Acquiring Software-Intensive Ground Systems with Evolving Requirements

Suellen Eslinger
Software Engineering Subdivision
Computers and Software Division

April 2, 2008

THE AEROSPACE CORPORATION

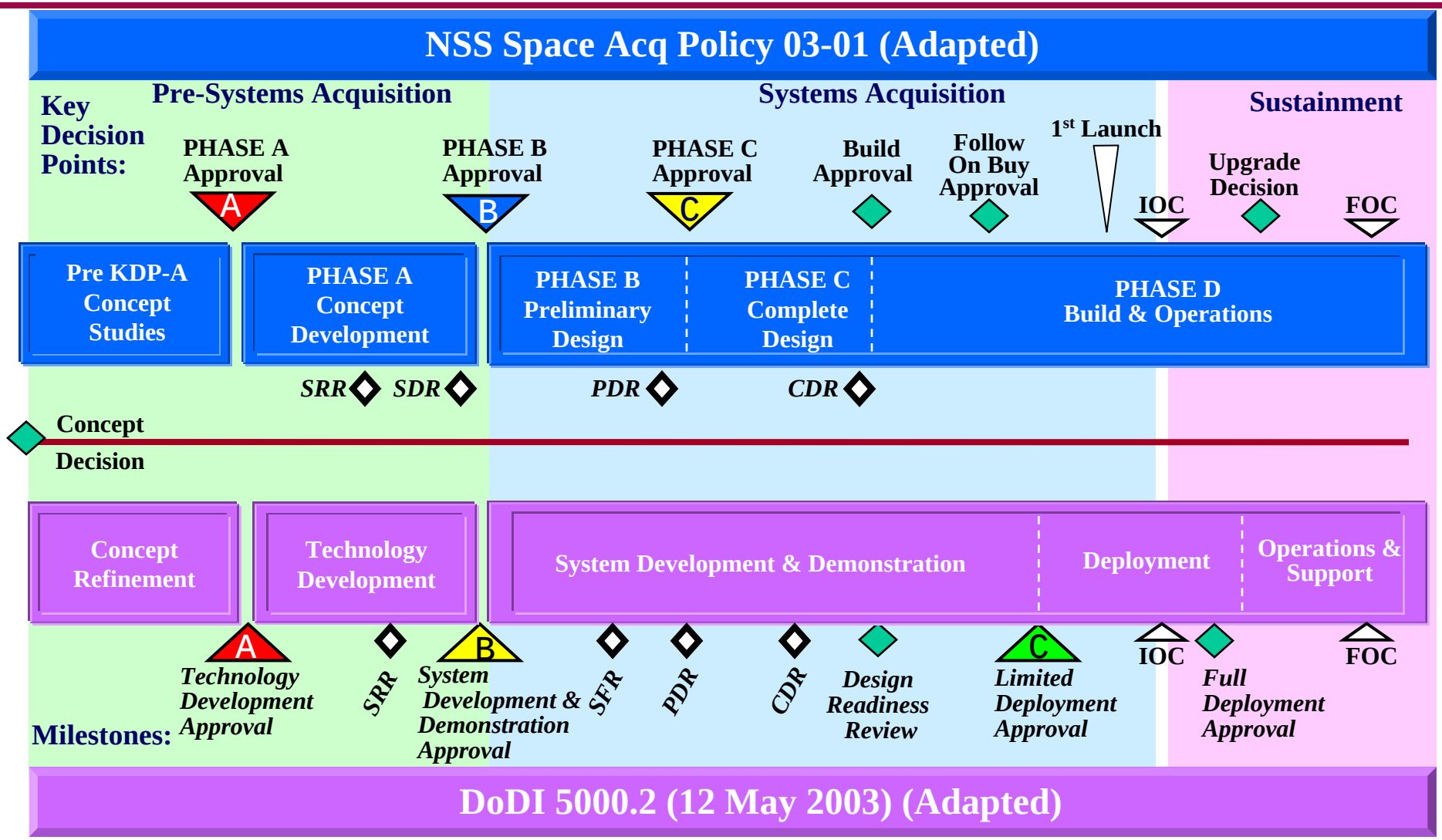
Outline

→ **Background**

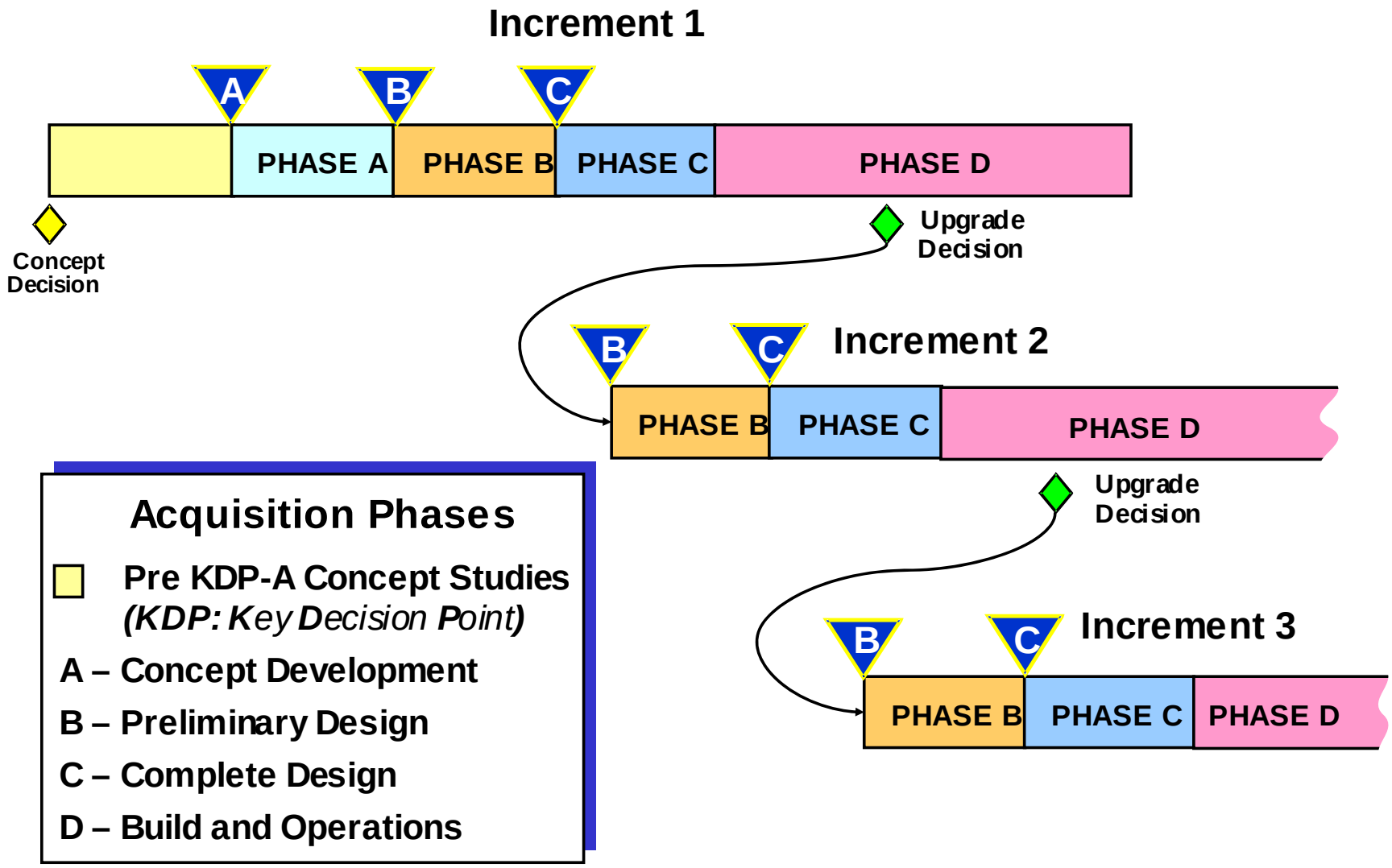
- **Pre-Systems Acquisition: Planning for Requirements Evolution**
  - ❖ Defining the program to accommodate evolving requirements
- **Systems Acquisition: Managing Requirements Change**
  - ❖ Using failure scenarios to help understand and manage volatility

The DoD and NSS Acquisition Models

Tailored for Software-Intensive Systems without Production



The Evolutionary Acquisition Pattern



Highlights of Evolutionary Acquisition

- **Evolutionary Acquisition is an approach to deliver capability in increments**
  - ❖ It is recognized up-front that new, improved capabilities will be needed in the future
- **Objective is to get mature technology rapidly to the user**
  - ❖ Implement/deliver early those aspects of required capabilities that already have their underlying mature technology foundation
    - – There is an implied recognition that technology can be both the driving and limiting force in software-intensive system development.
- **Evolutionary Acquisition is the preferred acquisition strategy for the DOD**
  - ❖ Its use helps in mitigating selected, anticipated risks, such as
    - – Requirements volatility
    - – Technology risks
    - – Dealing with system complexity
    - – etc.

Outline

- **Background**

→ **Pre-Systems Acquisition: Planning for Requirements Evolution**
  - ❖ Defining the program to accommodate evolving requirements

- **Systems Acquisition: Managing Requirements Change**
  - ❖ Using failure scenarios to help understand and manage volatility

Best Practices for Defining the Program Life Cycle

Use a software-friendly acquisition model

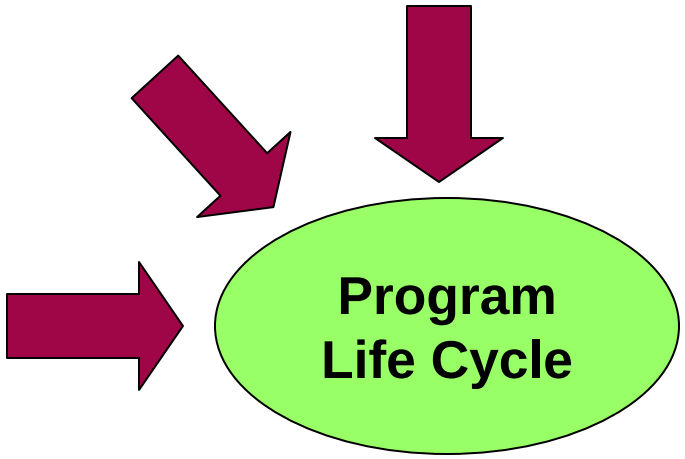
- Evolutionary acquisition is more suited to large, complex software-intensive systems

Tailor the acquisition model for software-intensive system

- Selection of a single contractor at appropriate point in software development life cycle
- With or without production phase

Choose software-friendly points in the life cycle for contract actions

- Avoid contract actions in the middle of software development spirals (e.g., post System PDR)
- Develop firm basis for software costing before MS B/KDP B



Best Practices for Developing the System Architecture

Perform software-inclusive architecture trade studies

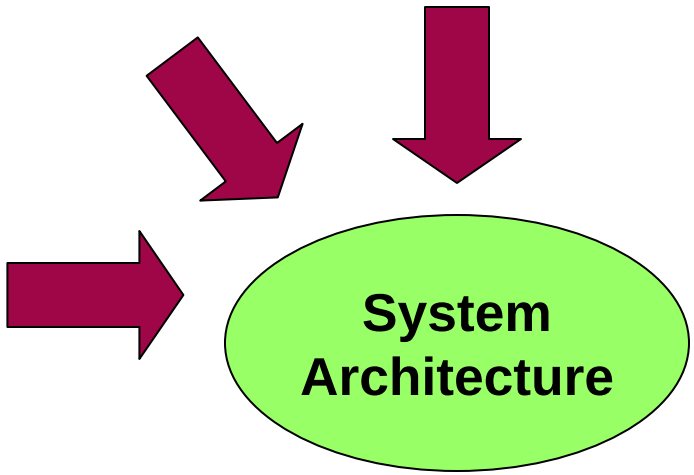
- With system architecture trades
- Identify and address critical HW/SW architecture issues
- Include major legacy components and COTS software

Select a set of integrated HW/SW architecture concepts

- Able to grow with each successive evolution with little expected rework
- Able to integrate each successive evolution with previous evolutions (and legacy system, as applicable)

Include software in evaluation of architecture concepts

- Evaluate software evolution and growth capability
- Include software in life cycle cost analysis (COTS software refresh, legacy and new software re-engineering and maintenance)



Best Practices for Developing the Government Cost and Schedule Baseline

Determine realistic SW size estimates for each evolution

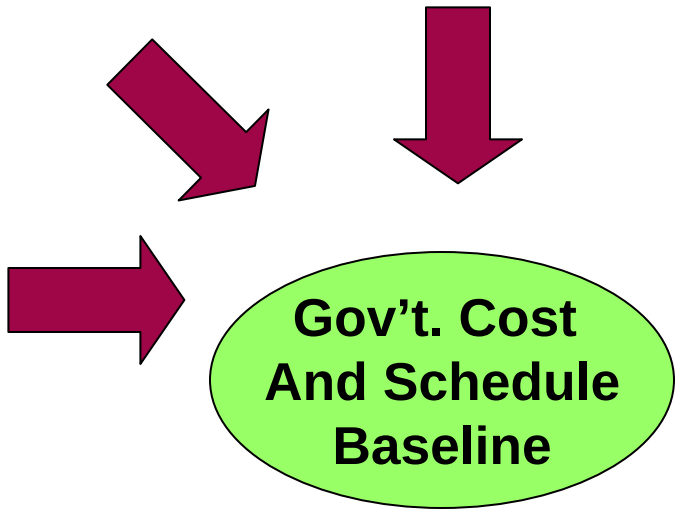
- Use Gov't. HW/SW architecture concept
- Include all SW functionality and infrastructure needed
- Use historical data from similar past programs & early concept study data

Determine realistic SW schedule estimates for each evolution

- Include all software effort in schedule
- Never compress software schedule >20% off nominal*

Determine realistic SW effort & cost estimates for each evolution

- Include COTS, reuse & new software
- Include tasks not reflected in cost models (e.g., integration of SW components costed separately, COTS)
- Estimate at least at 80% likelihood level



* Software Program Manager's Network, The Program Manager's Guide to Software Acquisition Best Practices, Version 2.31, p. 22.

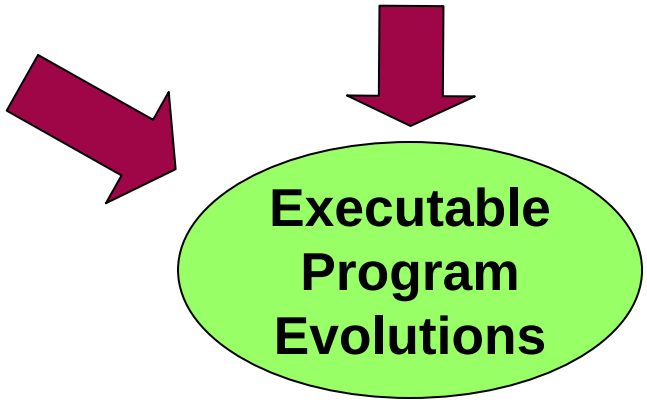
Best Practices for Defining Executable Program Evolutions

Consider SW implications when defining evolution capabilities

- Analyze feasibility of developing the required software for each evolution
  - Based on realistic software size, effort, cost & schedule estimates
  - Include software cost and schedule estimation risk
- Analyze feasibility of integrating the software in each evolution with all previous evolutions (and legacy system(s), as applicable)
  - Based on integrated hardware/ software architecture
- Plan for COTS hardware and software refresh, legacy software upgrades, and new technology insertion in each evolution

Consider SW implications when defining evolution schedules

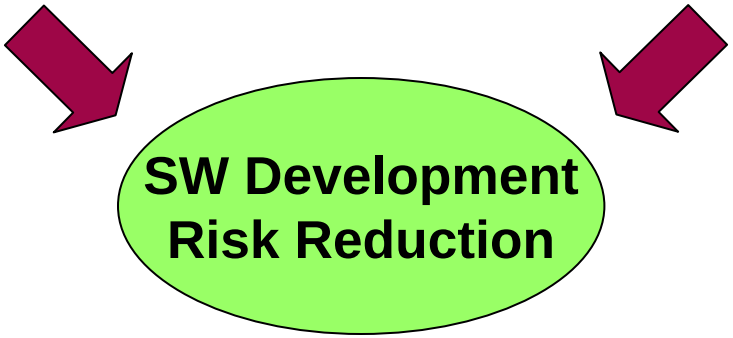
- Analyze feasibility of overlapping software development schedules for closely spaced evolutions
- Avoid plans that require developing subsequent evolutions on unknown software baselines
- Analyze feasibility of COTS refresh and legacy SW upgrade schedules



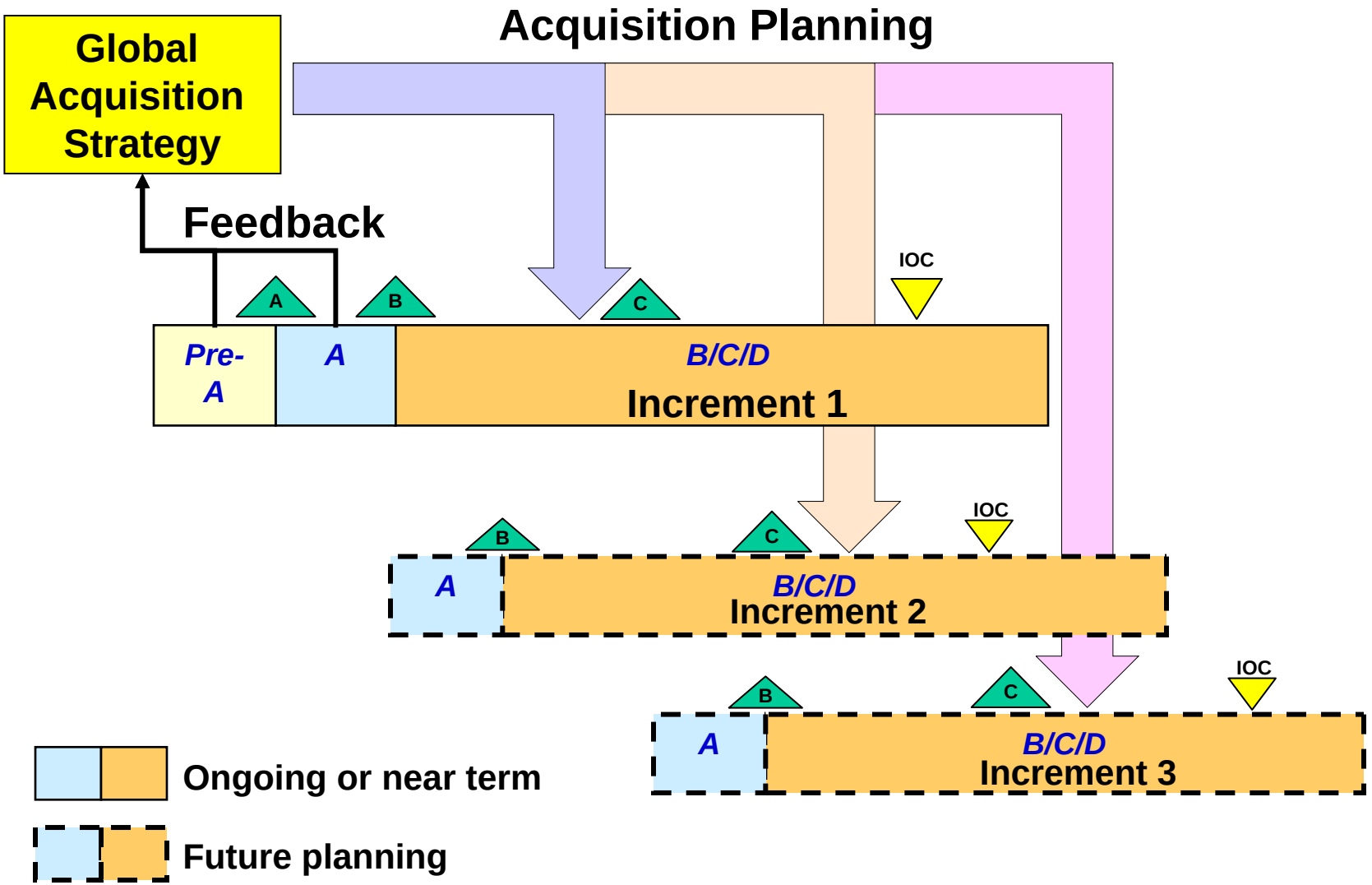
Best Practices for Reducing Software Development Risk

Contract for software product risk reduction	Contract for software process risk reduction

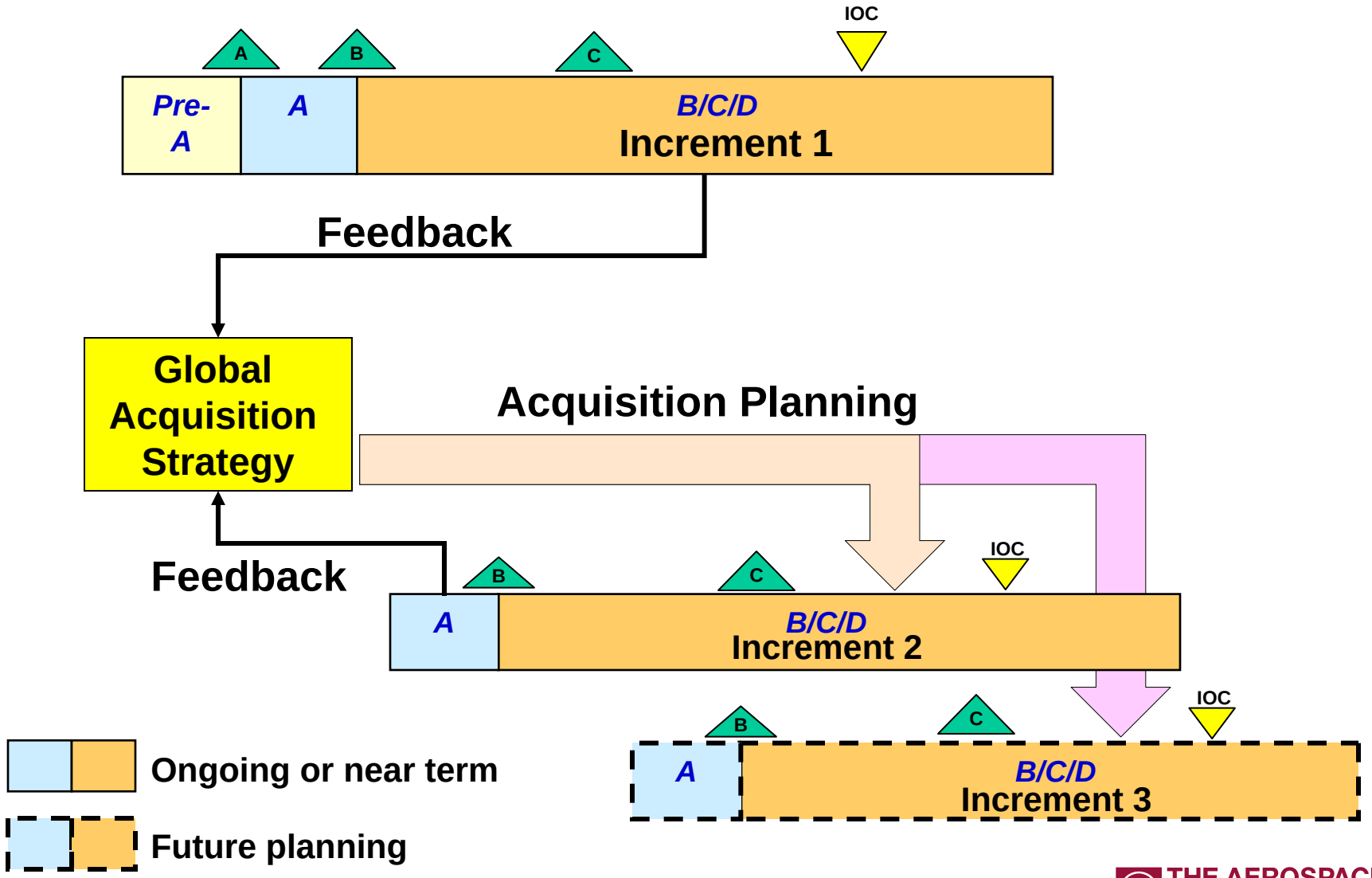
• Studies/prototyping of high risk areas for software, e.g.,
  • Mission data processing algorithms
  • Mission planning concepts
• Simulation development
• Increase readiness level of computer HW and SW technologies
	• Require delivery of Software Development Plan (DID DI-IPSC-81427a)
	• Require compliance with robust software development standard
	• Enable contractor team to prepare for software capability evaluation



Updating the Global Acquisition Strategy for Evolutionary Acquisition - 1



Updating the Global Acquisition Strategy for Evolutionary Acquisition - 2



Outline

- **Background**
- **Pre-Systems Acquisition: Planning for Requirements Evolution**
  - ❖ Defining the program to accommodate evolving requirements

→ **Systems Acquisition: Managing Requirements Change**
  - ❖ Using failure scenarios to help understand and manage volatility

Requirements Creep
IKIWISI

Description:

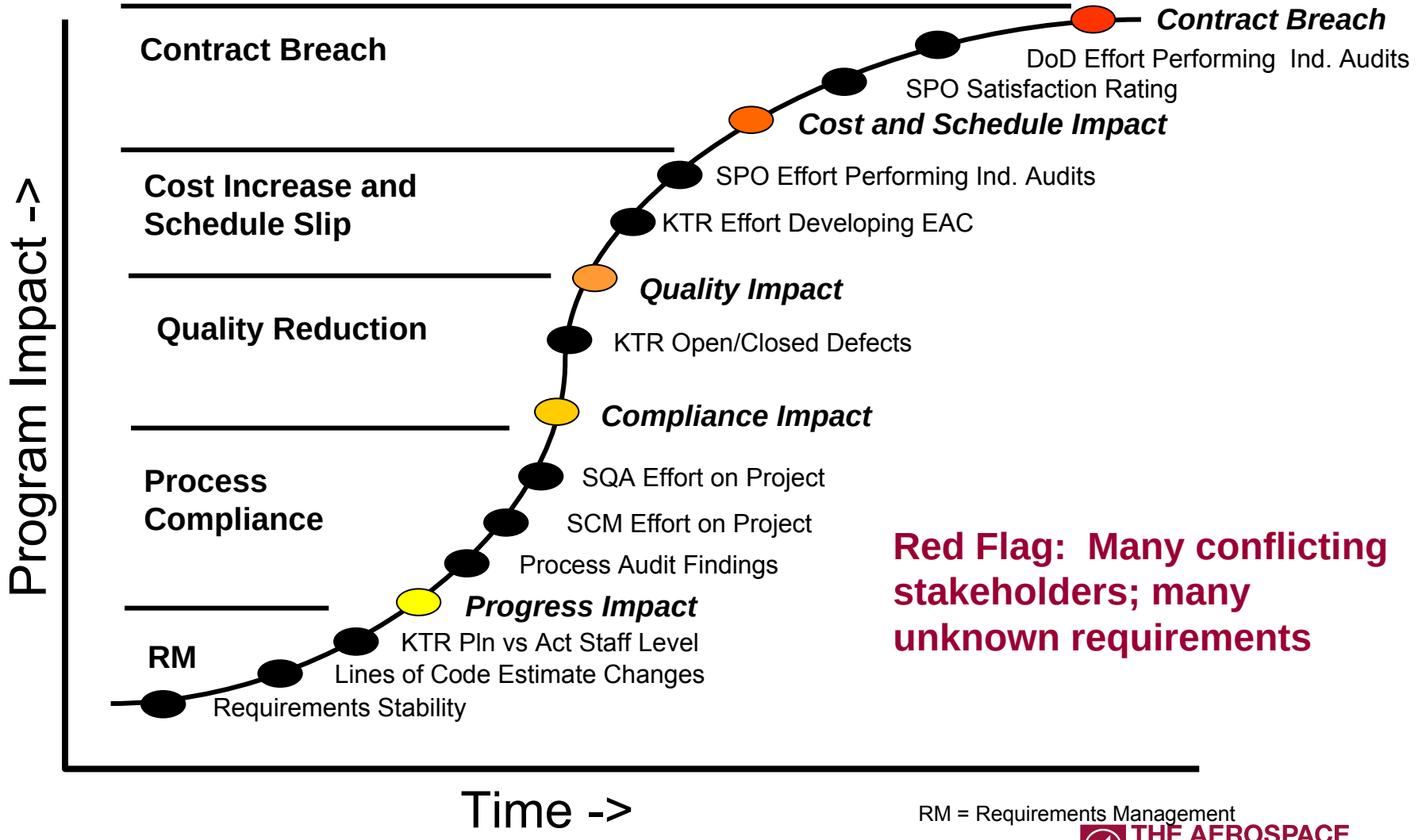
Requirements creep describes the activity of evolving the system to be what the user/customer/SPO desires it to be. This most often is witnessed during the development of an unprecedented system applying new technology. As the system definition evolves, the technical knowledge of the people involved also increases, providing opportunities for "improvement" in the initial system vision.

Red Flag: Many conflicting stakeholders; many unknown requirements

Scenario Summary:

- **Increasing requirements and requirements changes**
- **Increase in code estimates**
- **Additional personnel needs cannot be met due to unavailability**
- **Quality activities are reduced in an attempt to make up schedule**
- **Milestone slips occur as reduced productivity and rework show up**
- **More personnel are added, further reducing productivity**
- **Cost increases are experienced**
- **Schedule slip is experienced**
- **Finally, a contract breach occurs**

Requirements Creep Profile Indicators

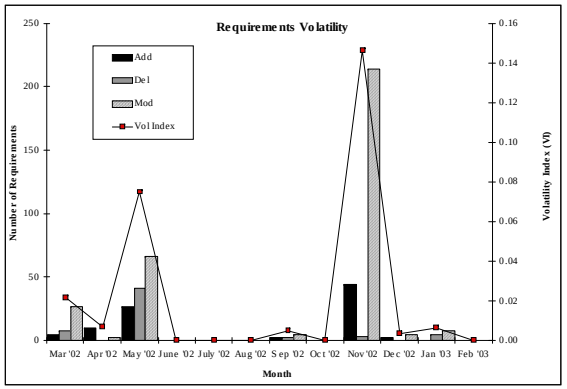


Red Flag: Many conflicting stakeholders; many unknown requirements

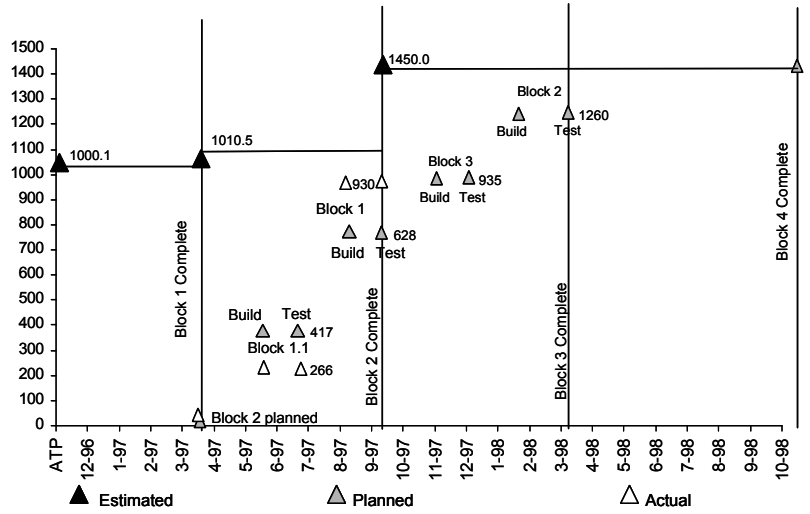
RM = Requirements Management

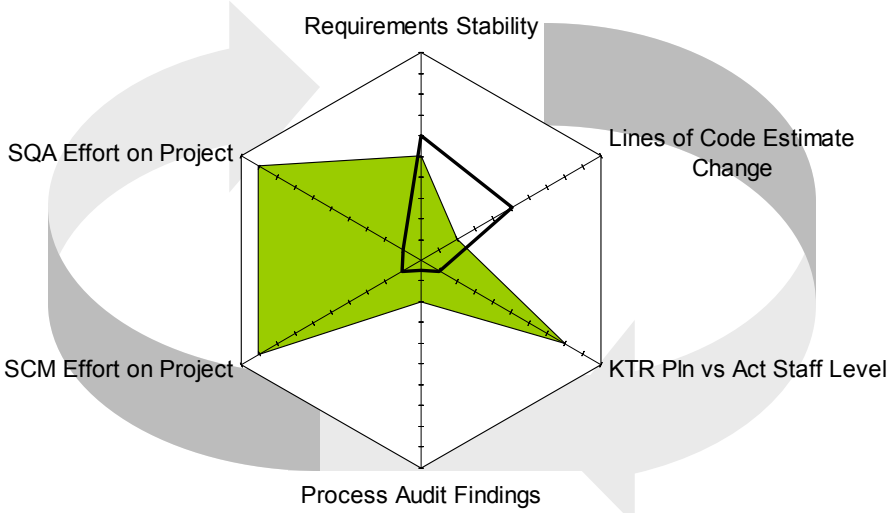
Critical Indicator Set
Requirements Creep Profile

Year												
Month	Mar '02	Apr '02	May '02	June '02	July '02	Aug '02	Sep '02	Oct '02	Nov '02	Dec '02	Jan '03	Feb '03
Activity												
Total Active Requirements	1791	1788	1798	1784	1784	1784	1784	1784	1784	1825	1827	1823
Added (A)	5	10	27	0	0	0	2	0	44	2	0	0
Deleted (D)	8	0	41	0	0	0	2	0	3	0	4	0
Modified (M)	26	2	66	0	0	0	5	0	214	4	8	0
VI =(A+ D+M)/Total	0.02	0.01	0.07	0.00	0.00	0.00	0	0.00	0.15	0.00	0.01	0.00



- Requirements Stability >1%[7] per month of unstable requirements for 3 consecutive months indicative of an unstable project
  EXAMPLE: Volatility Index indicates unstable condition for Mar/Apr/May
- Lines of Code Estimate Changes Growth in SLOC estimate is >20%[8] necessitates project rescope
  EXAMPLE: New SLOC Estimate-Old SLOC Estimate/Old SLOC Estimate = Lines of Code Estimate Changes (1450-1000.1/1000.1= 45%)

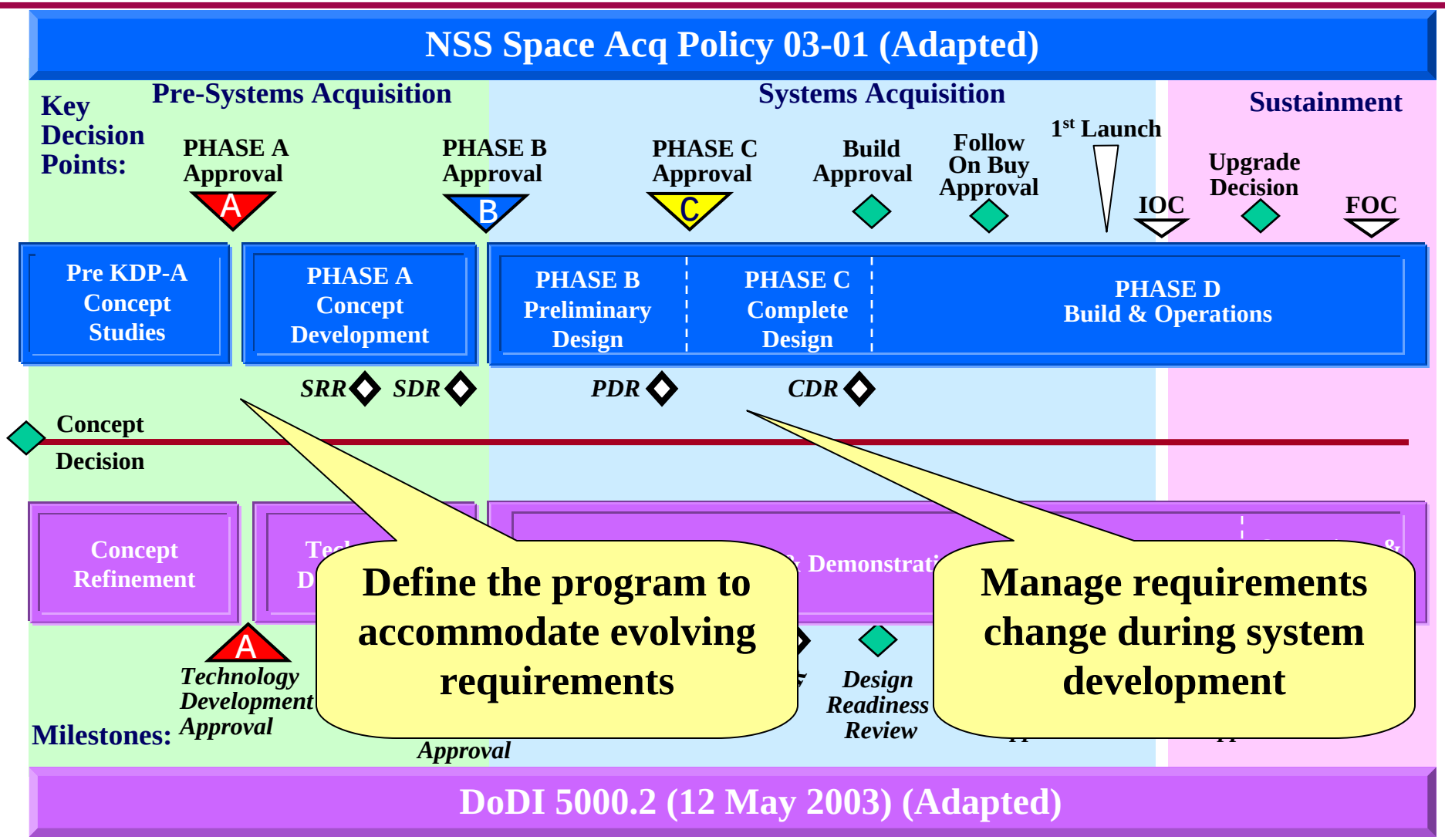




THE AEROSPACE CORPORATION

The DoD and NSS Acquisition Models

Tailored for Software-Intensive Systems without Production



Author Contact Information

- **Suellen Eslinger**
  - Distinguished Engineer
  - Software Engineering Subdivision
  - (310) 336-2906
  - Suellen.Eslinger@aero.org