Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering

Barry Boehm and Jo Ann Lane
University of Southern California
Center for Systems and Software Engineering

One of the top recommendations to emerge from the October 2006 Deputy Under Secretary of Defense (DUSD) Acquisition, Technology, and Logistics (AT&L) Defense Software Strategy Summit was to find ways of better integrating software engineering into the systems engineering and acquisition process. Concurrently, a National Research Council study was addressing the problem of better integrating human factors into the systems engineering and acquisition process. This paper presents a model that emerged from these and related efforts that shows promise of improving these integrations. This model, called the Incremental Commitment Model (ICM), organizes systems engineering and acquisition processes in ways that better accommodate the different strengths and difficulties of hardware, software, and human factors engineering approaches. It also provides points at which they can synchronize and stabilize, and at which their risks of going forward can be better assessed and fitted into a risk-driven stakeholder resource commitment process.

Introduction

Many projects have difficulties in integrating their hardware, software, and human factors aspects. This is basically due to differences between these aspects with respect to their underlying economics, evolution patterns, product subsetability (ability to deliver usable partial initial operational capabilities), user tailorability, adaptability, underlying science, and testing considerations, as summarized in Table 1.

This paper begins by summarizing trends that have caused difficulties for current systems engineering and acquisition processes (complex multi-owner systems of systems (SoS), emergent requirements, rapid change, reused components, high assurance of qualities) and principles underlying the ICM that better address these trends (stakeholder satisficing, incremental and evolutionary growth of system definition and stakeholder commitment, iterative development, concurrent definition and development, and risk management). It then presents several complementary views of the ICM, discusses their implications with respect to acquisition and engineering practices and personnel career paths, and assesses project performance with respect to use of the principles. The principles can also be used to avoid the negative effects of common misinterpretations of other current models such as the V, spiral, and Rational Unified Process (RUP) models.

Table 1. Underlying Differences Between Hardware, Software, and Human Factors Engineering.

Difference Area	Hardware	Software	Human Factors
Major Life-cycle Cost Source	Development, manufacturing	Life-cycle evolution	Training and operations labor
Ease of Changes	Generally difficult	Good within architectural framework	Very good, but people-dependent
Nature of Changes	Manual, labor-intensive, expensive	Electronic, inexpensive	Need personnel retraining, can be expensive
User-tailorability	Generally difficult, limited options	Technically easy; mission-driven	Technically easy; mission-driven
Subsetability	Inflexible lower limit	Flexible lower limit	Smaller increments easier to introduce
Underlying Science	Physics, chemistry, continuous mathematics	Discrete mathematics, linguistics	Behavioral sciences
Testing	By test organization; much analytic continuity	By test organization; little analytic continuity	By users

Summary of Difficulties and Some of Their Causes

Current systems engineering and acquisition practices (and the associated program management personnel) still rely heavily on their historical hardware engineering and acquisition legacy. An emphasis on reducing hardware development and manufacturing costs often leads to selection of components with incompatible software infrastructures and human interfaces, leading to much higher development, operations, and maintenance costs and associated system underperformance in the software and human engineering areas. A hardware-oriented fixed-price, build-to-specification contract may deliver a hardware initial operational capability (IOC) within its development budget, but may elect not to architect or upgrade the software to avoid excessive software maintenance or human operational costs.

The relative difficulty of modifying hardware installed in many places, as compared to electronic modification of software or modification of human operational procedures, may lead to added software costs for hardware shortfall workarounds or to fitting the people to the product rather than fitting the product to the people. And the limited subsetability of hardware systems (e.g., aircraft without landing gear or complete flight controls) as compared to partial software or human interface features often leads to incompatibilities between single-increment hardware acquisition practices and multiple-increment software and human interface practices on the same project.

If these hardware-software-human factors integration problems are difficult today, they will present formidable problems for the future if not adequately addressed. Some trends that will exacerbate such integration problems are

- *Complex, multi-owner systems of systems.* Current collections of incompatible, separately-developed systems cause numerous operational deficiencies such as unacceptable delays in

service, uncoordinated and conflicting plans, ineffective or dangerous decisions, and inability to cope with fast-moving events. Multiple owners of key interdependent systems make integration of systems of systems a major challenge, but the current alternative of just trying to mash them together will only become worse in the future [1].

- *Emergent requirements*. The most appropriate user interfaces and collaboration modes for a complex human-intensive system are not specifiable in advance, but emerge with system prototyping and usage. Forcing them to be prematurely and precisely specified generally leads to poor business or mission performance and expensive late rework and delays [2].
- *Rapid change*. Specifying current-point-in-time snapshot requirements on a cost-competitive contract generally leads to a big design up front, and a point-solution architecture that is hard to adapt to new developments. Each of the many subsequent changes then leads to considerable nonproductive work in redeveloping documents and software (or worse, hardware), and in renegotiating contracts [3].
- *Reused components.* Building all of one's own components from scratch will be economically infeasible for complex systems. However, reuse-based development has major bottom-up development implications, and is incompatible with pure top-down requirements-first approaches. Prematurely specifying requirements (e.g., hasty specification of a 1-second response time requirement when later prototyping shows that 4 seconds would be acceptable) that disqualify otherwise most cost-effective reusable components often leads to overly expensive, late, and unsatisfactory systems [4].
- *High assurance of qualities.* Future systems will need higher assurance levels of such qualities as safety, security, reliability/availability/maintainability, performance, adaptability, interoperability, usability, and scalability. Just assuring one of these qualities for a complex SoS will be difficult. Given the need to satisfice (not everybody gets everything they want, but everybody gets something they are satisfied with) among multiple system owners with different quality priorities, their complex sources of conflict and tradeoff relationships will make multi-attribute satisficing even more challenging [5].

Such concerns led to one of the top recommendations from the October 2006 Deputy Under Secretary of Defense (DUSD) Acquisition, Technology, and Logistics (AT&L) Defense Software Strategy Summit. This recommendation was to find ways of better integrating software engineering into the systems engineering and acquisition process [6]. Concurrently, a National Research Council (NRC) study was addressing the problem of better integrating human factors into the systems engineering and acquisition process [7].

We have performed several analyses to determine the kind of process that would satisfactorily address these challenges. As part of the NRC study, we analyzed the strengths and difficulties of current process models. Each had strengths, but needed further refinements to address all of the five challenges above. The most important conclusion, though, was that there were key process principles that address the challenges, and that forms of the models were less important than their ability to adopt the principles. These key principles are

1. *Stakeholder satisficing*. If a system development process presents a success-critical operational or development stakeholder with the prospect of an unsatisfactory outcome, the stakeholder will generally refuse to cooperate, resulting in an unsuccessful system. Stakeholder satisficing includes identifying the success-critical stakeholders and their value

propositions; negotiating a mutually satisfactory set of system requirements, solutions, and plans; and managing proposed changes to preserve a mutually satisfactory outcome.

2. *Incremental and evolutionary growth of system definition and stakeholder commitment*. This characteristic captures the often incremental discovery of emergent requirements and solutions via methods such as prototyping, operational exercises, and use of early system capabilities. Requirements and commitment cannot be monolithic or fully pre-specifiable for complex, human-intensive systems; increasingly detailed understanding, trust, definition and commitment is achieved through an evolutionary process.
3. *Iterative system development and definition*. The incremental and evolutionary approaches lead to cyclic refinements of requirements, solutions, and development plans. Such iteration helps projects to learn early and efficiently about operational and quality requirements and priorities.
4. *Concurrent system definition and development*. Initially, this includes concurrent engineering of requirements and solutions, and integrated product and process definition. In later increments, change-driven rework and rebaselining of next-increment requirements, solutions and plans occurs concurrently with stabilized development of the current system increment. This allows early fielding of core capabilities, continual adaptation to change, and timely growth of complex systems without waiting for every requirement and subsystem to be defined.
5. *Risk management – risk driven anchor point milestones*. The key to synchronizing and stabilizing all of this concurrent activity is a set of risk-driven anchor point milestones. At these milestones, the business, technical, and operational feasibility of the growing package of specifications and plans is evaluated by independent experts. Shortfalls in feasibility evidence are treated as risks and addressed by risk management plans. If the system's success-critical stakeholders find the risks acceptable and the risk management plans sound, the project will proceed to the next phase. If not, the project can extend its current phase, de-scope its objectives, or avoid low-return resource commitments by terminating the project. If the risks of proceeding straight into development are negligible, the project can skip one or more early phases.

Table 2 shows the relationships among the five critical challenges and the five key principles.

Overview of the ICM

The ICM builds on the strengths of current process models: early verification and validation concepts in the V-model, concurrency concepts in the Concurrent Engineering model, lighter-weight concepts in the Agile and Lean models, risk-driven concepts in the spiral model, the phases and anchor points in the RUP [8, 9, 10], and recent extensions of the spiral model to address SoS acquisition [11].

An overview of the ICM life cycle process is shown in Figure 1. In comparison to the software-intensive RUP, the ICM also addresses hardware and human factors integration. It extends the RUP phases to cover the full system life cycle: an Exploration phase precedes the RUP Inception phase, which is refocused on valuation and investment analysis. The RUP Elaboration phase is refocused on Architecting (a term based on [12] describing concurrent development of requirements, architecture, and plans), to which it adds feasibility evidence; the RUP Construction and Transition phases are combined into Development; and an additional Operations phase combines operations, production, maintenance, and phase-out.

Table 2. Relations Between Future Process Challenges and Process Principles

Future Process Challenges	Process Principles				
	Stakeholder satisficing	Incremental/evolutionary definition and commitment	Iterative development	Concurrent system definition and development	Risk management
Complex, multi-owner systems of systems	X	X	X	X	X
Emergent requirements	X	X	X	X	X
Rapid change			X	X	X
Reused components				X	X
High assurance of system qualities		X			X

Also, the names of the milestones are changed to emphasize that their objectives are to ensure stakeholder commitment to proceed to the next level of resource expenditure based on a thorough feasibility and risk analysis, and not just on the existence of a set of system objectives and a set of architecture diagrams. Thus, the RUP Life Cycle Objectives (LCO) milestone is called the Architecture Commitment Review (ACR) in the ICM and the RUP Life Cycle Architecture (LCA) milestone is called the Development Commitment Review (DCR).

In comparison to the sequential waterfall [13] and V-model [14], the ICM explicitly:

- Emphasizes concurrent engineering of requirements and solutions
- Establishes Feasibility Rationales as pass/fail milestone criteria
- Enables risk-driven avoidance of unnecessary documents, phases, and reviews
- Provides support for a stabilized current-increment development concurrently with a separate change processing and rebaselining activity to prepare for appropriate and stabilized development of the next increment.

These aspects can be integrated into a waterfall or V-model, enabling projects required to use such models to cope more effectively with systems of the future.

The overall lifecycle process divides naturally into two major stages. Stage I, Incremental Definition, covers the up-front growth in system understanding, definition, feasibility assurance, and stakeholder commitment leading to a larger Stage II commitment to a feasible set of specifications and plans for Incremental Development and Operations.

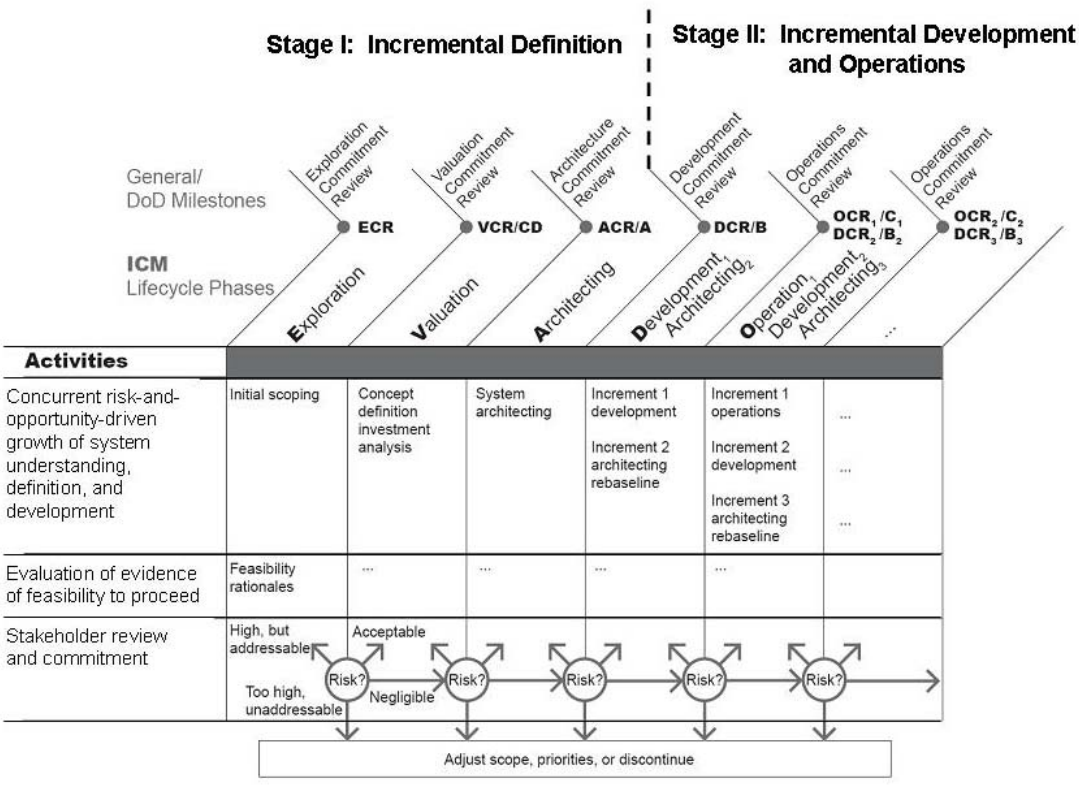

Figure 1. Overview of the Incremental Commitment Life Cycle Process.

Stage I: The duration of Stage I can be anywhere from one week to five years. The duration depends on such factors as the number, capability, and compatibility of the proposed system's components and stakeholders. A small, well-jelled agile-methods, developer-customer team operating on a mature infrastructure can form and begin incremental development using Scrum, XP, Crystal or other agile methods in a week. An ultra-large, unprecedented, multi-mission, multi-owner SoS project may take up to five years to progress from a system vision through sorting out needs, opportunities, and organizational roles; maturing key technologies; reconciling infrastructure incompatibilities, and evolving a feasibility-validated set of specifications and plans for Stage II. These specifications and plans would be at the build-to level for the initial increment, but only elaborated into detail for the later increments and the overall system where there were high-risk elements to resolve.

As shown in Figure 1, each project's activity trajectory will be determined by the risk assessments and stakeholder commitment decisions at its anchor point milestone reviews. The small agile project will follow the Negligible-risk arrows at the bottom of Figure 1 to skip the Valuation and Architecting phases and begin Stage II after a short exploratory phase confirms that the risks of doing so are indeed negligible. The ultra-large project could, for example, fund 8 small competitive concept-definition and validation contracts in the Exploratory phase, 4 larger follow-on Valuation contracts, and 2 considerably larger Architecting contracts, choosing at each anchor point milestone the best-qualified teams to proceed, based on the feasibility and risk evaluations performed at each anchor point milestone review. Or, in some cases, the reviews

might indicate that certain essential technologies or infrastructure incompatibilities needed more work before proceeding into the next phase.

Stage II: For Stage II, Incremental Development and Operations, a key decision that is made at the Development Commitment review is the length of the increments to be used in the system's development and evolution. A small agile project can use 2 to 4 week increments. However, an ultra-large SoS project with a couple of dozen system suppliers, each with a half-dozen levels of subcontractors, would need increments of up to two years to develop and integrate an increment of operational capability, although with several internal integration sub-increments. Some of the non-subsetable hardware systems would take even longer to develop their initial increments, and would be scheduled to synchronize their deliveries with later increments.

The features in each Stage II increment would be prioritized and the increment architected to enable what has variously been called timeboxing, time-certain development, or schedule-as independent variable, in which borderline-priority features are added or dropped to keep the increment on schedule. It would also be architected to accommodate foreseeable changes, such as user interfaces or transaction formats. For highly mission-critical systems, it would include a continuous verification and validation team analyzing reviewing, and testing the evolving product to minimize delayed-defect-finding rework.

While the stabilized development team is building the current increment and accommodating foreseeable changes, a separate system engineering team is dealing with sources of unforeseeable change and rebaselining the later increments' specifications and plans. Such changes can include new COTS releases, previous-increment usage feedback, current-increment deferrals to the next increment, new technology opportunities, or changes in mission priorities. Having the development team try to accommodate these changes does not work, as it destabilizes their schedules and carefully-worked-out interface specifications. At the end of each increment, the system engineering team also produces for expert review the feasibility evidence necessary to ensure low-risk, stabilized development of the next increment by the build-to-spec team.

ICM Commitment Milestones: The ICM commitment milestones correspond fairly closely with the Department of Defense acquisition milestones CD, A, B, and C as defined in DoDI 5000.2 [15]. The ICM commitment milestones occur at similar points in the acquisition life-cycle, but provide additional guidance and rigor for evaluating feasibility prior to commitment for the next stage. For example, the ICM DCR-milestone commitment to proceed into Development based on the validated LCA package (an Operations Concept Description, Requirements Description, Architecture Description, Life Cycle Plan, working prototypes or high-risk elements, and a Feasibility Rationale providing evidence of their compatibility and feasibility) corresponds fairly closely with DoD’s Milestone B commitment to proceed into the System Development and Demonstration phase.

ICM Metaphor: A simple metaphor to help understand the ICM is to compare ICM to poker games such as Texas Hold’em, to single-commitment gambling games such as Roulette. Many system development contracts operate like Roulette, in which a full set of requirements is specified up front, the full set of resources is committed to an essentially fixed-price contract, and one waits to see if the bet was a good one or not. With the ICM, one places a smaller bet to see whether the prospects of a win are good or not, and decides to increase the bet based on better information about the prospects of success.

What is Being Concurrently Engineered in the ICM?

Having addressed the stages, phases, and milestones in the ICM, let us now look at the activities. The top row of Figure 1 indicates that a number of system aspects are being concurrently engineered at an increasing level of understanding, definition, and development. The most significant of these aspects are shown in Figure 2, an extension of a similar view of concurrently engineered software projects developed as part of the RUP [9].

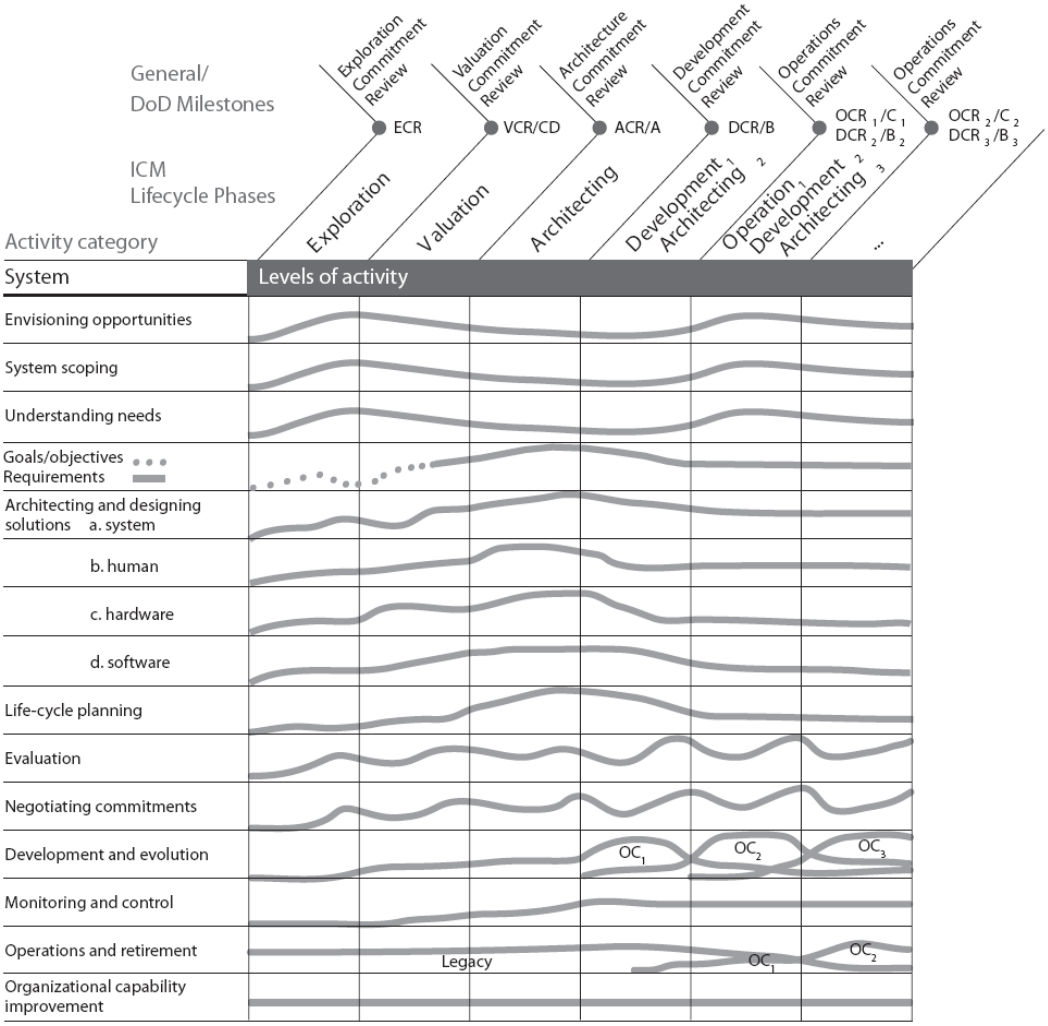

Figure 2. ICM Activity Categories and Level of Effort.

As with the RUP version, it should be emphasized that the magnitude and shape of the levels of effort will be risk-driven and likely to vary from project to project. In particular, they are likely to have mini risk/opportunity-driven peaks and valleys, rather than the smooth curves shown for simplicity in Figure 2. The main intent of this view is to emphasize the necessary concurrency of the primary success-critical activities shown as rows in Figure 2. Thus, in interpreting the Exploration column, although system scoping is the primary objective of the Exploration phase, doing it well involves a considerable amount of activity in understanding needs, envisioning opportunities, identifying and reconciling stakeholder goals and objectives, architecting solutions, life cycle planning, evaluation of alternatives, and negotiation of stakeholder commitments.

For example, if one were exploring the initial scoping of a SoS for a metropolitan area's disaster relief, one would not just interview a number of stakeholders and compile a list of their expressed mission needs. One would also envision and explore opportunities for reusing (parts of) other metropolitan-area disaster relief systems; for obtaining development funds from federal agencies; and for applying maturing virtual collaboration technologies. In the area of understanding needs, one would concurrently assess the capability and compatibility of existing disaster relief systems in the metropolitan area to determine which would need the most work to re-engineer into a SoS. One would also assess the scope of authority and responsibility of each existing system to determine whether the best approach would be a truly integrated and centrally-managed SoS or a best-effort interoperable set of systems. And one would explore alternative architectural concepts for developing and evolving the system, evaluate their relative feasibility, benefits, and risks for stakeholders to review, and negotiate commitments of further resources to proceed into a Valuation phase.

How is All This Concurrent Engineering Synchronized and Stabilized?

Figure 2 indicates that a great deal of concurrent activity occurs within and across the various ICM phases. To make this concurrency work, the anchor point milestone reviews are the mechanism by which the many concurrent activities are synchronized, stabilized, and risk-assessed at the end of each phase. Each of these anchor point milestone reviews, labeled at the top of Figure 2, is focused on developer-produced *evidence*, instead of PowerPoint charts and Unified Modeling Language (UML) diagrams, to help the key stakeholders determine the next level of commitment. For the Exploration Commitment Review (ECR), the focus is on a review of an Exploration Phase plan with the proposed scope, schedule, deliverables, and required resource commitment, by a key subset of stakeholders. The plan content is risk-driven, and could therefore be put on a single page for a small and non-controversial Exploration phase since there is minimal risk at this point—a much riskier Exploration phase would require a more detailed plan outlining how the risks will be re-evaluated and managed going forward. For the Valuation Commitment Review (VCR), the risk-driven focus is similar; the content includes the Exploration phase results and a valuation phase plan; and a review by all of the stakeholders involved in the Valuation phase. The Architecture Commitment Review (ACR) and the Development Commitment Review (DCR) reviews are based on the highly successful AT&T Architecture Review Board procedures described in [16]. For the ACR, only high-risk aspects of the Operational Concept, Requirements, Architecture, and Plans are elaborated in detail. And it is sufficient to provide evidence that at least one combination of those artifacts satisfies the Feasibility Rationale criteria shown in Figure 3 (similar to

Pass/Fail Feasibility Rationales

Evidence *provided by developer* and *validated by independent experts* that if the system is built to the specified architecture, it will:

- **Satisfy the requirements: capability, interfaces, level of service, and evolution**
- **Support the operational concept**
- **Be buildable within the budgets and schedules in the plan**
- **Generate a viable return on investment**
- **Generate satisfactory outcomes for all of the success-critical stakeholders**

All major risks resolved or covered by risk management plans

Serves as basis for stakeholders' commitment to proceed

Figure 3: Pass/Fail Feasibility Rationale Overview

the RUP LCO milestone), as compared to demonstrating this at the DCR for a particular choice of artifacts to be used for development.

The Operations Commitment Review (OCR) is different, in that it addresses the often much higher operational risks of fielding an inadequate system. In general, stakeholders will experience a factor of 2-to-10 increase in commitment level in going through the sequence of ECR to DCR milestones, but the increase in going from DCR to OCR can be much higher. These commitment levels are based on typical cost profiles across the various stages of the acquisition life-cycle. The OCR focuses on evidence of the adequacy of plans and preparations with respect to doctrine, organization, training, material, leadership, personnel, and facilities (DOTMLPF), along with plans, budgets, and schedules for production, fielding, and operations.

Taking this to the next level for SoS development, Figure 4 shows how these anchor point milestone reviews can be used to synchronize, stabilize, and manage risks across multiple supplier/vendor/strategic partner activities.

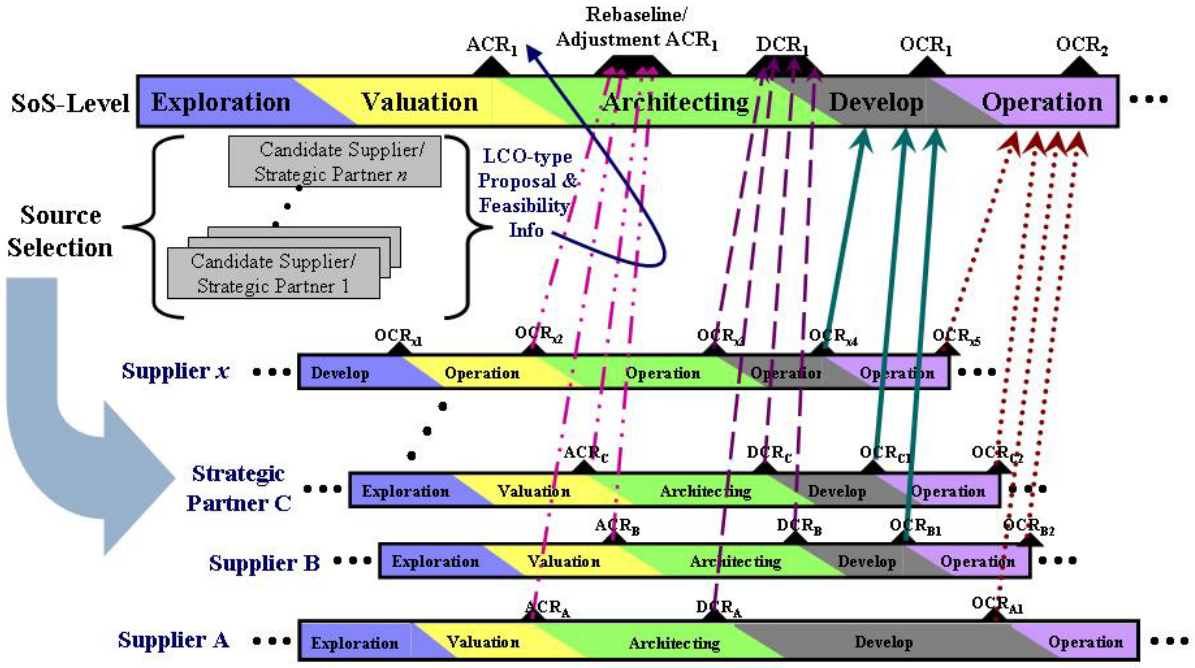

Figure 4. Combining SoS Engineering and Component Supplier Processes using ICM Anchor Point Reviews.

The major SoS-level milestones are compatible with those of Figure 1, but the realities of many SoSs involve some reinterpretation of their nature. Many SoSs will need to include COTS, legacy, or separately managed systems that are defined and incrementally released on different schedules than the SoS in Figure 4. The case that is shown in Figure 4 is one in which for reasons of training, provisioning, or operational stability, the main upgrades are batched into major SoS operational releases. Other SoS cases may require a more continual stream of upgrades such as security patches or electronic warfare countermeasures, in which parts of the SoS are more continuously vs. incrementally evolving.

Since not all of the source-selected component systems are defined on the same schedule, there will be delays in reconciling them into a common system architectural framework, requiring an additional SoS Rebaseline/Adjustment Architectural Commitment Review after the original SoS ACR used to drive source selection. The selected suppliers and partners will also need to participate in negotiating the particular SoS build-to architecture that will be baselined at the SoS DCR. And their incremental delivery schedules will not all be compatible with the incremental delivery schedule of the SoS in Figure 4.

Some of these suppliers, such as Supplier x, will already be operating and will feed incremental upgrade information into the SoS process during its definition and development stages. The SoS system engineers will try to predefine and anticipate these upgrades as much as possible, but will have to adapt to changes in Supplier x's capabilities or interfaces. If these are well-anticipated, the SoS development team in Figure 1 will accommodate them. If they are complex and unanticipated, threatening destabilization of the build-to-spec team, the SoS architecting rebaselining team will engineer an interim solution if necessary to keep the operational capability running. If a more comprehensive solution is needed for the longer term, the team will rebaseline the SoS architecture of the next increment to accommodate Supplier x's new capability. Changes from Strategic Partner C and Supplier B would be handled similarly.

In some additional cases, particularly for Supplier A who may be developing a longer-duration, non-subsetable, hardware-intensive initial operational capability, the SoS management will schedule Supplier A's IOC to synchronize with a later SoS increment (OCR_2 in Figure 4).

Chapter 2 of [7] provides several additional views of the ICM, including a more detailed version of Figure 1, some examples of how different risk patterns create different process sequences, a spiral view of the ICM phases and commitment milestones, and an illustration of how the concurrent increment development, increment Verification and Validation (V&V), and next-increment rebaselining activities address the need to simultaneously achieve high assurance and adaptiveness to rapid change.

Project Experience with ICM Principles

The ICM uses the critical success factor principles to extend several current spiral-related processes such as RUP, Win-Win Spiral, and Lean Development, in ways that more explicitly incorporate human-system factors into the system life cycle process. Some case studies of applying the ICM to remotely piloted vehicle, port security, and commercial medical infusion pump development are in Chapter 5 of [7]. Another good source of successful projects that have applied the critical success factor principles is the annual series of Top-5 software-intensive systems projects published in CrossTalk [17].

Table 3. Number of Top-5 Projects Explicitly Using ICM Principles

Year	Concurrent Engineering	Risk-Driven	Evolutionary Growth
2002	4	3	3
2003	5	4	3
2004	3	3	4
2005	4	4	5
Total (of 20)	16	14	15

The "Top-5 Quality Software Projects" were chosen annually by panels of leading experts as role models of best practices and successful outcomes. Table 3 summarizes each year's record with respect to usage of four of the five principles: concurrent engineering, risk-driven activities, and evolutionary and iterative system growth (most of the projects were not specific about stakeholder satisficing). Of the 20 top-5 projects in 2002 through 2005, 16 explicitly used concurrent engineering, 14 explicitly used risk-driven development, and 15 explicitly used evolutionary and iterative system growth, while additional projects gave indications of their partial use. Table 4 provides more specifics on the 20 projects (zero, one, or two stars correspond with minimal, partial, and strong application of the CSFs).

Table 4. Critical Success Factor (CSF) Aspects of Top-5 Software Projects

Software Project	CSF Degree	Concurrent Requirements/ Solution Development	Risk-Driven Activities	Evolutionary, Incremental Delivery
STARS Air Traffic Control	*	Yes	HCI, Safety	For multiple sites
Minuteman III Messaging (HAC/RMPE)	*	Yes	Safety	Yes; block upgrades
FA-18 Upgrades	*	Not described	Yes	Yes; block upgrades
Census Digital Imaging (DCS2000)	**	Yes	Yes	No; fixed delivery date
FBCB2 Army Tactical C3I	**	Yes	Yes	Yes
Defense Civilian Pay (DCPS)	**	Yes; risk-driven waterfall	Yes	Yes; over several years
Tactical Data Radio (EPLRS)	**	Yes	Yes	Yes
Joint Helmet-Mounted Cueing (JHMCS)	*	Yes; IPT-based	Not described	For multiple aircraft
Kwajalein Radar (KMAR)	**	Yes; IPT-based	Yes	For multiple radars
One SAF Simulation Testbed (OTB)	**	Yes	Yes	Yes
Advanced Field Artillery (AFATDS)	*	Initially waterfall; later concurrent	Not described	Yes; block upgrades
Defense Medical Logistics (DMLSS)		Initially waterfall	Not described	Yes; block upgrades
F-18 HOL (H1E SCS)		Legacy requirements-driven	Yes; COTS, display	No
One SAF Objectives System (OOS)	**	Yes	Yes	Yes
Patriot Excalibur (PEX)	**	Yes; agile	Yes; via short increments	Yes
Lightweight Handheld Fire Control	**	Yes	Yes	Yes
Marines Integrated Pay (MCTFS)		Initially waterfall	Not described	Yes; block upgrades
Near Imaging Field Towers (NIFTI)	**	Yes; RUP based	Yes	Yes
Smart Cam Virtual Cockpit (SC3DF)	**	Yes	Yes	Yes
WARSIM Army Training	**	Yes	Yes	Yes

Evidence of successful results of stakeholder-satisficing can be found in the annual series of University of Southern California (USC) e-services projects using the win-win spiral model as described in [18]. Since 1998 over 50 user-intensive e-services applications have used the win-win spiral model to achieve a 92% success rate of on-time delivery of stakeholder-satisfactory systems.

Conclusions

Future transformational, network-centric systems will have many usage uncertainties and emergent characteristics. Their hardware, software, and human factors will need to be concurrently engineered, risk-managed, and evolutionarily developed to converge on cost-effective system operations. They will need to be both highly dependable and rapidly adaptable to frequent changes.

The Incremental Commitment Model described in this article builds on experience-based critical success factor principles (stakeholder satisficing, incremental definition, iterative evolutionary growth, concurrent engineering, risk management) and the strengths of existing V, concurrent engineering, spiral, agile, and lean process models to provide a framework for concurrently engineering system-specific critical factors into the systems engineering and systems development processes. It provides capabilities for evaluating the feasibility of proposed solutions; and for integrating feasibility evaluations into decisions on whether and how to proceed further into systems development and operations. It presents several complementary views showing how the principles are applied to perform risk-driven process tailoring and evolutionary growth of a systems definition and realization; to synchronize and stabilize concurrent engineering; and to enable simultaneous high-assurance development and rapid adaptation to change. It analyzes the use of the critical success factor principles on the best-documented government software-intensive system acquisition success stories: the 2002-2005 *Cross Talk* Top-5 projects, and shows that well over half of them explicitly applied the principles.

Unfortunately, the current path of least resistance for a government program manager is to follow a set of legacy regulations, specifications, and standards that select, contract with, and reward developers for doing almost the exact opposite. Most of these legacy instruments emphasize sequential vs. concurrent engineering; risk-insensitive vs. risk-driven processes; early definition of poorly-understood requirements vs. better understanding of needs and opportunities; and slow, unscalable contractual mechanisms for adapting to rapid change.

This article has provided a mapping of the ICM milestones to the current DoD 5000.2 acquisition milestones that shows that they can be quite compatible. It also shows how projects could be organized into stabilized build-to-specification increments that fit current legacy acquisition instruments, along with concurrent agile change-adaptation and V&V functions that need to use alternative contracting methods. Addressing changes of this nature will be important, particularly if organizations are to realize the large potential value offered by investments in future network-centric SoSs. However, as military planners have had to recognize that there is no longer a Forward Edge of the Battle Area (FEBA) marking the boundary between Blue and Red forces, acquisition and financial planners will need to recognize that there is no longer a Cutover-type milestone marking the boundary between Research & Development and Operations & Maintenance. Instead, future life cycles will need to adapt to continuous incremental and evolutionary acquisition and development.

References

1. Krygiel, A., Behind the Wizard's Curtain; CCRP Publication Series, July, 1999.
2. Highsmith, J. (2000) Adaptive Software Development. Dorset House, New York, NY
3. Beck, K. (1999) Extreme Programming Explained, Addison Wesley, Reading, MA
4. Boehm, B. (2000) "Unifying Software Engineering and Systems Engineering", Computer, March 2000, pp. 114-116.
5. Clements, P. et al. (2002). Documenting Software Architectures, Addison Wesley.
6. Baldwin, K. "DoD Software Engineering and System Assurance: New Organization, New Vision," http://csse.usc.edu/events/2007/ARR/presentations/Baldwin.ppt, February 14, 2007.
7. Pew, R. W., and Mavor, A. S., (2007). Human-System Integration in the System Development Process: A New Look, National Academy Press.
8. Royce, W. E. (1998) Software Project Management. Addison Wesley.
9. Kruchten, P. (1999). The Rational Unified Process, Addison Wesley.
10. Boehm, B. (July 1996). Anchoring the Software Process. Software. pp. 73-82.
11. Boehm, B., and Lane, J. (May 2006). 21st Century Processes for Acquiring 21st Century Systems of Systems. CrossTalk. pp. 4-9.
12. Rechtin, E. (1991). Systems Architecting, Prentice Hall.
13. Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques, Proceedings, Wescon 1970.
14. Patterson, F. (1999). System engineering life cycles: life cycles for research, development, test, and evaluation; acquisition; and planning and marketing, in Sage, A., and Rouse, W., ed., (1999) Handbook of Systems Engineering and Management, Wiley, pp. 59-111.
15. U.S Department of Defense (2003). DoD Instruction 5000.2, Operation of the Defense Acquisition System.
16. Marenzano, J. et al. (2005) Architecture reviews: Practice and experience. IEEE Software, March/April 2005, pp. 34-43.
17. CrossTalk (2002-2005), "Top Five Quality Software Projects," January 2002, July 2003, July 2004, September 2005.
18. Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., and Madachy, R. (July 1998). Using the WinWin Spiral Model: A Case Study. Computer. pp.33-44.

Barry Boehm, Ph.D., is the TRW professor of software engineering and director of the Center for Systems and Software Engineering at the University of Southern California. He was previously in technical and management positions at General Dynamics, Rand Corp., TRW, and the Defense Advanced Research Projects Agency, where he managed the acquisition of more than $1 billion worth of advanced information technology systems. Dr. Boehm originated the

spiral model, the Constructive Cost Model, and the stakeholder win-win approach to software management and requirements negotiation.

University of Southern California
Center for Systems and Software Engineering
941 W. 37th Place, SAL Room 326
Los Angeles, CA 90089-0781
E-mail: boehm@usc.edu

Jo Ann Lane is currently a Principal at the University of Southern California Center for Systems and Software Engineering conducting research in the area of SoS engineering. In this capacity, she is currently working on a cost model to estimate the effort associated with system-of-system architecture definition and integration. She is also a part time instructor teaching software engineering courses at San Diego State University. Prior to this, she was a key technical member of Science Applications International Corporation's Software and Systems Integration Group responsible for the development and integration of software-intensive systems and systems of systems.

University of Southern California
Center for Systems and Software Engineering
941 W. 37th Place, SAL Room 328
Los Angeles, CA 90089-0781
E-mail: jolane@usc.edu