“They Keep Moving the Cheese”

A Framework for Evolutionary Acquisition of Large Software Intensive Systems

Cecilia Albert
Lisa Brownword

Sponsored by the U.S. Department of Defense
© 2003 by Carnegie Mellon University
Who Moved My Cheese?
A Story…

Program goal: provide a tool for strategic, operational, and tactical planners from all services and defense agencies to support joint and coalition engagements and peace keeping efforts

- Run on existing enterprise backbone (managed by another agency)
- Interface with multiple existing and developing systems
- Operate across multiple security levels

<table>
<thead>
<tr>
<th>Program Start (late ’90s)</th>
<th>2003</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Automate manual process</td>
<td>• New planning processes</td>
</tr>
<tr>
<td>• Client-server architecture</td>
<td>• Web-based architecture</td>
</tr>
<tr>
<td>• Support 2-3 day planning cycle</td>
<td>• Dynamic planning cycles</td>
</tr>
<tr>
<td>6 increments delivered across 6-7 years</td>
<td>• Collaborative planning</td>
</tr>
<tr>
<td>• First release in 18-24 months</td>
<td>• Increment 1 is obsolete</td>
</tr>
<tr>
<td></td>
<td>• Struggling to build/field increment 2</td>
</tr>
<tr>
<td></td>
<td>• Users have built “interim” solutions</td>
</tr>
<tr>
<td></td>
<td>• Future is uncertain</td>
</tr>
</tbody>
</table>
## Size Matters!

<table>
<thead>
<tr>
<th>Project Size</th>
<th>People</th>
<th>Time (mos)</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; $750K</td>
<td>6</td>
<td>6</td>
<td>55%</td>
</tr>
<tr>
<td>$750K-$1.5M</td>
<td>12</td>
<td>9</td>
<td>33%</td>
</tr>
<tr>
<td>$1.5M-$3M</td>
<td>25</td>
<td>12</td>
<td>25%</td>
</tr>
<tr>
<td>$3M-$6M</td>
<td>40</td>
<td>18</td>
<td>15%</td>
</tr>
<tr>
<td>$6M-$10M</td>
<td>+250</td>
<td>+24</td>
<td>8%</td>
</tr>
<tr>
<td>&gt;$10M</td>
<td>+500</td>
<td>+36</td>
<td>0%</td>
</tr>
</tbody>
</table>

Source: The Standish Group, 1999
Definitions

A *software-intensive system* is one that
- Relies on software to provide core/priority mission function(s)

A *large software-intensive system* is one whose software
- Takes longer than 6 months to implement
- Takes more than 6 people to implement
- Takes more that $750,000 to implement
  and/or
- Is comprised of multiple interrelated systems or independently developed components implemented in software (system of systems, family of systems, etc)
Outline

Change Happens

Adapting to Change

Be Ready to Change Quickly
Change Happens

Large software-intensive systems change at a rate faster than the full system capability can be implemented – and they change during development and operation.

Sources of change

- Enterprise priorities shift
- Business or operational needs change
- New technologies introduce new opportunities
- COTS products add and delete key features
- Participants rotate
- …
Adapt to Change

DoD 5000* provides mechanisms for coping with change

**Evolutionary Acquisition**
Delivers capability in increments, recognizing, up front, the need for future capability improvements
- Success of the strategy depends on the consistent and continuous definition of requirements and maturation of technologies that lead to disciplined development and production of systems that provide increasing capability towards a material concept.

**Spiral Development**
A desired capability is identified but the end-state requirements are not known at program initiation
- Those requirements are refined through demonstration and risk management; there is continuous user feedback; and each increment provides the user the best possible capability. The requirements for future increments depend on feedback from users and technology maturation.

* The Operation of the Defense Acquisition System, 30 Oct 02
Lessons Learned

- Going after “low hanging fruit” in the absence of an overarching architecture and coherent plan results in incompatible and stove-piped solutions.

- System requirements defined without sufficient insight into what can be realistically built, results in systems that cannot be built.

- There are no “silver bullets” that avoid disciplined system and software engineering (doing the right engineering correctly).
Be Ready To Change Quickly

Consciously apply spiral development practices at 2 (or more) discrete levels – with continuous interaction between the levels

• Program or system level
  - Evolve definition and implementation plan for system end-state
  - Define and spawn increments of useful capability that will build to full system functionality and performance

• Project or increment level
  - Define and implement plan for delivering the defined increment in the context of the system end-state
Disciplined Spiral Development

- Continuously determine a compatible and feasible set of: business processes, requirements, plans, architecture, COTS products and other components
- Enterprise business objectives drive solution definition
- Risk considerations drive degree of detail
- Executable representations demonstrate current understanding and agreements

Spiral development facilitates evolving a viable solution – at both system and increment levels
Phases Bounded by Anchor Points

Simultaneous Definition and Tradeoffs

Scope

Design

Build

Field

Multiple iterations per phase

© 2003 by Carnegie Mellon University

26 January 2003
Disciplines* Extend Across Phases

*adapted from Kruchten; shows partial set of disciplines
Keep a Long View in Systems Planning
Evolving System Definition

New technology
Modified environment
Changed mission

Evolving System Definition

Current state
Increment 1
Increment 2
Future state

A → B → C → D → D1

reassess and replan

A → B → B1 → D1

actual

Z1

vision

© 2003 by Carnegie Mellon University
26 January 2003
Take a Short View on Increment Planning

Allows a stable development environment – if a short timeframe (6-18 months)

Allows focused discovery, experimenting, and learning on a manageable scale to find optimum way to understand and meet user needs
Increment Activity Mapping

**Scope**
- Define feasible scope
  - Survey/try components
  - Agree to business changes
- Establish project plan
- Develop business case
- Outline candidate architectures
- Study COTS market; screen candidates
- Prepare demos of candidate solutions
- Identify key risks
- Determine business changes

**Design**
- Refine, experiment, & select solution
  - Try/select components
  - Prototype business changes
- Update project plan
- Define, baseline and demonstrate solution
- Evaluate COTS products and components
- Stabilize requirements and architecture
- Develop plan to manage business process change

**Build**
- Implement selected solution
  - Apply/track components
  - Prepare to change business processes
- Update project plan
- Build production quality solution for beta test
- Continue market/COTS surveys and evaluation
- Prepare end users for initial fielding

**Field**
- Rollout and support solution
  - Use/track components
  - Change business processes
- Complete rollout
- Fix bugs, adjust features, make minor enhancements
- Achieve user satisfaction / self-supportability
- Continue market/COTS surveys and evaluation
- Support solution until retirement

6 to 18 months
Leverage Feedback between Long- and Short-Term

Maintain long-term strategy (system level) aligned with enterprise improvement

Make short-term implementation decisions (increment level) aligned with long-term strategy

Use knowledge gained in short-term increments to evolve long-term strategy

Anticipate continuous change
Plan and Manage *Efficient* Feedback

Decisions take place simultaneously at both levels – one informs the other.
Managing Continuous Evolution

System level

Scope
- Business model
- Scope
- Constraints
- Market study

Design
- Critical use cases at system level (what)
- Architecture (how)
- Available and projected technology

Increment #1

Scope
Design
Build
Field

LCO
LCA
LCO
LCA

6-18 months

© 2003 by Carnegie Mellon University
Scenarios of Multiple Increments

Several increments for different areas of system capability running concurrently

Several increments for same area of system capability where successive generations provide greater capability
The Handwriting on the Wall

Change Happens

Adapt To Change Quickly
- Anticipate Change
- Monitor Change
- Change
- Enjoy Change!

Be Ready To Change Quickly And Enjoy It Again

Reprinted through the courtesy of CIO © 2002 CXO Media Inc.
Contact Information

Lisa Brownsword
llb@sei.cmu.edu

Cecilia Albert
cca@sei.cmu.edu
Lisa Brownsworth is a senior member of the technical staff in the Commercial-off-the-shelf- (COTS)-Based Systems (CBS) Initiative at the Software Engineering Institute (SEI). Before joining the SEI, Lisa was on staff at Computer Sciences Corporation in support of NASA/Goddard’s Software Engineering Lab. Prior to that, she was employed at Rational Software Corporation providing consulting to managers and technical practitioners in the use of and transition to software engineering practices, including architecture-centered development, product lines, object technology, Ada, and CASE.

Cecilia Albert is a senior member of the technical staff in the Commercial-off-the-shelf- (COTS)-Based Systems (CBS) Initiative at the Software Engineering Institute (SEI). Before joining the SEI, Cecilia was in the Air Force where she served in a variety of information technologies related positions including: developing major software programs for simulation, command and control, and mission processing of national satellite systems; teaching acquisition and leading an industry study on telecommunications and information systems at the Industrial College of the Armed Forces; and managing the archive and dissemination programs at the National Imagery and Mapping Agency.