Principles for Successful Systems Engineering
Barry Boehm\textsuperscript{a}, Supannika Koolmanojwong\textsuperscript{*a}, Jo Ann Lane\textsuperscript{a}, and Richard Turner\textsuperscript{b}

\textsuperscript{a}University of Southern California (USC), Los Angeles, California, 90089, USA
\textsuperscript{b}Stevens Institute of Technology, Hoboken, New Jersey, 07030, USA

Abstract
This paper summarizes several iterations in developing a compact set of four key principles for successful systems engineering, which are 1) Stakeholder Value-Based System Definition and Evolution 2) Incremental Commitment and Accountability 3) Concurrent Multidiscipline System Definition and Development, and 4) Evidence-Based and Risk-based Decisionmaking. It provides a rational for the principles, including short example case studies of failed projects that did not apply the principles, and of successful projects that did. It will compare the principles with other sets of principles such as the Lean Systems Engineering and the Hitchins set of principles for successful systems and systems engineering.

© 2012 Published by Elsevier Ltd. Selection

Keywords: The Incremental Commitment Spiral Model; Systems Engineering Principles; Stakeholder Value Proposition; Feasibility Evidence; Risk-Based Decisionmaking; Concurrent Multidiscipline System Development

1. Introduction
Systems engineering and development processes and principles were created 40 to 50 years ago. At that time their users could generally assume the system would operate as a standalone device, its requirements could be specified up front, its behavior would seldom change, it could be designed top-down, and starting from a clean blackboard or sheet of paper. Given current trends in system development, these assumptions are becoming less valid. Unfortunately, the original processes and principles have been slow to change. The Incremental Commitment Spiral Model (ICSM) is a life cycle
process model generator that has been evaluated to be a flexible but robust framework for system development. The ICSM combines the strengths of various current process models and limits their weaknesses. The ICSM, like the Vee model, emphasizes early verification and validation, but allows for multiple-incremental interpretation and emphasizes concurrent rather than sequential engineering. Compared to the Spiral Model, the ICSM also focuses on risk-driven activity prioritization, but offers an improvement by adding well-defined in-process milestones. While ICSM, RUP, and MBASE perform concurrent engineering that stabilizes the software process at anchor point milestones, ICSM also supports integrated hardware-software-human factors oriented development. Comparing with Agile methods, ICSM embraces adaptability to unexpected change and at the same time allows scalability.

The followings are the four key trends that require changes and how the ICSM suggests in process changes if we are to develop successful systems:

- **Increasingly complex, global systems of systems**: The Internet and personal communication devices are connecting most everything and everyone together. Developers of 21st century systems will need to consider how they fit, not only within their own enterprise, but often within multiple networks of independently evolving, codependent systems. The ICSM commitment milestones require evidence that the system be scalable to operate with its intended full-up environment, and be interoperable with its codependent systems for global collaborative processes.

- **Emergent requirements**: Asking people what they would like in their user interface usually results in a response such as “I’m not sure, but I’ll know it when I see it” (IKIWISI). The most appropriate user interfaces and collaboration modes for a complex human-intensive system are not specifiable in advance, but emerge with usage. Forcing users to specify them precisely in advance of development generally leads to poor business or mission performance and expensive late rework and delays. The ICSM provides support for incremental and concurrent definition of system requirements and solutions, including competitive prototyping approaches.

- **Rapid change**: Trying to stay competitive in a world of increasingly rapid changes requires new levels of agility, and shorter times between new releases of products and services. The ICSM’s incremental definition and development stages directly support shorter increments, more agile methods, and evolutionary development.

- **High assurance of qualities**: At the same time that systems engineering and development need to become more agile, the growing interdependence of systems and people requires systems to have higher assurance levels. Just assuring any of the quality attributes for a complex system of systems is difficult. It is nearly impossible to get agreement among multiple system owners with widely disparate quality priorities. The term “satisficing” means not everybody gets everything they want, but everybody gets something they are satisfied with. The ICSM’s principles of stakeholder satisficing and evidence-based commitment milestones help ensure that key stakeholders’ primary quality concerns are addressed.

2. Key Principles

To understand and apply the process model appropriately, the process users should understand the core concepts of the model. The followings are the key principles of the ICSM and example case studies.

*Principle 1: Stakeholder Value-Based System Definition and Evolution.* An interdisciplinary approach and means are enabling the realization of successful systems. A system will be successful if and only if it makes winners of its success-critical stakeholders. Thus, in order to create a successful system, you need to identify which stakeholders are success-critical, to determine their value propositions or win conditions, and to define, design, develop, and evolve a mutually satisfactory or win-win system with respect to their value propositions. If a project fails to include and address the value propositions of its
success-critical stakeholders such as end-users, maintainers, interoperators, or suppliers, these stakeholders will frequently feel little commitment (or active hostility) to the project and either underperform, decline to use, or block the use of the results.

**Principle 2: Incremental Commitment and Accountability.** Without key personnel commitment and accountability for the system under development, there is no way to build trust among the system’s stakeholders. It is too easy to overpromise and depart. And there must be clear visibility of progress versus plans up and down the supplier chain. If success-critical stakeholders are not accountable for their commitments, lack of commitments, and associated consequences (good or bad), they may not provide necessary commitments or decisions in a timely manner and are likely to be drawn away to other pursuits when they are most needed.

**Principle 3: Concurrent Multidiscipline System Definition and Development.** The fundamental assumptions underlying sequential processes, prespecified requirements, and functional-hierarchy product models began to be seriously undermined in the 1970s and 1980s. On the other hand, the increasing pace of change in technology, competition, organizations, and life in general made assumptions about stable, prespecifiable requirements unrealistic. The existence of cost-effective, competitive, incompatible commercial products or other reusable non-developmental items (NDIs) made it necessary to evaluate and often commit to solution components before finalizing the requirements. The difficulty of adapting to rapid change with brittle, optimized, point-solution architectures generally made optimized first-article design to fixed requirements unrealistic. The ICSM emphasizes the principle of concurrent rather than sequential work on understanding needs, envisioning opportunities, system scoping, system objectives and requirements determination, architecting and designing of the system and its hardware, software, and human elements, life cycle planning, and development of feasibility evidence. So, it is important to do everything in parallel especially in the early phases. If definition and development of requirements and solutions; hardware, software, and human factors; or product and process definition are done sequentially, the project is likely both to go more slowly, and to make early, hard-to-undo commitments that cut off the best options for project success.

**Principle 4: Evidence-Based and Risk-based Decision making.** Having evidence serves as the principal decision criterion at milestone decision reviews is a considerable step forward from traditional schedule-based or event-based reviews. This is better, but frequently leads to “Death by PowerPoint and SysML” reviews, which present much design detail, but there is little time to determine whether or not the design will meet the system’s key performance parameters. Such evidence of feasibility is generally desired, but is considered as an optional appendix and not a project deliverable. Thus, it is often neglected. In an ICSM evidence-based review, the feasibility evidence is a first-class deliverable. As such, its planning and preparation becomes subject to earned value management and is factored into progress payments and award fees. Investments in feasibility evidence have been found to pay off significantly in development rework avoidance [Boehm-Valerdi-Honour, 2008]. The link between evidence-based and risk/opportunity-based decisionmaking is that shortfalls in evidence are uncertainties or probabilities of loss or gain. If the Opportunity Exposure OE is high and the window of opportunity is closing rapidly, proceeding at least incrementally with a small amount of evidence can be the best decision. Thus, we can see that Principle 4 brings all of the other principles together. It involves concerns with the stakeholders’ value propositions in making decisions as in Principle 1; with proceeding incrementally as in Principle 2; and with synchronizing and stabilizing the concurrent activity prescribed in Principle 3.

3. Case Studies

3.1. Failure Stories
**Principle 1: The Too-Good Road Surface Assessment Robot.** The Carnegie Mellon University Road Surface Assessment Robot project seemed to have done everything right. It delivered a system with radically higher accuracy, efficiency, and labor savings with respect to its manual predecessor system. However, its end product still sits in a storage room unused [Latimer, 2008]. The project was executed over three years. The first year explored alternative operational concepts for operating a robotic vehicle that would identify and mark deviations from roughness standards on the roadway. The business case for the winning design indicated a likely 100:1 time and cost savings for inspection. The first year ended with a specification for the system’s functional and performance requirements that had been thoroughly validated. The second year involved selection of outsourced components and detailed build-to specifications for the robot vehicle. The robot vehicle was successfully developed in the third year, and passed an acceptance test on a test track that had representative deviations built into it. It also passed the test of reducing inspection time by a factor of 100. The system operated entirely according to the specifications, but not according to expectations. A post-analysis by the lead engineer for surface assessment determined that all of the deviations were correctly reported by the system, but over 99% of them were minor, not ride-quality threatening. The roadway subcontractor felt that its quality reputation was being unfairly degraded. As a result, the best-acceptable management solution was to discontinue the robot project and to continue to use manual surface assessment methods. The lesson learned is the operational concept analyses were good, but the operational scenarios were focused only on the technical performance of the robot vehicle, and not on the effect of off-nominal outcomes or sociotechnical aspects on the stakeholders involved.

**Principle 3: Sequential RPV Systems Engineering and Development.** A sequential approach that is representative of several recent government acquisition programs would use the demo results to create the requirements for a proposed program that used the agent-based technology to develop a 4:1 ratio system that enabled a single operator to control 4 RPVs in battlefield-based, sea-based, and home-country based RPV operations. A number of assumptions were made to sell the program at an optimistic cost of $1 billion and schedule of 40 months. These requirements were included in a Request for Proposals (RFP). The winning bidder provided an even more impressive demo of agent technology and a proposal indicating that all of the problems were well understood, that a preliminary design review (PDR) could be held in 120 days, with $800M. At the System Functional Requirements Review (SFRR), the items reviewed were transcriptions and small elaborations of the requirements in the RFP. They did not include any functions for coordinating the capabilities, and included only sunny-day operational scenarios. However, at the PDR, the contractor could not show feasible solutions for several critical and commonly-occurring scenarios. Since the schedule was tight and the contractor had almost run out of systems engineering funds, their management proposed to address the problems by using a “concurrent engineering” approach of having the programmers develop the off-nominal capabilities while the systems engineers were completing the detailed design. Having no other face-saving alternative to declaring the PDR to be failed, the customers declared the PDR to be passed. Actually, this is a pernicious misuse of “concurrent engineering,” since there is not time to produce feasibility evidence and to synchronize and stabilize the numerous off-nominal approaches taken by the programmers and the detailed designers. Eventually, the 1:1 capability was achieved and the system delivered, but with reduced functionality, a cost of $3 billion, a schedule of 80 months and components integration problem. Even worse, the hasty patching to get the first article delivered left the customer with a brittle, poorly documented, poorly-tested system that would be the source of many expensive years of system ownership and sub-par performance.

### 3.2. Success Stories

**Principle 2: Software Productivity Project** [5]. In 1980s, TRW was about to introduce a new project whose objective was to revolutionize the way the company developed software. The project was to
introduce a new work environment where all project members has individual offices with workstations and communicated electronically via a Local Area Network (LAN). The challenge was to convince the typical project personnel to adopt a new culture and technologies that they were bringing. The project did not run smoothly all the time, being a guinea pig trying a new culture created a lot of resistance and jealousy. Various new processes and new tools are not welcomed by all. But the management tried to involve all stakeholders and ensure that they understand about the project. Team building is another critical activity. It needs trusting and caring that break the glass shell. New technologies such as email that speed up the communication but at the same time it can incur negativity productivity such as over-reliance on email and accidentally “Reply All”. That led to some team issues that had to be solved by an all-hands where management encouraged the team to “talk” to each other rather than defaulting to email as a means of communication. At the end, the productivity project was fortunate to have a number of the features that enable successful software projects such top management support, capable and enthusiastic team members, realistic budgets and schedules. In comparing the productivity project with some of other projects, the key beautiful team enablers were: identifying and involving all of the success-critical stakeholders, a lot of work upfront on listening, exploring, and team building, developing a shared vision for the product and its results, identifying a manager with an open mind and good listening and team building skills, encouraging creative ideas from outside and within, paying attention and addressing the team’s needs, respectfully redeploying incompatible performers, negotiating win-win resolutions of stakeholder conflicts, and carefully monitoring progress and proactively addressing win-lose threats.

Principle 4: CCPDS-R. The Command Center Processing and Display System Replacement (CCPDS-R), a project to re-engineer the command center aspects of the US early missile warning system. It covered not only the software but also the associated system engineering and computing hardware procurement. The software effort involved over 1 million lines of Ada code, across a family of three related user capabilities. The core capability was developed on a 48-month fixed price contract between 1987 and 1991. One of the high risk elements is the extremely high dependability requirements for a system of this nature. Others were the ability to re-engineer the sensor interfaces, the commander situation assessment and decision-aid displays, and the critical algorithms in the application. The usual DoD-STD-2167A Preliminary Design Review (PDR) to review paper documents and briefing charts around Month 6 was replaced by a PDR at Month 14 that demonstrated working software for all the high-risk areas, particularly the network operating system, the message-passing middleware and the graphic user interface (GUI) software. The PDR also reviewed the completeness, consistency, and traceability of all of the Ada package interface specifications, as verified by the Rational Ada compiler and R-1000 toolset. Thus, a great deal of system integration was done before the software was developed. Evidence of achievable software productivity was provided via a well-calibrated cost and schedule estimation model, in this case an Ada version of the Constructive Cost Model (Ada COCOMO). Since the CCPDS-R plans and specifications were machine processable, the project was able to track progress and change at a very detailed level. This enabled the developers to anticipate potential downstream problems and largely handle them via customer collaboration and early fixes. USAF/ESC and TRW agreed that the contract award fee for good performance would not just go into the TRW corporate profit coffers, but also for individual project performer bonuses. This not only enhanced motivation and teamwork, but made the CCPDS-R project personnel turnover the lowest in TRW large-project history.

4. Comparison to other principles

A comparison of the four key ICSM principles with characteristics of the Lean principles [4] and principles for successful systems and systems engineering by Hitchins [3] was conducted and a summary of the results are shown in Table 1. Although there are differences in the level of detail in the way each set of principles is specified, there are no substantial differences with respect to a guidance to a successful
system. All focus on team working, efficiently performing value-adding activities at the appropriate point in the development life cycle and eliminating activities that don’t add value.

Table 1. Key Principles Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Stakeholder value-based system definition and evolution</td>
<td>- See the whole</td>
<td>- Holism “Consider whole when making decisions</td>
</tr>
<tr>
<td></td>
<td>- Empower the team</td>
<td>- Systems Approach “Consider System of Interest in context”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Progressive Satisfying “system success equals stakeholder success”</td>
</tr>
<tr>
<td>2. Incremental commitment and accountability</td>
<td>- Amplify learning</td>
<td>- Organism Analogy “Consider systems to have dynamic behavior”</td>
</tr>
<tr>
<td></td>
<td>- Decide as late as possible</td>
<td>- Adaptive Optimizing “Solve problems progressively over time”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Progressive Entropy Reduction “Continue to make systems work over time”</td>
</tr>
<tr>
<td>3. Concurrent multidiscipline system definition and</td>
<td>- Deliver as fast as possible</td>
<td>- Synthesis “Bring parts together to create solutions”</td>
</tr>
<tr>
<td>development</td>
<td>- Empower the team</td>
<td>- Adaptive Optimizing “Solve problems progressively over time”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Evidence and risk-driven decisionmaking</td>
<td>- Build integrity in</td>
<td>- Holism “Consider whole when making decisions</td>
</tr>
<tr>
<td></td>
<td>- Eliminate waste</td>
<td>- Progressive Satisfying “system success equals stakeholder success”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Progressive Entropy Reduction “Continue to make systems work over time”</td>
</tr>
</tbody>
</table>

5. Conclusion

If things aren’t changing much in your domain, and you already have a way to create successful systems, you should keep on using it. But you will be in a shrinking minority as the 21st century pace of change increases. Wherever we look, things are rapidly changing: technology, competition, the nature of systems’ users, and in the infrastructure used to create systems. The four key ICSM principles of the Incremental Commitment Spiral Model (ICSM) guide you to adapt to the changes and lead you to successful systems engineering development. Case studies show that the projects yield satisfactory results when they have a successful usage of the four principles, on the other hand, the failure case studies failed to apply the principles.

References