25th anniversary
annual INCOSE
international symposium
Seattle, WA
July 13 - 16, 2015


# Differentiating System Architectures: Applying Architecture Measures

**Dr. Ron Carson, ESEP, Fellow, International Council on Systems Engineering** ronald.s.carson@gmail.com

International Council on Systems Engineering, Bellevue, WA July, 2015.

# Outline

- **What is an architecture?**
- **Why do we need to differentiate architectures?**
- **How do we measure an architecture?**
- **How do we determine goodness?**
- **When is one architecture better than another?**

# What is an Architecture?

- ISO/IEC/IEEE 42010:2011 – Systems and software engineering – Architecture description
  - **"Architecture** (system) – fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution"
  - **Elements:** Requirements, behavior, logical/physical elements, data, procedures
  - **Relationships:** Interfaces among internal and external Elements
  - **Principles:** Architecture rules, patterns, and overarching guidance

- Collectively these comprise an Integrated Product Architecture – the Enterprise Boeing project for model-based systems engineering (MBSE)

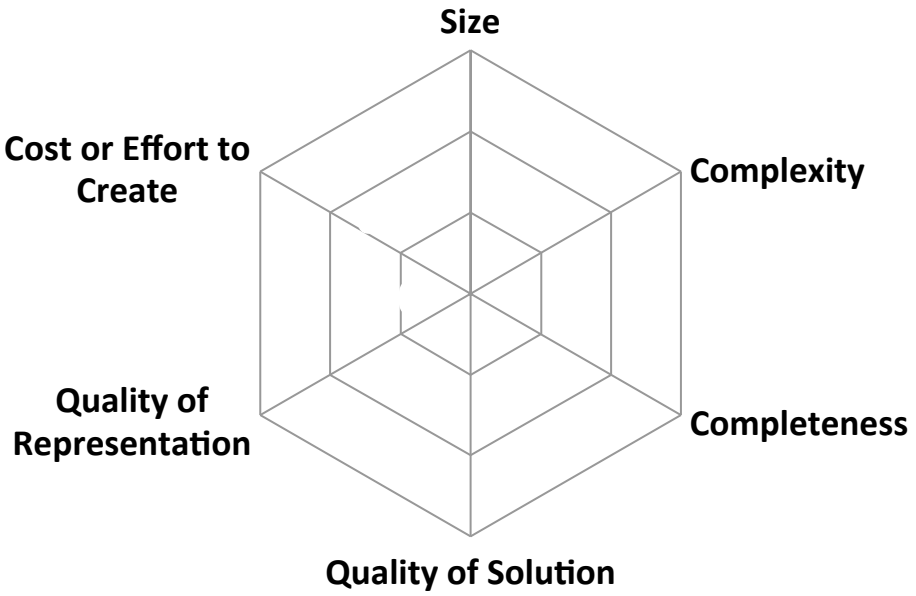
# Why do we need to differentiate architectures? Why is it important?

- *System performance may depend critically on architecture*
  - Weight (range, speed): discrete wiring (3 lbs/ft) vs. data bus (0.015 lbs/ft) ; integrated vs. distributed line-replaceable units; network performance vs. demand based on functional allocations
  - Fault tolerance and reliability: redundancy of critical systems
  - Fault isolation: distributed functions complicate fault isolation because of allocation of functionality (1 element vs. 20 elements for a specific functional failure)
- *There needs to be consistency of program (budget, activities) and architecture*
  - Different architectural patterns have different assumptions and consequences
  - Assumptions need to be validated and consistent with the requirements and program
- *The wrong architecture can doom the system and the program because of such technical and programmatic impacts*

# How do we measure an architecture?*

- *This is necessary if we want to determine "better than" or "good/bad"*

  - Technical measures
    - Size(t)
    - Complexity(t)
    - Completeness(t)
    - Quality(t)
  - Stability ≡ *f*(t)
    - Measuring stability is most important once a baseline is achieved
  - Cost/effort are measured by other processes, e.g., earned-value management
  - Measures in each category are outlined in the following slides




*Carson & Kohl, "New Opportunities for Architecture Measurement", Proceedings of INCOSE 2013

# Size, Complexity and Stability*

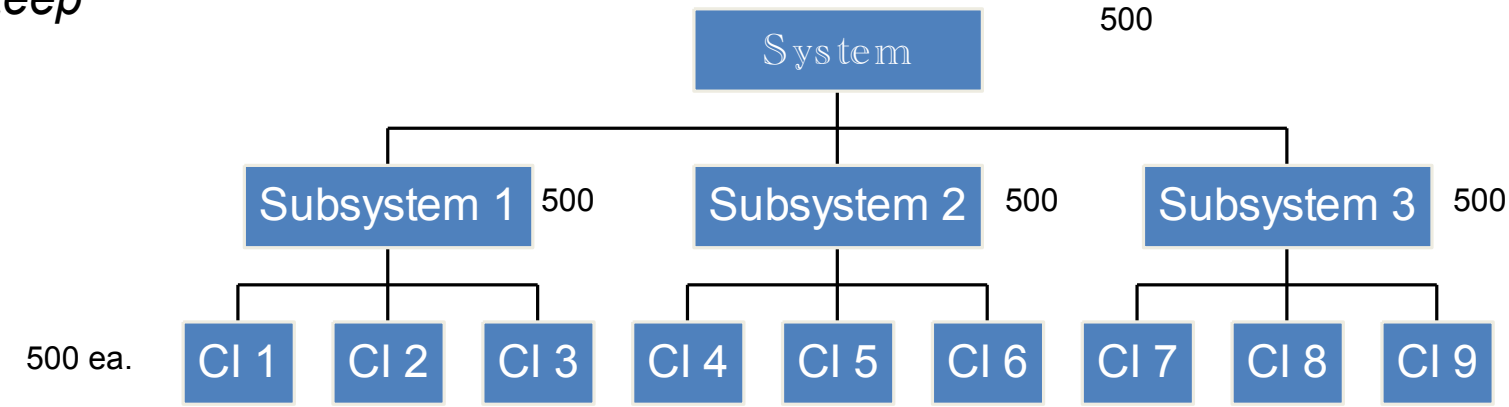
## Which measures can differentiate architectures?

| Proposed Measures | Definition/Description | Architecture Discriminator? |
|---|---|---|
| Number of elements | Count of constituent parts to be bought or developed, at each architectural level vs. time (stability) | Perhaps. The absolute number is generally not as significant as organization and relationships among elements (parts, subsystems, etc.) |
| Number of external interfaces | Count of logical and physical interfaces vs. time | Yes. Though external interfaces are largely controlled by context and ConOps independently of internal architecture, the number and details may be affected by architectural choices (e.g., hybrid vehicles require two energy/power interfaces) and associated infrastructure. |
| Number of external relationships | Count of organizational relationships (stakeholders) vs. time | Perhaps. Stakeholders may arise because of technology or architectural choices that require certification. |
| Number of requirements | Count of requirements at each architectural level vs. time | Yes. Requirements at each level depend on architectural choices; "steeper" architecture yields more levels and total requirements. |
| Number of internal interfaces | Count of logical and physical interfaces/ element vs. time | Yes. More interfaces require more interface management to address information exchanges and distributed functions. |
| Number of interactions | Transaction types or messages, frequency/ element vs. time | Yes. More interaction or data sharing across subsystems requires more analysis and test/ evaluation to understand behavior and failure effects. |
| Number of states | Count of number of defined states and/or modes vs. time | Perhaps. This may influence the number of requirements because of unique behaviors in defined states/modes. |

*Carson & Kohl, "New Opportunities for Architecture Measurement", Proceedings of INCOSE 2013

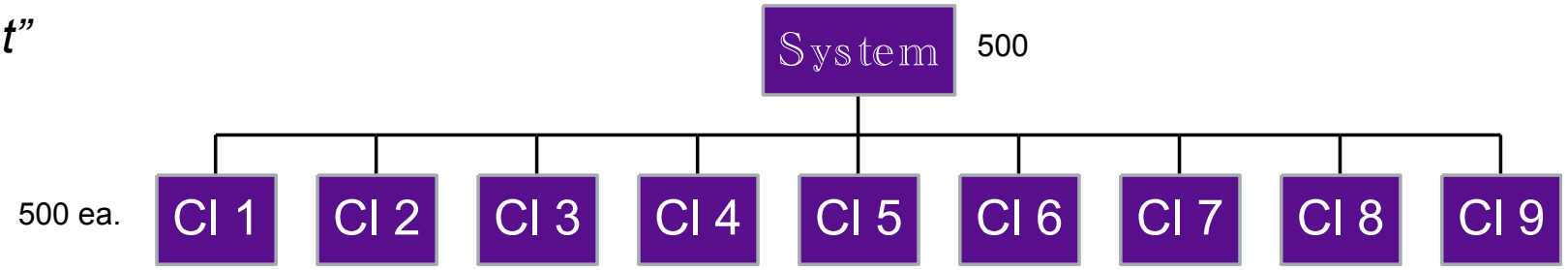
# Example: "Count of requirements at each architectural level" – Affects Requirements Management (RM)

- *A: "Steep"*




  - # Requirements = 500 + 3 x 500 + 9 x 500 = 6500

- *B. "Flat"*

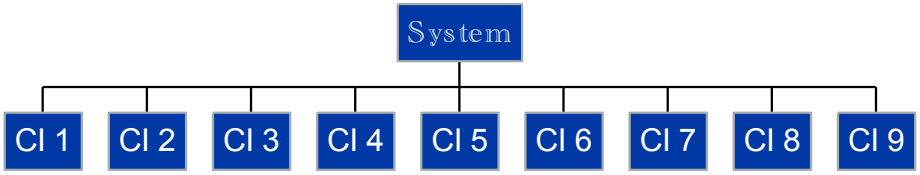


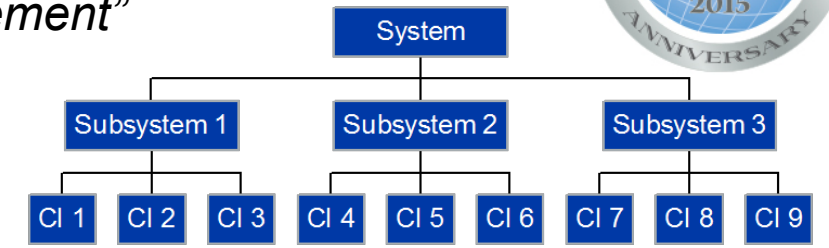

  - # Requirements = 500 + 9 x 500 = 5000
    - 1500 fewer requirements to manage and verify
    - RM workload is less; analysis to substantiate 1:9 fan-out is more complex

- *Each alternate has benefits and issues – no consistent preference*

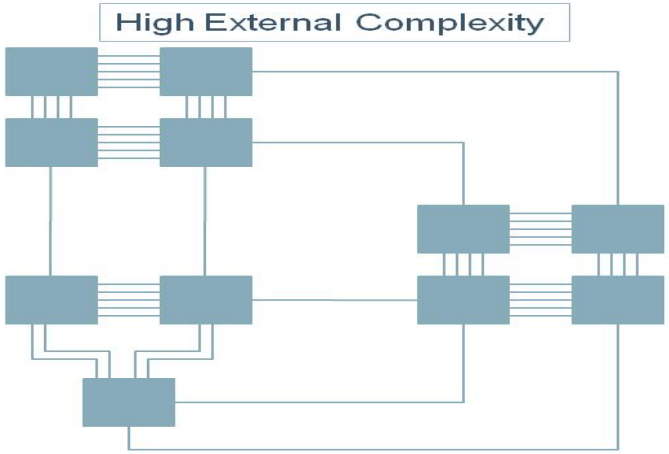
# Flatter or Steeper Hierarchy: Which is better?

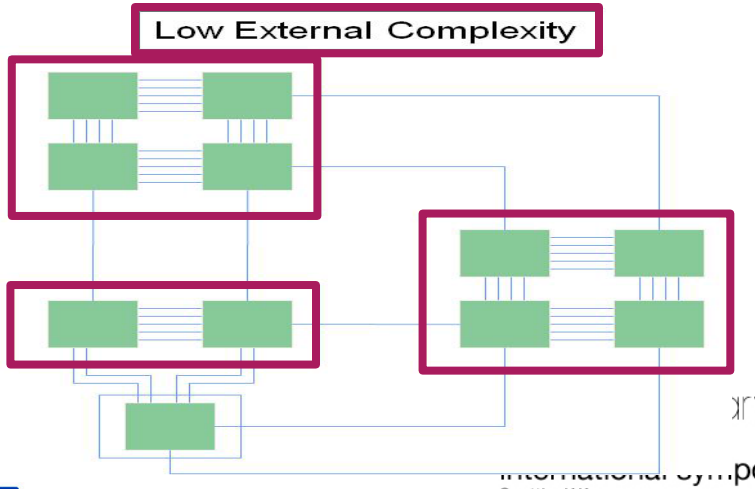
- *"Count of logical and physical interfaces/element"*




| Flatter | Steeper |
|---|---|
| Same # requirements at top and bottom | Same # requirements at top and bottom |
| More appropriate if intermediate specifications are unnecessary<br>• Internal development<br>• Off-the-shelf CIs (no specifications)<br>• Highly federated; fewer interfaces | Needed when intermediate specifications are necessary<br>• Outside organizations<br>• Phased development<br>• Subsystems with high internal complexity |




Carson & Kohl, "New Opportunities for Architecture Measurement", Proceedings of INCOSE 2013

# Completeness Measures*

## Which measures can differentiate architectures?

| Proposed Measures | Definition/Description | Architecture Discriminator? |
|---|---|---|
| Requirements addressed | Count of number of top-level requirements addressed by the architecture (traced to architecture element) | Yes; all architectures being considered should address all top-level requirements. |
| Artifacts produced | Count of number of architecture artifacts (e.g., viewpoints) produced vs. time | Not generally; the process should be the same regardless of architecture. |
| Artifacts expected | Count of number of architecture artifacts (e.g., viewpoints) needed vs. time | Not generally; the process should be the same regardless of architecture. |

*Carson & Kohl, "New Opportunities for Architecture Measurement", Proceedings of INCOSE 2013

# Quality Measures*

## Which measures can differentiate architectures?

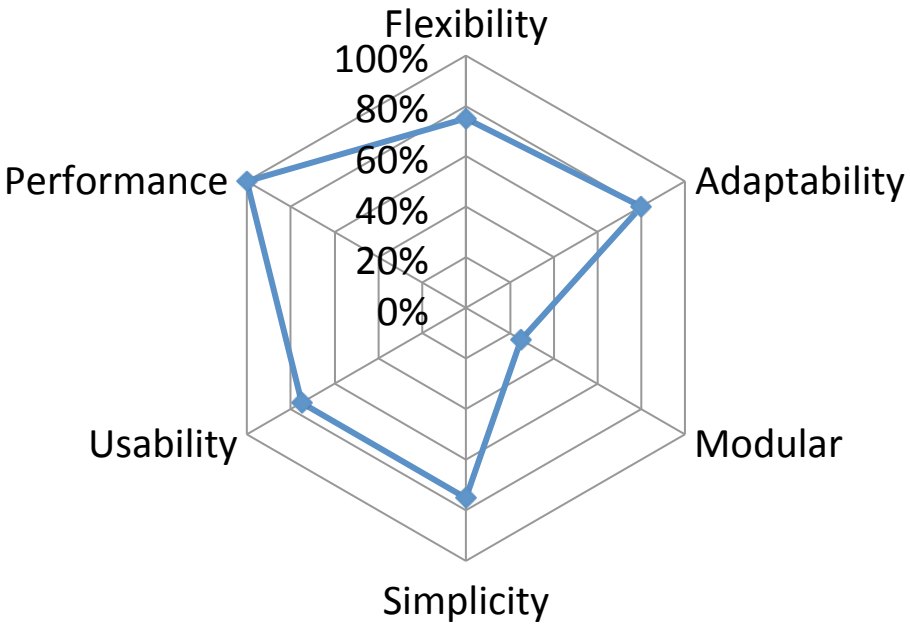
| Proposed Measures | Definition/Description | Architecture Discriminator? |
|---|---|---|
| Degree of requirements satisfaction | Count of number of requirements satisfied, normalized by the number of requirements | Yes. All architectures being considered should satisfy all top-level requirements. |
| Degree of Suitability | User/program-defined multivariate function of weighted suitability attributes (more on next slide) | Yes. Key discriminator when all solutions are compliant with requirements. |
| Degree of consistency of representation | User/program defined measure of adherence to internal standards or templates (data content and format), by artifact vs. time | Not generally. The process should be the same regardless of architecture. |
| Degree of standards compliance | Measures of adherence to external standards(data content and format), by artifact vs. time | Not generally. The process should be the same regardless of architecture. |

*Carson & Kohl, "New Opportunities for Architecture Measurement", Proceedings of INCOSE 2013

# Quality of Solution – Suitability

**Key attributes**

- ***Suitability*** **is a weighted utility function**
- **For criteria with thresholds (requirements)**
  - ***Threshold*** **requires "compliance" (binary)**
  - ***Suitability*** **portion is the degree of exceeding requirements**




| Attribute | % of Objective Value above Threshold | Weight | Weighted Value |
|---|---|---|---|
| Flexibility | 75% | 25% | 19% |
| Adaptability | 80% | 10% | 8% |
| Modular | 25% | 15% | 4% |
| Simplicity | 75% | 10% | 8% |
| Usability | 75% | 10% | 8% |
| Performance | 100% | 30% | 30% |
| **Total** | | **100%** | **77%** |

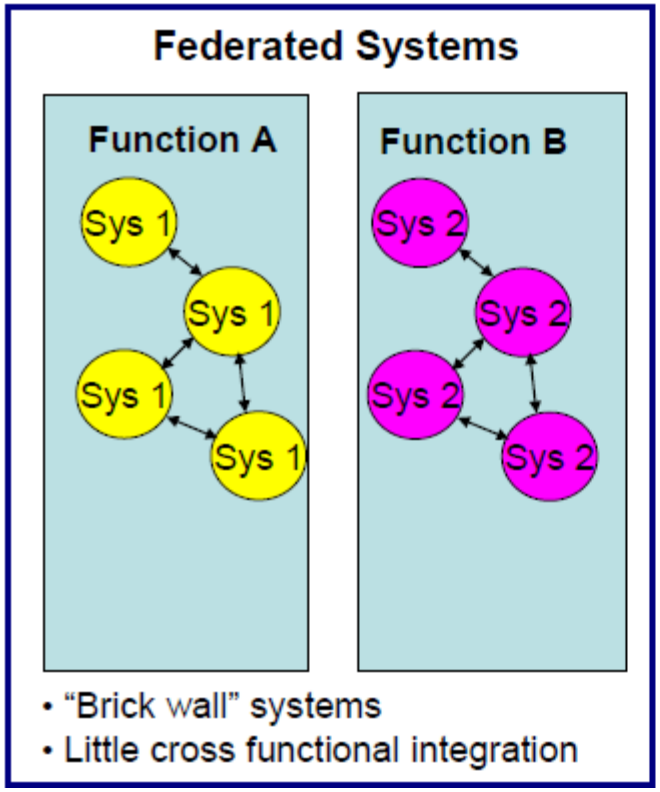
$$Suitability = \sum_i W_i \frac{|V_i - T_i|}{|O_i - T_i|}$$

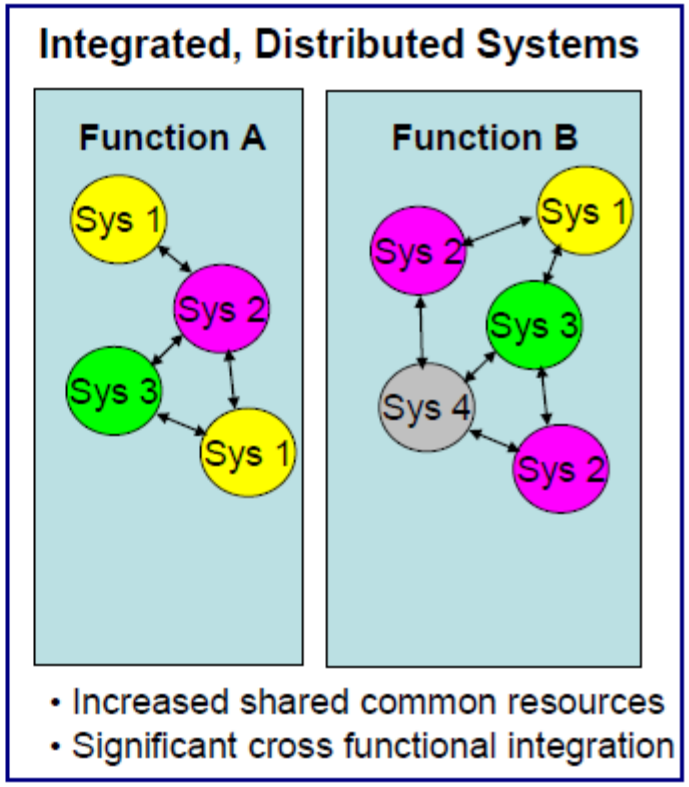
- If the only way to meet threshold performance is less modularity, then the better architecture is less modular (more integrated).



BOEING

# Federated (modular) vs. Integrated Architectures




- Architecture is a choice that depends on selection criteria
- "Modularity" simplifies integration
- More "cross-functional integration" requires better program integration

# Example: Performance and Modularity

- Performance (weight = 75%)
  - 550 nm range (threshold)
  - 600 nm (objective)
- Modularity (weight = 25%)
  - Federated is preferred (1.0) vs. fully Integrated (0.0)

$$Suitability = \sum_i W_i \frac{|V_i - T_i|}{|O_i - T_i|}$$

| Criterion | Value: A | W(V-T)/(O-T): A | Value: B | W(V-T)/(O-T): B |
|---|---|---|---|---|
| Performance | 550 nm | 0.75*0.0 = 0 | 600 nm | 0.75*1.0 = 0.75 |
| Modularity | Federated | 0.25*1.0 = 0.25 | Integrated | 0.25 * 0.0 = 0.0 |
| Suitability | | 0.25 | | 0.75 |

**Alternative B is preferred because (in this case) "Performance above threshold" is more valuable than "Modularity".**

**If weights are exchanged, Alternate A is preferred: "modularity" is more important than "range".**

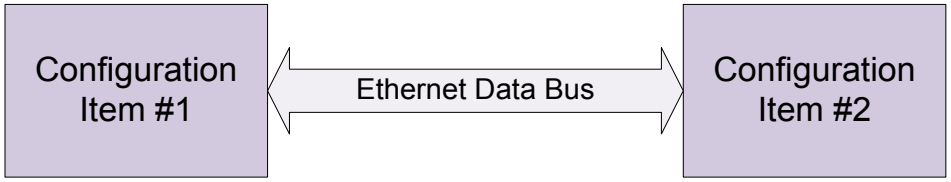
# Which is more complex?

- *"Transaction types or messages, frequency/element"*

## Physical complexity

1 signal/wire; 1000 wires

## Functional complexity




| Message Number | Message Name | Source | Destination |
|---|---|---|---|
| 001 | Signal 1 | 1 | 2 |
| 002 | Signal 2 | 1 | 2 |
| 003 | Signal 3 | 2 | 1 |
| 004 | Signal 4 | 2 | 1 |
| 005 | Signal 5 | 1 | 2 |

1 wire pair; 1000 signals

# Complexity Comparison

- *Notional information*

| Measure ↓ | Discrete Wires | Databus |
|---|---|---|
| Count of logical and physical interfaces/ element | 1000 | 1 (twisted, shielded pair) |
| Transaction types or messages, frequency/ element | 1 | 1000 |
| Total Complexity (∑) | 1001 | 1001 |
| Role affected | Physical / wire designer; Flight (payload/range) | Software engineer / bus message designer / integrator |
| Weight (22 AWG) | 3 lbs/ft | 0.015 lbs/ft |
| Reliability (failure rate) | ∑ λ (all wires) | λ (data bus) |
| Failure effects | Isolated and more predictable | Harder to predict & isolate; common mode failure effects |
| Life-cycle management | More difficult physical repair (e.g., aging wiring) or additions | More integration and regression testing for any changes (technology insertion or upgrade) |

*Complexity* depends on your role

# When is one architecture better than another?

- *An architecture is better when*
  - It meets all requirements (and another may not), and
  - It satisfies architecture selection criteria better than another, and
  - It has lower life-cycle costs (may be part of *Suitability*)
    - Lower non-recurring costs (development, technology upgrades, DMS)
    - Lower recurring costs (production, maintenance, spares)
- *The only absolutes are the requirements*
- *Architecture evaluation/selection is another system trade-off study that should be conducted during concept development and Analysis of Alternatives*