Architecture Centric Evolution

A Personal Perspective

Dr. Charles (“Bud”) Hammons

Software Engineering Institute
Acquisition Support Program
cbh@sei.cmu.edu

Ground Systems Architecture Workshop 2005

Agenda

- Thesis
- Complex Systems
- Complexity Drivers
- Architecture and Evolvability
- Q&A

Thesis

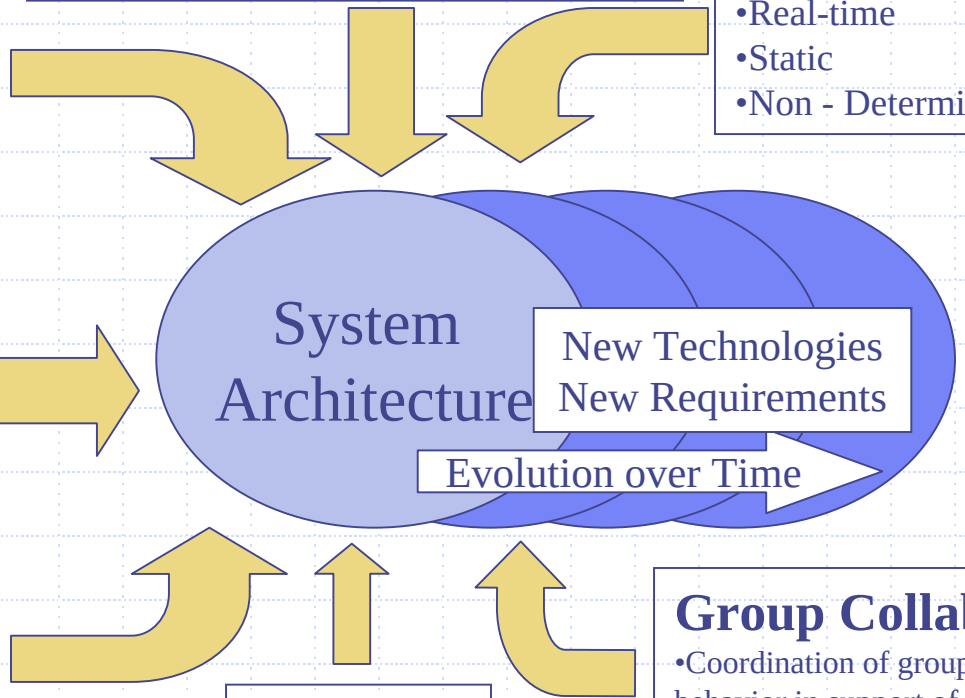
- We are learning to live with fewer fixed assumptions about systems than in past acquisitions
  - Changing threats induce evolution/change in CONOPS
  - Changing requirements often invalidate point-solution designs and interact strongly with CONOPS
  - Desire to exploit rapidly evolving technology removes prior assumptions about relatively deterministic technical base
  - Policy/economic motivations to exploit COTS/Legacy elements complicate interfaces and system operation
  - Network-centric systems embody numerous open issues, notably enterprise integration and interface proliferation
- Evolvable System/Software Architecture is a necessary, but not sufficient component of a strategy to master the new environment

Complex Systems

- Unprecedented Requirements
  - Performance (timeline, capacity, ..)
  - Complexity of Human-Machine task division
  - System autonomy
  - Scope of program
- Non-linear requirements interaction, induced by
  - Size, weight, power constraints
  - Schedule and/or cost constraints
- Dependence upon “maturing” technology
- Integration challenges presented by unprecedented configurations
- Evolution complicated by long service life

Complexity Drivers

Complex System Requirements (FCS Phase 1)



"Illities"
- Dynamic Reconfigurability, Fault Tolerance
- Graceful Degradation, Information Assurance
- Quality of Service, Security
- Evolvability

Communications & C4ISR Attributes
- Asynchronous
- Non Real-time
- Dynamic
- Deterministic
- Location Transparency

Robotic Attributes
- Synchronous internal control
- Real-time
- Static
- Non - Deterministic

System of Systems Attributes
- Asynchronous
- Real-time
- Dynamic
- Location Awareness
- Location Transparency
- Non - Deterministic

Embedded Attributes
- Synchronous
- Real-time
- Static
- Deterministic

Scenarios
- Mission
- Support
- Change Cases

Group Collaboration
- Coordination of group and individual behavior in support of mission(s)
- Adapt to dynamic, hostile environment
- Variable levels of interaction among agents, robots, humans, collections

Credit: Raytheon, Network Centric Systems

Complexity Drivers
Technical

- Performance
  - Highly distributed (ALP, TCIMS, FCS)
  - Difficulty of composing verifiable requirements that trace directly to operational performance (Javelin focal plane)
  - Hard Real-Time + Enterprise complexity (FCS)
- Dynamic Human-Machine task division (Pilot's Associate)
- Increasing of external interfaces to other evolving systems (TSAT)
- Need for comprehensive domain model to comprehend complex behavior (ALP, PA, TCIMS, FCS)

Architecture and Evolvability

- Recognize Fluidity of the Environment
  - CONOPS
  - Requirements
  - Technology
  - Human-machine task division
  - Number and kind of external interfaces
- The architecture must serve as basis for a family of similar systems that evolve, one from the other, over time.
  - →it is no longer a “point solution”
  - →it defines a “capability envelope” for all system instances

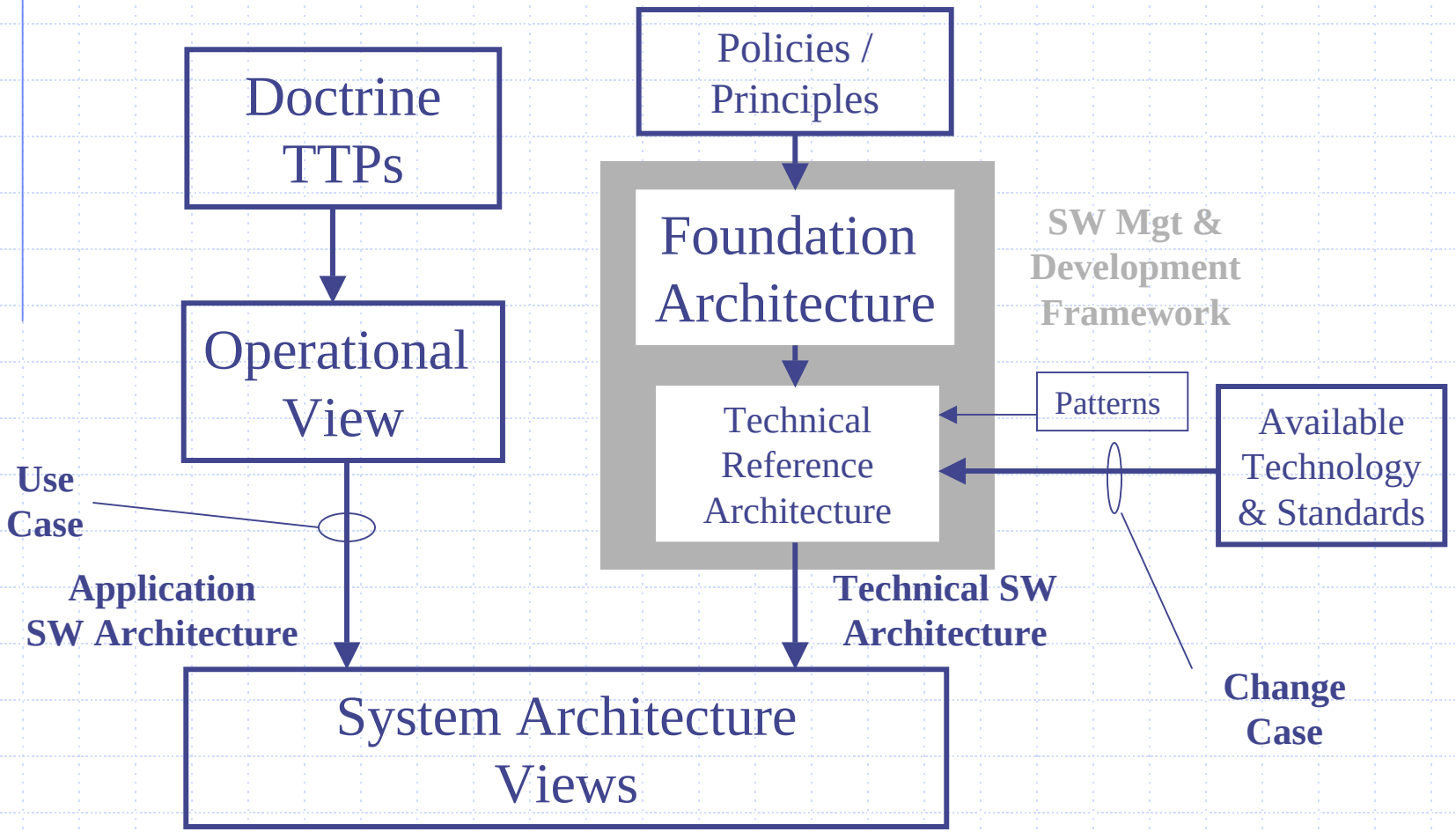
Architecture and Evolvability

With so much in flux, where do we find fixed points?

- Basic Principles employed to understand problem domain and technology
  - Policies that constrain any design
  - Separation of concerns
  - Design patterns
  - ..
- Synthesized foundation architecture that supports broadest range of data, behavior, and technology anticipated in solution space.
- Processes to capture and adjudicate interactions between operational and technical architecture.
- Mechanism(s) for evolution.

Architecture and Evolvability

Notional Evolution Process (Software)



Credit: Raytheon, Network Centric Systems

Probable Features

of Foundation Software Architecture
for contemporary systems

- Layered system with loose coupling
- Inherently distributed
- Flexible common messaging model
- Flexible administrative control
- Integration-friendly observability of data
- Simulation-friendly interfaces
- Formalized flexible task allocation to humans and/or automation
- Conscious employment of design patterns

What else is needed?

- Socialization/acceptance of the concept based on value argument
- Integrate evolvability into document packages and the acquisition cycle
  - Start early on, in parallel with Initial Concept Document
  - Refine as acquisition moves forward
  - Engage contractor community early and often
- Suggests even more emphasis on cross IPT coordination
- Continued dependence upon individuals/groups with deep understanding
  - Domain
  - Technology
  - Integration
  - Programmatics

Summary

- Evolvable architecture augments existing architecture models, is not a replacement
- To make an impact, the concept needs to be integrated into the acquisition cycle

Questions/Comments?