When Does Requirements Volatility Stop All Forward Progress?

Practical Software and Systems Measurement
User's Group Conference
Golden, Colorado
July 2007

Jo Ann Lane and Barry Boehm
University of Southern California
Center for Systems and Software Engineering
http://csse.usc.edu

Overview

- ***Requirements: what are they and what are their characteristics?***
- **Requirements volatility: all changes are not "equal"**
- **Quantitative observations about requirements volatility**
- **Conclusions**

Applies to systems, complex systems, and systems of systems (SoSs)

What is a Requirement

- **IEEE Std 1220-1998: Standard for Application and Management of the Systems Engineering Process**

  A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines).

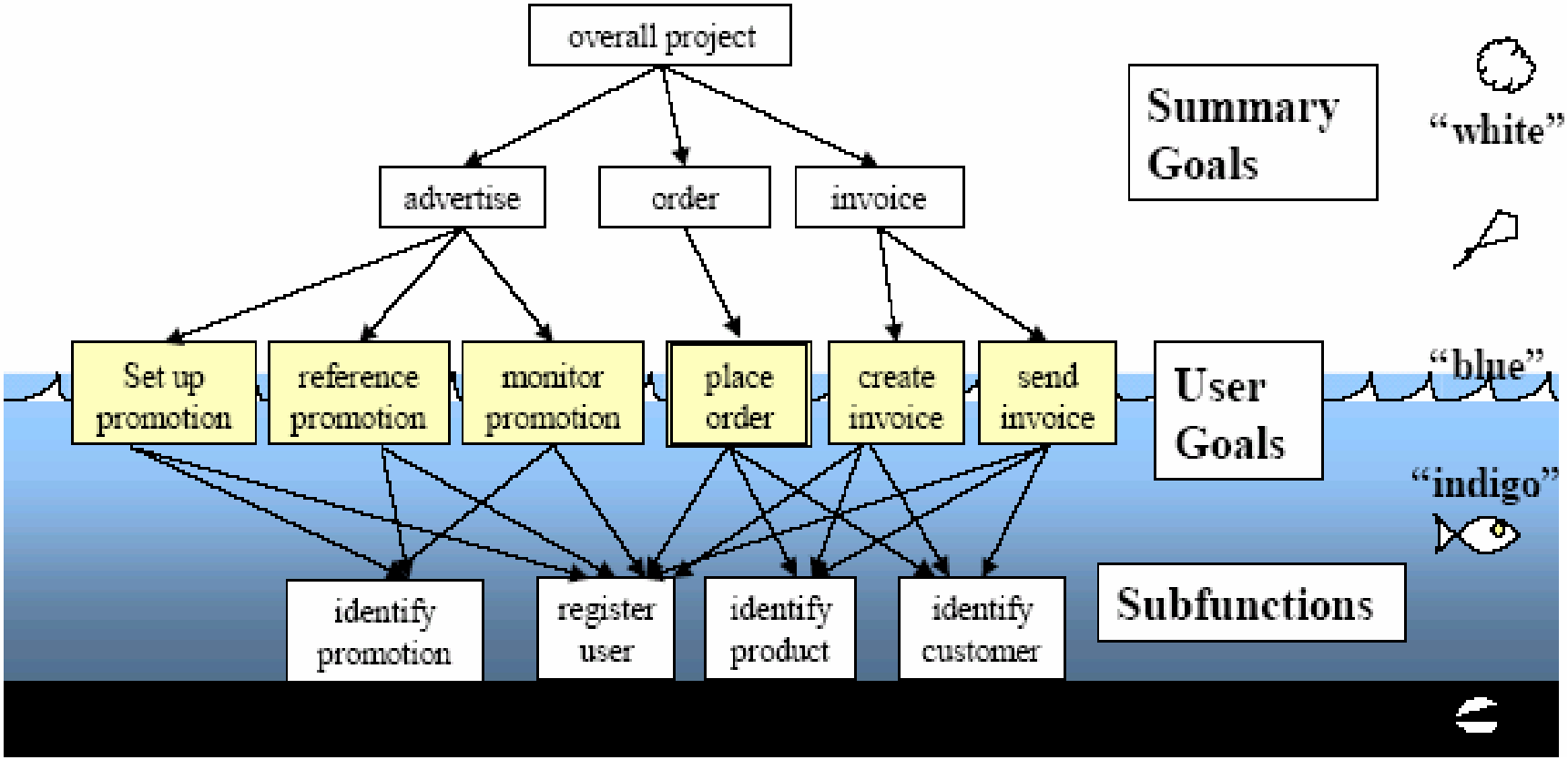
- **SEI [CMMI 2001]:**

  (1) A condition or capability needed by a user to solve a problem or achieve an objective.

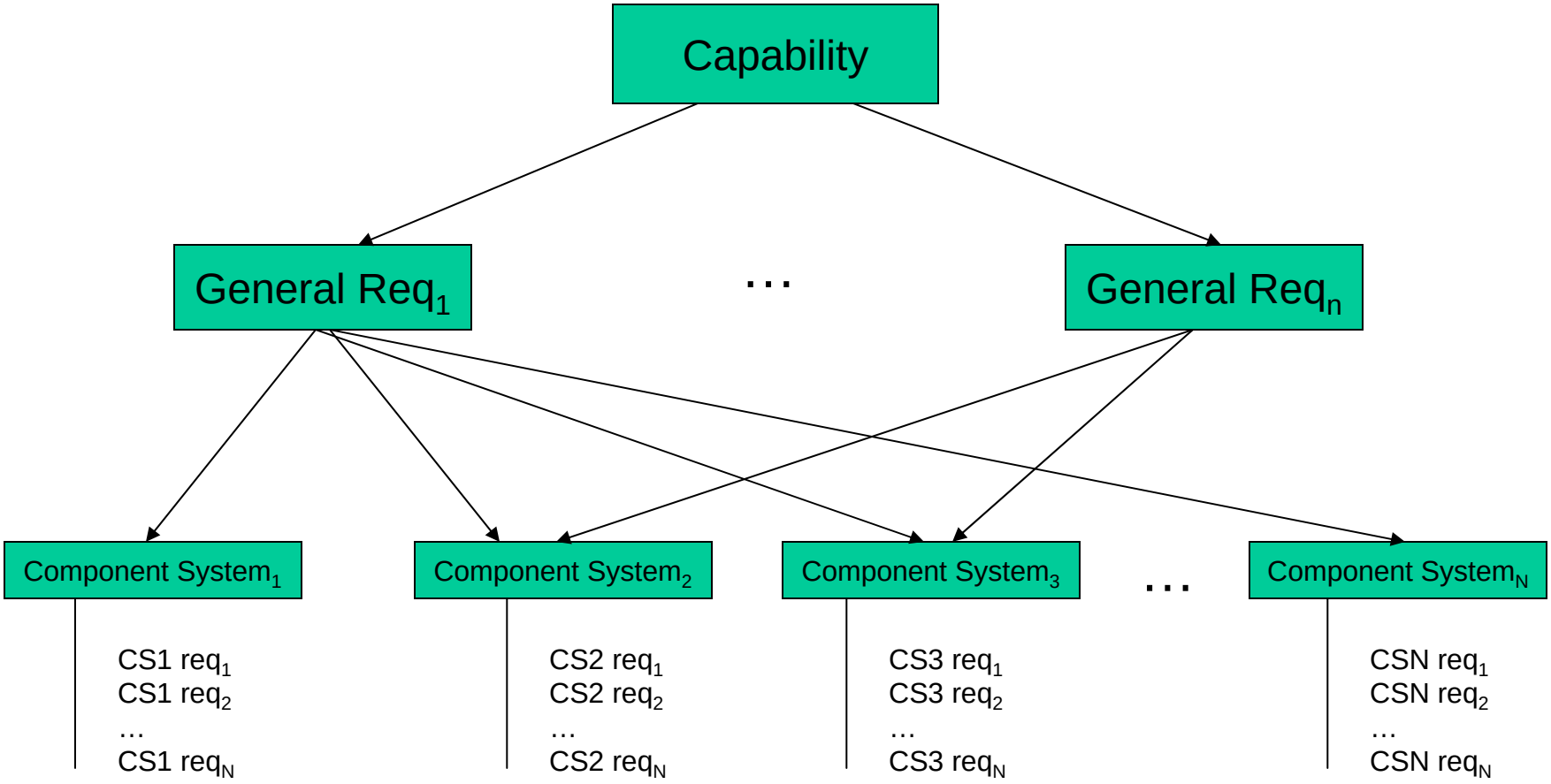
  (2) A condition or capability that must be met or possessed by a product or product component to satisfy a contract, standard, specification, or other formally imposed documents.

  (3) A documented representation of a condition or capability as in (1) or (2). [IEEE 610.12-1990]

Cockburn Hierarchy as it Relates to Requirements



Hierarchy of Requirements



Types of Requirements

- **Functional**
- **Interfaces**
- **Level of service (e.g., performance targets, interoperability*, security*, safety)**
- **Design constraints**
- **Quality attributes**
- **Acquisition (e.g., cost and schedule)**
- **Process**

* Cited as the most important areas for SoSs [Kriegel, 1999].

Some Key Purposes for Requirements

- Specify needed system capabilities
- Coordinate work performed by multiple organizations/vendors (or to prevent incompatible design decisions within the system architecture)
- Ensure interoperability and compatibility between system components
- Control cost/schedule
- Establish acceptance criteria for development work performed

Why Do Requirements Change*?

- Changing business/user needs
  - Environment changes – Market trends
  - Legislative changes – New technology
- Incorporation of COTS upgrades
- Resolve requirements conflicts
- Specify missing requirements
- Manage cost/schedule
- Adjustment of requirements in response to design decisions
- Derivation of lower level requirements as solution evolves

** "Requirements change" as investigated here is the evolution of requirements over time, not the resolution of defective requirements*

Overview

- Requirements: what are they and what are their characteristics?
- *Requirements volatility: all changes are not "equal"*
- Quantitative observations about requirements volatility
- Conclusions

Requirements Volatility Definitions

- Requirements change
  - Change to a baselined set of requirements
  - For projects where requirements are not baselined (e.g., agile projects), change to an operational capability
- Volatility
  - Rate of requirements change over time or per increment of development
- Impact of volatility
  - Effort and schedule changes other than those associated with actual effort/schedule required to implement the requirement
  - Includes
    - Rework
      - Work already completed for current increment
      - Increased defect densities associated with incomplete change analysis/attempted schedule compression
    - Delays due to related approval and contract modification activities
    - Productivity impacts due to project staff frustration

Influences on Effort to Change a Capability/Requirement

- Scope of change
- Level of change
- Number of components affected by requirement change
- Targeted increment for requirement implementation (current vs. future)
- Impact of change for each affected component
  - Number of component levels affected
  - Number of lower level suppliers affected
- How tightly coupled requirements are to supplier contracts at various levels

Influences on Schedule Required to Change a Capability/Requirement

- Time to assess impact of proposed requirement change
- Time to approve proposed requirement change (e.g., number of approvers)
- Time to flow down requirement change (e.g., number of required contract changes)
- Time to implement requirement change (e.g., scope of requirement change/required rework)

Influences on System Requirements Volatility

- Number of system missions/objectives
- Stability of system missions/objectives (e.g., business needs)
- System architecture stability/maturity
- Stability/maturity of system components
- Technology maturity/changes

Overview

- Requirements: what are they and what are their characteristics?
- Requirements volatility: all changes are not “equal”
- *Quantitative observations about requirements volatility*
- Conclusions

Scenarios for Analysis of Impacts

1. **Early: Proposed requirement change received during requirements identification/analysis phase**
   a. Limited scope
   b. Pervasive scope/no outside suppliers affected
   c. Pervasive scope/outside suppliers affected
2. **Middle: Proposed requirement change received during implementation phase**
   a. Limited scope
   b. Pervasive scope/no outside suppliers affected
   c. Pervasive scope/outside suppliers affected
3. **Late: Proposed requirement change received during integration and test phase**
   a. Limited scope
   b. Pervasive scope/no outside suppliers affected
   c. Pervasive scope/outside suppliers affected

Findings of System Dynamics Models Used to Evaluate Requirements Volatility

- **Ferreira Model***
  - Evaluates the effects of requirements volatility on a software project's cost, schedule, and quality
  - Based on survey data from 232 projects
    - Over 78% of respondents experienced some level of requirements volatility
    - Average increase in software size due to volatility: 32%
  - Once the design process begins, the impact of requirements change is progressively greater
  - Captures low morale impacts (reduced productivity, higher error rates)
- **Madachy et al** Model**
  - Reduction of impacts by deferring as much change as possible to future increments
  - Effort and schedule impacts when using various size teams in a hybrid agile/plan-driven approach

* Ferreira S, Collofello J, Shunk D, Mackulak G, Wolfe P. Utilization of Process Modeling and Simulation in Understanding the Effects of Requirements Volatility in Software Development. *Proceedings of the 4th International Workshop on Software Process Simulation and Modeling*, Portland OR, 2003., 2002.

** Madachy, R., Boehm, B., Lane, J. (2006); "Assessing Hybrid Incremental Processes for SISOS Development", USC CSSE Technical Report USC-CSSE-2006-623.

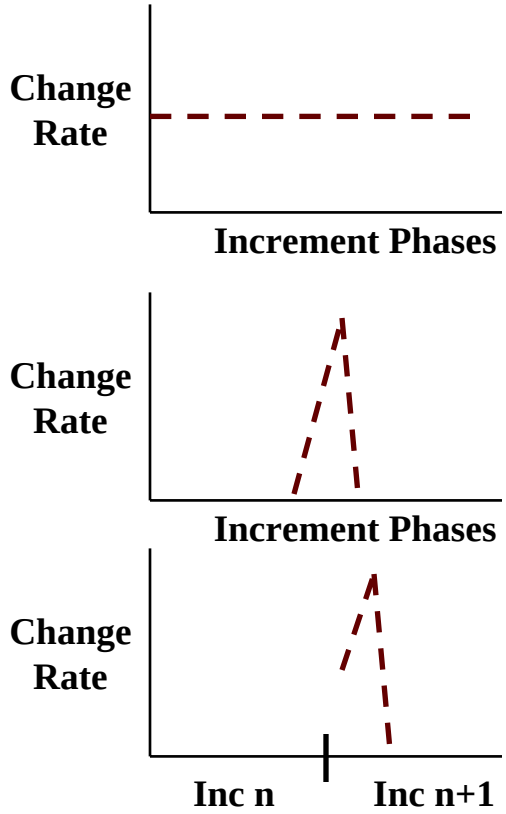
Findings of System Dynamics Models Used to Evaluate Requirements Volatility

- **Brooks' Law Model***
  - Adding more people late in the game can make the project later
  - Due to
    - Reduced productivity of initial staff to train new staff
    - Reduced productivity of new staff
- **Repenning's Model****
  - Impact of fire fighting techniques to handle late changes
  - Leads to
    - Increased overtime
    - Staff burn-out and turnover
    - Continued fire fighting to work new issues introduced in previous fire fighting activities

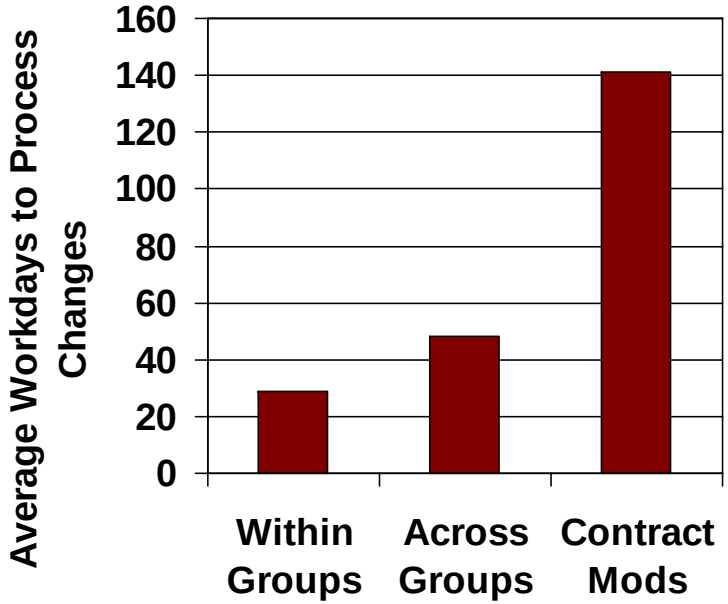
* Madachy, R., Software Process Dynamics, Wiley/IEEE Computer Society Press, 2007.
** Repenning, N., "Understanding Fire Fighting in New Product Development", Journal of Product Innovation Management, 18, pp. 285-200, 2001.

Range of Requirements Volatility Profiles

- Continual periodic change across increment
- Single mid-increment re-alignment
- Deferral to next increment

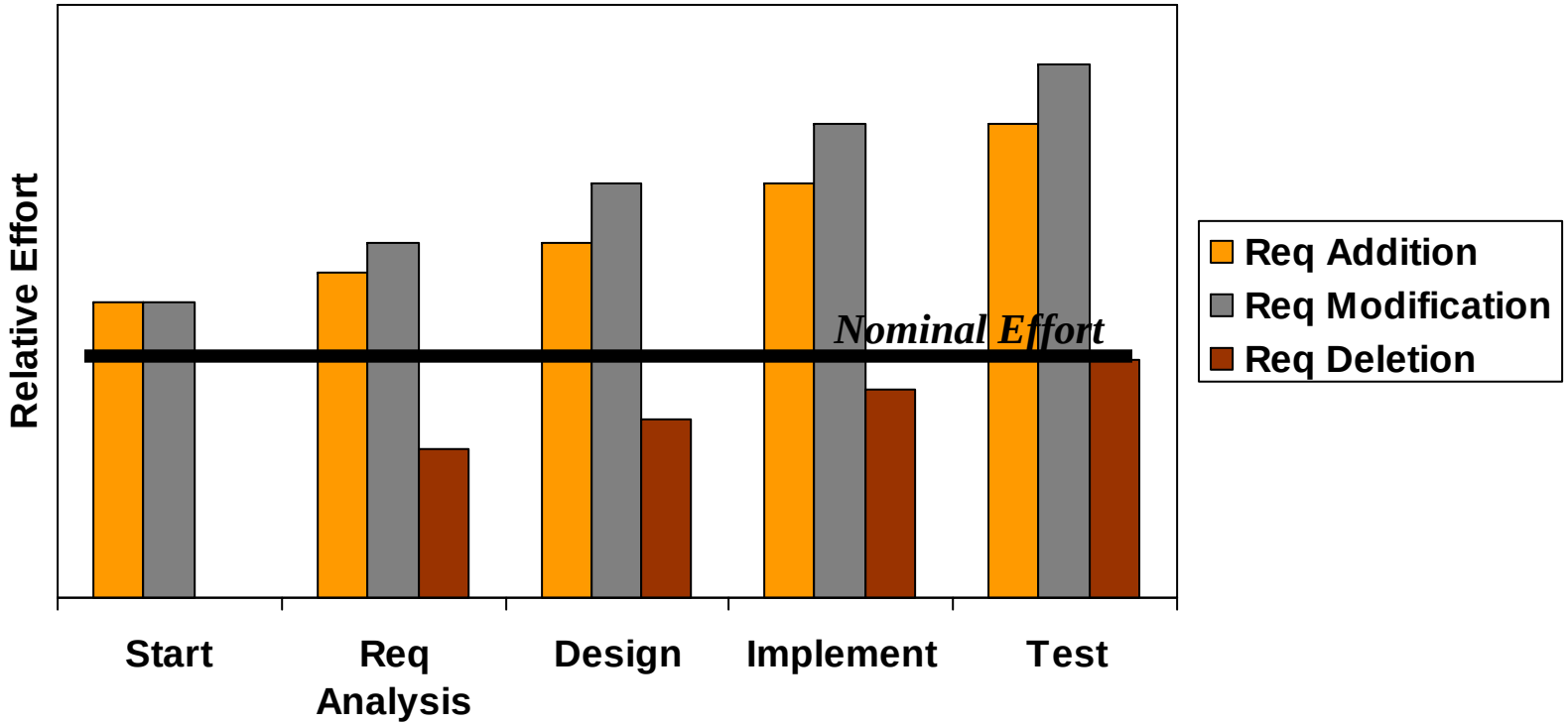


Average Change Processing Time: Based on Data From Two SoSs



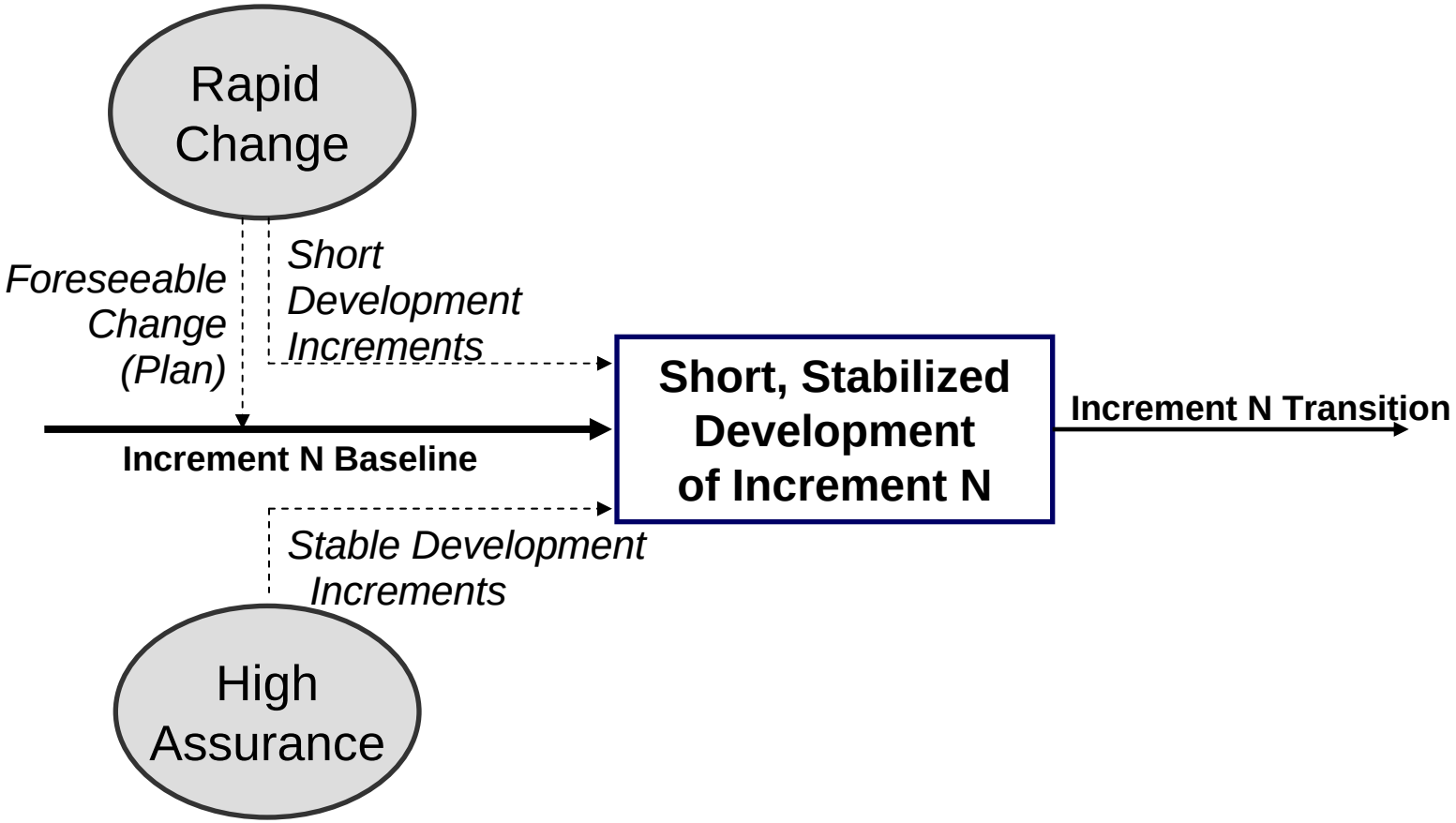
- Plan for continual change and the development of future baselines
- Most SoS changes are typically across groups and may also require contract modifications to flow down changes to multiple suppliers and vendors
- Must also negotiate changes with strategic partners
- Need to minimize impacts to increment currently under development
- Need to continually monitor evolution (changes in) the component systems for potential SoS impacts

"Cost" to Change a Requirement with Relatively Local Scope

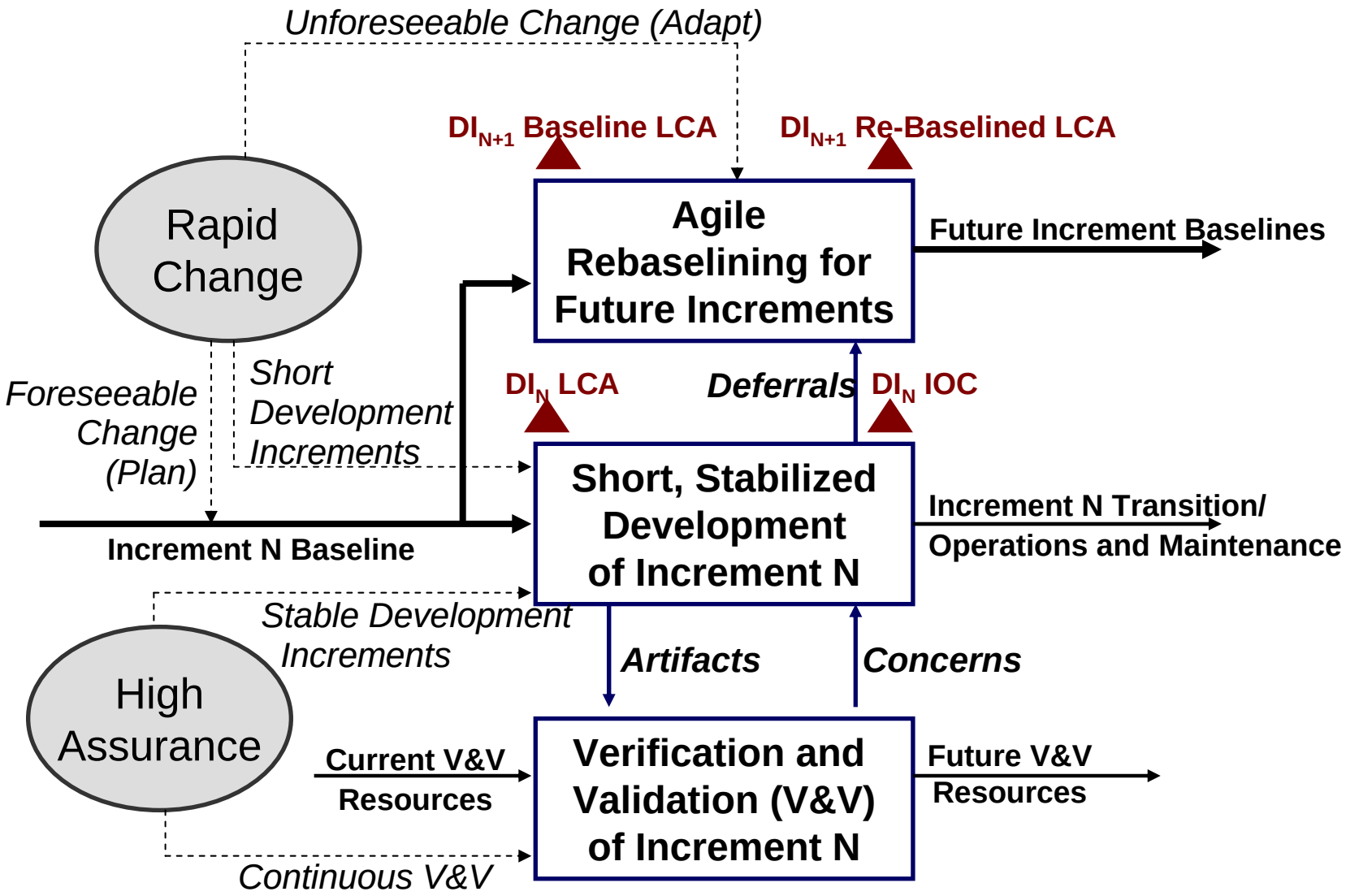


When comprehensive regression tests required to verify change (e.g., re-execution of acceptance tests), costs can exceed 100x the nominal effort to change the requirement

Risk-Driven Scalable Spiral Model: Increment View



Hybrid Process for Managing Increments



Overview

- Requirements: what are they and what are their characteristics?
- Requirements volatility: all changes are not “equal”
- Quantitative observations about requirements volatility
- *Conclusions*

Conclusions

- **Initial Question: When does requirements volatility stop all forward progress?**
- **Answer: It depends…**
  - **Continual, unending change: Probably for projects with higher change rates**
  - **A "few" controlled bursts: Maybe, but not for long**
  - **Deferral to next increment: Probably not**

Conclusions *(continued)*

- "Change" is required to evolve systems in needed directions
- How change is handled can affect impact to cost, schedule, and developer productivity
  - Architecting for change
  - Having adequate staff very familiar with the system
  - Immediate change vs. deferral to future increments
- Business processes that can significantly add to change "overhead"
  - Starting development before key stakeholders have agreed on core requirements
  - Starting detailed development before determining architecture feasibility
  - Requiring contract modifications to implement changes
  - Adding changes late in a development cycle