LAI

LEAN
AEROSPACE
INITIATIVE

# Spiral Development

Presented By
Eric Rebentisch
MIT/LAI

Product Development Meeting
October 7, 2003

# March ‘03 Plenary Breakout Session: Spiral Development

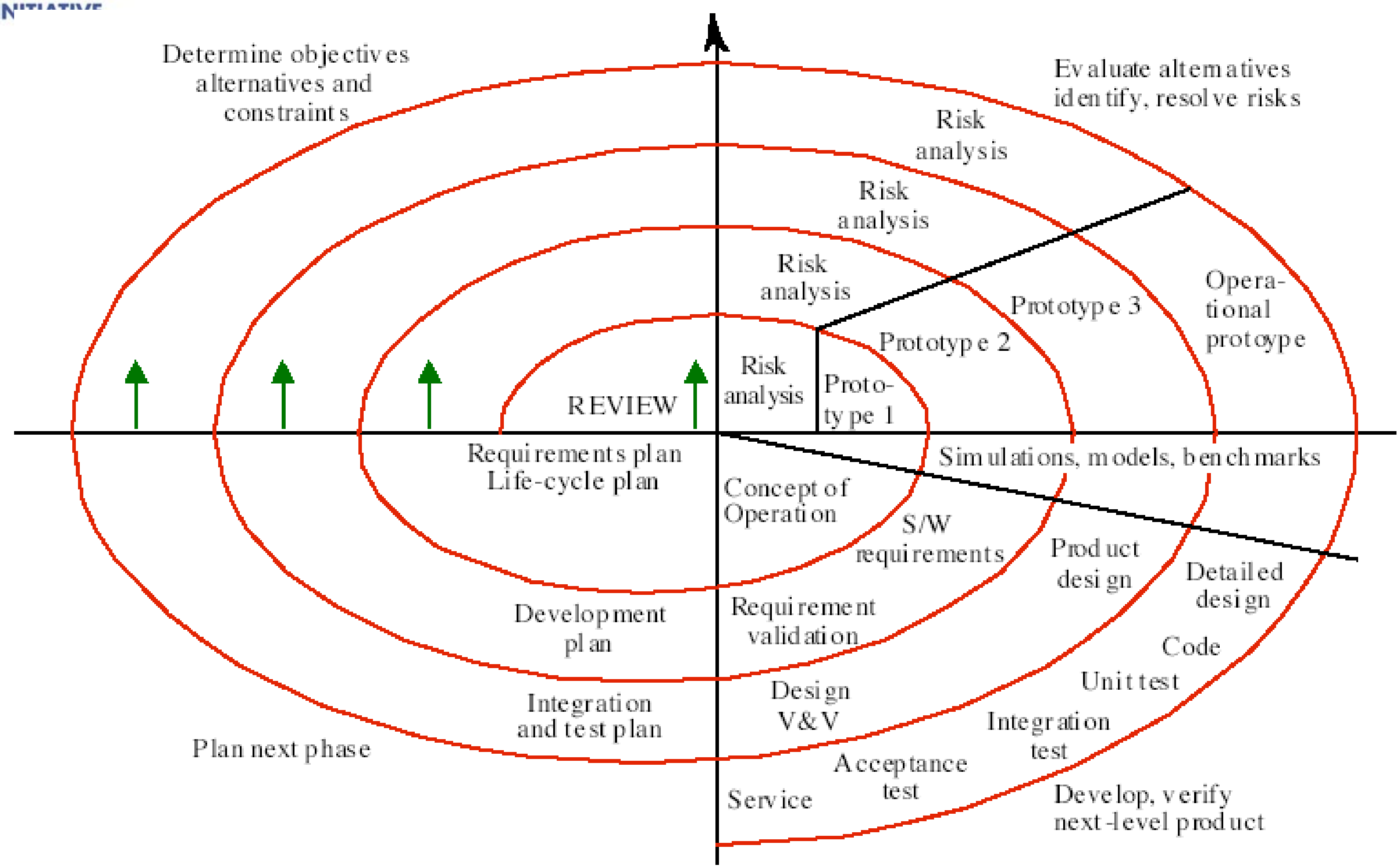
- Bobak Ferdowsi (MIT) “Evolutionary product development strategies”
- Panel: "Putting Spiral Development into practice"
- Dr. Beryl Harmon (DAU)
- Ms. Tina James (SAF/ACE)
- CDR Rick McQueen (Globalhawk SPO)
- Lt. Tim Spaulding (MIT/Harvard)
- Jeremy Tondreault (BAE Systems) “Iterating development to produce affordable military avionics systems”
- LtCol Rob Dare (ESC/ACE) “Collaborative Requirements Development”

**First Look at LAI Spiral Development Work**

# Evolutionary Acquisition

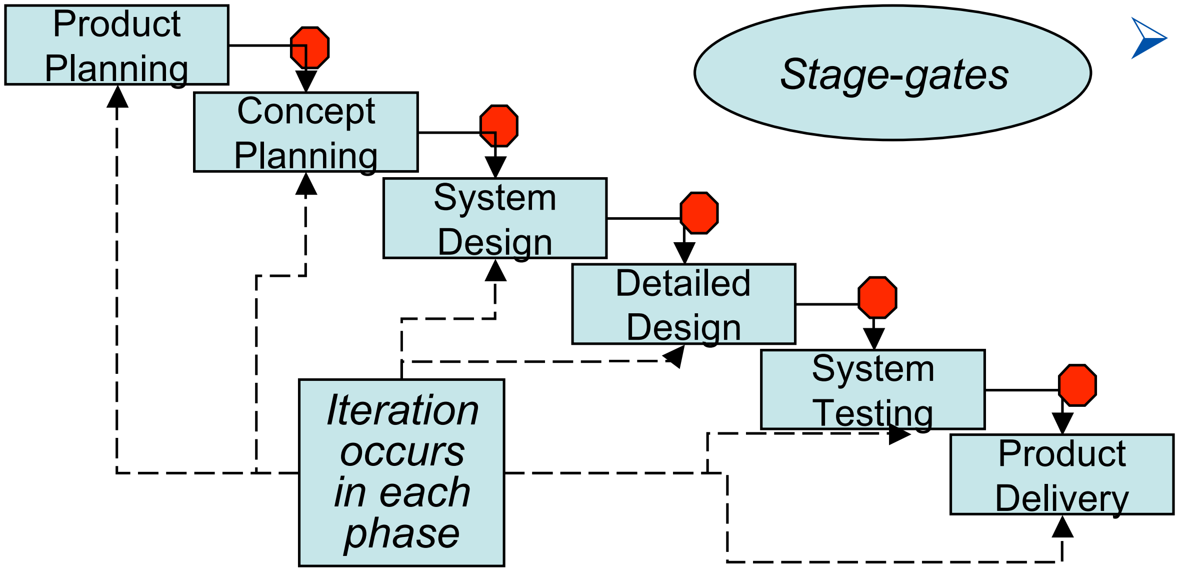
- Air Force realized the need for better development strategies
  - Increasing costs and cycle times for new products
    - Technology innovation cycle times shorter than product cycle times
    - “When it takes so long, it just can’t be state of the art” --Dr. Sambur, Assistant SAF/AQ
- Evolutionary Acquisition with Spiral Development
  - Use increments and/or spirals to quickly grow the system capability
  - Increase user feedback

# Spiral Development



- Addresses user requirements uncertainty
  - “I’ll know it when I see it”

# Waterfall Processes



Implies static requirements

- **Variants:**
  - Parallel waterfall
  - Overlapping waterfall
  - Evolutionary prototyping & delivery
  - Design to schedule & budget

- Primarily mitigates technical risks
  - Can address some user uncertainty
  - Can address schedule and cost risks

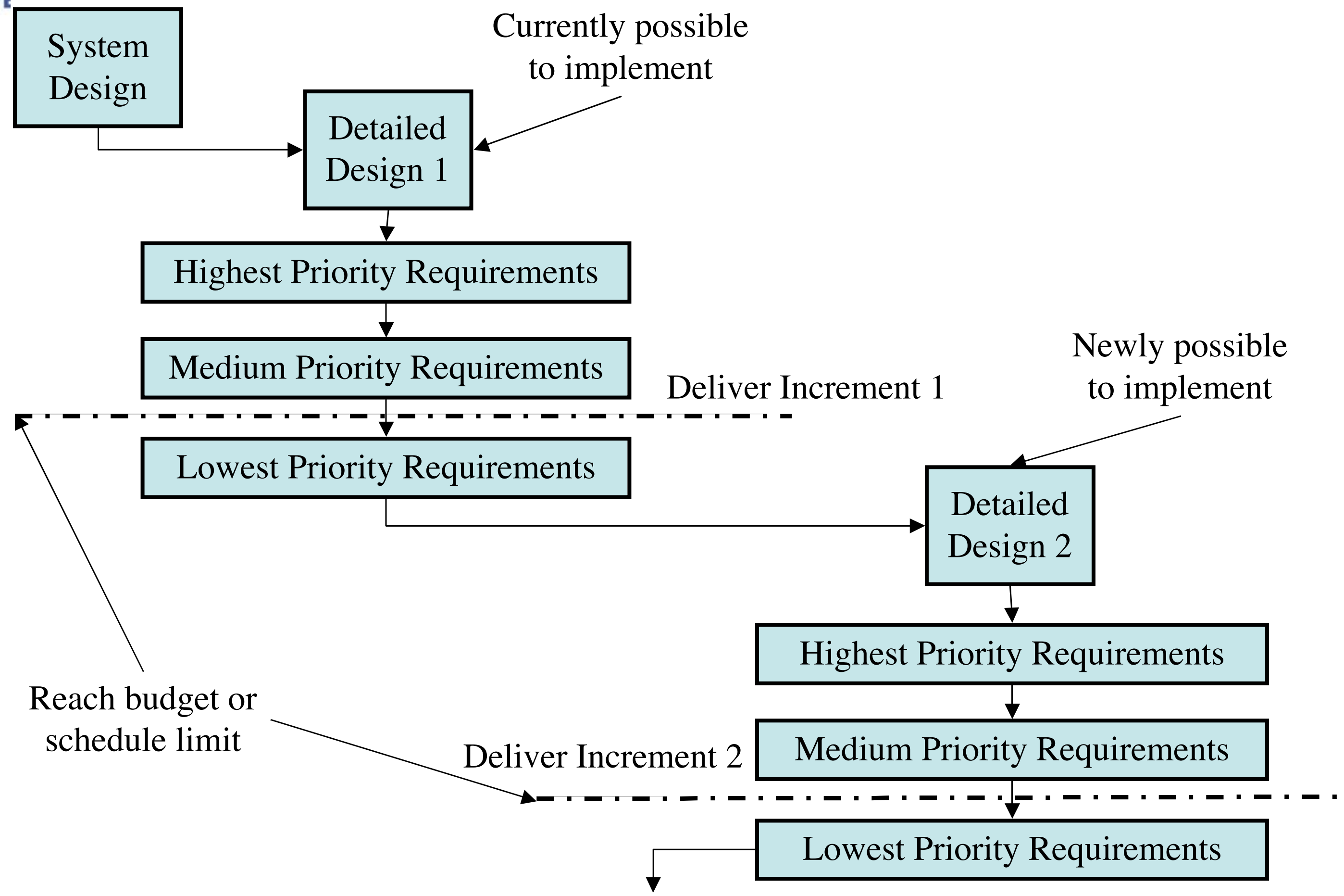
# MATECON Spiral Development Research

- Several recent theses used MATECON method to assess applicability to spiral development (Derleth, Spaulding, Roberts, Shah)
- Findings:
  - Once a MATECON model is constructed, it is readily adapted to explore evolutionary architectures
  - Individual architectures can be identified that have persistent superior performance over multiple increments (and "*one hit wonders*" eliminated)
  - Evolutionary pathways can be mapped (in specific discrete steps) to take an existing sub-optimal legacy design to a performance frontier
  - Modeling process' strengths lie in creating a communication medium (system representation) and developing intuition for system behavior over multiple increments

# Research Approach

- Deliver useful tool for program managers to select the ‘best’ process
- Six case studies with program managers and chief engineers
  - Programs identified as Evolutionary Acquisition leaders
  - Mix of software and hardware
  - Various degrees of development
  - Different approaches to Evolutionary Acquisition
- Broad-based survey of program managers in review at SAF/AQ

# Acquisition Strategy

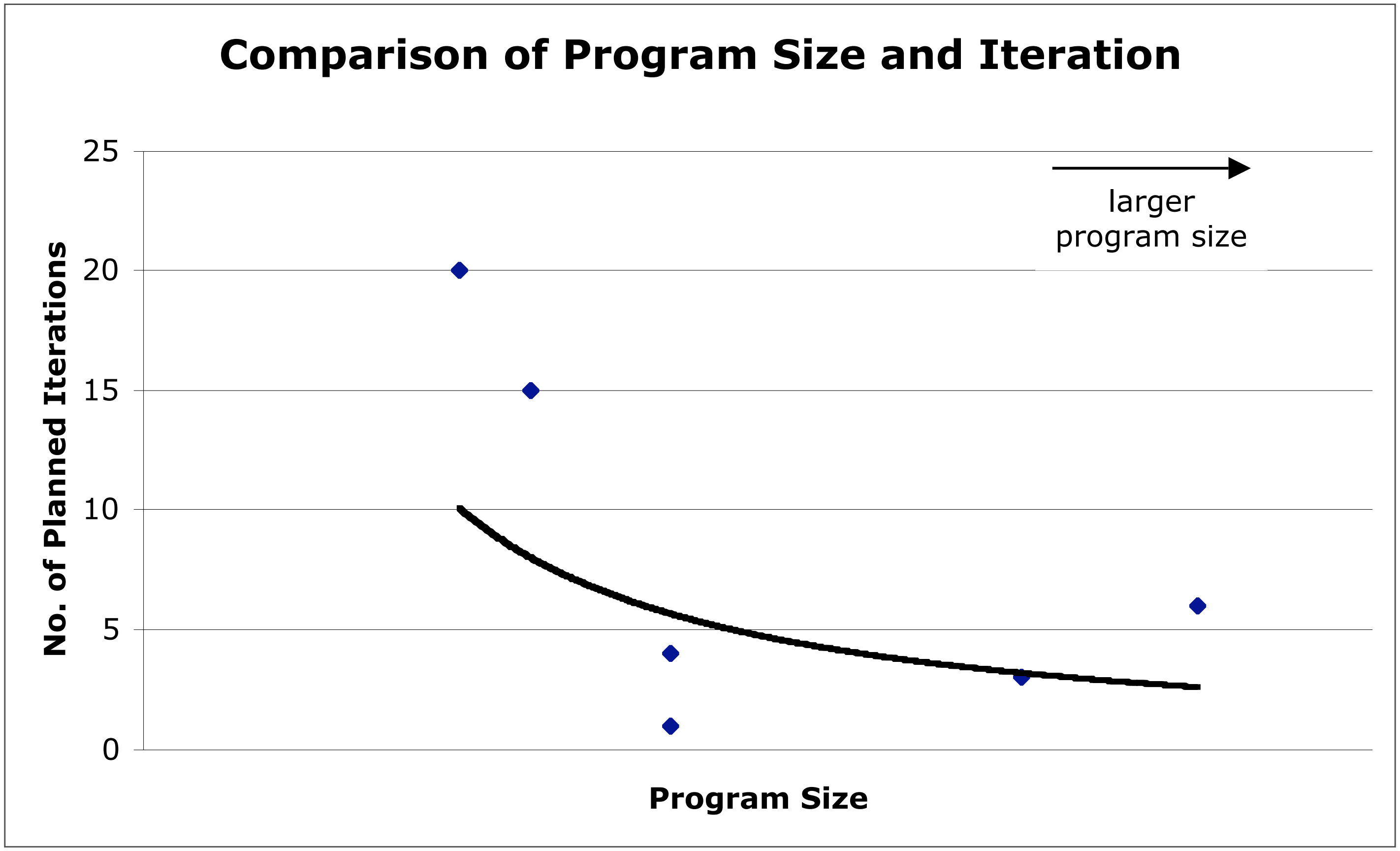

System Design
Detailed Design 1
Currently possible to implement
Highest Priority Requirements
Medium Priority Requirements
Deliver Increment 1
Lowest Priority Requirements
Newly possible to implement
Detailed Design 2
Highest Priority Requirements
Medium Priority Requirements
Reach budget or schedule limit
Deliver Increment 2
Lowest Priority Requirements

**Design to schedule/budget most commonly observed strategy in these evolutionary acquisition cases**

# Evolving Requirements a Continuing Challenge

- Program managers tended to want to freeze requirements early on so as to better plan the process and execute to a predictable schedule
  - Prioritizing requirements with the user
  - Allowing requirements changes only when additional funding is provided
- Difficulty with user expectations and understanding of EA
  - Too many requirements in the first increment
  - Program managers addressed this in two ways
    - Used demonstrators to show capability and gather feedback
    - Agents within the user community as disseminators
- The majority of program managers were primarily budget constrained with the prospect of budget cuts
  - Programs could not keep budget reserves
  - Used the requirements as reserves and cut requirements accordingly

# Acquisition Strategies Varied by Program Size

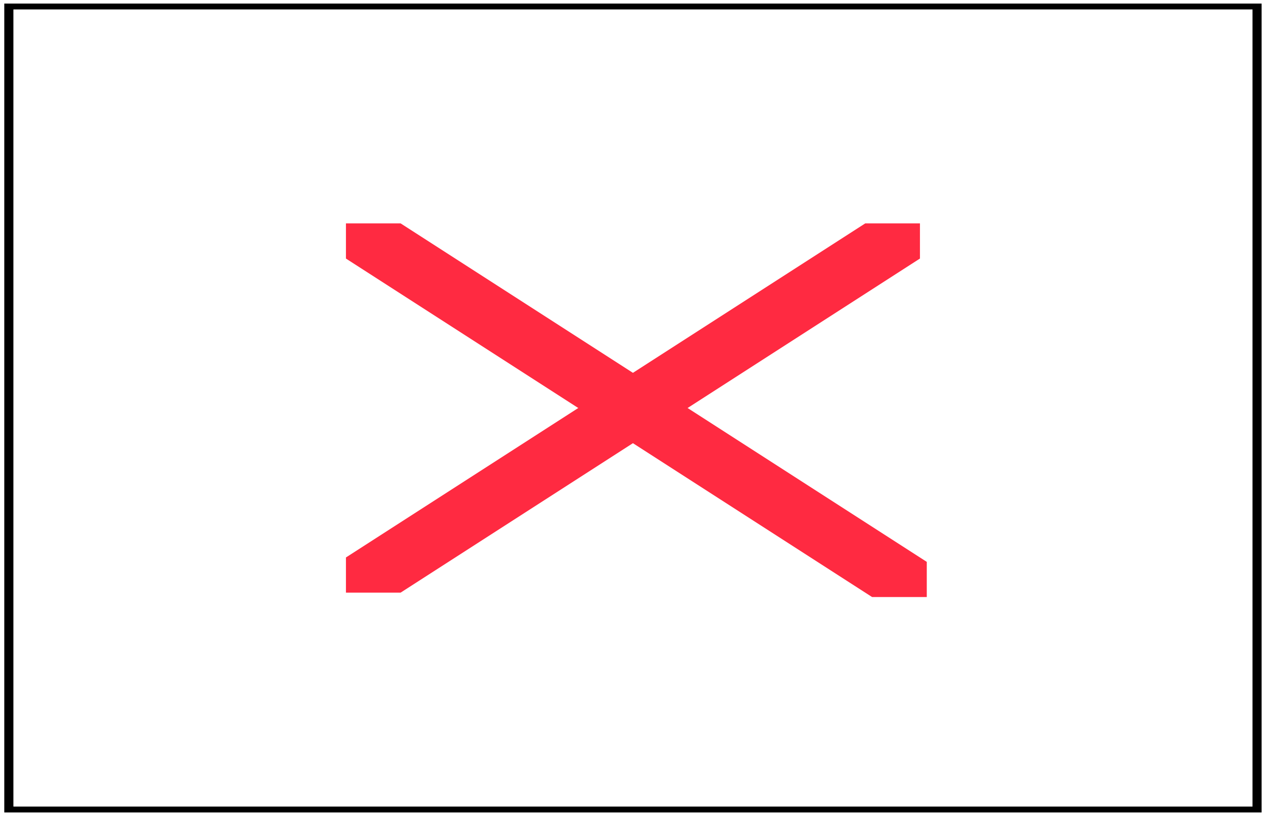


**Larger (more expensive?) programs had fewer planned iterations**

# Concurrency Increased Workload of Enterprise *Enabling* Infrastructure

- Planning
  - High concurrency in programs meant managers were working on one increment while planning for the next
- Contracting
  - More increments meant more contracts
  - Contracts were not as flexible as the programs
- Engineering
  - Concurrency often meant that testing for one phase was going on at the same time as engineering for another--engineers were no longer available to address testing finds
- Logistics
  - Multiple configurations of the same system
  - Upgrading existing systems to new standards was not always easy
- Testing
  - Increased testing loads associated with multiple increments
  - Increments are tested as if they were completely new systems

# Value of Modularity to Program Adaptability



**Modularity didn’t offset challenges of making changes to highly interdependent programs**

# Use of Open Architectures Not Without Challenges

- Open architectures possible only if interfaces are standard, and data is not proprietary
  - Cannot use systems from various vendors
- Implications:
  - Own interfaces and data rights between modules and subsystems
    - Develop own standards based on commercial or otherwise
    - Purchase data rights from commercial companies so that the Air Force owns and operates the data transfer between systems
  - Use off-the-shelf components only after lifecycle analysis

LAI
LEAN
AEROSPACE
INITIATIVE

# Observations (so far…)

- Program/system complexity still a significant issue
    - Program size a barrier to responsiveness and adaptability
    - Bigger programs look more like traditional incremental waterfalls
    - Simple modularity vs. complex modularity
- Evolutionary acquisition currently involves increasing concurrency
    - Lean an important enabler to create additional needed capacity
- Enterprise perspective important to ensure enabling infrastructure doesn't become the system constraint
- The testing process must be updated to apply to evolving systems
    - Full scale testing for each increment or deliverable is not practical
    - Regressive testing of changes in the system for sufficiently decoupled systems