# Enabling Agility on Complex System Developments

## Getting to Agile Implementation as Fast as Possible; a Systems Perspective

# Authors

**Michael Coughenour**

*Lockheed Martin IS&GS, System Engineering Technologist;*
*INCOSE Agile SE Working Group Co-Chair*
*mike.coughenour@lmco.com 618-910-0133*

**Jim Brake**

*Lockheed Martin IS&GS, Senior Manager*
*jim.brake@lmco.com 719-277-5438*

**Brad Newman**

*Lockheed Martin IS&GS, MBSE SME, CSEP*
*bradford.j.newman@lmco.com 719-277-4118*

# Topics

- What is the Challenge?
- What is Complexity
- What's Stopping Us?
- How do we Solve a Problem?
- Which Engineering Approaches Should we Use?
- How do we Architect to Enable Agility?
- How Much is Enough - How do Architects Know/Decide?
- How do we Currently Architect a Solution?
- What are Four Dimensions of Architectural Considerations?
- What is the Framework for Enabling the Architectural Considerations?

# What is the Challenge?

- **How do we Get to Usable Solutions as Fast as Possible?**
  - We have to enable Agile Development ASAP without adding significant risk or unacceptable technical debt
- **A different systems development environment exists today than even a few years ago**
  - Several forces are influencing this environment
    - Complex, Adaptive and Quickly Evolving Threats
      - Self-Adapting Systems
      - Improvised Explosive Devices (IEDs)
      - Cyber Attacks
    - Budgetary Pressures
      - Reduced budgets
      - Firm Fixed Price contracts
    - Interoperating with Legacy Systems
      - Impedes the delivery of rapid capabilities

# What is Complexity?

- **Detailed Complexity (Complicated)**
  - A system that is composed of a great number of different parts
    - The relationships between the parts are known
    - The system has predictive behavior
- **Dynamic Complexity (Complex)**
  - "A system presents dynamic complexity when cause and effect are subtle, over time (Peter Senge, "The Fifth Discipline")
    - Different behavior in the short term vs. the long term
    - Obvious stimuli produce non-obvious results
      - » Unpredictable and uncertain →

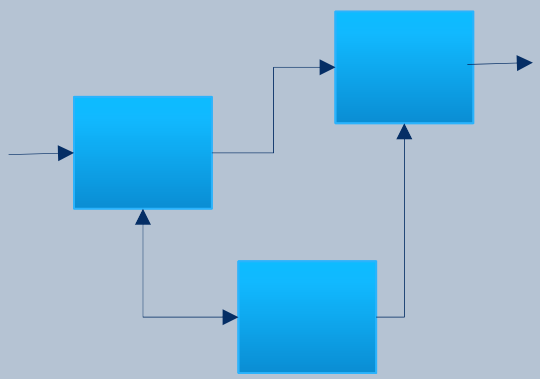
Rolls-Royce Derwent turbojet engine, Mark Nicolson, Acutance 2008

Public Domain: President George H.W. Bush, 1991 Addresses Joint Session of Congress by Susan Biddle (NARA) National Archives via pingnews

# Increasing Complexity

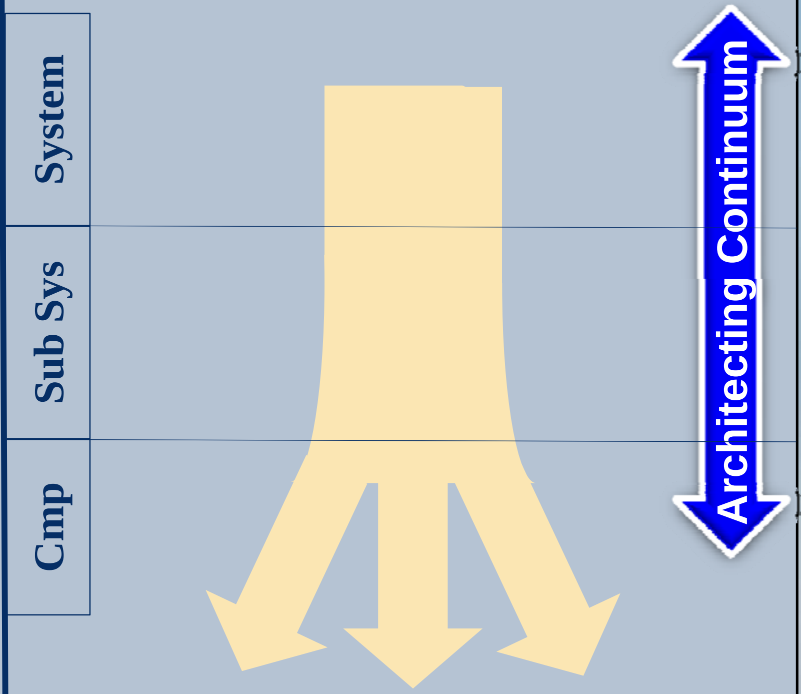
## Complex (Dynamic Complexity)

### Composable Design



## Complicated (Detailed Complexity)

### Decompositional Design



# Types of Problem Complexity

- **Technical**
  - Dynamic Operating Environment (e.g. underwater vehicle)
  - Interface Complexity (e.g. amount of information flow and physical characteristics)
  - Behavioral Complexity
  - Non-deterministic (emergent) Behaviors (e.g. unknown feedback loops)
  - Random / Unpredictable Data
  - Threats (e.g. security, environmental)
- **Programmatic**
  - Dynamic Stakeholder Environment (rapidly changing needs)
  - Threats (e.g. political, funding, regulatory)
  - Technology Changes (e.g. disruptive, end-of-life)
- **Mission**
  - Dynamic Mission Environment (e.g. red/blue force layouts)
  - Threats (e.g. adaptive adversarial capabilities)

# Complexity and SE

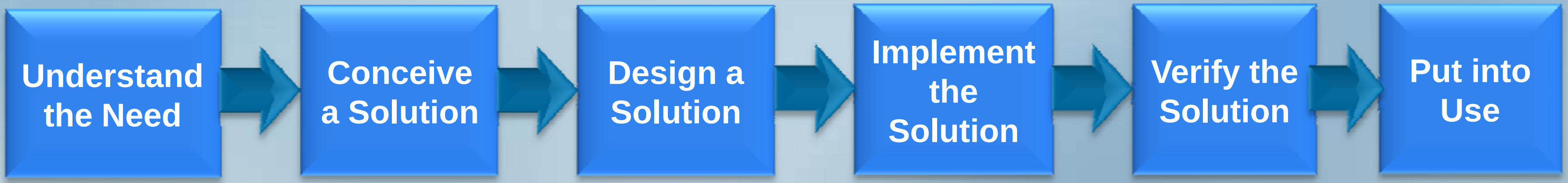
***“Historically, systems engineering has been successful in bringing order to the development of systems as they have become increasingly complicated. But there is a big difference between complicated and complex," he says. "Complicated is decomposable, which is what systems engineering is based on. Complex systems are no longer strictly decomposable, and systems engineering has to adapt.”***

***Aviation Week Interview with Jeff Wilcox: Is It Time To Revamp Systems Engineering ? (http://www.aviationweek.com/Article.aspx?id=/article-xml/AW_11_01_2010_p72-265541.xml)***

# What’s Stopping Us?

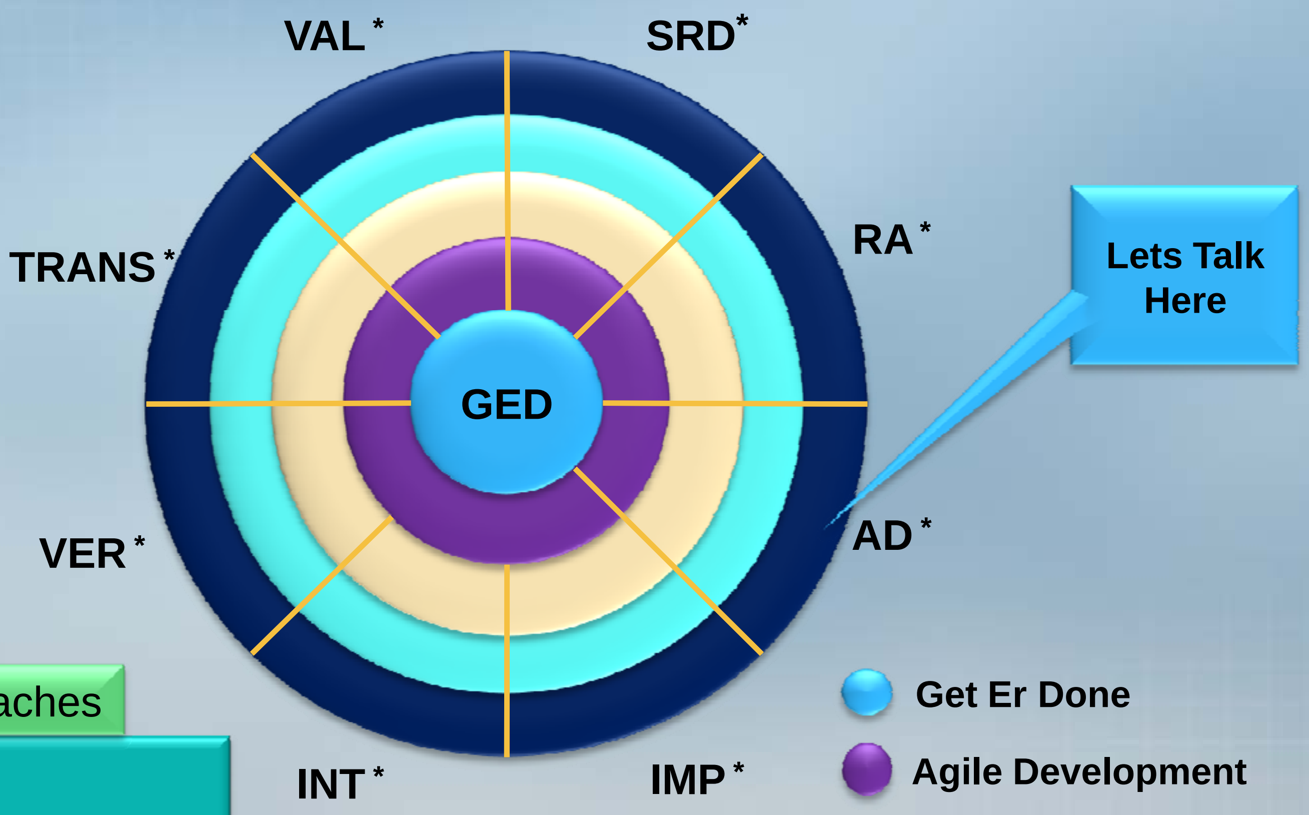
- **Why Can’t we Just Start Building?**
  - We learned years ago that when competing/warring constraints/needs and complexity are present, we can’t just throw caution to the wind and start building
    - Need for Oversight
    - Low Risk Tolerance (e.g. mission criticality, human lives – medical)
- **What May Happen if we Do?**
  - Poor interface alignment (interoperability issues)
  - Unanticipated behaviors
  - Missing functionality
  - Poor performance
  - Difficult to monitor and manage
  - Integration issues
  - Increased costs and/or schedule slips due to redesign and rework
- **Sometimes we Can!**

# How do we Solve a Problem?



Understand the Need → Conceive a Solution → Design a Solution → Implement the Solution → Verify the Solution → Put into Use

# Which Engineering Approach Should I Use?



● Get Er Done

● Agile Development

● Precedented but Complex

● Unprecedented

● Unprecedented with Untried Technology

## Engineering Approaches

- *Traditional - Formal Methods*
- *Agile*
- *Architected Agile*
- *Expedited SE*
- *Lean SE*
- *System Composition*
- *Rapid Prototype*
- *Hybrid Agile-Plan Driven*

* INCOSE SE Processes

# Architecting to Enable Implementation

- **Some prescription of the solution design is necessary to guide the implementation (Imp, Int, Ver)**
  - When there is enough you “launch” the implementation
- **To realize value the quickest, we need to launch ASAP**

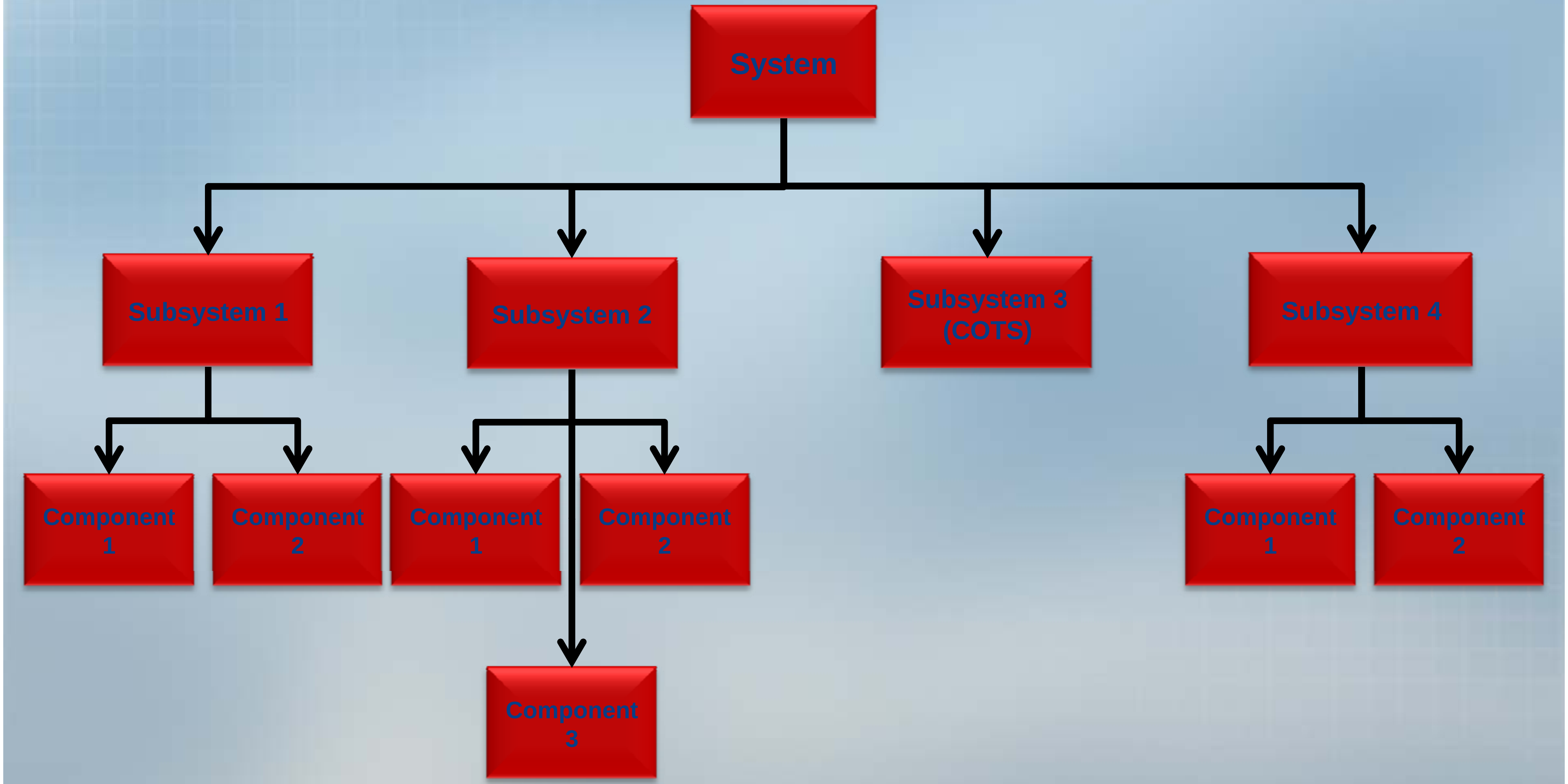
**So How do we Know?**

- **Rules of thumb for earliest launch**
  - Behaviors defined
  - Physical interfaces defined
  - Data consumed/produced defined
- **Be able to answer these questions**
  - How are the components connected?
  - What do the components do?
  - What data is exchanged?
  - What do the components produce?
  - How well must the components perform?
- **This is the subject so let’s move towards our objective**

# How Much is Enough - How do Architects Know/Decide?

- **Before we answer that, let's make sure we're all on the same page**
- **What are the basic practices of Architecting?**
  - Decomposition and iteration (used here for illustration)
    - Architecting, Synthesizing, and Evaluating
  - Composable design ( a discussion to have another day)
- **What are the core aspects of an architecture?**
  - Structure
  - Behavior
  - Data
  - Relationships
  - Procedures

# The Basic Practice of Architecting



# How do we Currently Architect a Solution?

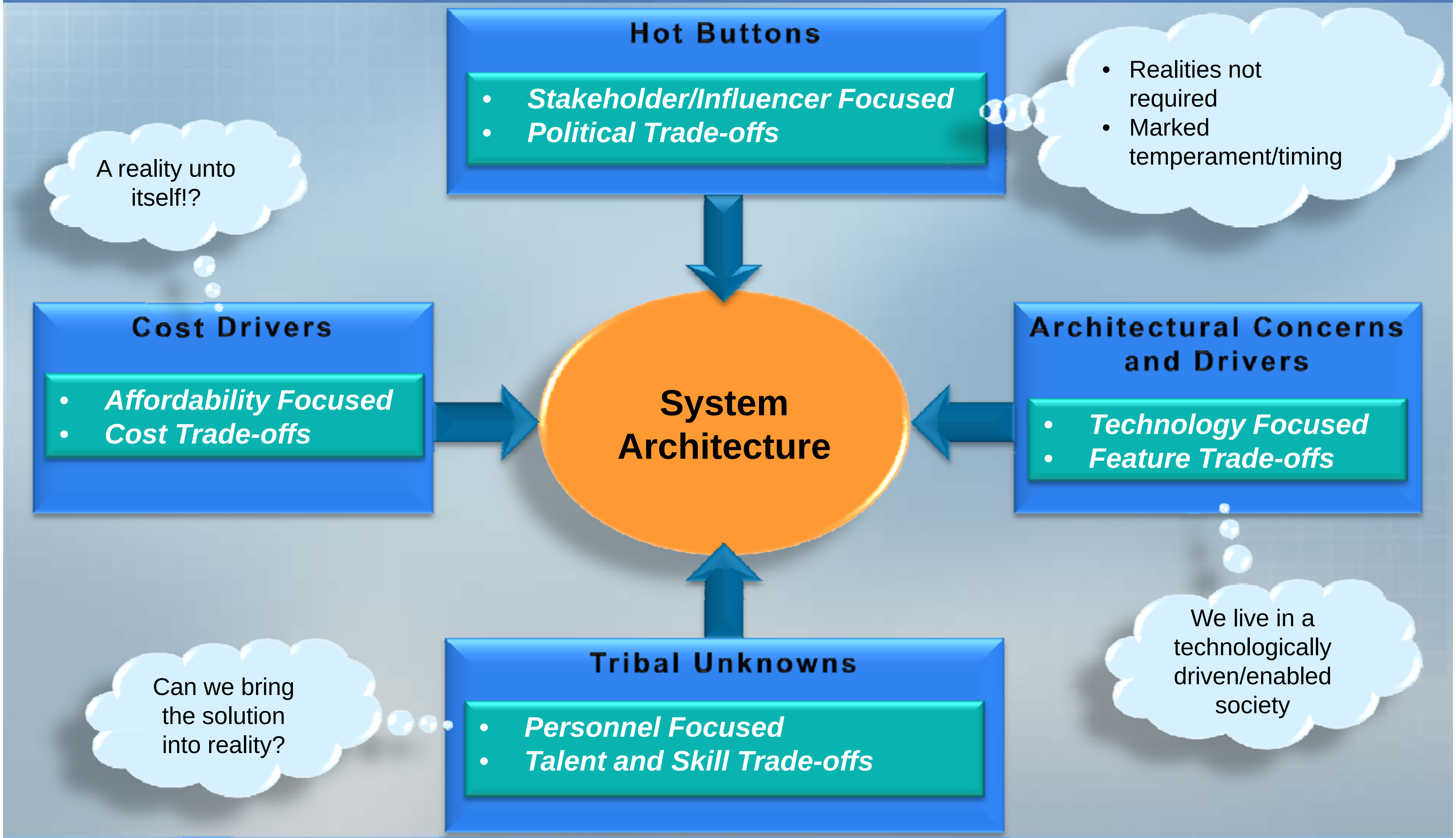
- **Primarily through applications of:**
  - Commonly accepted architectural principles
    - Partitioning (logical and physical)
      - At highest level possible
      - Based on different groupings (e.g. functional, physical, operational) to achieve the optimal design
      - High cohesion
      - Loose coupling
    - Apportionment
    - Equivalence
      - Reference Architectures
    - Understanding drivers and constraints
    - Minimize Interfaces between logical segments
  - Common quality attributes (see following slides)
  - Subjective context and domain specific sets of constraints applied by well experienced architects

# Architecture Quality Attributes

- **Suitability -** Solution meets it's need
- **Affordability** - When the value is worth the cost (time, money)
- **Flexibility -** Can change without making a mess
- **Extensibility** - Can add more
- **Scalability** - Can do more
- **Configurability** - Control and modify behavior without modifying HW/SW
- **Interoperability** - Works with other things
- **Producibility** - Can be produced
- **Deployability** - Can get it where it will be used
- **Reliability** - Doesn't fail too often
- **Availability** - Functions when needed
- **Maintainability** - The ease of maintenance

# A Simple Classical Aristotelian Classification Matrix For Architects

# Four Dimensions of Architectural Considerations



**Hot Buttons**

- *Stakeholder/Influencer Focused*
- *Political Trade-offs*

• Realities not required
• Marked temperament/timing
A reality unto itself!?

**Cost Drivers**

- *Affordability Focused*
- *Cost Trade-offs*

**System Architecture**

**Architectural Concerns and Drivers**

- *Technology Focused*
- *Feature Trade-offs*

We live in a technologically driven/enabled society
Can we bring the solution into reality?

**Tribal Unknowns**

- *Personnel Focused*
- *Talent and Skill Trade-offs*

# Solution Taxonomy

| | |
|---|---|
| SPACE | • Satellites<br>• Missiles<br>• Probes/Robots<br>• Manned Vehicles |
| AIR | • Manned Aircraft<br>• Unmanned Aircraft<br>• Airships |
| GND | • Fixed Platforms<br>• Mobile Platforms<br>• Transport Systems |
| SEA | • Manned Ships<br>• Unmanned Ships<br>• Senor Systems (e.g. radars) |
| UNDER SEA | • Manned Vehicles<br>• Unmanned Vehicles |
| IT | • Surveillance<br>• Information Systems<br>• Command and Control (C2)<br>• Services |

# FLAME – A Fast Launch Agile Implementation Enabler

| *Solution Type* / *Consideration* | Satellites | Missiles | Probes | Manned Vehicles | Robots | Manned Aircraft | Unmanned Aircraft | Airships | Fixed Platforms | Mobile Platforms | Transport | Manned Ships | Unmanned Ships | Sensors | Manned Vehicles | Unmanned Vehicles | Surveillance | Info Systems | C2 | Services |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Hot Buttons** | | | | | | | | | | | | | | | | | | | | |
| **Cost Drivers** | | | | | | | | | | | | | | | | | | | | |
| **Architectural Drivers** | | | | | | | | | | | | | | | | | | | | |
| **Tribal Unknowns** | | | | | | | | | | | | | | | | | | | | |

Content Not Releasable Yet

- SPACE
- AIR
- GND
- SEA
- UNDERSEA
- IT

# Next Step

- Goal is to develop a data set based on the knowledge of a community of experienced Architects

**Any Volunteers?**

# Questions and/or Comments?