

The Schedule as Independent Variable (SAIV) Process for Acquisition Software-Intensive Systems

Barry Boehm, Winsor Brown, LiGuo Huang
University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781

Dan Port
University of Hawaii

Abstract

In this article, we show how you can use the MBASE process framework to generate a family of acquisition process models for delivering user-satisfactory systems under schedule, cost, and quality constraints. We present the six major steps of the Schedule/Cost/Schedule-Cost-Quality as Independent Variable (SAIV/CAIV/SCQAIV) process using SAIV and a representative Department of Defense (DoD) Command, Control, and Communications Interoperability application as context. We then summarize our experience in using SAIV on 26 University of Southern California electronic services projects, followed by discussions of SAIV/CAIV/SCQAIV application in the commercial and defense sectors, of model application within the DoD acquisition framework, and of the resulting conclusions.

Introduction

In our earlier Cross Talk articles on the Spiral Model (Boehm, Hansen 2001) and Model-Based (System) Architecting and Software Engineering (MBASE) (Boehm, Port 2001), we showed that these were actually process model generators for the acquisition of software intensive systems. They use risk considerations to determine the most appropriate sequence of activities to perform (among specification, prototyping, simulation, benchmarking, increments of development, etc.) in order to achieve the most cost-effective system capability within various resource constraints such as cost, schedule, personnel, and platform characteristics.

In this article, we show how you can use the MBASE process framework to generate a particularly attractive family of acquisition process models for delivering user-satisfactory systems under such constraints. Many software-intensive systems acquisitions in DOD and elsewhere try to fix the required capabilities, and then hopefully deliver them within a pre-committed set of cost and schedule constraints. This has been shown to be a highly risky approach.

For example, the Standish Report (Standish) found that 84% of the software-intensive system projects it surveyed either overran their budgets and schedules or were cancelled before completion. The average overruns on these projects were 189% of planned cost and 222% of planned schedule. And the completed overrun projects delivered an average of only 61% of the originally specified features.

The risk-driven MBASE-Spiral approach uses this overrun risk to invert the software-intensive-system acquisition process. Either schedule, cost, or some combination of

cost, schedule, and quality becomes the independent variable, and the lower-priority features become the dependent variable. This requires several sub-processes:

1. Determination of a top-priority core capability and quality level strongly assured to be achievable within the schedule-cost-quality constraints;
2. User expectations management and continuing update of feature priorities;
3. Architecting the system for ease of dropping borderline-priority features and future addition of lower-priority features;
4. Careful progress monitoring and corrective action to keep within cost-schedule-quality constraints.

Given current commercial time-to-market pressures, a number of more-and-less-successful versions of the Schedule as Independent Variable (SAIV) approach are practiced in the commercial sector. Within DOD, the Cost as Independent Variable (CAIV) approach is currently used to determine a set of required features achievable for a given cost, but the resulting cost and feature set is then usually fixed during system acquisition. With evolutionary acquisition, the ability to use spiral development approaches is encouraged, and often a fixed time period of 18 or 24 months is desired for early increments. The SAIV model is particularly well-suited for this objective. However, further adjustments in system acquisition, source selection, and contracting processes are needed to enable its use in competitive procurements.

In this article, we next review the SAIV-relevant portions of the MBASE process framework. We then present the six major steps of the SAIV/CAIV/SCQAIV process, and relate them to previous rapid-development processes such as design-to-schedule and timeboxing. We then summarize our experience in using SAIV on 26 USC electronic services projects, 24 of which have successfully delivered systems with high client-satisfaction ratings on a fixed schedule. This is followed by discussions of SAIV/CAIV/SCQAIV application in the commercial and defense sectors, of model limitations and extensions, and of the resulting conclusions.

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 (Boehm, Port 1999, Boehm et. al 2000b).

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 (Boehm 1998) to include stakeholder win-win model negotiation and a set of common anchor point milestones (Boehm 1996): key life-cycle decision points at which a project verifies that it has feasible life-cycle objectives (LCO); a feasible life-cycle architecture and plan (LCA); and a product ready for operational use (IOC). More detailed descriptions and examples of the MBASE

process framework are available in our previous Cross Talk article, "Balancing Discipline and Flexibility with the Spiral Model and MBASE"(Boehm, Port, 2001).

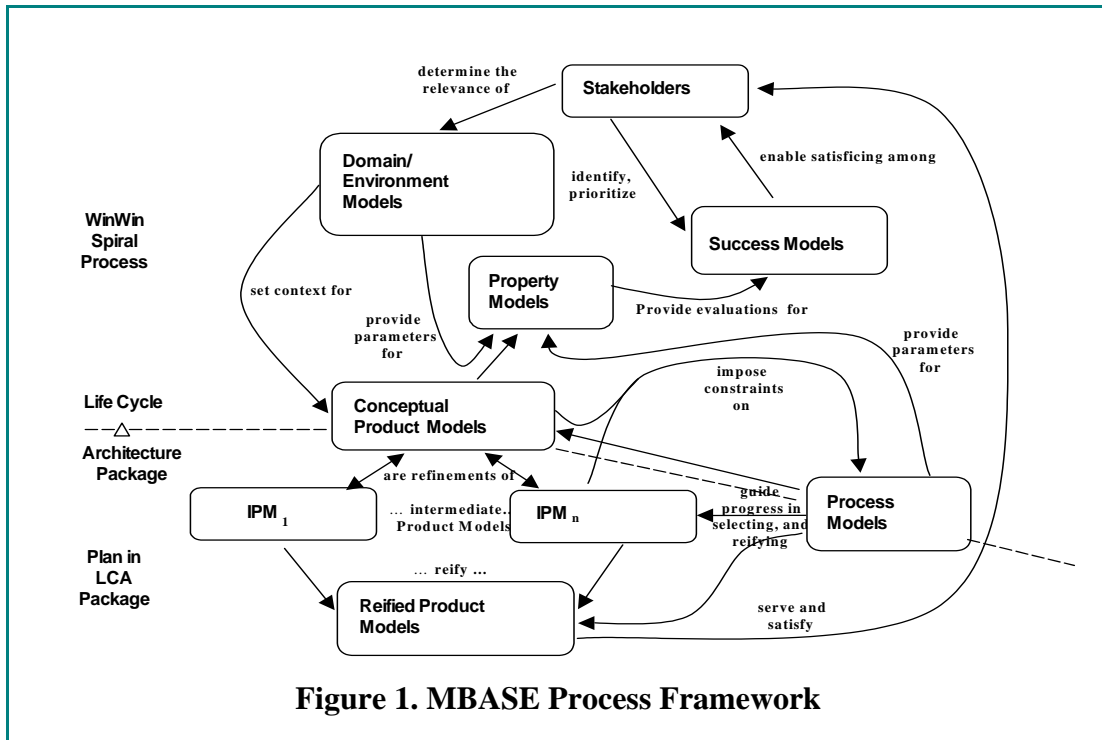


Figure 1. MBASE Process Framework

MBASE differs from Model-Based Systems Engineering (MBSE) (Fisher 1998), 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 (Gargaro, Peterson 1996) and Honeywell’s Model-Based Software Development (Honeywell 1998) approaches.

MBASE is most compatible with the Rational Unified Process (Jacobson et al. 1999, Kruchten 1998, Royce 1998), 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.

The SAIV Process Model

The SAIV Process model is described in terms of a representative set of SAIV applications: USC’s annual series of e-services projects (Boehm et. al 1998, Boehm et. al 1999b). These projects are largely web-based applications developed by 5-person MS-student teams, using the MBASE Guidelines (Boehm et. al 2001a) and the MBASE Electronic Process Guide (Metha 1999).

The teams’ main challenges are to develop a Life Cycle Objectives (LCO) package and a Life Cycle Architecture (LCA) package, described below, for a USC Information Services Division client’s application in 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.

Figure 2 shows how the general MBASE process framework in Figure 1 is mapped onto the SAIV version of the win-win spiral model. The process models for CAIV and SCQAIV are essentially the same except for the definition of the radial dimension of the spirals. For CAIV projects, the spiral's traditional radial dimension of cumulative cost is used. For SAIV projects, the radial dimension is cumulative calendar time, with the USC SAIV project milestones occurring at 7 weeks for LCO, 12 weeks for LCA, 24 weeks for IOC, and a Core Capability milestone around 18-20 weeks, depending on the application. The USC e-services projects are pre-screened and expectations-managed with clients to ensure that acceptable capabilities can be developed within these constraints.

Figure 2 also shows the major SAIV process elements to be described next. These are executed concurrently within the spirals. As discussed in our spiral model article (Boehm, Hansen 2001), feedback and iteration of previous-cycle results are part of the spiral process, but are omitted from Figure 2 for simplicity.

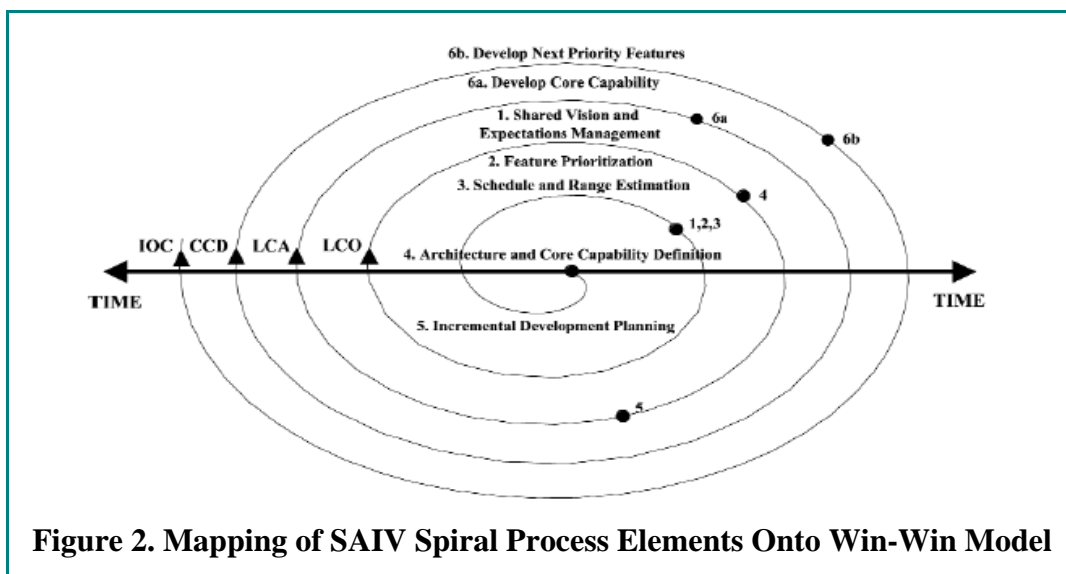


Figure 2. Mapping of SAIV Spiral Process Elements Onto Win-Win Model

Shared Vision and Expectations Management. As graphically described in Death March (Yourdon 1997), 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 or success model in Figure 1, 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 USC librarians and other Information Services personnel and the 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 (Boehm et. al 1999a). The resulting shared vision enables the stakeholders to rapidly renegotiate the requirements as they encounter changing conditions.

Feature Prioritization. With MBASE at USC, stakeholders use the USC/GroupSystems.com EasyWinWin requirements negotiation tool (Boehm et. al 2001b) to converge on a mutually

satisfactory (win-win) set of project requirements. One step in this process involves the stakeholders prioritizing 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, and that the domain model and product model in Figure 1 are feasibly scoped.

Schedule Range Estimation. The developers then use a mix of expert judgement and parametric cost modelling (the property models in Figure 1) 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 (Boehm et. al 2000a). Other models such as SLIM (Putnam 2001), SEER (Galorath 2001), and Knowledge PLAN (Jones 2001) provide similar capabilities.

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 use the Parnas information-hiding approach to encapsulate the foreseeable sources of change within modules (Parnas 1979). The architecting process may take two or more win-win spiral cycles of prototyping, COTS product evaluation, and stakeholder renegotiation to reconcile the system's product, process, property, and success models into a Life Cycle Architecture Package as in Figure 1.

Incremental Development. The LCA package includes an incremental development plan (item 5a in figure 2) indicating the schedules and pass/fail criteria for the core capability (item 5b), IOC (item 5c), and perhaps other milestones.

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 have to stretch 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. A rapid, "cold-turkey" cutover to Information Services Division operations and maintenance is extremely important and challenging for the USC applications, since the students disappear after IOC.

Change and Progress Monitoring and Control. This process element is not shown in Figure 2,

as it is applied continuously in each spiral cycle. 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.

Related Approaches. Design-to-cost and design-to-schedule processes have been used for a long time, particularly for such activities as rapid prototyping. The 1989 book Software Risk Management defines them as "prioritizing the desired system capabilities and organizing the architecture to facilitate dropping lower-priority capabilities as one finds that their development does not fit within one's available budget or schedule." (Boehm 1989, p.437)

The related Timeboxing approach developed at Du Pont in the late 1980's by Scott Schulz and others was a highly successful application of the SAIV approach to moderate-size business applications (Martin 1991). Steve McConnell's book, Rapid Development, has a particularly good description of Timebox Development (McConnell 1996, pp.575-583). It includes such key features as prioritizing requirements, realistic schedules, end-user involvement, not sacrificing quality, and use of small, skilled, highly motivated teams. Its focus is on fixed 60-120 day timeboxes with a binary accept/reject decision at the end. It has less emphasis than SAIV/CAIV/SCQAIV on such practices as expectations management, schedule range estimation, architecting for ease of adding and dropping marginal features, and core capability determination. The more adaptive and scaleable form of timeboxing used in Jim Highsmith's Adaptive Software Development (Highsmith 1999) 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.

SAIV Experience

USC Electronic Services Projects. The USC electronic services projects (Boehm et. al 1998, Boehm et. al 1999b) use the MBASE approach as detailed in (Boehm et. al 2001a) and (Metha 1999). To elaborate on its top-level description in Figure 1, it involves the concurrent development of several initial artifacts: 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 (Port, Boehm 2001). 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 sensitize them to the risks of overspecifying their set of desired IOC features, and to emphasize the importance of prioritizing their desired capabilities. This generally leads to a highly collaborative win-win negotiation of prioritized 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 24 of the 26 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 24 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 synchronize product transition on a specific fixed date with their other transition activities. Another pleasant surprise was the effect on clients' review timeliness: "You mean if I evaluate the prototype right away, I'll get more features in my IOC?" The e-services project artifacts can be reviewed on the class web page, <http://sunset.usc.edu/classes>.

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 (Madachy et. al 2001).

The CAIV and SCQAIV Process Models. Simply substituting "cost" for "schedule" in the SAIV process model described above provides you with an equally effective way to use CAIV as a process model. The SCQAIV model is a straightforward extension of CAIV and SAIV. It involves setting the system's quality goals (e.g., a delivered defect density of 0.3 nontrivial defects per thousand source lines of code (KSLOC), or of 0.03 nontrivial defects per function point), and tracking progress with respect to achieving the desired combination of schedule, cost, or quality goals. If any of these goals becomes unachievable in delivering the current feature set, the project must drop enough lower-priority features to make the combination of goals achievable. There may be limits to the project's ability to do this, which are discussed next.

Applying SAIV, SAIV, and SCQAIV

SAIV/CAIV/SCQAIV Limitations

The primary limitation arises in applications which have a minimum essential core capability (e.g., aircraft control) whose implementation within the quality goals and available schedule and budget is unachievable. This limitation can be visualized and addressed in terms of the project's production function relating the value of the product to the resources required to create it, as shown in Figure 3. It is slightly modified and updated from (Boehm 1981, 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 3 with respect to two example project durations, $T = 12$ months and $T = 6$ months.

The COCOMO II model produces Most Likely (50% confidence) and Pessimistic (90% confidence) estimates, enabling you to determine how much software can be developed to these confidence levels in 6 or 12 months. Again, it is good practice to cross-check the COCOMO estimates by using expert judgement and/or another cost model.

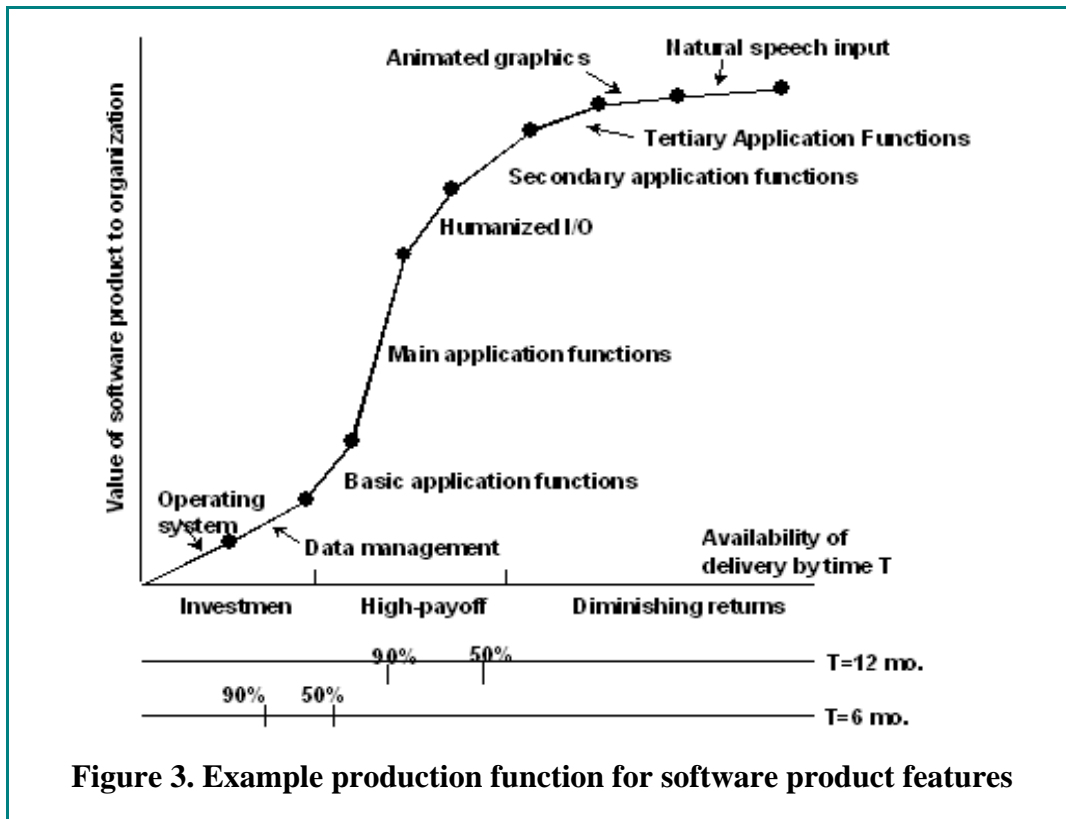


Figure 3. Example production function for software product features

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 is used to determine whether a project is worth pursuing within a given fixed schedule or budget.

Realizing SAIV/CAIV/SCQAIV Within the DoD Acquisition Framework

Stable Post-Deployment Support. In situations such as post-deployment upgrades and pre-planned product improvements, DoD can and often has implemented versions of SAIV/CAIV/SCQAIV as smoothly as they are done commercially. Frequently, in such situations, the organization's software maintenance budget and release cycle are relatively fixed. The biggest risk is to promise too much within these constraints, leaving the sacrifice of quality as the only way

to meet budget and schedule. This inevitably leads to degradations of the software's maintainability, operational fitness, and future maintenance productivity.

Thus, a form of SCQAIV is the best option for software maintenance, in which quality standards are set, infrastructure upgrades are given appropriate priorities, and lower-priority features are shed to meet cost, schedule, and quality objectives.

This approach is workable because DoD's operations and maintenance acquisition practices are similar to their commercial counterparts. Budgets are generally not tied to premature promises of delivered features, and there is usually a long-term customer-supplier relationship with a shared product vision among the customer, supplier, and users. This continuing relationship usually increases mutual trust, the ability to share and respect each other's win conditions and to negotiate mutually satisfactory or win-win agreements and priorities.

Competitive Development. In some cases, such as the Air Force Electronic Systems Center's Command Center Product Line (CCPL) and within classified-application organizations, DoD customers have been able to create development arrangements similar to the stable post-deployment support situation described above. CCPL, for example, developed a flexible contractual instrument focused on creating user value rather than prespecified features, and allowing in-process renegotiation of priorities. It selected three contractors via competitive source selection. The evaluation criteria included track record on similar developments, software CMM process maturity, technical and management approach, and demonstration of the approach via a representative exercise.

Once selected, the contractors operated as a team with the customer, developing a strong shared vision for the product line, and taking on new assignments based on best-matched available expertise, ensuring effective employment of all three contractors' resources. In this situation, SAIV/CAIV/SCQAIV-type approaches were highly feasible and preferable.

In many cases, however, DoD organizations must develop a new system vision and set of acquisition parameters (schedule, cost, quality attribute levels, feature scope) within a competitive acquisition framework. Here, complete multi-contractor shared vision development is impractical, as developers will be unwilling to share their competitive-discriminator technology solutions with competing developers. Frequently, this leads acquisition organizations to exclude developers from participating in the creation of the shared vision. This is highly risky, as the resulting decision may exclude attractive developer technology solutions. And it may leave serious vision mismatches between the customer, user, and selected developers, making SAIV/CAIV/SCQAIV system scoping and feature prioritization difficult to achieve, particularly if the program's funding and schedule have been tied to a particular set of delivered capabilities.

Unfortunately, there is no ideal solution to this dilemma. The most attractive near-solution involves the use of multiple competitive spiral cycles of system definition, with the number of competitors being reduced from one cycle to the next. The earlier cycles are shorter and less expensive, making a larger number of participants affordable. They can be run as SAIV/CAIV procurements with equal opportunity for each competitor. Some care is necessary to avoid leaking competitors' key discriminators, but many competitive DoD concept definition efforts have achieved this.

These earlier cycles enable overall system scoping and trade-off analysis to be performed, along with the evaluation of readiness levels of key technologies via prototyping, benchmarking, modeling and simulation, etc. This also enables the acquirers to evaluate the competing developers' capabilities and understanding of the system context and objectives. Some similar criteria to those used by CCPL (track record, technical and management capabilities, concept

definition and evaluation performance) are used for initial competitor selection and early downselection. Later downselection criteria increasingly involve development capabilities such as process maturity and realism of development plans, schedules, and budgets. Here again, the final development competition can fix the cost and/or schedule, provide a prioritized feature set, and compete on scope and realism of feature set delivery plans.

All three Services are making progress toward mastering this kind of evolutionary acquisition in the context of the new DoD 5000-series of acquisition regulations (DoD 2000). These include the extensive use of simulation and modeling in the Army's SMART program, the Air Force's Instruction 63-123, "Evolutionary Acquisition of Command and Control Systems" (US Air Force 2000), use of downselected contractors' expertise in other system life-cycle roles, and Service use of new contractual vehicles such as Cooperative Research and Development Agreements (CRADAs) and the DARPA-originated "Other Transactions" approach (US Air Force 2000). All of these are compatible with and have been used successfully with SAIV/CAIV/SCQAIV-type approaches.

Conclusions

The six-step SAIV process model presented here has been used successfully on 24 of 26 e-services 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 92% success rate compares favorably with the 16% success rate in the Standish Group's survey of current practice. 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 prioritized 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 borderline 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, "Schedule, Cost, Quality: Pick Any Two," to "Schedule, Cost, Quality: Pick All Three," via the SCQAIV version of the model, whenever your project is able to shed lower-priority features to meet its SCQ objectives. Proven strategies are also available for applying SAIV/CAIV/SCQAIV to competitive DOD system acquisitions.

References

- Boehm, B., Software Engineering Economics, Prentice Hall, 1981.
- Boehm, B., Software Risk Management, IEEE-CS Press, 1989.
- Boehm, B., "Anchoring the Software Process," IEEE Software, July 1996, pp. 73-82.
- Boehm, B., "A Spiral Model of Software Development and Enhancement," IEEE Computer, May 1998, pp. 61-72.
- 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.

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.

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.

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.

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.

Boehm, B., Port, D. and Al-Said, M., "Avoiding the Software Model-Clash Spiderweb," IEEE Computer, November 2000, pp. 120-122.

Boehm, B., and Hansen, W., "The Spiral Model as a Tool for Evolutionary Acquisition," CrossTalk, May 2001, pp. 3-11.

Boehm, B., and Port, D., "Balancing Discipline and Flexibility with the Spiral Model and MBASE," Cross Talk, 2001

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>

Boehm, B., Gruenbacher, P., and Briggs, R. "Developing Groupware for Requirements Negotiation: Lessons Learned," IEEE Software, May/June 2001, pp. 46-55

Department of Defense Instruction 5000.2, "Operation of the Defense Acquisition System," September 2000, www.acq.ord.mil/ap/i50002p.doc

Fisher, J. et al., "Model-Based Systems Engineering: A New Paradigm," INCOSE INSIGHT, October 1998, pp. 3-16.

Galorath, D., "SEER-SEM," Galorath, Inc., 2001, <http://www.galorath.com>

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.

Highsmith, J., Adaptive Software Development, Dorset House, 1999.

Honeywell Technology Center, "Model-Based Software Development," Course Announcement, Minneapolis, MN, 1998.

Jacobson, I., Booch, G. and Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999.

Jones, C., "Knowledge PLAN," Artemis/SPR, 2001, <http://www.spr.com>

Kruchten, P., The Rational Unified Process, Addison-Wesley, 1998.

Madachy, R., Pan, A. and Arboleda, A., "Processes for Rapid Development of Internet Applications," LA SPIN Presentation, January 24, 2001.

Martin, J., Rapid Application Development, Prentice Hall, 1991.

McConnell, S., Rapid Development, Microsoft Press, 1996

Mehta, N., "MBASE Electronic Process Guide," USC-CSE, September 1999, <http://sunset.usc.edu/Research/MBASE>

Parnas, D., "Designing Software for Ease of Extension and Contraction," IEEE Trans. Software Engr., March 1979, pp. 128-137.

Port, D. and Boehm, B. "Educating Software Engineering Students to Manage Risk," Proceedings, ICSE 2001, May 2001.

Putnam, L. "Software Life Cycle Model (SLIM)," QSM, 2001, <http://www.qsm.com>

Royce, W.E., Software Project Management: A Unified Framework, Addison-Wesley, 1998.

Standish Group, "CHAOS," <http://www.standishgroup.com/chaos.htm>
U.S. Air Force Instruction 63-123, "Evolutionary Acquisition for C2 Systems," April 2000,
<http://afpubs.hq.af.mil/>
Yourdon, E., Death March, Prentice Hall, 1997.