Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

"They Keep Moving the Cheese"

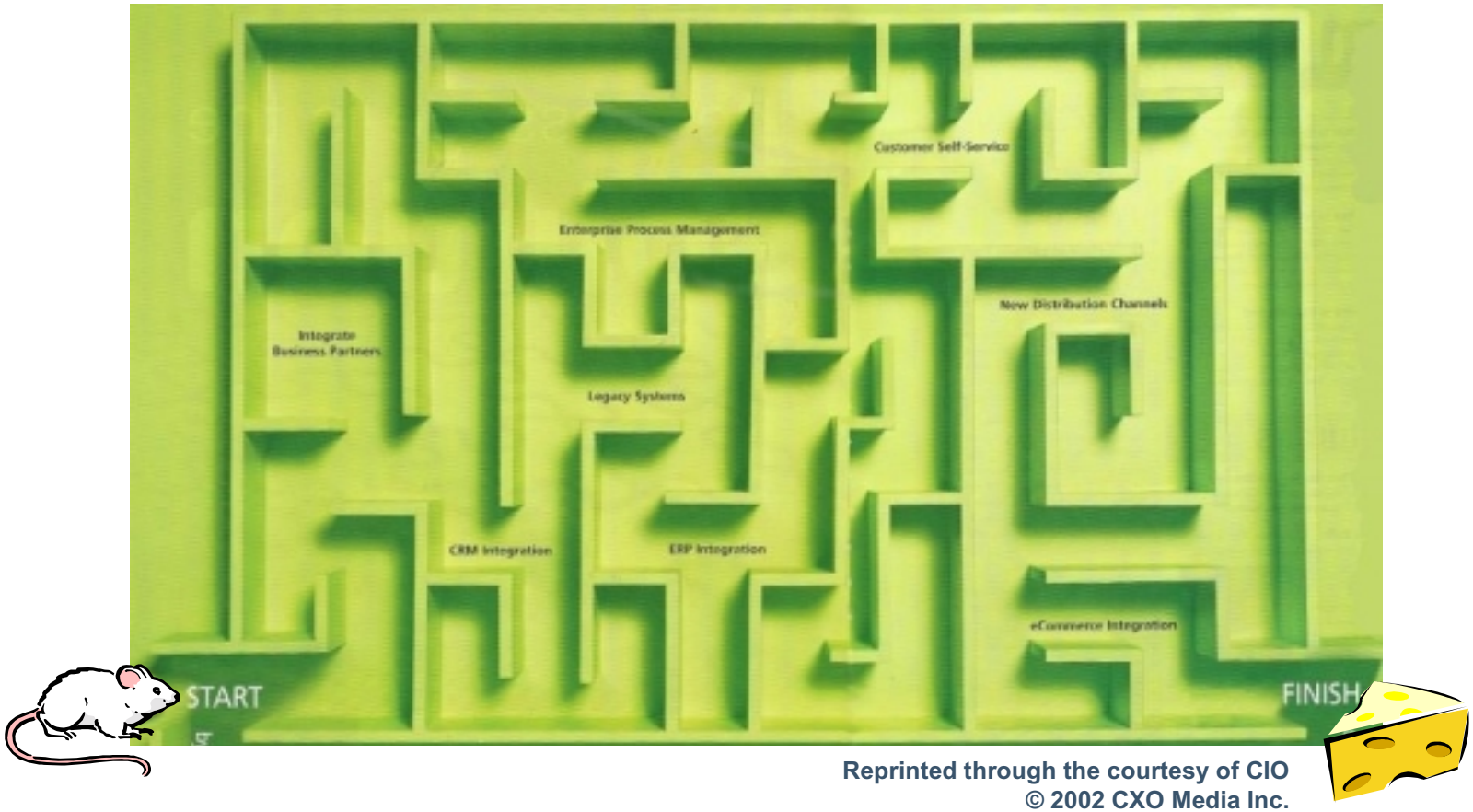
A Framework for Evolutionary Acquisition of Large Software Intensive Systems

Cecilia Albert
Lisa Brownsword

Sponsored by the U.S. Department of Defense

Carnegie Mellon
Software Engineering Institute

Who Moved My Cheese?



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

Carnegie Mellon
Software Engineering Institute

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

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6 increments delivered across 6-7 years • First release in 18-24 months	• Increment 1 is obsolete • Struggling to build/field increment 2 • Users have built “interim” solutions • Future is uncertain

Carnegie Mellon
Software Engineering Institute

Size Matters!

Project Size	People	Time (mos)	Success Rate
< $750K	6	6	55%
$750K-$1.5M	12	9	33%
$1.5M-$3M	25	12	25%
$3M-$6M	40	18	15%
$6M-$10M	+250	+24	8%
>$10M	+500	+36	0%

Source:The Standish Group, 1999

Carnegie Mellon
Software Engineering Institute

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)

Carnegie Mellon
Software Engineering Institute

Outline

Change Happens

Adapting to Change

Be Ready to Change Quickly

Carnegie Mellon
Software Engineering Institute

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
- …

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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)

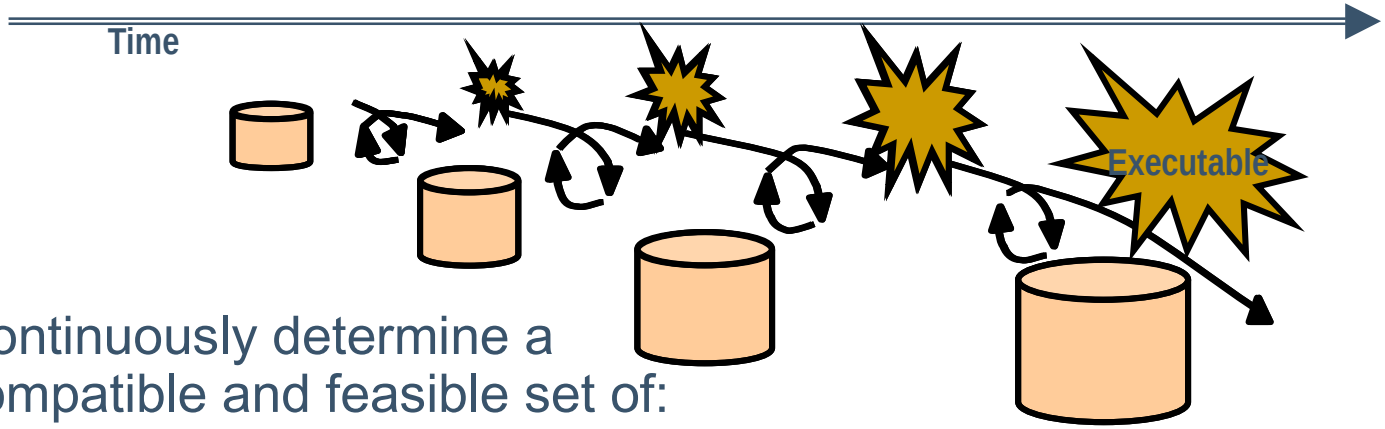
Carnegie Mellon
Software Engineering Institute

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

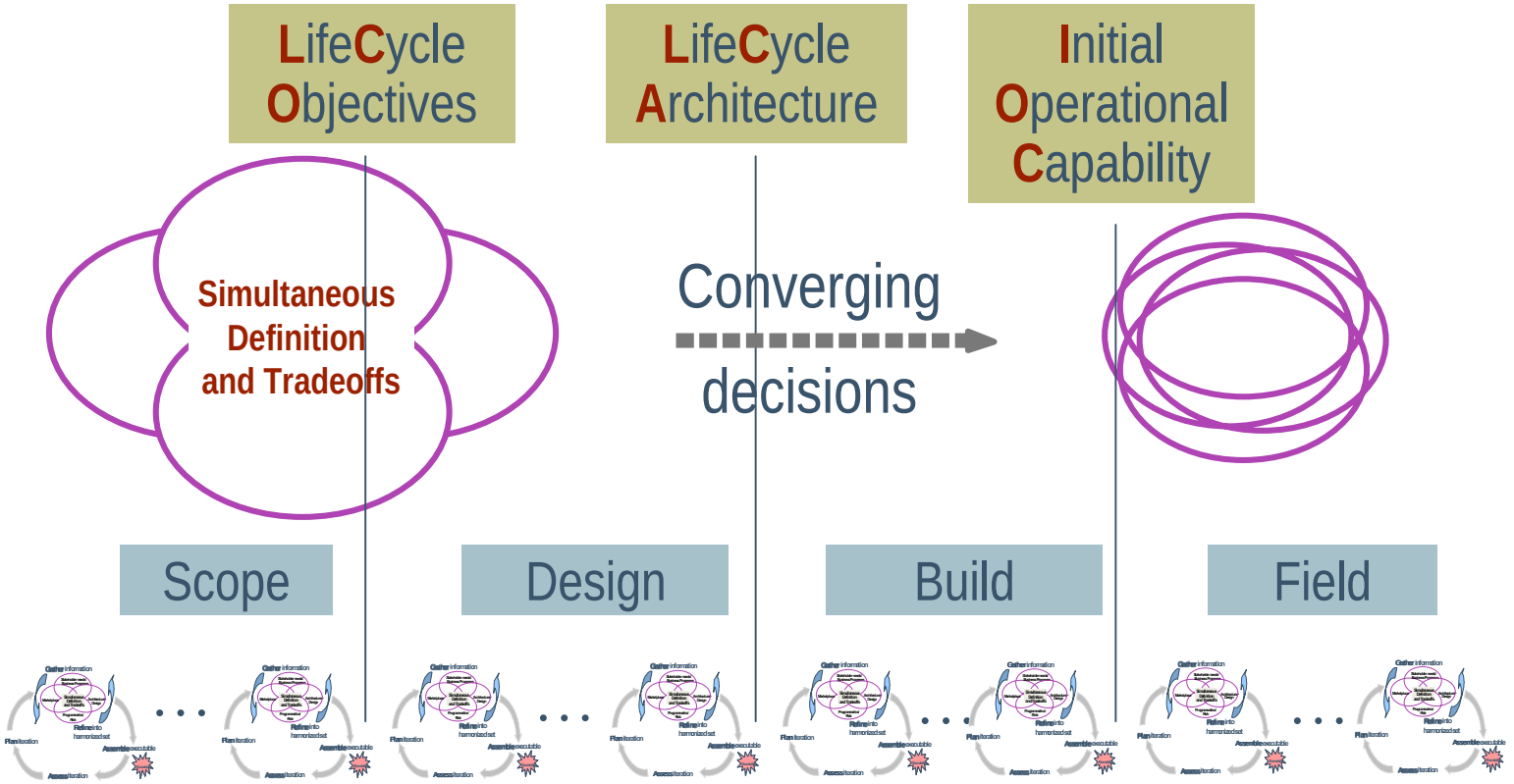
Disciplined Spiral Development



- 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

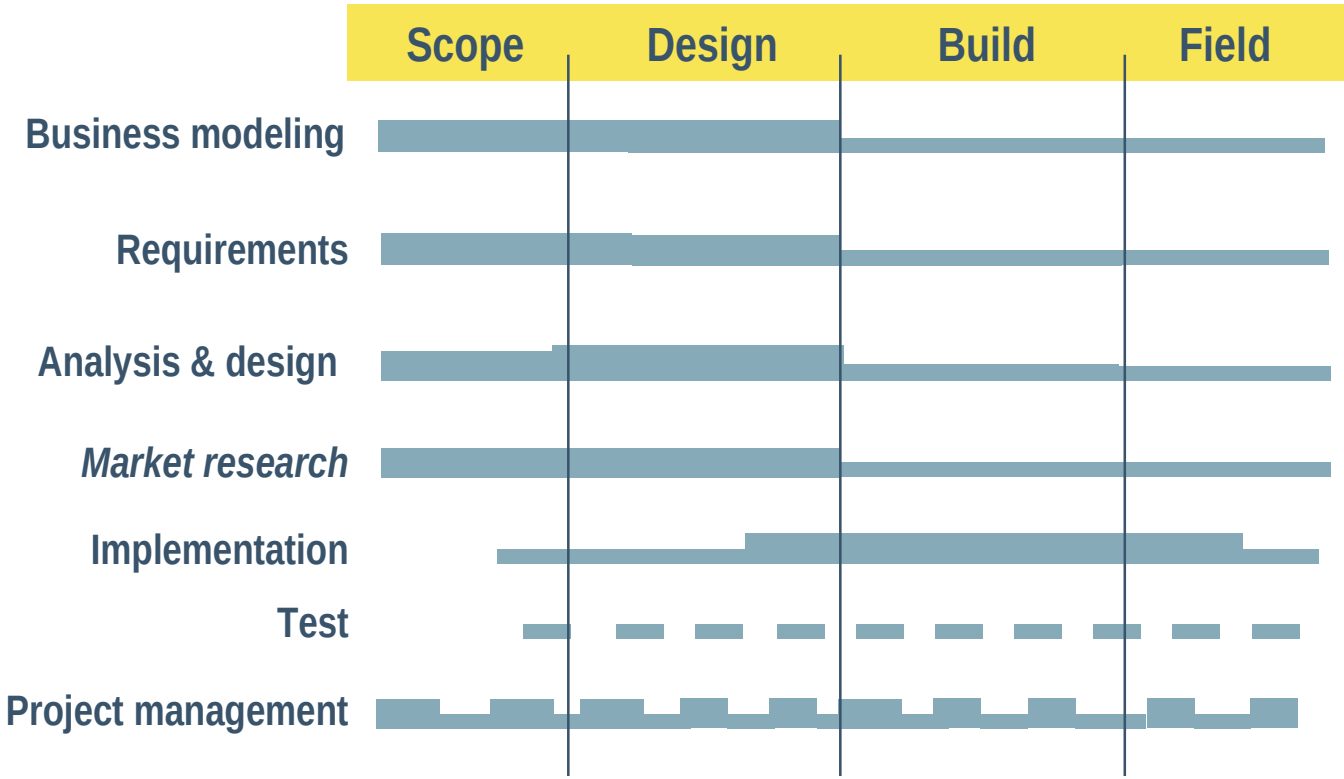
Phases Bounded by Anchor Points



Multiple iterations per phase

Carnegie Mellon
Software Engineering Institute

Disciplines* Extend Across Phases



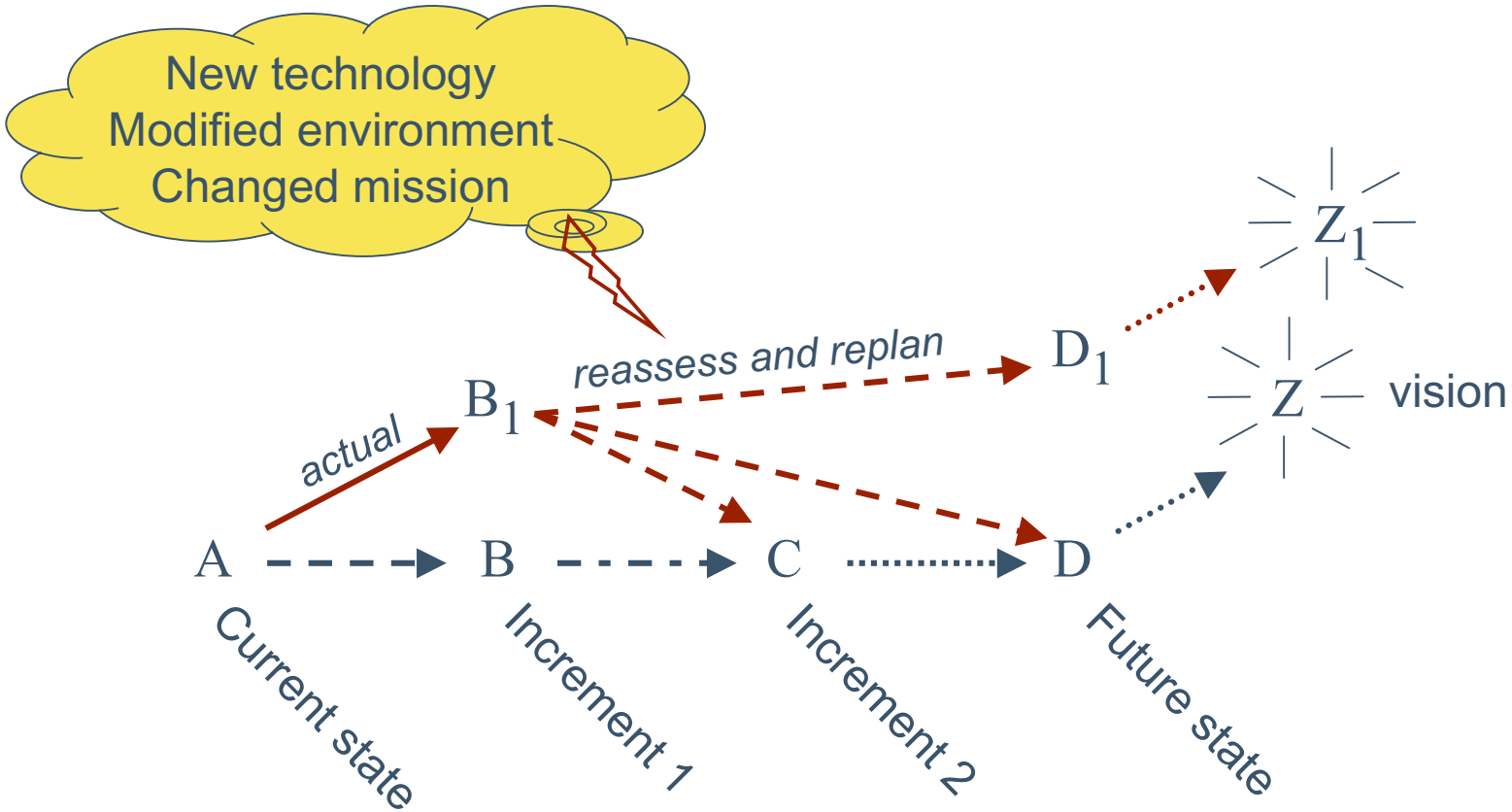
*adapted from Kruchten; shows partial set of disciplines

Carnegie Mellon
Software Engineering Institute

Keep a Long View in Systems Planning

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

Evolving System Definition



Take a Short View on Increment Planning

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

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

Carnegie Mellon
Software Engineering Institute

Increment Activity Mapping

Scope	Design	Build	Field
Define feasible scope	**Refine, experiment, & select solution**	**Implement selected solution**	**Rollout and support solution**
Survey/try components	***Try/select components***	***Apply/track components***	***Use/track components***
Agree to business changes	***Prototype business changes***	***Prepare to change business processes***	***Change business processes***
Establish project plan	Update project plan	Update project plan	Complete rollout
Develop business case	Define, baseline and demonstrate solution	Build production quality solution for beta test	Fix bugs, adjust features, make minor enhancements
Outline candidate architectures	Evaluate COTS products and components	Continue market/COTS surveys and evaluation	Achieve user satisfaction / self-supportability
Study COTS market; screen candidates	Stabilize requirements and architecture	Prepare end users for initial fielding	Continue market/COTS surveys and evaluation
Prepare demos of candidate solutions	Develop plan to manage business process change		Support solution until retirement
Identify key risks			
Determine business changes			

6 to 18 months

Leverage Feedback between Long- and Short-Term

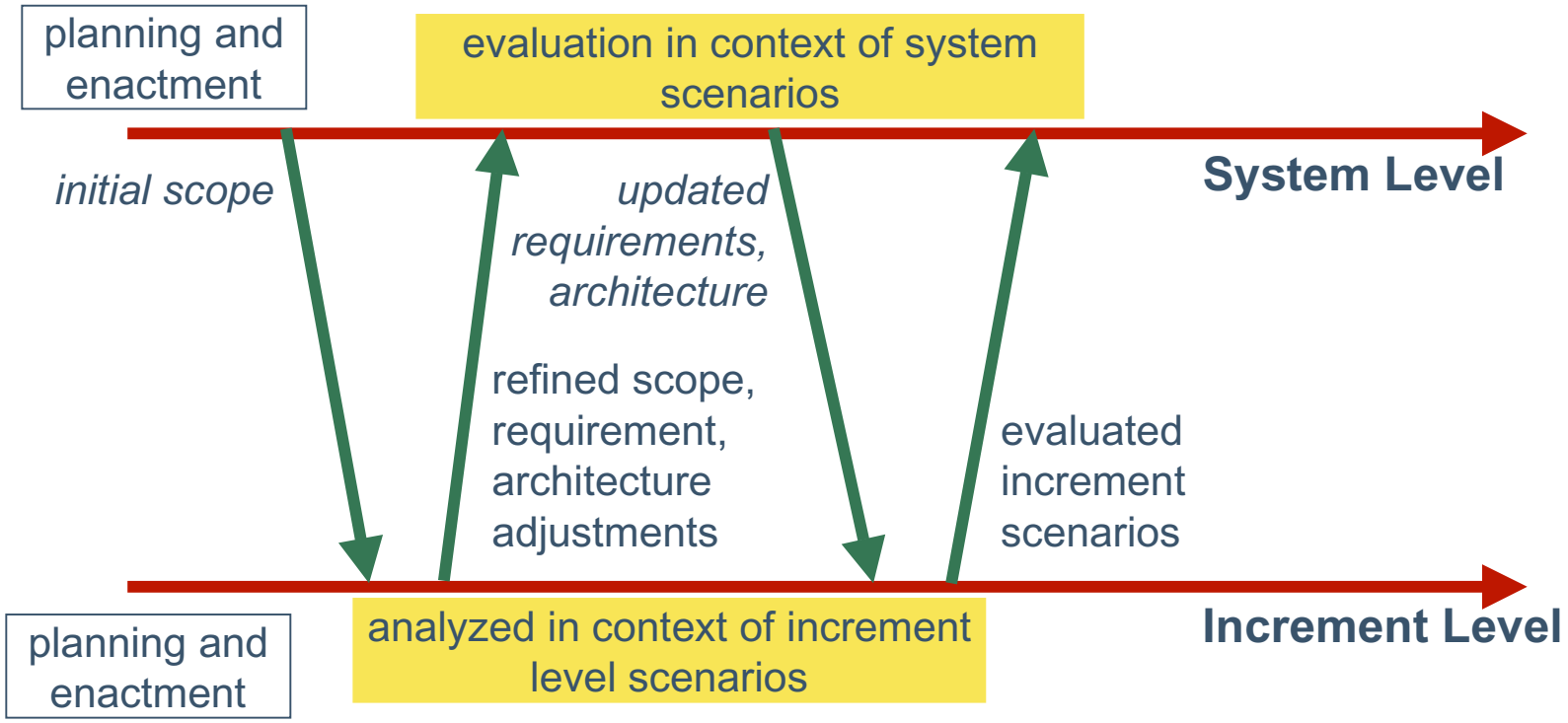
Maintain long-term strategy (system level) aligned with enterprise improvement

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

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

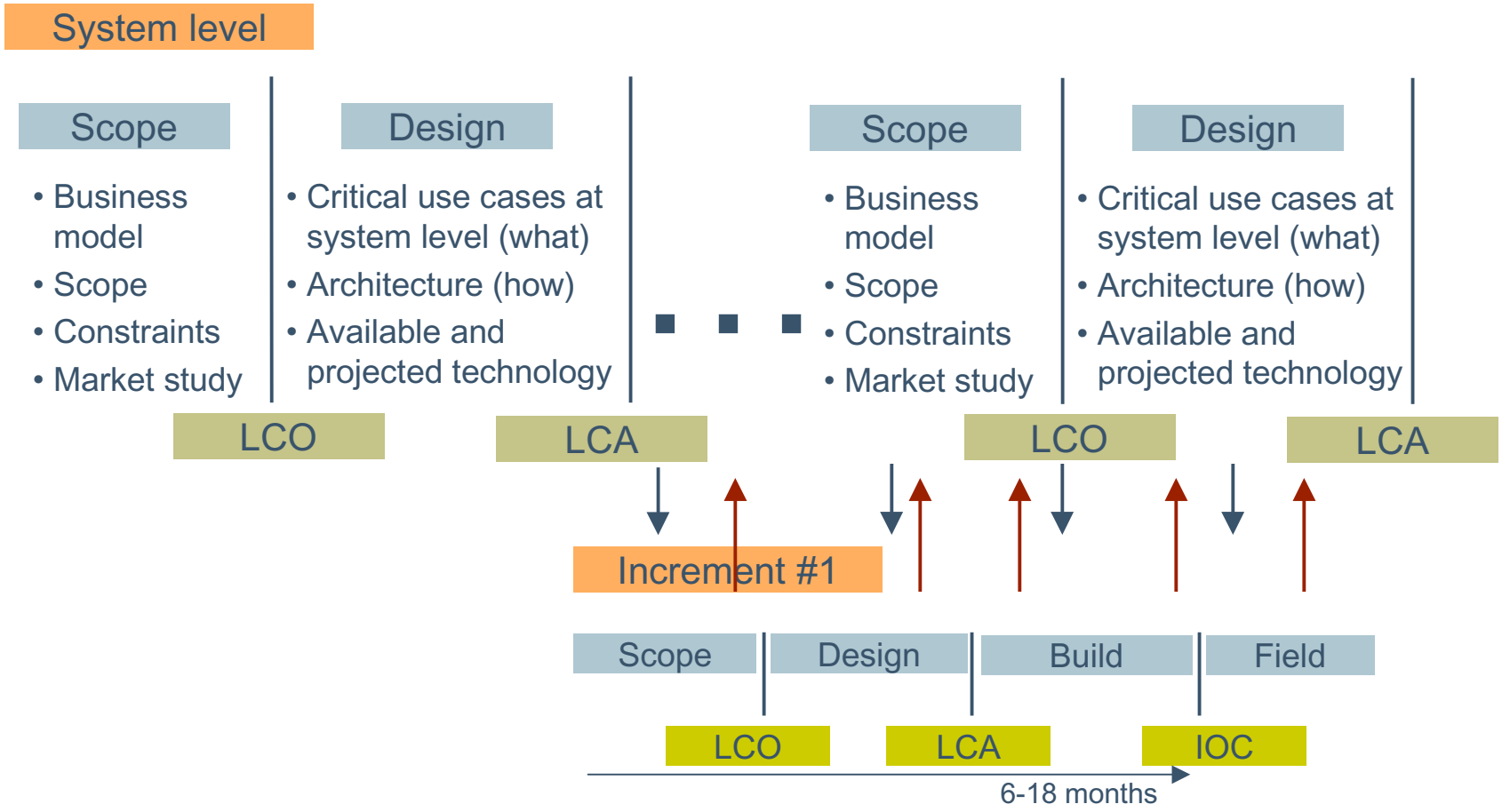
Anticipate continuous change

Plan and Manage *Efficient* Feedback

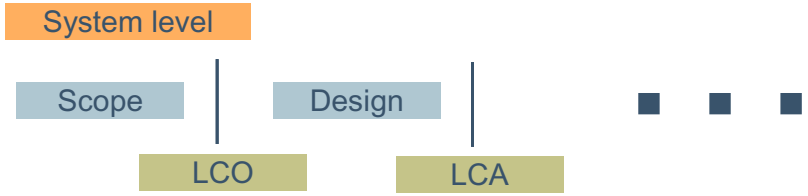


Decisions take place simultaneously at both levels – one informs the other

Managing Continuous Evolution

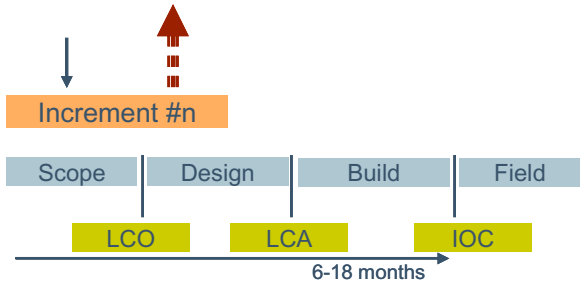


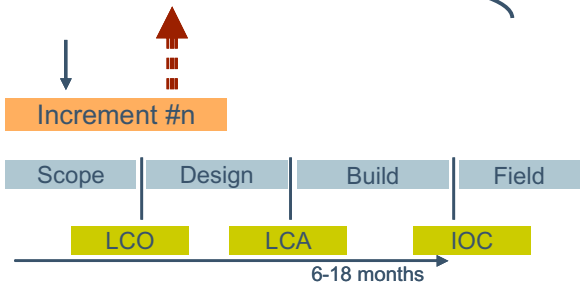
Scenarios of Multiple Increments

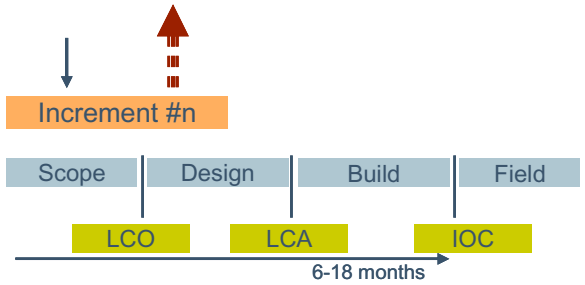


Several increments for same area of system capability where successive generations provide greater capability

Several increments for different areas of system capability running concurrently







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

Carnegie Mellon
Software Engineering Institute

Contact Information

Lisa Brownsword
llb@sei.cmu.edu

Cecilia Albert
cca@sei.cmu.edu

Carnegie Mellon
Software Engineering Institute

Lisa Brownsword 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.