority requirements.
3. Project changes. These may include staffing changes, COTS changes, or new marketing-related tasks (e.g., interim sponsor demos).

In some cases, these changes can be accommodated within the existing plans. If not, there is a need to rapidly renegotiate and restructure the plans. If this involves the addition of new tasks on the project's critical path, some other tasks on the critical path must be reduced or eliminated. There are several options for doing this, including dropping or deferring lower-

priority features, reusing existing software, or adding expert personnel. In no cases should new critical-path tasks be added without adjustments in the delivery schedule.

### **3.7. SAIV and Time-Boxing**

The SAIV process model differs from classic time-boxing, in which small fixed increments of capability are assigned to fixed-length time boxes. This works fine in the nominal case in which no unforeseen difficulties or significant changes are encountered. But it has no flexibility built into its stakeholder expectations, its feature prioritisation, or its product architecture to enable it to adapt to difficulties which break one of the time boxes. The more adaptive form of time boxing used in Adaptive Software Development [21] comes closer to the SAIV approach, but is considerably more informal. The SAIV approach described here tries to steer the project toward a balance of flexibility and discipline with a low risk of failure.

## **4. SAIV Experience**

### **4.1. USC Digital Library Projects**

The USC digital library projects [1,22] use the MBASE approach. To elaborate on its top-level description in Figure 1, it involves the concurrent development of several initial artefacts: an Operational Concept Description, a Requirements Definition, an Architecture Description, a Life Cycle Plan, a Feasibility Rationale, and one or more prototypes. These are evaluated at two major pass/fail points, the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) milestones. Both milestones use the same primary pass-fail criterion:

- If we build the system to the given architecture, it will satisfy the requirements, support the operational concept, be faithful to the prototypes, and be buildable within the processes, budgets, and schedules in the plan.

For the LCO milestone, this criterion must be satisfied for at least one choice of architecture, along with demonstration of a viable business case for the system and the expressed concurrence of all the success-critical stakeholders. For the LCA milestone, the pass-fail criterion must be satisfied for the specific choice of architecture and COTS components to be used for the system, along with continued business case viability and stakeholder concurrence, plus elimination of all major project risks or coverage of the risks in a risk management plan.

One of our primary goals in the project course is to give the students experience in risk management [23]. Our risk management lectures and homework exercises emphasize a list of the ten most serious risk items: personnel risks are number 1, and budget-schedule risks are number 2. The student projects' risk management plans must show how their team will avoid the risks of delivering an unsatisfactory Life Cycle Architecture package in the first 12 weeks (fall semester), and of unsatisfactorily delivering and transitioning an Initial Operational Capability (IOC) in the second 12 weeks (spring semester). The MBASE Guidelines recommend that they adopt the SAIV model described in Section 3; so far, all the projects have done this.

Also, we work in advance with the USC Library clients to sensitise them to the risks of overspecifying their set of desired IOC features, and to emphasize the importance of prioritising their desired capabilities. This generally leads to a highly collaborative win-win negotiation of prioritised capabilities, and subsequently to a mutually satisfactory core capability to be developed as a low-risk minimal IOC.

The projects' monitoring and control activities include:

- Development of a top-N project risk item list which is reviewed and updated weekly to track progress in managing risks (N is usually between 5 and 10).
- Inclusion of the top-N risk item list in the project's weekly status report.
- Management and technical reviews at several key milestones
- Client reviews at other client-critical milestones such as the Core Capability Demonstration.

The use of SAIV and these monitoring and control practices have led to on-time, client-satisfactory delivery and transition of 17 of the 19 products developed to date. One of the two failures was in our first year, when we tried to satisfy three clients by merging their image archive applications into a single project, and underestimated the complexity of the merge. As a result, "merging multiple applications" has become one of the major sources of project risk that we consider.

The second failure happened recently when a project which appeared to be on track at its Transition Readiness Review, simply did not implement its transition plan when its client suddenly had to go out of town. We were not aware of this until the client returned after the semester was over and the students had disappeared to graduation and summer jobs. We have since revised our system of closeout reviews to eliminate this "blind spot" and related problem sources.

On the other 17 projects, client evaluations have been uniformly quite positive, averaging about 4.4 on a scale of 1 to 5. A particularly frequent client evaluation comment has been their pleasure in being able to synchronise product transition on a specific fixed date with their other transition activities. The digital library artefacts can be reviewed on the class web page, <http://sunset.usc.edu/classes>.

## 4.2. E-Commerce Projects

One of our industrial affiliates, C-Bridge, Inc., uses a very similar SAIV process model, which enables them to consistently deliver e-commerce systems on fixed schedules between 16 and 26 weeks. Their Rapid Value™ approach uses milestones very similar to MBASE's LCO, LCA, and IOC milestones; their counterpart phases are named Define, Design, Develop, and Deploy. They use similar approaches in working in advance with their clients to ensure a workable SAIV scope and schedule, and in anticipating and pre-working potential transition problems to client-based operations and maintenance [24].

## 5. SAIV Limitations and Extensions

### 5.1 SAIV Limitations

The primary limitation of the SAIV approach can be visualised in terms of the canonical software production function shown in Figure 2. It is slightly modified and updated from [25, p. 193]. As with most production functions, it is an S-shaped curve with three segments:

- An investment segment, in which necessary infrastructure capabilities are developed, but very little value-generating applications capability is developed.
- A high-payoff segment, in which incremental investments in applications capability produce significant added value to the organisation.
- A diminishing returns segment, in which incremental investments in applications capability produce decreasingly small added value to the organisation.

For any given project duration T, and for any given set of project parameters such as the COCOMO II cost drivers, there is a smaller set of capabilities that can be developed with 90% confidence of completion, and a larger set of capabilities that can be developed with

50% confidence of completion. These are shown in Figure 2 with respect to two example project durations, T = 12 months and T = 6 months.

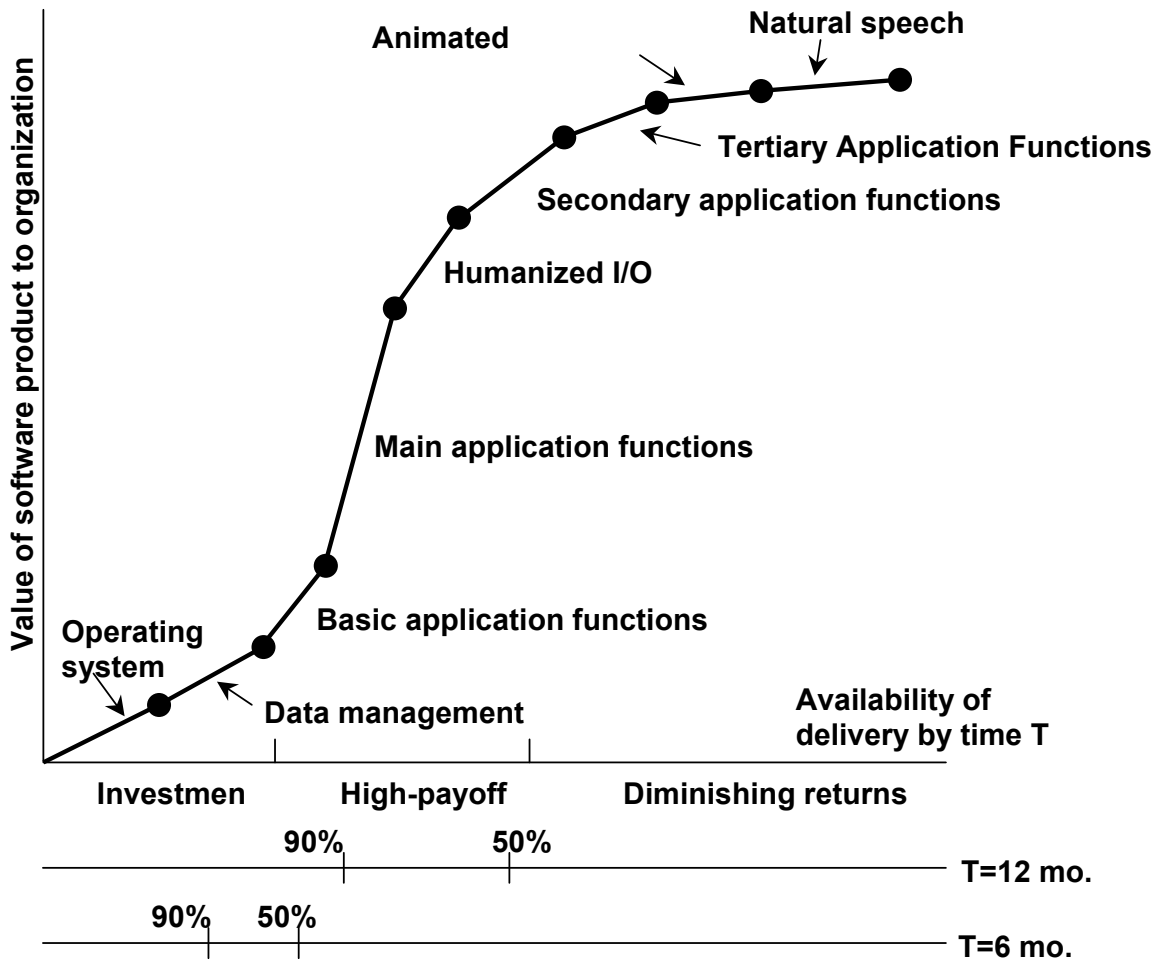


Figure 2. Canonical production for software product features

For the 12-month project duration, the value profile is compatible with a SAIV approach, since even at a conservative 90% completion confidence level, enough applications software is achievable within 12 months to produce appreciable value, and the value goes up significantly at the 50% confidence level. Thus, one could plan for an increment 1 core capability that (sometimes with some extra effort or descoping) would produce acceptable value even in the worst case.

However, for the 6-month project duration, even the 50% confidence level feature set does not reach an adequate value to be worth the investment, and the 90% core capability does even worse. Thus, we see that Step 3 of the SAIV approach, Schedule Range Estimation, actually includes a decision branch that terminates the process if an inadequate core capability results from the analysis. With both the USC digital library projects and the C-Bridge e-commerce applications, an exploratory effort with the clients determines whether a project is worth pursuing within a desired time period.

## 5.2 SAIV Extensions

In some cases, particularly if the infrastructure development is considered strategically important and the specific applications choices are unclear, an organisation may still consider the options furnished by the infrastructure to be of sufficient value to invest. Providing a decision framework for such options is one of the attractive prospective extensions of the SAIV approach [26].

Another attractive prospective extension discussed in [26] is the concept of using realised value as the basis of the project monitoring and control feedback cycle. In this case, not only is project progress monitored, but also the counterpart initiatives and assumptions linking the developed software applications to the realised benefits. These benefits are quantified in the business case for the application; the relation of the business case to counterpart initiatives and assumptions can be developed using the Results Chain in the DMR Benefits Realisation Approach [27]. This would enable projects to use earned-value systems related to real value for project monitoring and control, as compared to current “earned value” systems, which actually monitor and control budgeted cost and schedule rather than realised business value.

## 6. Conclusions

The six-step SAIV process model presented here has been used successfully on 17 of 19 digital library applications at USC, and on a similar percentage of e-commerce applications at C-Bridge, to deliver highly client-satisfactory applications on a fixed schedule in a climate of rapid change. Its critical success factors are:

- Working with stakeholders in advance to achieve a shared product vision and realistic expectations;
- Getting clients to develop and maintain prioritised requirements;
- Scoping the core capability to fit within the high-payoff segment of the application’s production function for the given schedule;
- Architecting the system for ease of adding and dropping features;
- Disciplined progress monitoring and corrective action to counter schedule threats.

The approach can also be applied to its counterpart Cost As Independent Variable (CAIV) process model. It can also provide a way to transform the current dilemma, “Cost, Schedule, Quality: Pick Any Two,” to “Cost, Schedule, Quality: Pick All Three.” Attractive future extensions include options-based software design and development and realised-value based monitoring and control.

## 7. References

- [1] Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A. and Madachy, R., “Using the WinWin Spiral Model: A Case Study”, *IEEE Computer*, July 1998, pp. 33-44.
- [2] Boehm, B. and Port, D., “Escaping the Software Tar Pit: Model Clashes and How to Avoid Them,” *ACM Software Engineering Notes*, January 1999, pp. 36-48.
- [3] Boehm, B., Port, D., Abi-Antoun, M. and Egyed, A., “Guidelines for Model-Based Architecting and Software Engineering (MBASE)” version 2.2, USC-CSE, (Feb.2001), <http://sunset.usc.edu/Research/MBASE>
- [4] Mehta, N., “MBASE Electronic Process Guide,” USC-CSE, September 1999, <http://sunset.usc.edu/Research/MBASE>
- [5] Boehm, B., Port, D. and Al-Said, M., “Avoiding the Software Model-Clash Spiderweb,” *IEEE Computer*, November 2000, pp. 120-122.

- [6] Boehm, B., "A Spiral Model of Software Development and Enhancement," IEEE Computer, May 1998, pp. 61-72.
- [7] Boehm, B., "Anchoring the Software Process," IEEE Software, July 1996, pp. 73-82.
- [8] Fisher, J. et al., "Model-Based Systems Engineering: A New Paradigm," INCOSE INSIGHT, October 1998, pp. 3-16.
- [9] Gargaro, A. and Peterson, A.S., "Transitioning a Model-Based Software Engineering Architectural Style to Ada 95," SEI Technical Report CMU/SEI-96-TR-016, 1996.
- [10] Honeywell Technology Center, "Model-Based Software Development," Course Announcement, Minneapolis, MN, 1998.
- [11] Jacobson, I., Booch, G. and Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999.
- [12] Kruchten, P., The Rational Unified Process, Addison-Wesley, 1998.
- [13] Royce, W.E., Software Project Management: A Unified Framework, Addison-Wesley, 1998.
- [14] Yourdon, E., "Death March," Prentice Hall, 1997.
- [15] Boehm, B., Abi-Antoun, M., Kwan, J., Lynch, A. and Port, D., "Requirements Engineering, Expectations Management, and the Two Cultures," Proceedings, 1999 International Conference on Requirements Engineering, June 1999.
- [16] Boehm, B., Abts, C., Brown, A.W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D. and Steece, B., Software Cost Estimation with COCOMO II, Prentice Hall, 2000.
- [17] Putnam, L. "Software Life Cycle Model (SLIM)," QSM, 2001, <http://www.qsm.com>
- [18] Galorath, D., "SEER-SEM," Galorath, Inc., 2001, <http://www.galorath.com>
- [19] Jones, C., "Knowledge PLAN," Artemis/SPR, 2001, <http://www.spr.com>
- [20] Parnas, D., "Designing Software for Ease of Extension and Contraction," IEEE Trans. Software Engr., March 1979, pp. 128-137.
- [21] Highsmith, J., "Adaptive Software Development," Dorset House, 1999.
- [22] Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J. and Madachy, R., "A Stakeholder Win-Win Approach to Software Engineering Education", Annals of Software Engineering, April 1999.
- [23] Port, D. and Boehm, B. "Educating Software Engineering Students to Manage Risk," Proceedings, ICSE 2001, May 2001.
- [24] Madachy, R., Pan, A. and Arboleda, A., "Processes for Rapid Development of Internet Applications," LA SPIN Presentation, January 24, 2001.
- [25] Boehm, B., "Software Engineering Economics," Prentice Hall, 1981.
- [26] Boehm, B., and Sullivan, K., "Software Economics: A Roadmap," in A. Finkelstein (ed.), "Software Engineering Futures," Proceedings, ICSE 2000, June 2000.
- [27] Thorp, J., "The Information Paradox," McGraw Hill, 1998.