



Balancing Discipline and Flexibility With the Spiral Model and MBASE

Dr. Barry Boehm and Dr. Daniel Port

University of Southern California, Center for Software Engineering

This article details how the spiral model and its recent extension, Model-Based Architecting and Software Engineering (MBASE) can be used to tailor a project's balance of discipline and flexibility via risk considerations. It also describes and rationalizes the major MBASE extensions to the spiral model – model clash avoidance, stakeholder win-win – and elaborates on the use of these extensions and risk considerations in the anchor-point milestones used in MBASE and the spiral model.

In his keynote address at the 1996 International Conference on Software Engineering, Tom De Marco summarized the work of the great military analyst Karl Von Clausewitz on the interplay of armor and mobility in military conflict. De Marco said Clausewitz proposed that at times, armor would dominate mobility, as with heavily armed medieval knights dominating lightly armed peasantry. But if over-optimized, one strategy will lose to advances in the other, as the ponderous French knights found in their inability to dominate the lightly armed and mobile English longbowmen in their watershed loss to the English at Crecy in 1346.

De Marco then drew a parallel between *armor-intensive* software strategies such as the software Capability Maturity Model® (CMM®) and the *mobility-intensive* lightweight processes that were emerging at the time. He was inferring that the software CMM was too ponderous to cope with the need for rapid development and rapid change characteristic of such sectors as electronic commerce and Web-based systems. In the ensuing discussion, software CMM advocates cited the high mortality rates of lightweight process organizations, and their frequent inability to cope with success when they need to scale up their process and architectures to deal with more complex services and heavier workloads.

Underlying this point/counterpoint is a key software-engineering question: How much discipline is enough, and how much flexibility is enough?

In *Understanding the Spiral Model as a Tool for Evolutionary Acquisition* [1], we showed that the risk exposure considerations used as spiral model decision criteria could be used to address *how-much-is-enough* questions. There, we showed how a

© Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

SM Capability Maturity Model-Integrated and CMMI are service marks of Carnegie Mellon University.

how-much-testing-is-enough question could be addressed by balancing the risks of doing too little testing (alienating your users) and the risks of doing too much testing (unavailable combat capability, missed market windows).

In this article, we show how the spiral model and its recent extension, Model-Based (system) Architecting and Software Engineering (MBASE), can be used to tailor a project's balance of discipline and flexibility via risk considerations. We also describe and rationalize the major MBASE extensions to the spiral model (model clash avoidance, stakeholder win-win), and elaborate on the use of these extensions and risk considerations in the anchor-point milestones used in MBASE and the spiral model.

In subsequent articles, we will present an attractive special case of MBASE, the Schedule as an Independent Variable (SAIV) process model, and present an integration of the MBASE project approach with the University of Maryland's Experience Factory approach, which facilitates an organization's transition to the Capability Maturity Model®-IntegratedSM (CMMISM).

MBASE and Model Clash Avoidance

Particularly for an invisible product such as software, projects make use of various process, product, property, and success models to guide its progress. *Process models* can include the waterfall model (sequential determination of the system's requirements, design, and code); evolutionary development (development of an initial core capability, with full definition of future increments deferred); incremental development; spiral development; rapid application development; adaptive development; and many others.

Product models can include various ways of specifying operational concepts,

requirements, architectures, designs, and code, along with their interrelationships; for example, the object-oriented design models specified within the Rational Unified Process, e.g., class and sequence diagrams, use-cases, etc.

Property models can include models of desired or acceptable cost, schedule, performance, reliability, security, portability, evolvability, reusability, etc., and their tradeoffs, e.g., Constructive Cost Model II (COCOMO II).

Success Models can include correctness (satisfying the specified requirements), organization and project goals, stakeholder win-win, business-case, Goal-Question-Metric (GQM), or others such as IKI-WISI (I'll know it when I see it: a frequent response when users are asked to specify user-interface requirements).

Software architects and product developers are generally familiar with the concept that trying to integrate two or more arbitrarily selected products or product models can lead to serious conflicts and disasters. Some examples are mixing functional and object-oriented components, or the architectural style clashes you can encounter when integrating commercial off-the-shelf (COTS) products (see [2] for a well-described case study involving factors of four and five overruns in schedule and effort). Software process people are similarly familiar with the serious effects of trying to coordinate organizations with clashing process models such as top-down/bottom-up or CMM Level 1/Level 5.

However, relatively few people are aware of the extent to which projects can run into trouble because they choose incompatible combinations of software process, product, property, and success models. Such clashes are not only frequent and severe, but they are also hard to diagnose because they derive from different sources and involve mismatched assumptions lying deep below the project's written

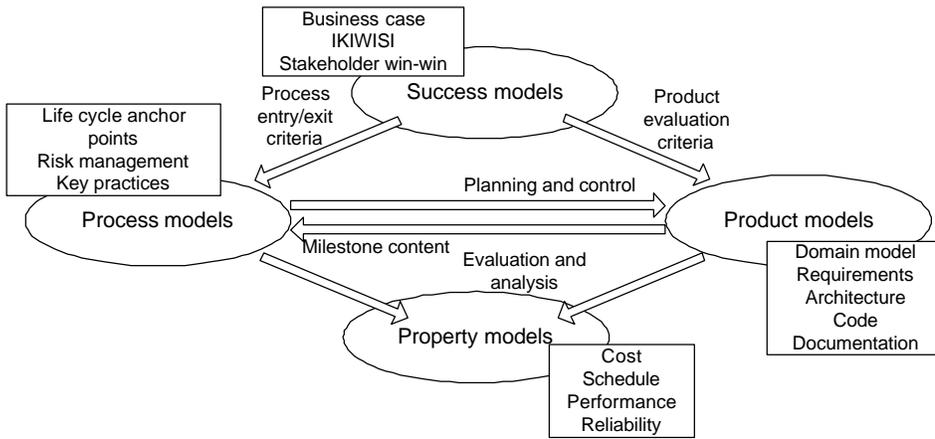


Figure 1: Model-Based Architecting and Software Engineering Model Integration Framework

plans and specifications.

A good Department of Defense (DoD) example was a deeply troubled project encountered during the National Research Council *Ada and Beyond* study [3]. This project inherited a commitment to a waterfall process model via its organization’s commitment to DoD-STD-2167. It also inherited a commitment to COTS-based product model through the need to comply with a secretary of defense mandate.

The waterfall model assumes that the requirements determine the capabilities, and the project had contracted for a two-second response time requirement. However, the contractor found that none of the available COTS capabilities could process the system workload with a two-second response time, and was proceeding to develop an expensive custom software system to meet the two-second requirement. The customer wanted the COTS-directive product model to take precedence, and to have the available COTS capabilities determine the performance requirements.

Besides this difficult model clash, the

project had also inherited an Ada-based product model via the DoD Ada mandate. This clashed with the COTS model, since some attractive COTS solutions did not have adequate Ada bindings. Further, none of the approaches were compatible with the project’s success model of producing an initial operational capability in 36 months; or with an additional property model, which was being adopted by the organization. This model was *Cost As Independent Variable*, which would have been difficult to satisfy when the project already had at least four existing independent variables.

This example covers only a small subset of the model clashes a project can encounter. Further discussions, examples, and case studies can be found in [4] and [5]. We have been able to use MBASE to help other large government and commercial projects to diagnose and avoid serious model clashes, and have worked with smaller companies to develop lightweight versions of MBASE to balance discipline and flexibility on rapid-development projects. After describing MBASE in the next

four sections, we will summarize how its use could have avoided the DoD project model clash dilemmas noted, followed by a summary of MBASE usage to date.

MBASE Model Integration Framework and Process Framework

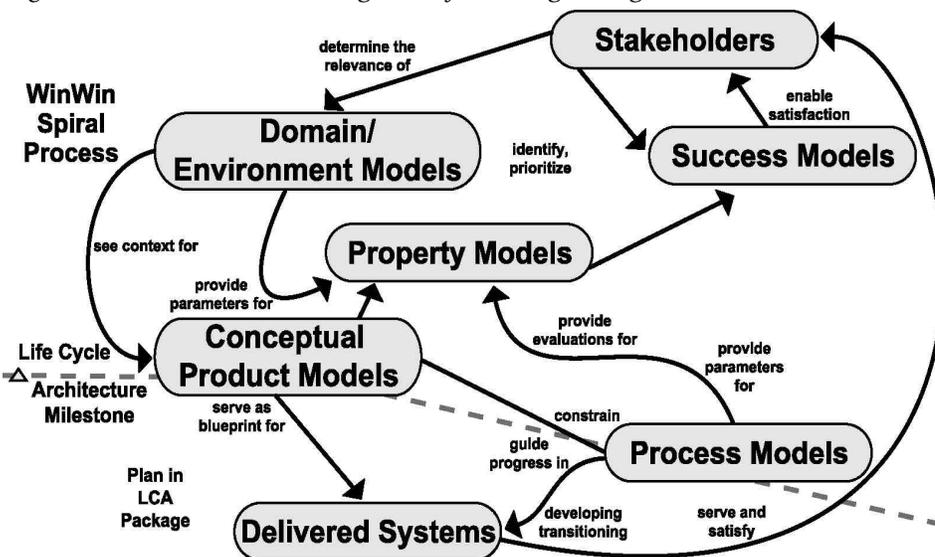
Figure 1 summarizes the overall model integration framework used in the MBASE approach to ensure that a project’s success, product, process, and property models are consistent and well integrated. At the top of Figure 1 are success models, illustrated with several examples, whose priorities and consistency should be considered first as they tend to drive the selection and use of other models (including other success models).

Thus, if the overriding top-priority success model is to “demonstrate a competitive agent-based electronic commerce system on the floor of the COMDEX trade show in nine months,” this constrains the ambition level of other success models. It would be a major schedule risk to insist on provably correct code or a fully documented system. The nine-month schedule constraint is most critical because the system will lose most of its value if it is not available to compete for early market share at COMDEX.

The risk schedule overrun also determines many aspects of the product model (architecture designed to easily shed lower-priority features if necessary to meet schedule), the process model (SAIV), and various property models (portable and reliable enough to achieve a successful demonstration). The achievability of the success model needs to be evaluated with respect to the other models. Figure 1 shows that the choices of process and product models need to be evaluated by having the success model provide evaluation criteria for the product milestone artifacts, and provide preconditions and post-conditions (entry and exit criteria) for the process milestones.

In the nine-month COMDEX demonstration example, a cost-schedule estimation model would relate 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 nine months would be sufficiently competitive for the success models. Thus, as shown at the bottom of Figure 1, a cost and schedule property model would be used for the

Figure 2: Model-Based Architecting and Software Engineering Process Framework



evaluation and analysis of the consistency of the system's product, process, and success models.

In other cases, the success model would make a process model or a product model the primary driver for model integration. An IKIWISI success model might initially establish a prototyping and evolutionary development process model leaving most of the product features and property levels to be determined by the evolutionary development 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.

Figure 2 provides an overall process framework for the MBASE approach. The primary drivers for any system's (or product line's) characteristics are its key stakeholders. These generally include the system (taken below to mean *system or product-line*) users, customers, developers, and maintainers. Key stakeholders can also include strategic partners, marketers, operators of closely coupled systems, and the general public for such issues as safety, security, privacy, or fairness.

The critical interests of these stakeholders determine the priorities, desired levels, and acceptable levels of various system success criteria. These are reflected in the success models for the system such as stakeholder win-win, business case, organization and project goals, operational effectiveness models, or IKIWISI. These in turn determine which portions of an applications domain and its environment are relevant to consider in specifying the system and its development and evolution process. The particular objective is to determine a system boundary, within which the system is to be developed and evolved; outside of which is the system environment (and context).

For example, in our COMDEX electronic commerce application, the driving success model is the nine-month schedule. The system boundary would be determined by the most cost-effective set of capabilities that could be developed in nine months. Thus, a credit card verification capability might be considered outside the initial system boundary, although it would be needed later.

This latter point illustrates how boundaries might (and will likely) change over time, particularly in the face of evolving success models (e.g., the nature of a

competitive e-commerce system). For example, if a compatible COTS credit card verification capability became available and easy to integrate, it could be added within the system boundary. Thus the domain scope for the demo system would be very much determined by the available COTS products that could be tailored, integrated, and built upon.

Determining the appropriate combination of COTS products and extensions could take several win-win spiral cycles of experimental prototyping and risk resolution, in concert with cost-schedule modeling to determine how much capability would be feasible to develop in nine months. The appropriate process model would be SAIV, which adds further product model constraints, such as the need to prioritize features and to design the system architecture for ease of adding or dropping marginal-priority features in order to minimize the risk of not meeting the nine-month schedule.

A Different Balance of Discipline and Flexibility: Safe Air Traffic Control

With a different set of stakeholders and success models, the same MBASE process framework in Figure 2 will produce a different balance of discipline and flexibility. In an air traffic control system, for example, the key stakeholders will include the airplane passengers and various regulatory bodies whose success models involve a very high level of system safety.

In this case, the success models will reject high-risk product models, including unreliable COTS products. The process models will include considerably more dis-

cipline to eliminate safety risks at the requirements, architecture, design, and code levels. The key property model will focus on safety rather than schedule, although schedule considerations might still affect the timing of various increments of system capability. Thus, the different risk patterns imposed by the stakeholders and their success models will produce different sequences of product capabilities and processes with different balances of discipline and flexibility.

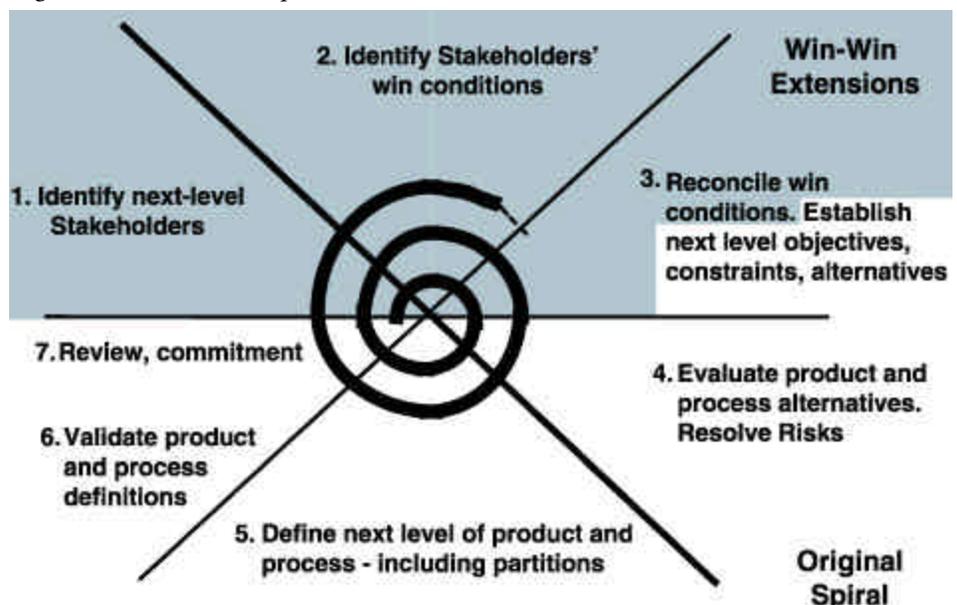
MBASE and Stakeholder Win-Win

A common element in the e-commerce and air traffic control examples is the need to reconcile the key stakeholders' success models. Thus, a stakeholder win-win negotiation process becomes a key step in each spiral cycle of the MBASE approach, as shown in Figure 3.

In the COMDEX application, for example, the initial spiral cycle would focus on evaluating COTS products and scoping the overall system to be buildable in nine months. In a subsequent spiral cycle, the next-level stakeholders would include representative users of the e-commerce system, and the reconciliation of their win conditions would include prototyping of the user interface to eliminate the risk of showing up at COMDEX with an unfriendly user interface. The MBASE tool support includes a groupware system called Easy WinWin, which enables distributed stakeholders to enter their win conditions and to negotiate mutually satisfactory (win-win) agreements with other stakeholders [6].

The win-win spiral model in Figure 3

Figure 3: *The Win-Win Spiral Model*



Spiral Model Essentials	
1.	Concurrent determination of key artifacts.
2.	Each cycle does objectives, constraints, alternatives, risks, review, and commitment to proceed.
3.	Level of effort driven by risk considerations.
4.	Degree of detail driven by risk considerations.
5.	Use of anchor point milestones: LCO, LCA, IOC.
6.	Emphasis on system and life-cycle activities and artifacts.

Table 1: *Essentials of the Spiral Model*

provides another view of how risk considerations are used to reconcile stakeholder success conditions in terms of product, process, and property models. A complementary view was shown in Figure 2 (see page 24), which also identifies the win-win spiral model's role in guiding the early feedback cycles involved in defining and reconciling the system's domain, product, process, and property models.

MBASE and Life-Cycle Anchor Points

MBASE also adopts and extends the six Spiral Model Essentials presented in our May 2001 CROSSTALK article [1] and summarized in Table 1. The *stakeholder commitment to proceed* in Essential 2 is implemented via the win-win spiral model as shown in Figure 3 (see page 25). MBASE adopts Essential 5 by using its life-cycle anchor points as critical review and management decision points. It adopts Essentials 3 and 4 on risk management via continuous risk identification, risk assessment, and risk exposure reduction, and Essentials 1 and 6 via a concurrent engineering approach to both system and software issues.

The life-cycle anchor points are described further in the side bar located on page 27. As shown in Figure 2, one of them is the Life Cycle Architecture milestone. It includes a product definition, a process definition, and a feasibility ration-

ale ensuring the compatibility of the system's product, process, property, and success models. From this base, the project can continue to construct the system by refining the product models into an executing product up through a third milestone, the Initial Operating Capability.

The specific content of the first two anchor point milestones is summarized in the sidebar. It includes increasingly detailed, risk-driven definitions of the system's operational concept, prototypes, requirements, architectures, life-cycle plan, and feasibility rationale. For the feasibility rationale, property models are invoked to help verify that the project's success models, product models, process models, and property levels or models are acceptably consistent.

The first milestone is the Life Cycle Objectives (LCO) milestone, at which management verifies the basis for a business commitment to proceed at least through an architectural stage. This involves verifying that there is at least one system architecture and choice of COTS/reuse components that is shown to be feasible to implement within budget and schedule constraints to satisfy key stakeholder win conditions and to generate a viable investment business case.

The second milestone is the Life Cycle Architecture (LCA) milestone, at which management verifies the basis for a sound commitment to product development and evolution. This is a particular system architecture with specific COTS and reuse commitments that is shown to be feasible with respect to budget, schedule, requirements, operations concept and business case; identification and commitment of all key life-cycle stakeholders; and elimination of all critical risk items. The AT&T/Lucent Architecture Review Board technique [7] is an excellent management review approach involving the LCO and LCA milestones. It is similar to the highly successful recent DoD best practice of software Independent Expert Program Reviews [8].

The third anchor point is the system's Initial Operational Capability (IOC), defined further in [9]. The LCO, LCA, and IOC have become the key milestones in the Rational Unified Process [10, 11, 12]. There are many possible minor milestones (adjusted to the particular project as needed) that may lie between LCO and IOC and several important post-deployment milestones beyond IOC. Table 2 summarizes the pass/fail criteria for the LCO, LCA, and IOC anchor points.

The focus of the LCO review is to ensure that at least one architecture choice is viable from a business perspective. The focus of the LCA review is to commit to a single detailed definition of the review artifacts. The project must have either eliminated all significant risks or put in place an acceptable risk-management plan. The focus of the IOC review, also called the Transition Readiness Review, is to ensure that the initial users, operators, and maintainers (generally equivalent to beta-testers) are fully prepared to successfully operate the delivered system. If the pass/fail criteria for any review are not satisfied, the package should be reworked.

We determined these anchor point milestones as common commitment points across commercial, aerospace, and government organizations when searching with our University of Southern California (USC) Center for Software Engineering Affiliates for a set of common milestones for referencing COCOMO II cost and schedule estimates. They work well as common commitment points across a variety of process model variants because they reflect similar commitment points during one's lifetime.

The LCO milestone is the equivalent of getting engaged, and the LCA milestone is the equivalent of getting married. As in life, if you marry your architecture in haste, you and your stakeholders will repent at leisure (if, in Internet time, any leisure time is available). The third anchor point milestone, the IOC, constitutes an even larger commitment: It is the equivalent of having your first child with all the associated commitments of care and feeding of a legacy system.

To return to our DoD 2167/ COTS/Ada/deadline/CAIV model clash example on page 24, at the latest it would have failed its LCO milestone review by being unable to demonstrate that a COTS-based architecture could satisfy the two-second response time requirement. Even earlier, though, this model clash would have been picked up by the MBASE process framework in Figure 2, in feeding back to the stakeholders the need

Table 2: *LCO, LCA, and IOC Pass/Fail Criteria*

LCO	LCA	IOC
For at least one architecture, a system built to that architecture will: <ul style="list-style-type: none"> • Support the core operational concept. • Satisfy the core requirements. • Be faithful to the prototype(s). • Be buildable within the budgets and schedules in the plan. • Show a viable business case. • Have its key stakeholders committed to support the Elaboration Phase (to LCA). 	For a specific detailed architecture, a system built to that architecture will: <ul style="list-style-type: none"> • Support the elaborated operational concept. • Satisfy the elaborated requirements. • Be faithful to the prototype(s). • Be buildable within the budgets and schedules in the plan. • Have all major risks resolved or covered by a risk management plan. • Have its key stakeholders committed to support the full life cycle. 	An implemented architecture, an operational system that has: <ul style="list-style-type: none"> • Realized the operational concept. • Implemented the initial operational requirements. • Prepared a system operation and support plan. • Prepared the initial site(s) in which the system will be deployed for transition. • Prepared the users, operators, and maintainers to assume their operational roles.

to revise their success models to permit a clash-free solution. This would involve additional win-win spiral cycles to determine a mutually satisfactory (win-win) combination of features, budgets, schedules, increments, and COTS choices.

MBASE Usage Experience

For the past five years, USC has used and refined MBASE extensively within its two-semester graduate software-engineering course. The students work on a Web-based electronic services project for a real USC client (frequently a digital library application for the university information services division) from initial system definition through transition, utilizing a specialized form of MBASE. This specialization includes particular tools and models such as Easy WinWin, Rational Rose, MSPProject, and elements of the Rational Unified Process. More than 100 real-client projects have used MBASE, and over 90 percent have delivered highly satisfactory products on very short fixed schedules. The annual lessons learned have been organized into an extensive set of usage guidelines and an Electronic Process Guide [13], all accessible at <<http://sunset.usc.edu/research/MBASE>>. In the spring of 1999, MBASE was used in both the undergraduate and graduate software engineering courses at Columbia University. Although these are single semester courses, MBASE was successfully adapted to help student teams complete a full project life cycle for real clients.

Within industry, Xerox has adopted many elements of MBASE to form its *time-to-market* process, including the use of the LCO and LCA anchor points as synchronization points for the hardware and software portions of their printer product definitions.

As mentioned previously, Rational has adopted the LCO, LCA, and IOC anchor points within their Rational Unified Process while MBASE adopted Rational's Inception-Elaboration-Construction-Transition phase definitions.

C-Bridge has mapped their *define, design, develop, deploy* rapid development methodology for e-commerce systems to the MBASE spiral model.

The Internet startup company Media Connex adopted MBASE and used Easy WinWin to establish win-win relationships among their key stakeholders. Each of these companies converged on different balances of discipline and flexibility to satisfy their stakeholders' success models.

Additionally, there are numerous companies and organizations directly making

The Spiral Model Essential Life-Cycle Anchor Points

Milestone Element	Life-Cycle Objectives (LCO)	Life-Cycle Architecture (LCA)
Definition of Operational Concept	<ul style="list-style-type: none"> • Top-level system objectives and scope: <ul style="list-style-type: none"> - System boundary. - Environment parameters and assumptions. - Evolution parameters. • Operational concept. 	<ul style="list-style-type: none"> • Elaboration of system objectives and scope by increment. • Elaboration of operational concept by increment.
System Prototype(s)	<ul style="list-style-type: none"> • Exercise key usage scenarios. • Resolve critical risks. 	<ul style="list-style-type: none"> • Exercise range of usage scenarios. • Resolve major outstanding risks.
Definition of System Requirements	<ul style="list-style-type: none"> • Top-level functions, interfaces, quality attribute levels, including: <ul style="list-style-type: none"> - Growth vectors. - Priorities. • Stakeholders' concurrence on essentials. 	<ul style="list-style-type: none"> • Elaboration of functions, interfaces, quality attributes by increment: <ul style="list-style-type: none"> - Identification of TBDs (to-be-determined items). • Stakeholders' concurrence on their priority concerns.
Definition of System and Software Architecture	<ul style="list-style-type: none"> • Top-level definition of at least one feasible architecture: <ul style="list-style-type: none"> - Physical and logical elements and relationships. - Choices of COTS and reusable software elements. • Identification of infeasible architecture options. 	<ul style="list-style-type: none"> • Choice of architecture and elaboration by increment: <ul style="list-style-type: none"> - Physical and logical components, connectors, configurations, constraints. - COTS, reuse choices. - Domain-architecture and architectural style choices. • Architecture evolution parameters.
Definition of Life-Cycle Plan	<ul style="list-style-type: none"> • Identification of life-cycle stakeholders: <ul style="list-style-type: none"> - Users, customers, developers, maintainers, interpreters, general public, others. • Identification of life-cycle process model: <ul style="list-style-type: none"> - Top-level stages, increments. • Top-level WWWWWHH* by stage. 	<ul style="list-style-type: none"> • Elaboration of WWWWWHH* for Initial Operational Capability (IOC). <ul style="list-style-type: none"> - Partial elaboration, identification of key TBDs for later increments.
Feasibility Rationale	<ul style="list-style-type: none"> • Assurance of consistency among elements above: <ul style="list-style-type: none"> - Via analysis, measurement, prototyping, simulation, etc. - Business case analysis for requirements, feasible architectures. 	<ul style="list-style-type: none"> • Assurance of consistency among elements above. • All major risks resolved or covered by risk management plan.

* WWWWWHH: Why, What, When, Who, Where, How, How Much

use of MBASE elements within their project development efforts. For example, the U.S. Army Tank and Automotive Command has used Easy WinWin and other MBASE elements to reconcile its software technology organizations' process and product strategies.

Conclusions and Future Directions

The ability to balance discipline and flexibility is critical to developing highly dependable software-intensive systems in a rapidly changing environment. The MBASE integration framework, process framework, and associated guidelines provide a set of risk-driven techniques that elaborate on the spiral model and the Rational Unified Process enabling an organization to achieve an appropriate balance of discipline and flexibility for each of its projects.

However, this requires a large number of guidelines to keep all of a complex software system's process, product, property, and success models well integrated across all of the phases and activities in the software-system life cycle. In some ways, we have been able to reduce this complexity. One way is by providing tools and templates for MBASE artifacts via the MBASE Electronic Process Guide [13].

Another way is to develop special domain-specific models such as for the digital library domain that enables student teams to learn the development principles and successfully develop moderate-sized Web-based applications in 24 weeks [14].

Third is to develop specialized models for particular situations, such as the SAIV process model we will discuss in an upcoming CROSSTALK article.

A future challenge is to extend the project-oriented MBASE approach to address organization-level software and system process issues. In January CROSSTALK, we will present an integration of MBASE with the University of Maryland's Experience Factory approach [15], and show how it can help organizations transition to the CMMI.

Acknowledgements

We would like to acknowledge the support of the Defense Advanced Research Projects Agency and the National Science Foundation in establishing and refining MBASE, the DoD Software Intensive Systems Directorate in supporting its application to DoD projects and organizations, and the affiliates of the USC Center for Software Engineering for their contributions to MBASE. ♦

References

- Boehm, B., and W. Hansen, eds. "The Spiral Model as a Tool for Evolutionary Acquisition." *CROSSTALK* May 2001, pp. 4-9.
- Garlan, D., R. Allen, and J. Ockerbloom, eds. "Architectural Mismatch: Why Reuse Is So Hard." *IEEE Software* November 1995, pp. 17-26.
- Boehm, B., et al. "Ada and Beyond: Software Policies for the DoD." *National Academy Press*, 1997.
- Boehm, B., and D. Port, eds. "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them." *ACM Software Engineering Notes* January 1999, pp. 36-48.
- Boehm B., D. Port, and M. AlSaid, eds. "Avoiding the Software Model Clash Spider Web." *IEEE Software* November 2000, pp. 120-122.
- Boehm, B., P. Gruenbacher, and R. Briggs, eds. "Developing Groupware for Requirements Negotiation: Lessons Learned." *IEEE Software* May/June 2001, pp. 46-55.
- Marenzano, J., "System Architecture Validation Review Findings," in D. Garlan (ed.). *ICSE-17 Architecture Workshop Proceedings*. CMU, Pittsburgh, PA, 1995.
- Report of the Defense Science Board Task Force on Defense Software. Defense Science Board. OUSD (A&T), November 2000.
- Boehm, B. "Anchoring the Software Process." *IEEE Software* July 1996, pp. 73-82.
- Royce, W.E. *Software Project Management: A Unified Framework*. Addison-Wesley, 1998.
- Jacobson, I., G. Booch, and J. Rumbaugh, eds. *The Unified Software Development Process*. Addison-Wesley, 1999.
- Kruchten, P. *The Rational Unified Process* (2nd ed.). Addison-Wesley, 2000.
- MBASE Guidelines and MBASE Electronic Process Guide. USC-CSE. <<http://sunset.usc.edu/research/MBASE/>>.
- Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, eds. "Using the WinWin Spiral Model: A Case Study." *Computer* July 1998, M. 33-44
- Basili, V., G. Caldeira, and H. Rombach, eds. "The Experience Factory," in J. Marciniak (ed.). *Encyclopedia of Software Engineering*. Wiley, 1994.

About the Authors



Barry Boehm, Ph.D., is the TRW professor of software engineering and director of the Center for Software Engineering at the University of Southern California. He was previously in technical and management positions at General Dynamics, Rand Corp., TRW, the Defense Advanced Research Process Agency, and the Office of the Secretary of Defense as the director of Defense Research and Engineering Software and Computer Technology Office. Dr. Boehm originated the spiral model, the Constructive Cost Model (COCO-MO), and the stakeholder win-win approach to software management and requirements negotiation.

University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781
Phone: (213) 740-8163
Fax: (213) 740-4927
E-mail: boehm@sunset.usc.edu



Daniel Port, Ph.D., is a research assistant professor of Computer Science and an associate of the Center for Software Engineering at the University of Southern California. He received a doctorate degree from the Massachusetts Institute of Technology, and a bachelor's degree from the University of California, Los Angeles. His previous positions were assistant professor of Computer Science at Columbia University, director of Technology at the USC Annenberg Center EC2 Technology Incubator, co-founder of Tech Tactics, Inc., and a project lead and technology trainer for NeXT Computers, Inc.

University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781
Phone: (213) 740-7275
Fax: (213) 740-4927
E-mail: dport@sunset.usc.edu