Introduction to the ICSM

What Is the ICSM?

The ICSM is a principles-based, process framework or meta-model that supports the generation of lifecycle processes based on the characteristics, constraints and risks associated with a given project or program. It incorporates the system success definition described above and so is both technology and stakeholder aware. It handles projects of all sizes, but is particularly applicable to projects requiring multiple life cycle models for various components – new or legacy; software, hardware, or cyber-physical systems; unprecedented or common; critical or mundane. Like the original spiral model, the ICSM is incremental and concurrent in nature, has specific approaches to establishing commitment from stakeholders for those increments before moving forward, and is dependent on evidence and risk-based decisions.

Given that the real world is infinitely complex, the ICSM does not present itself as an all-or-nothing model. Organizations can and do adopt its key practices incrementally, such as its evidence-based decision milestones, its stabilized increment development and concurrent off-line change traffic handling, and its continuous verification and validation activity. They can also develop internal decision tables to fit their most common cases and decision criteria. This is described in more detail later.

Principles Trump Diagrams

Most of the problems in using the 1988 spiral model came from looking at the diagram and constructing processes that had nothing to do with the underlying concepts. This is true of many process models: people just adopt the diagram and neglect the principles that need to be followed. For that reason, before we start to mesmerize you with diagrams and views, it is important to carefully state the ICSM’s four underlying principles. The four principles are:

1. **Stakeholder value-based system definition and evolution.** 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.

2. **Incremental commitment and accountability.** 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.

3. **Concurrent hardware-peopleware-software system definition and development.** If definition and development of a) requirements and solutions; b) hardware, software, and human factors; or c) product and processes 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.

4. **Evidence and risk-based decision-making.** If key decisions are made based on assertions, vendor literature, or meeting an arbitrary schedule without access to evidence of feasibility, the project is building up risks. And in the words of Tom Gilb, “If you do not actively attack the risks, the risks will actively attack you.”
The four principles and their influence on the model are discussed in much more detail in Part I of the Practitioner’s Guide.

**Metaphors for Using the ICSM Components**

To illustrate these principles, it is appropriate to follow Kent Beck’s eXtreme Programming practice of establishing one or more metaphors for the work to be accomplished. Here are two different but useful ways of thinking about why risk-driven, concurrent incremental commitment can bring about successful systems.

**Playing the Odds.**

Our first simple metaphor to help understand the ICSM is to compare the ICSM to incremental-commitment gambling games such as Poker or Blackjack, rather than to single-commitment gambling games such as Roulette. Many system development projects 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. The ICSM is more like Poker or Blackjack, in that one places an increasing series of bets to see whether the prospects of a win are good or not, and decides whether or not to continue betting based on better information about the prospects of success.

**Life Together.**

Our second metaphor is somewhat more personal. The major ICSM life cycle phases and milestones work well as common commitment points across a variety of process model variants because they reflect similar commitment points during one’s lifetime. Exploration is the equivalent of investing time in going out on dates (Yes, the object of desire looks good and seems reasonably intelligent, but so do some others). The Valuation leads to the decision point at which you and a partner decide to go steady, investing some of your degrees of freedom in exploring the prospects of a more serious commitment (We’re interested enough to forgo other dalliances for awhile to see if this could really be “it”). The Foundations phase results in the investment in rings, bringing in other stakeholders, and possibly making domicile arrangements. (Someone says yes to a “Will you marry me?” question, but it is not yet an “until death do us part” commitment). That happens at the Development Commitment Review (the equivalent of officially, legally and in sight of God and everybody getting married). As in life, if you marry your system architecture and plans in haste, you and your stakeholders will repent at leisure (if, in Internet time, any leisure time is available). Finally, the Operations Commitment Review constitutes an even larger commitment (Having your first child, with all the associated commitments of care and feeding of a legacy system).

As with most metaphors, the correspondence of these decision points is not perfect. But the analogies of failure are pertinent: prematurely committing to infeasible requirements and spending lots of resources to create a failed outcome is similar to out-of-sequence life experiences such as dating, creating a child, getting married, and then finding that you can’t live with each other.

**ICSM Diagrams and Views**

Knowing you have been waiting breathlessly to see all the exotic diagrams that are the standard fare in process models, it is now appropriate to introduce the ICSM in graphical splendor.

**The Revised Clarified Spiral View**

The view of the ICSM shown in Figure 2. is helpful in clarifying that the ICSM is not a single one-size-fits-all process model, but actually a process generation model, in which a project’s particular risk patterns and
process drivers steer it toward achieving success with respect to its particular objectives, constraints, and priorities.

Each of the numbered decision nodes in Figure 2. has four possible exits, as seen in the expanded Risk-Based Decisions node at the lower left (the fourth dotted-line exit is not visible in the numbered nodes; it is pointing into the third dimension into the paper or into your screen).

Figure 2. The Incremental Commitment Spiral Model

Rather than the traditional sequential approaches, each spiral will be concurrently rather than sequentially addressing all of the activities of product development. Every spiral considers:

- requirements (objectives and constraints)
- solutions (alternatives)
- products and processes
- hardware
- software
- human factors aspects
- business case analysis of alternative product configurations
- product line investments

All of this concurrency is synchronized and stabilized by the development team producing not only management and engineering artifacts, but also evidence of their combined feasibility. This evidence is assessed at the various stakeholder commitment decision milestones by independent experts. Any shortfalls in evidence are considered as uncertainties or probabilities of loss Prob(Loss), and a level of risk exposure RE calculated as $RE = \text{Prob}(\text{Loss}) \times \text{Size}(\text{Loss})$, where Size(Loss) is the loss of value to success-critical stakeholders. Any such significant risks should then be addressed by a risk mitigation plan.

The stakeholders then consider the risks and risk mitigation plans, and decide on a course of action:

- If the *risks are acceptable* and well covered by risk mitigation plans, the project would proceed into the next spiral.
- If the *risks are high but addressable*, the project would remain in the current spiral until the risks are resolved. An example would be working out safety cases for a safety-critical system, or producing acceptable versions of missing risk mitigation plans.
- If the *risks are negligible*, there would be no need to perform separate Valuation and Foundations spirals, and the project could go straight into a combined Valuation/Foundations spiral followed by a Development spiral. For example, this can happen if the Exploration spiral finds that the solution can
be easily produced by tailoring a COTS package rather than developing a custom solution. The combined Valuation and Foundation spiral would conduct COTS product trades to identify the best COTS alternative for the solutions based on suitable functionality, costs, and evolvable and then proceed into the Development spiral. In the Development spiral, the engineering team will tailor and initialize the COTS product(s) and integrate it into the enterprise.

- If the risks are too high and unaddressable (for example, if it is found at any point that the market window for such a product has already closed), the project should be terminated or rescoped, perhaps to address a different market sector whose market window is clearly still open. Thus, the ICSM provides a set of project off-ramps that make it more acceptable to discontinue the project in order to save the resources for more valuable projects, instead of continuing the project to avoid the stigma of becoming a Standish Group cancelled project.

There are complementary views of the ICSM that provide additional insights that are harder to see from the spiral view. These appear in the next section to illustrate how the model is used throughout the lifecycle.

**Complementary Views of the ICSM**

The standard view of the ICSM shown in Figure 2 is helpful in clarifying that the ICSM is not a single one-size-fits-all process model, but actually a process generation model, in which a project’s particular risk patterns and process drivers steer it toward achieving success with respect to its particular objectives, constraints, and priorities. There are complementary views of the ICSM that provide additional insights that are harder to see from the spiral view. These are:

- **Phased View.** This view details the two primary ICSM stages—Incremental System Definition and Incremental System Development—along with their internal phases and an example of their use and value in a competitive prototyping example.
- **Evolution View.** This view, elaborated in the discussion of Stage II of the Phased View below, shows a different set of concurrent activities that are performed by three teams in Stage II under conditions of emergent requirements, rapid change, and the need for high assurance.
- **Concurrency View.** This view illustrates the activities that are concurrently performed across the system life cycle.
- **Swimlanes View.** This view, used in Chapter 10, shows how the agile rebaselining team in Stage II assesses the incoming change traffic, and proposes a triage of changes to be handled within the current increment, deferred to the next or future increments, or rejected.

**The Phased View**

The Phased View (Figure 3) shows how 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 implement a feasible set of specifications and plans for Incremental Development and Operations of the desired system.

**Stage I**

The duration of the generic 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, eXtreme Programming (XP), Crystal or other agile methods in a week.

An ultra-large, unprecedented, multi-mission, multi-owner system-of-systems project needing to integrate with numerous independently-evolving legacy or external systems may take up to five years. Such an endeavor takes significant effort:

- defining a system vision through sorting out needs, opportunities, and organizational roles
- maturing key technologies
- reconciling infrastructure incompatibilities
- reducing uncertainties and risks via models, simulations, prototypes, advanced technology demonstrations, and operational exercises
- evolving a feasibility-valuated set of specifications and plans for Stage II at the build-to level for the initial increment, but only elaborated for later increments where there were high-risk elements to resolve or expected downstream changes in mission objectives, technology, and interoperating systems to accommodate.

Figure 3. The Incremental Commitment Model: Phased View

As shown in Figure 3 each project's activity trajectory will be determined by the arrows chosen based on 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 3 to skip the Valuation and Foundations 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, apply a form of competitive prototyping to fund four small competitive concept-definition and validation contracts in the Exploratory phase, three larger follow-on Valuation contracts, and two considerably larger Foundations 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 two-to-four week increments. However, an ultra-large SoS project such as a metropolitan-area crisis management system with independently evolving elements or external systems 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 very large, non-subsettable hardware systems would take even longer to develop their initial increments, and would be scheduled to synchronize their deliveries with concurrently-evolving infrastructure or software increments.

**The Evolution View**

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 (SAIV), 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. Figure 4 provides a graphic representation of this evolutionary approach.

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. Towards the end of each increment, the system engineering team also produces for expert review the specifications, plans, and feasibility evidence necessary to ensure low-risk, stabilized development of the next increment by the build-to-spec team.

**The Concurrency View**

Having addressed the stages, phases, and milestones in the generic ICSM, let’s look at the activities. The top row of Activities in Figure 3 indicates 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 5, an extension of a similar “hump chart” drawn by Philippe Kruchten to illustrate the Rational Unified Process. 

![Figure 5. ICSM Concurrency View](image)

As with the Kruchten version, this is a conceptual illustration, not a result of some complex, data-driven analysis. The magnitude and shape of the levels of effort are risk-driven and will almost certainly 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 5. For example, in 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, evaluating alternatives, and negotiating stakeholder commitments.

**How is All This Concurrent Engineering Synchronized and Stabilized?**

The evidence-based anchor point milestone reviews are the mechanisms used in the ICSM to manage the concurrency at the end of each phase. Each of these anchor point milestone reviews, labeled at the top of Figure 5, is focused on evidence - developer-produced and independent-expert reviewed - instead of typical briefings, assertions and assumptions. This provides accurate information to help the key stakeholders determine the next level of commitment.
The review processes and use of independent experts are based on the highly successful AT&T Architecture Review Board procedures described in. ii Figure 6. shows the content of the Feasibility Evidence Description, the focus of each of the concurrency stabilization reviews: Exploration Commitment Review (ECR), Valuation Commitment Review (VCR), Foundations Commitment Review (FCR), and the Development Commitment Review (DCR).

Feasibility Evidence Description Content

- 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
  - Resolve all major risks by treating shortfalls in evidence as risks and covering them by risk management plans
  - Serve as basis for stakeholders’ commitment to proceed

Figure 6. Feasibility Evidence Description Content

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 two-to-ten 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, as can be the increase in going from DCR to the Production Commitment Review (PCR). 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 hardware, software, personnel, facilities, and operational procedures, along with plans, budgets, and schedules for fielding and operations.
Using the ICSM

Common Case Process Patterns and Decision Tables

As illustrated in the four example paths through the ICSM in Figure 7, the ICSM is not a monolithic one-size-fits-all process model. As with the spiral model, it is a risk-driven process model generator. However, the ICSM makes it easier to visualize how different risks create different processes.

Example A is a simple business application based on an already-available Enterprise Resource Planning (ERP) package. There is no need for a Valuation or Architecting activity if the ERP package has already been purchased and its architecture has already proved cost-effective in supporting more complex applications. Thus, the project can go directly into Stage II, using an agile method such as a combination of Scrum and Extreme Programming. There is no need for “Big Design Up Front” activities or artifacts because an appropriate architecture is already present in the ERP package. Nor is there a need for heavyweight waterfall or V-model specifications and document reviews. The fact that the risk at the end of the Exploration phase is negligible implies that the ERP package’s human interface risks have been sufficiently mitigated.

Example B involves a risky but innovative system such as adding a smart camera to the next model of a cellphone product. There are a number of uncertainties and risks/opportunities to resolve, such as camera hardware integration and simplifying the smart camera’s user controls. But the new capability is needed quickly and there is a fallback (deferring its introduction to the following model), so proceeding to address the risks and develop the system is acceptable.

Example C is a system that is defined as safety critical. The stakeholders responsible for the safety of the proposed system find at the Foundations Commitment Review that the proposers have provided inadequate safety evidence. It is better to have the proposers develop such evidence through architecture-based safety cases, fault tree analyses, and failure modes and effects analyses before proceeding into the Foundations phase. This is indicated by the arrow back into the Valuation phase.
In Example D, the developers are simply too late to play. It is discovered before entering the Development phase that a superior product has already entered the marketplace, leaving the current product with an infeasible business case. Here, unless a viable business case can be made by adjusting the project’s scope, it is best to discontinue it. It is worth pointing out that it is not necessary to proceed to the next major milestone before terminating a clearly non-viable project; however, stakeholder concurrence in termination is essential.
As the ICSM is used, different risk patterns lead to the development of corresponding common cases of the ICSM. Figure 8 highlights some of the more common process patterns, in terms of their five key risk characteristics: the project’s size and complexity; its change rate in percentage per month; its degree of mission-criticality; the degree to which the project can be realized via non-developmental item (NDI) support; and the relative organization and personnel capability to develop systems of this nature.

For example, Row 1 indicates that if a small accounting application can be completely supported by an NDI (commercial-off-the-shelf, open-source or purchased-service) package, its life cycle process is to acquire and tailor the NDI package. Row 2 corresponds to the agile ERP-based project described as Example A in Figure 7: a small, fairly rapidly changing, low mission-criticality, strong ERP NDI support, and high organization and personnel capability to develop systems of this nature. Row 3 corresponds to the cellphone project described as Project B in Figure 7: medium sized and complexity; rapid change; medium-high mission criticality, fairly strong NDI support, and high organizational capability, but with a need to establish and architectural solution before implementation.

### Figure 8. The ICSM Risk-Driven Common Cases

#### ICSM Risk-Driven Common Cases

<table>
<thead>
<tr>
<th>Special Case</th>
<th>Example</th>
<th>Size, Complexity</th>
<th>Change Rate (%/Month)</th>
<th>Criticality</th>
<th>NDI Support</th>
<th>Org, Personnel Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Use NDI</td>
<td>Small Accounting</td>
<td></td>
<td></td>
<td></td>
<td>Complete</td>
<td></td>
</tr>
<tr>
<td>2. Agile</td>
<td>Simple business application</td>
<td>Low</td>
<td>1 – 30</td>
<td>Low-High</td>
<td>Good; in place</td>
<td>Agile-ready High</td>
</tr>
<tr>
<td>3. Architected Agile</td>
<td>Competitive cellphone feature</td>
<td>Med</td>
<td>1 – 10</td>
<td>Med-High</td>
<td>Good; most in place</td>
<td>Agile-ready High</td>
</tr>
<tr>
<td>4. FORMAL METHODS</td>
<td>Security kernel; Safety-critical LSI chip</td>
<td>Low</td>
<td>Low</td>
<td>Extra High</td>
<td>None</td>
<td>Strong formal methods experience</td>
</tr>
<tr>
<td>5 SW embedded HW component</td>
<td>Multisensor control device</td>
<td>Low</td>
<td>0.3 – 1</td>
<td>Med-Very High</td>
<td>Good; In place</td>
<td>Experienced; med-high</td>
</tr>
<tr>
<td>6. Indivisible IOC</td>
<td>Complete vehicle platform</td>
<td>Med – High</td>
<td>0.3 – 1</td>
<td>High-Very High</td>
<td>Some in place</td>
<td>Experienced; med-high</td>
</tr>
<tr>
<td>7. NDI-Intensive</td>
<td>Supply Chain Management</td>
<td>Med – High</td>
<td>0.3 – 3</td>
<td>Med- Very High</td>
<td>NDI-driven architecture</td>
<td>NDI-experienced; Med-high</td>
</tr>
<tr>
<td>9. Multi-owner system of systems</td>
<td>Crisis management</td>
<td>Very High</td>
<td>Mixed parts: 1 – 10</td>
<td>Very High</td>
<td>Many NDI; some in place</td>
<td>Related experience, med-high</td>
</tr>
<tr>
<td>11. Brownfield Major Upgrade</td>
<td>Incremental legacy phase-out</td>
<td>High-Very High</td>
<td>Low</td>
<td>Med-High</td>
<td>NDI as legacy replacement</td>
<td>Legacy re-engineering</td>
</tr>
<tr>
<td>12. Maintenance</td>
<td>On-going maintenance/ small hardware and/or software upgrades to legacy system</td>
<td></td>
<td>Changes: Low System: Low – Very High</td>
<td>Low-medium average over time</td>
<td>Changes: Low-Very High</td>
<td>Depends on legacy system</td>
</tr>
</tbody>
</table>
An extended table in Chapter 7 identifies further project characteristics, such as the common-case project’s key Stage I and Stage II activities, and a rough estimate of the length of its internal builds and deliverable increments. In general, the risk patterns will be well enough established during the ICSM Exploration phase to enable the project to determine and switch to the appropriate familiar process without having to go through the full generality of the ICSM framework.

**Examples of Successful ICSM Use**

During the National Research Council Human-Systems Integration study, it was found that the ICSM processes and principles corresponded well with best commercial practices. A good example, documented in Chapter 5 of the study, showed its application to a highly successful commercial medical intravenous infusion pump development. The Hospira, Inc. Symbiq IV Pump won the 2006 Human Factors and Ergonomics Society’s Best New Design Award. It is a general-purpose intravenous infusion pump (IV pump) designed primarily for hospital use with secondary, limited-feature use by patients at home. The device is intended to deliver liquid medications, nutrients, blood, and other solutions at programmed flow rates, volumes, and time intervals via intravenous and other routes to a patient. It offers medication management features, including medication management safety software through a programmable drug library and has sufficient memory to support extensive tracking logs and the ability to communicate and integrate with hospital information systems. It is a good example of concurrent hardware-software-human factors engineering, including a large color touchscreen display and the ability to be powered by either A/C power or rechargeable batteries.

During the Symbiq IV pump Exploration Phase, the project activities included stakeholder needs interviews, field observations, initial user interface prototypes, competitive analysis, and system scoping, resulting in positive results and a commitment to proceed. During the Valuation Phase, the activities included feature analysis and prioritization, display vendor option prototyping and analysis, top-level life cycle plans, business case analysis, safety and business risk assessment, again resulting in a commitment to proceed while addressing risks. During the Foundations Phase, the activities included definition of the modularity of pumping channels, safety feature and alarms prototyping and iteration, programmable therapy types, touchscreen analysis, failure modes and effects analyses (FMEAs), and prototype usage in a teaching hospital.

After the commitment to proceed into the Development Phase, further risk/opportunity activities included extensive usability criteria and testing, iterated FMEAs and safety analyses, patient-simulator testing and adaptation to concerns, followed by a commitment to an Operations and Production Phase.

Beyond the Symbiq IV, Chapter 5 of the Study includes two further case studies. One shows how the ICSM would help address several challenges in adapting current unmanned aerial systems to work with fewer operators. The other describes the risk-driven incremental commitment approach used in the development of large-scale port security systems.

A well-documented successful government-acquisition project using the ICSP principles was the CCPDS-R project. Its US Air Force customer and contractor reinterpreted the traditional defense regulations, specifications, and standards. They held a Preliminary Design Review, but it was not a PowerPoint show at Month 4. Instead, it had a fully validated architecture and demonstration of the working high-risk user interface and networking capabilities at Month 14. The resulting system delivery, including over a million lines of software source code, exceeded customer expectations and was delivered within budget and schedule.

A further source of successful projects that applied the ICSM principles is the annual series of Top-5 software-intensive systems projects published in CrossTalk. The “Top-5 Quality Software Projects” were chosen annually by panels of leading experts as role models of best practices and successful outcomes. 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 incrementally-committed, iterative system evolution. Additional projects gave indications of their partial use. Unfortunately, the project summaries did not include discussion of stakeholder involvement.

Evidence of successful results of stakeholder-satisficing can also be found in the annual series of University of Southern California (USC) e-Services projects using the Win-Win Spiral model. Since 1998, over 100 user-intensive e-Services applications have used precursor and current versions of the ICSM to achieve a 92%
success rate of on-time delivery of stakeholder-satisfactory systems. Its use on the software portion of the ultralarge Future Combat Systems program enabled the sponsors to much better identify and deal with particularly the software-intensive program risks and identify improved courses of action.

**Incremental ICSM Adoption Approaches**

Many system development programs are constrained to support continuity of service with respect to existing legacy systems and business processes. For such programs, it is often not possible to make major process changes all at once. Rather, it is better to incrementally change processes over time. Similarly, existing-system support and upgrade programs may benefit from adopting some of the ICSM principles and practices. Problem programs may also find some ICSM practices helpful in recovering viability.

The five practices discussed below can be adopted individually without requiring parts of the others. They are also compatible, so that organizations can choose their own approach to incrementally adopting parts or all of the ICSM. Finally, they are synergetic, in that the benefits of the whole are greater than the sum of the parts. For examples, performing continuous integration and V&V helps develop evidence for decision milestones; committing to develop the next increment based on strong evidence of feasibility ensures a sound set of plans and specifications for stabilizing its development; prioritizing features helps the continuous integration and V&V team focus its efforts on high-value activities.

The practices are:

- Adding an evidence-based Feasibility Rationale to the content of current reviews such as system and software requirements and design reviews
- Using a Schedule as Independent Variable or timeboxing process and prioritizing features to be delivered
- Stabilizing development increments by diverting most change traffic to a concurrent systems engineering effort to incorporate the changes into future increment baselines
- Performing continuous vs. back-end integration, verification, and validation; and
- Using risk to determine appropriate common cases of the ICSM to apply to familiar situations.

**Adding Evidence-Based Decision Milestones**

Projects and organizations are often constrained by sequential, document-driven waterfall-model decision milestones, such as System Functional Reviews and Preliminary Design Reviews. Sometimes, the review guidance asks for evidence of feasibility of achieving these functions, but the evidence is not a first-class deliverable, and is asked for in an optional appendix. Frequently, the constraints are reinforced by being tied to contractual progress payments and award fees, and the early-phase budgets and schedules are based on optimistic assumptions.

In such cases, even experienced project managers find that there is only time and effort available to document the functions or the design, and the evidence of feasibility is minimal. This may not be a problem for small, non-critical projects, but an analysis of 161 software-intensive system data done for the over-10-million source lines of code (SLOC) Future Combat Systems project found that the rework penalty for proceeding without evidence of feasibility went from 18% for a 10,000 SLOC project to 91% for a 10-million SLOC project (details are provided in Chapter 3).

Using such data, it is often possible to convince the stakeholders to reinterpret the waterfall-model decision milestones to include having feasibility evidence ready, and to provide sufficient resources to produce the evidence as well as the documented functions and design. An example is found in the CCPDS-R example discussed in Section 0, A combination of expert judgment and a parametric model such as the Constructive Systems Engineering Cost Model (COSYSMO)ix will generally produce a viable estimate for the added systems engineering effort.
Using Schedule as Independent Variable and Feature Prioritization

Even with evidence-based commitment processes, there will be “devils in the details” that will cause delays. Simultaneously fixing the budget, schedule, and promised feature set in such situations is unrealistic. A good strategy for dealing with this is to define a core capability that is essential (one can’t deliver an aircraft without the landing gear or engines), and prioritize features that are easy to add or defer to the next increment in order to meet the current increment’s schedule or budget (such as on-board payload electronics or software features). The borderline features thus act as a buffer or risk reserve; their being tied to actual capabilities makes it easier to defend against budget cuts than an undefined risk reserve. The priorities are also useful in prioritizing testing and peer-review efforts to focus on the more valuable parts.

Stabilizing Development Increments

Using concurrent teams to enable stabilized increment development (subject to adding or dropping borderline features) has been summarized under Stage II in Section 0.2, and is elaborated in considerable detail in Chapters 3 and 10. Having a separate team handle the unplanned-for change traffic avoids destabilizing the development team, which otherwise would overrun or deliver partial features and undermine planning and budgeting for the next increment. Given the increasing acceleration of rapid change discussed in Section 0.1, this practice is increasingly important for 21st century projects.

Performing Continuous Integration, Verification, and Validation

Besides the concurrent agile change-handling and next-increment rebaselining team, a third concurrent team to help the stabilized development team achieve high levels of assurance is often warranted, especially for the increasingly software-reliant systems of the 21st century. Numerous studies have documented that lengthening the time between defect insertion and defect removal leads to an exponential increase in rework effort. Further, the savings multiplier in total cost of ownership for eliminating defects in safety-, security-, or financially-critical applications is much higher than the rework savings. Investing in teams of people well qualified to identify defects and work with developers to resolve them will thus have a high payoff. For large projects, additional investments in such capabilities as an early system integration facility, workload models and simulations, and an executing architecture skeleton will enable the equivalent of daily or weekly system builds that expedite both early system integration and early defect reduction.

Using risk to determine appropriate common cases of the ICSM [Revision also TBD by Jo Ann]

Another initiative that organizations can apply incrementally vs. going to full adoption of the ICSM right away is to use the ICSM common-case decision table exemplified in Figure 8, and elaborated in Chapter 7. This helps the organization understand the factors that may affect its choices of process models for different types of projects, and provides shortcuts in getting to the right process for a given situation.

References for Introduction


\[iii\] Pew and Mavor, ibid