



21st Century Processes for Acquiring 21st Century Software-Intensive Systems of Systems

Dr. Barry Boehm and Jo Ann Lane
University of Southern California



Monday, 1 May 2006
Track 2: 8:00 – 11:15 a.m.
Ballroom B

Our experiences in helping to define, acquire, develop, and assess 21st century software-intensive systems of systems (SISOS) have taught us that traditional 20th century acquisition and development processes do not work well on such systems. This article summarizes the characteristics of such systems, and indicates the major problem areas in using traditional processes on them. We also present new processes that we and others have been developing, applying, and evolving to address 21st century SISOS. These include extensions to the risk-driven spiral model to cover broad (many systems), deep (many supplier levels), and long (many increments) acquisitions needing rapid fielding, high assurance, adaptability to high-change traffic, and complex interactions with evolving commercial off-the-shelf products, legacy systems, and external systems.

Between now and 2025, the ability of organizations and their products, systems, and services to compete, adapt, and survive will depend increasingly on software and the ability to integrate related software-intensive systems into systems of systems (SOS). As is being seen in current products (automobiles, aircraft, radios) and services (financial, communication, defense), software provides both competitive differentiation and rapid adaptability to competitive change. It facilitates rapid tailoring of products and services to different market sectors and rapid and flexible supply chain management.

The resulting software-intensive systems and SOS face ever-increasing demands to provide safe, secure, and reliable systems; provide competitive discriminators in the marketplace; support the coordination of multi-cultural global enterprises; enable rapid adaptation to change; and help people cope with complex masses of data and information. These demands will cause major differences in the processes currently used to define, design, develop, deploy, and evolve a diverse variety of software-intensive systems and software-intensive SOS (SISOS).

SISOS Trends and Their Influence on Systems and Software Engineering Processes

Today's trend towards larger, software-intensive systems and SOS often require much more complex systems and software engineering processes and better integration of these processes across the systems engineering and software engineering organizations. This section provides an overview of key SISOS historical trends, features, development organizations, and potential pitfalls.

Historical Evolution of Processes

Historically (and even recently for some forms of agile methods), systems and software development processes and maturity models were recipes for standalone stovepipe systems with high risks of inadequate interoperability with other stovepipe systems. Experience has shown that such collections of stovepipe systems cause unacceptable delays in service, uncoordinated and conflicting plans, ineffective or dangerous decisions, and an inability to cope with rapid change.

During the 1990s and early 2000s, standards such as the International Organization for Standardization (ISO)/

International Electrotechnical Commission (IEC) 12207 [1] and ISO/IEC 15288 [2] began to emerge that situated systems and software project processes within an enterprise framework. Concurrently, enterprise architectures such as IBM Zachman Framework [3], Reference Model for Open Distributed Processing (RM-ODP), [4] and the U.S. Federal Enterprise Architecture Framework [5], have been developing and evolving along with a number of commercial Enterprise Resource Planning (ERP) packages.

These frameworks and support packages are making it possible for organizations to reinvent themselves around transformational, network-centric SOS. As discussed in [6], these are necessary SISOS that have equally tremendous opportunities for success and risks of failure. Examples of successes are Federal Express; Wal-Mart; and the U.S. Command, Control, Intelligence, Surveillance, and Reconnaissance (C2ISR) system in Iraq. Examples of failures are the Confirm travel reservation system; K-Mart; and the U.S. Advanced Automation System for air traffic control. ERP packages have been the source of many successes and many failures, implying the need for considerable risk/opportunity assessment before committing to an ERP-based solution.

Table 1: *Software-Intensive Systems of Systems (SISOS) Solution Spaces*

| Characteristic | Range of Values |
|--|------------------------------|
| Size | 10-100 million lines of code |
| Number of external interfaces | 30-300 |
| Number of <i>coopetitive</i> suppliers | 20-200 |
| Depth of supplier hierarchy | 6-12 levels |
| Number of coordination groups | 20-200 |

Key SISOS Features

There are many definitions of SOS [7]. For this article, the distinguishing features of SOS are not only that they integrate multiple, independently developed systems, but also that they are very large, dynamically evolving, and unprecedented with emergent requirements and behaviors, and complex socio-technical issues to address.

Table 1 provides some additional characteristics of SISOS.

SISOS Development Organization Trends and Issues

There is often a lead system integrator who is responsible for developing SOS architecture, identifying the suppliers and vendors to provide various SOS components, adapting the architecture to meet evolving requirements and selected vendor limitations or constraints, overseeing the implementation efforts, and planning and executing the SOS level integration and test activities.

Keys to successful SOS development are the ability to: achieve timely decisions with a potentially diverse set of stakeholders; quickly resolve conflicting needs; and coordinate the activities of multiple vendors who are currently working together to provide capabilities for the SOS, but are often competitors on other system development efforts (sometimes referred to as “cooperative” relationships).

Potential SISOS Pitfalls

Our work in supporting SISOS development programs has shown that using a risk-driven spiral process with early attention to SISOS risks and systems architecting methods can avoid many of the SISOS development pitfalls [8]. A prioritized list of the top 10 SISOS risks we have encountered includes the following:

1. Acquisition management and staffing.
2. Requirements/architecture feasibility.
3. Achievable software schedules.
4. Supplier integration.
5. Adaptation to rapid change.
6. Systems and software quality factor achievability.
7. Product integration and electronic upgrade.
8. Commercial off-the-shelf (COTS) software and reuse feasibility.
9. External interoperability.
10. Technology readiness.

Strategies for addressing these risks are described in [8].

A Scalable Spiral Process Model for 21st Century SISOS

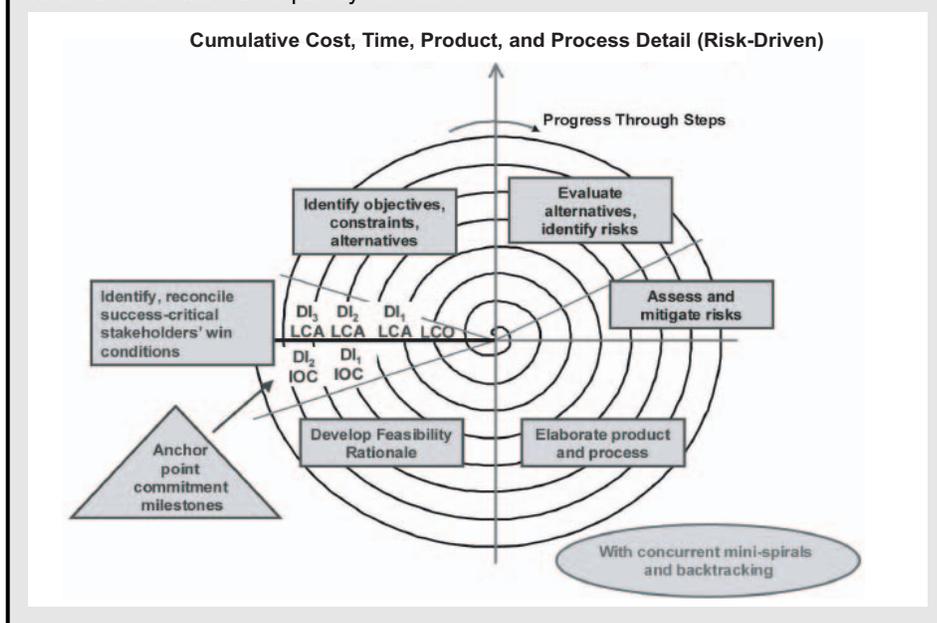
In applying risk management to the set of risks described above, the outlines of a hybrid plan-driven/agile process for developing SISOS product architecture are emerging. To keep SISOS developments from becoming destabilized from large amounts of change traffic, it is important to organize development into plan-driven increments in which the suppliers develop to interface specifications that are kept sta-

Spiral 2005 Process Model

The spiral model, first introduced in 1986, has continued to evolve to meet the needs of our evolving development processes. The figure below shows the latest version of the spiral model. Keys to this model are the use of success-critical stakeholder win-win negotiations to determine the system's objectives, constraints, and primary solution alternatives; and the life cycle anchor points that are the common system/software stakeholder commitment points for concurrent engineering. (These often replace the more traditional sequential Department of Defense system development milestone reviews such as System Requirements Review, Preliminary Design Review, and Initial Operational Capability.) The life cycle anchor points include:

- **Life Cycle Objectives (LCO):** The stakeholders' commitment to support system architecting.
- **Life Cycle Architecture/Development Increments ($\frac{DI_N}{LCA}$):** The stakeholders' commitment to support the full life-cycle development.
- **Incremental Operational Capabilities/Development Increments ($\frac{DI_N}{IOC}$):** The stakeholders' commitment to support operations.

The 2005 spiral framework provides a common frame of reference for the process views that we will subsequently describe.



ble by deferring changes, so that the systems can plug and play at the end of the increment (nobody has yet figured out how to do daily builds for these kinds of systems).

However, for the next increment to hit the ground running, an extremely agile team needs to be concurrently doing a continuous market, competition, and technology watch; change impact analysis; COTS refresh; and renegotiation of the next increment's prioritized content and the interfaces between the suppliers' next-increment interface specifications. This requires new approaches not only to process management, but also to staffing and contracting. The following sections elaborate on this emerging process architecture and its challenges.

21st Century SISOS Development and Evolution Modes

In the next 10 to 20 years, several 21st century system and software development and

evolution modes will have emerged as the most cost-effective ways to develop needed capabilities in the context of the trends discussed earlier. The four most common modes are likely to be exploratory development of unprecedented capabilities, business model-based user programming, hardware and software product lines, and network-centric SOS that will necessarily be software-intensive [6]. There are new challenges for organizations in the process of transforming themselves from collections of weakly coordinated, vertically integrated stovepipe systems into seamlessly interoperable network-centric SOS (NCSOS).

Architectures of these NCSOS are highly software-intensive and need to be simultaneously robust, scalable, and evolvable in flexible but controllable ways. The NCSOS development projects need processes such as the Internet spiral development process [9], but due to competitive pressures, their

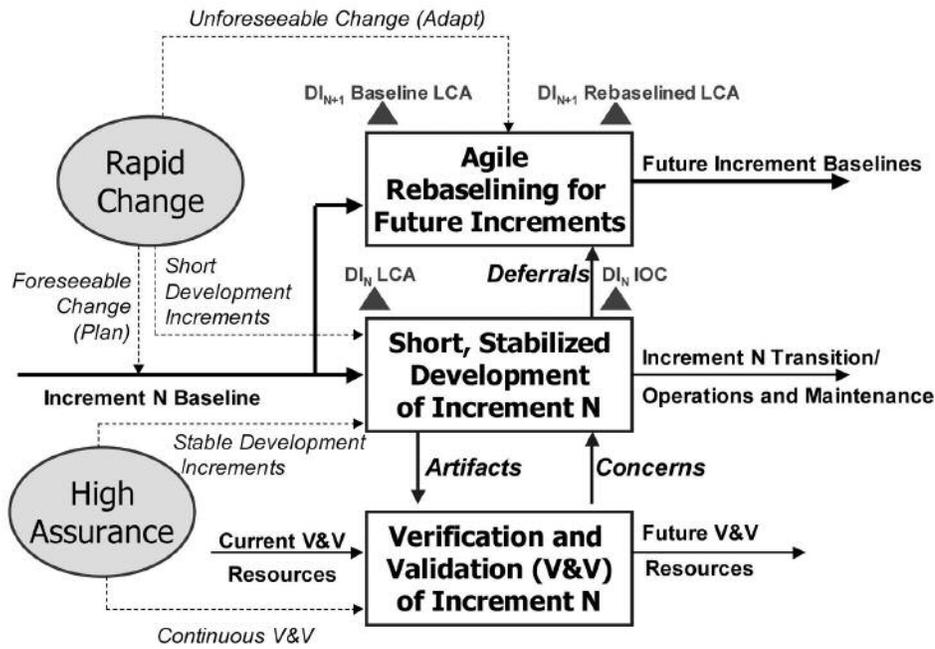


Figure 1: The Scalable Spiral Process Model – Increment Activities

processes must generally operate on much tighter timescales than were involved in the early evolution of the Internet.

Evolutionary development, business model-based user programming, and hardware and software product line development as described in [10] are key trends for the development and evolution of SISOS components. In the following sections, we describe an emerging, scalable spiral process model for developing and evolving 21st century product lines and NCSOS.

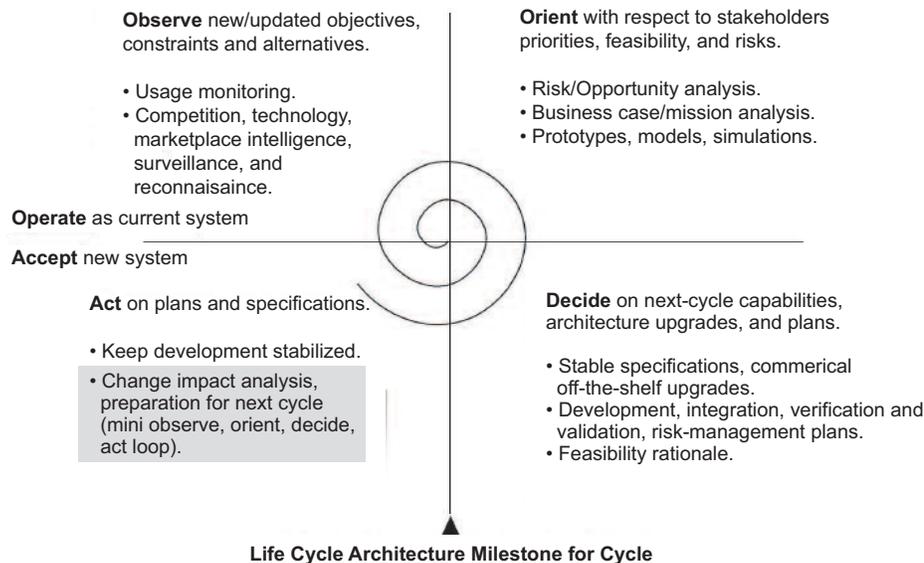
Overview of the Scalable Spiral Process Model

Complex systems and complex processes require multiple views to understand various aspects of the system and its development processes. And these multiple views

require some type of anchor points to help relate one view to another. Our first view begins with the spiral model view shown in the sidebar (see page 5).

Based on our experiences in adapting the spiral model to the development of SISOS representative of the 21st century trends discussed earlier, we have been converging on a scalable spiral process model. This model has shown in partial implementation to date to scale well from small e-services applications to super-large defense SOS and multi-enterprise supply chain management systems. The model contains familiar elements, but organizes them in ways that involve new approaches to enterprise organization, contracting, incentives, staffing, education, and career development. Figure 1 shows a single increment of

Figure 2: Observe, Orient, Decide, Act (OODA) Loop



the development and evolution portion of the model. It assumes that the organization has developed the following:

- A best-effort definition of the system’s steady-state capability.
- An incremental sequence of prioritized capabilities culminating in the steady-state capability.
- A feasibility rationale providing sufficient feasibility evidence for each increment and the overall system. This evidence should show that system architecture will support the incremental capabilities, that each increment can be developed within its available budget and schedule, and that the series of increments create a satisfactory return on investment for the organization and mutually satisfactory outcomes for the success-critical stakeholders.

As seen in Figure 1, the model is organized to simultaneously address conflicting 21st century challenges of rapid change and high assurance of dependability. It also addresses the need for rapid fielding of incremental capabilities with a minimum of rework, and the other major 21st century trends involving integration of systems and software engineering, COTS components, legacy systems, globalization, and user value considerations [10].

The need to deliver high-assurance incremental capabilities on short, fixed schedules means that each increment needs to be kept as stable as possible. This is particularly the case for a large SOS with deep supplier hierarchies (often six to 12 levels) in which a high level of rebaselining traffic can easily lead to chaos. In keeping with the use of the spiral model as a risk-driven process model generator, the risks of destabilizing the development process make this portion of the project into a waterfall-like, build-to-specification subset of the spiral model activities. The need for high assurance of each increment also makes it cost-effective to invest in a team of appropriately skilled personnel to continuously verify and validate the increment as it is being developed.

However, the previous discussion on *deferring change* does not imply deferring change impact analysis, change negotiation, and rebaselining until the beginning of the next increment. With a single development team and rapid rates of change, this would require a team optimized to develop stable plans and specifications to spend much of the next increment’s scarce calendar time performing tasks better suited to agile teams. Instead, Figure 1 shows how the spiral project would organize itself as follows:

- A plan-driven team transforms a build-

to DI₁ life-cycle architecture (LCA) package of validated specifications and plans (using one or more spiral cycles or intermediate builds) into a completed initial operational capability deliverable.

- Meanwhile, an independent verification and validation (IV&V) team continually verifies and validates the plan-driven increment under development.
- Meanwhile, an agile team adjusts and rebaselines the build-to specifications and plans for the next increment (DI₂) for hand-off to the plan-driven team.

The process in Figure 1 is then applied similarly to the subsequent cycles in the spiral chart.

The appropriate metaphor for addressing rapid change is not a build-to-specification metaphor or a purchasing-agent metaphor, but is an adaptive C2ISR metaphor as shown in Figure 2. It involves an agile team performing the first three activities of the C2ISR *Observe, Orient, Decide, Act* (OODA) loop for the next increments, while the plan-driven development team is performing the *Act* activity for the current increment. *Observing* involves monitoring changes in relevant technology and COTS products in the competitive marketplace, in external interoperating systems, and in the environment; and monitoring progress on the current increment to identify slowdowns and likely scope deferrals. *Orienting* involves performing change impact analysis, risk analysis, and trade-off analysis to assess candidate rebaselining options for upcoming increments. *Deciding* involves stakeholder renegotiation of the content of upcoming increments, architecture rebaselining, and the degree of COTS upgrading to be done to prepare for the next increment. It also involves updating the future increments' feasibility rationales to ensure that renegotiated scopes and solutions can be achieved within budget and schedule. The LCA milestone at the bottom of Figure 2 corresponds with the DI_{N+1} Rebaselined LCA increment in Figure 1.

A successful rebaseline means that the plan-driven development team can hit the ground running at the beginning of the Act phase of developing the next increment, and the agile team can hit the ground running on rebaselining definitions of the increments beyond. Figure 3 shows how this three-team cycle (lean, plan-driven, stabilized developers; thorough IV&V people; and agile, proactive rebaseline people) plays out from one increment to the next, including the early product line or SOS inception and elaboration phases with their pass-fail, life-cycle objectives and LCA exit milestones. The

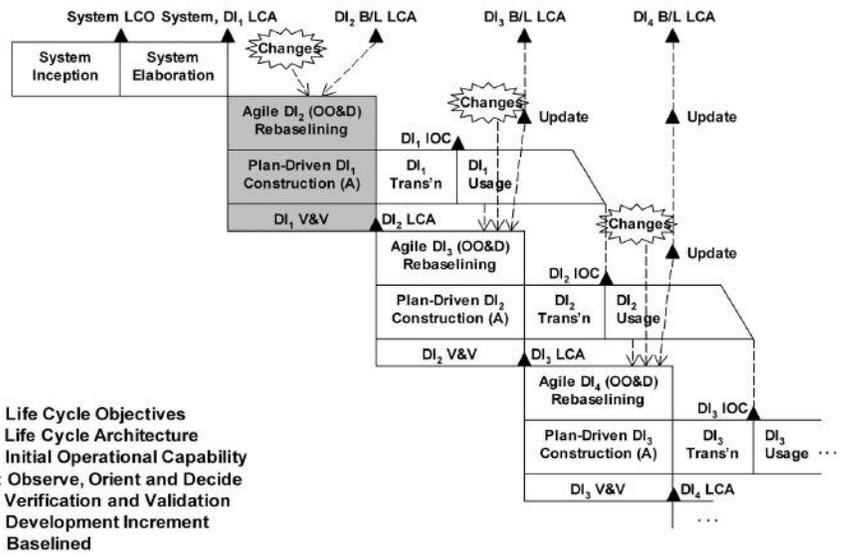


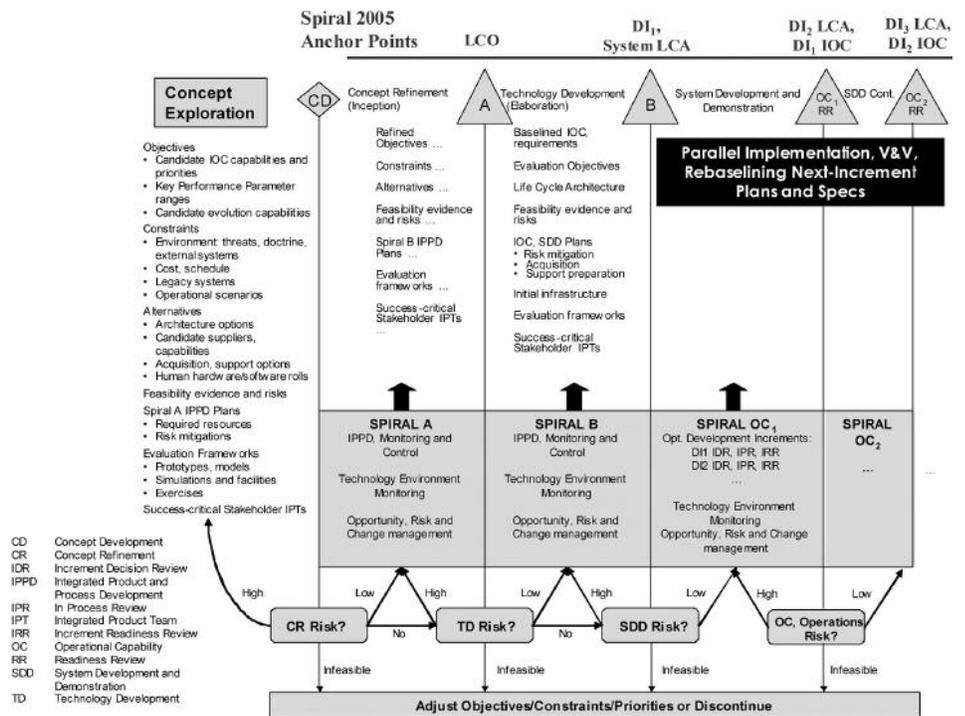
Figure 3: The Scalable Spiral Process Model: Life Cycle View

shaded activities in Figure 3 are the same set of activities that are shown in detail in Figure 1. Note that OO&D in each agile rebaselining increment stands for observe, orient, and decide, and *not* object-oriented design. The (A) below it stands for the *Act* portion of the OODA loop for the current increment. Note also that, as much as possible, usage feedback from the previous increment is not allowed to destabilize the current increment, but is fed into the definition of the following increment. Of course, some level of mission-critical updates will need to be fed into the current increment, but only when the risk of not doing so is greater than the risk of destabilizing the current increment.

As with command and control, the OO&D rebaselining portion of the project is not a sequential waterfall process. Instead, it is a risk-driven set of concurrent prototyping, analysis, and stakeholder renegotiation activities that lead to a best-possible redefinition of plans and specifications to be used by the stabilized development team for the next increment. For people familiar with the Department of Defense 5000 series of acquisition milestones, Figure 4 provides a mapping of them onto the Spiral 2005 anchor points.

Acquisition as C2ISR Versus Purchasing
The 20th century purchasing agent or contracts manager is most comfortable with a

Figure 4: Spiral 2005 Anchor Points in Relation to Department of Defense 5000 Milestones



fixed procurement to a set of pre-specified requirements; selection of the least-cost, technically adequate supplier; and a minimum of bothersome requirements changes. Many of our current acquisition institutions – regulations, specifications, standards, contract types, award fee structures, reviews and audits – are optimized around this procurement model.

Such institutions have been the bane of many projects attempting to deliver successful systems in a world of emerging requirements and rapid change. The project people may put together good technical and management strategies to do concurrent problem and solution definition, teambuilding, and mutual-learning prototypes and options analysis. Then they find that their progress payments and award fees involve early delivery of complete functional and performance specifications. Given the choice between following their original strategies and getting paid, they proceed to marry themselves in haste to a set of premature requirements then find themselves repenting at leisure for the rest of the project (if any leisure time is available).

Build-to-specification contract mechanisms still have their place, but it is just for the stabilized increment development. If such mechanisms are applied to the agile rebaselining teams, then frustration and chaos ensues. What is needed for the three-team approach are separate contracting mechanisms for the functions, under an overall contract structure, enabling them to be synchronized and rebalanced across the life cycle. Also needed are source-selection mechanisms more likely to choose the most competent supplier, using such approaches as competitive exercises to develop representative system artifacts using the people, products, processes, methods, and tools in the offeror's proposal.

A good transitional role model is the Command Center Processing and Display-Replacement (CCPDS-R) project described in [11]. Its U.S. Air Force customer and TRW contractor (selected using a competitive exercise such as the one described earlier) reinterpreted the traditional defense regulations, specifications, and standards. They held a preliminary design review: This was not a PowerPoint show at month four, but a fully validated architecture and demonstration of the working, high-risk user interface and networking capabilities at month 14. The resulting system delivery, including more than one million lines of software source code, exceeded customer expectations within budget and schedule.

Other good acquisition approaches are the Scandinavian Participatory Design approach [12], Checkland's Soft Systems Methodology [13], lean acquisition and development processes [14], and Shared Destiny-related contracting mechanisms and award fee structures [15, 16]. These all reflect the treatment of acquisition using a C2ISR metaphor rather than a purchasing-agent metaphor.

Model Experience to Date and Conclusions

The scalable spiral model has been evolving with experience and has not yet been fully implemented on a large, completed project. However, its principles and practices build on many successful project experiences and unsuccessful project lessons learned. Specific examples of projects that have successfully balanced agile and plan-driven methods are the agile-based ThoughtWorks lease management project [17] and the plan-based CCPDS-R project [11]. More generally, J. Collins' book, "Good to Great" [18] identifies 11 companies with exceptional performance records as having successfully transformed themselves into having both a strong ethic of entrepreneurship and a strong culture of discipline.

The use of concurrent IV&V teams has been successfully practiced and evolved since the 1970s [19]. More recent successful continuous IV&V practices include the continuous build practices at Microsoft [20] and in agile methods [21]. Proactive investments in agile next-increment teams are successfully used in exploiting disruptive technologies at companies such as Hewlett Packard (HP), Seagate, and Johnson and Johnson [22]; and in practicing open innovation in companies such as HP, IBM, Intel, and Lucent [23]. Successful use of the anchor point milestones and evolutionary development using the Rational Unified Process [16] and the WinWin Spiral model [24] has been experienced on numerous small, medium, and large software projects and on hardware projects at such companies as Xerox and Johnson and Johnson. Partial implementations of the model are also providing improvement and are being evolved on the large-scale U.S. Army Future Combat Systems program, large space systems, and commercial supply chain systems [8].

Experience to date indicates that the three teams' activities are not as neatly orthogonal as they look in Figures 1 and 3. Feedback on development shortfalls from the IV&V team either requires a response

from the development team (early fixes will be less disruptive and expensive than later fixes), or deferral to a later increment, adding work and coordination by the agile team. The agile team's analysis and prototypes addressing how to accommodate changes and deferred capabilities need to draw on the experience and expertise of the plan-driven development team, requiring some additional development team resources and calendar time. Additional challenges arise if different versions of each increment are going to be deployed in different ways into different environments. The model has sufficient degrees of freedom to address such challenges, but they need to be planned within the project's schedule and budget.

In working with our commercial and aerospace affiliates on how they can best evolve to succeed as 21st century enterprises, we have found several 20th century process-related institutions that need to be significantly rethought and reworked to contribute to success. Two key leading areas for SISOS development that need rethinking are acquisition practices and human relations [10]. Other institutions that also need rethinking and rework are continuous process improvement (repeatability and optimization around the past versus adaptability and optimization around the future), supplier management (adversarial win-lose versus team-oriented win-win), internal research and development strategies (core capability research plus external technology experimentation versus full-spectrum self-invention), and enterprise integration (not-invented-here stovepipes versus enterprise-wide learning and sharing). ♦

References

1. International Organization for Standardization. Information Technology – Software Life Cycle Processes. ISO/IEC 12207: 1995. Geneva, Switzerland: ISO, 1995.
2. International Organization for Standardization. Systems Engineering – System Life Cycle Processes. ISO/IEC 15288: 2002. Geneva, Switzerland: ISO, 2002.
3. Zachman, J. "A Framework for Information Systems Architecture." IBM Systems Journal 26.3 (1987): 276-292.
4. Putman, J. Architecting With RM-ODP. Prentice Hall, 2001.
5. Federal Chief Information Officer Council. A Practical Guide to Federal Enterprise Architecture Vers. 1.0. Washington, D.C.: FCIO, Feb. 2001.
6. Harned, D., and J. Lundquist. "What Transformation Means for the

- Defense Industry.” The McKinsey Quarterly 3 Nov. 2003: 57-63.
7. Lane, J., and R. Valerdi. “Synthesizing System-of-Systems Concepts for Use in Cost Estimation.” IEEE SMC, 2005.
 8. Boehm, B., A.W. Brown, V. Basili, and R. Turner. “Spiral Acquisition of Software-Intensive Systems of Systems.” CROSSTALK May 2004: 4-9 <www.stsc.hill.af.mil/crosstalk/2004/05/0405boehm.html>.
 9. U.S. Air Force-Scientific Advisory Board. Information Architectures Study. Washington, D.C.: U.S. Air Force, 1994.
 10. Boehm, B. “Some Future Trends and Implications for Systems and Software Engineering Processes.” USC-CSE-TR-2005-507. Los Angeles, CA: University of Southern California, 2005.
 11. Royce, W.E. Software Project Management. Addison-Wesley, 1998.
 12. Ehn, Pelle. Work-Oriented Design of Computer Artifacts. Lawrence Erlbaum Assoc., 1990.
 13. Checkland, P. Systems Thinking, Systems Practice. 2nd ed. Wiley, 1999.
 14. Womack, J., and D. Jones. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Simon & Schuster, 1996.
 15. Deck, M., M. Strom, and K. Schwartz. “The Emerging Model of Co-Development.” Research Technology Management Dec. 2001.
 16. Rational, Inc. Driving Better Business With Better Software Economics. Rational Software Corp., 2001.
 17. Elssamadisy, A., and G. Schalliol. Recognizing and Responding to ‘Bad Smells’ in Extreme Programming. Proc. of the 24th International Conference on Software Engineering, Orlando, FL, May 2002: 617-622.
 18. Collins, J. Good to Great. Harper Collins, 2001.
 19. Rubey, J.R., J.A. Dana, and P.W. Biché. “Quantitative Aspects of Software Validation.” IEEE Transactions on Software Engineering June 1975: 150-155.
 20. Cusumano, M., and R. Selby. Microsoft Secrets. Harper Collins, 1996.
 21. Beck, K. Extreme Programming Explained. Addison-Wesley, 1999.
 22. Christensen, C. The Innovator’s Dilemma (Harper Business Essentials). Harper Collins, 2000.
 23. Chesbrough, H. Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business School Press, 2003.
 24. Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy. “Using the WinWin Spiral Model: A Case Study.” IEEE Computer July 1998: 33-44.

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, and the Defense Advanced Research Projects Agency, where he managed the acquisition of more than \$1 billion worth of advanced information technology systems. 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 Software Engineering
941 W 37th PL
SAL RM 328
Los Angeles, CA 90089-0781
E-mail: boehm@usc.edu**



Jo Ann Lane is currently a research assistant supporting software engineering and system-of-systems research activities at the University of Southern California’s Center for Software Engineering. In this capacity, she is currently working on a cost model to estimate the effort associated with system-of-systems architecture definition and integration. Prior to this, she was a key technical member of Science Applications International Corporation’s Software and Systems Integration Group. She has over 28 years of experience in the development of software-intensive systems.

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

COMING EVENTS

June 17-21

33rd Annual International Symposium on Computer Architecture (ISCA 2006)



Boston, MA
www.ece.neu.edu/conf/isca2006

June 19-22

CISC 2006 Combat Identification Systems Conference

Orlando, FL
www.usasymposium.com/cisc

June 25-30

18th Annual FIRST (Forum of Incident Response and Security Teams)

Conference on Computer Security and Incident Handling
Baltimore, MD
www.first.org/conference/2006

June 26-29

*SERP 2006
International Conference on Software Engineering Research and Practice*
Las Vegas, NV

<http://people.cs.und.edu/~reza/SERP06.html>

June 26-30

2006 Better Software Conference and Expo
Las Vegas, NV
www.sqe.com/bettersoftwareconf.org

June 27-29

*CMSE Europe
Components for Military and Space Electronics Workshop and Conference*
Portsmouth, U.K.

www.cmse-eur.com

April 16-19, 2007

2007 Systems and Software Technology Conference



Salt Lake City, UT
www.sstc-online.org