Architecture Decisions and Risk Management
Agenda

- Design Heuristics and Principles
- Evaluating an Architecture
- Evaluating an Architecture with DFSS
Design Heuristics and Principles
Common Challenges in Architecture

- Architecture is a long-term investment, while projects often have aggressive short-term goals.
- Projects that suffer from schedule compression implicitly prioritize functional requirements over the non-functional requirements.
  - We often under-spend on architecture to meet delivery commitments.
Common Challenges in Architecture

Reuse is a worthy goal, but cultural and organizational barriers keep it from achieving its potential

- Cultural barriers:
  - “Not-invented-here” syndrome
  - Perceived long-term risks of open-source

- Organizational barriers:
  - Need a product-line architecture approach built into the organizational structure to succeed
Common Pitfalls in Architecture

- **Performance**
  - Not understanding the relative priorities between performance and other quality attributes

- **Security**
  - Not being up to speed on modern techniques
  - Not having an architecture roadmap for a product
  - Not thinking from hacker’s perspective

- **Usability**
  - Allowing for some customer configuration with excessive parameterization
  - Lack of backwards compatibility

- **Availability**
  - Creating single points of failure
  - Artificially tight constraints

- **Modifiability**
  - Excessive reliance on third party products
  - Overshooting on reusability
  - Not thinking enough about reusability
  - Not investing in refactoring
  - Excessive reliance on experts instead of documentation

- **Testability**
  - Too many fault messages results in information overload
  - Not considering the cost of defect characterization
What Do Good Architects Do?

Good architects can satisfy the functional requirements while meeting non-functional requirements (i.e. quality attributes). How do they achieve that?

By applying design heuristics and principles such as:

<table>
<thead>
<tr>
<th>Separation</th>
<th>Resource sharing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolates a portion of a system’s functionality into a component</td>
<td>Encapsulation of either data or services</td>
</tr>
<tr>
<td>Abstraction</td>
<td>Sharing among multiple independent consumers</td>
</tr>
<tr>
<td>Is the operation of creating a virtual machine and hiding its underlying implementation</td>
<td>Replication</td>
</tr>
<tr>
<td>Compression :</td>
<td>Operation of replicating a component.</td>
</tr>
<tr>
<td>Removing layers or interfaces (i.e. the opposite of Separation)</td>
<td>Decomposition</td>
</tr>
<tr>
<td></td>
<td>Separating a large system into smaller components</td>
</tr>
<tr>
<td></td>
<td>Part-Whole : each subcomponent represents non overlapping portions of the functionality</td>
</tr>
<tr>
<td></td>
<td>Is-a : Each subcomponent represents a specialization of its parent’s functionality</td>
</tr>
</tbody>
</table>
What Do Good Architects Use?

- The SEI has started collecting a catalog of proven solutions that help address the quality attributes – they call these proven approaches “tactics”.

**Tactics** = Fundamental design decisions employed to achieve the quality attributes.
Good Architects Never Stop Learning

- Good architects are continuously learning
- Good architects stay on top of industry trends
- Good architects balance the risk of change against the value of the new ideas they may apply to a product
- Good architects maintain the architecture documentation to help the next generation of architects to learn
- Good architects strive to optimize the system behavior, not the product behavior
  - Or, similarly, optimize the product more than the component…
Evaluating an Architecture
Evaluating an Architecture

- How do you know whether the application of the tactics will really pull the critical parameters within spec?
  - Create executable models of your architecture specification
  - Leverage an iterative development lifecycle
  - Develop prototypes that implement parts of the architecture

- Somehow, you’ll want to get some measurable feedback on the effectiveness of your architecture…
Evaluating with Iterative Development

**Waterfall Lifecycle**
- Breadth-First Delivery
- Phase-based Development
- End-of-phase Handoffs

**Iterative Lifecycle**
- Depth-First Delivery
- Feature-Set-based Development
- Full-lifecycle Collaboration

- **Reqs/Arch**
  - Models, Diagrams
  - Modules, Classes
  - Libraries, Objects
  - Binaries, Executables

- **Design**
  - Features
  - Set

- **Code**
  - Features
  - Set

- **Build**
  - Features
  - Set

- **Test**
  - Features
  - Set
Evaluating with Prototyping

Evolutionary Design
- An initial prototype is produced and refined
- Iterate through a number of stages to the final system

Outline Requirements & CPs

Create Prototyping

Evolutionary Design

Delivered System

Design Attributes

Throw-away prototyping
- A prototype is produced to help discover attributes associated with a concept
- The results are discarded when design selection is completed.
- Resist management's tendency to use/release the throw away prototype

Throw-away prototype after design selection

For Analyze, Use Throw-away techniques

Never change a Throw-away prototype to an Evolutionary Design
Prototyping Techniques & Tools

- Prototypes are created as models presented in a format that is immediately recognizable to the users.
- Normally a mock-up is a user interface model that may or may not be skeletal in terms of functional capability.
- Prototypes are created through the use of specification languages that are directly machine-interpretable. (Flash)
- Prototypes are developed by using a high-level language that is application oriented;
Evaluating using Risk Analysis
What is a Risk

- According to the Defense Acquisition University:
  "Risk is a measure of the potential inability to achieve overall program objectives within defined cost, schedule and technical constraints"

- ISO Defines Risk as the:
  "combination of the probability of an event and its consequence"
The Risk Management Process

- Risk Management is a continuous closed loop process that captures new risks as they emerge, tracks the status of already identified risks, retires risks through successful actions, or realizes risks through unsuccessful actions.

- 5 steps - derived from a process developed by the Defense Acquisition University, and other sources such as Carnegie Mellon’s Software Engineering Institute and the Open Systems Initiative.
  1. Plan
  2. Identify Risks
  3. Assess and Prioritize Risks
  4. Develop and Implement Risk Handling Approaches
  5. Track and Report

Risk Management is an ongoing process, not an event.
Evaluating using Risk Analysis

Risk is …the Possibility Of Suffering A Loss, the uncertainty of attaining a future goal – it hasn’t happened yet.

Every Risk has Two Elements
  • Probability: the chance that an event will occur. If it’s a sure thing, then it’s a problem (not a risk)
  • Consequence: A negative impact on Cost, Schedule, Performance or a combination of all three…”then”

Probability: The likelihood that an event will occur ( <100% ).

Undesirable Consequence: The negative impact if the risk occurs: Cost will increase, Schedule will be delayed or Performance will be degraded. AKA ‘Impact Severity’
Risk Assessment

- Risk Factor is an evaluation of a Risk’s probability of occurrence (Pf) and severity of consequence (Cf) to determine its seriousness
  \[(Pf \times Cf) = R_f\]

- Risk Factor is used to prioritize the list of risks
  - High, Med, Low

RISK FACTOR (Rf)

| HIGH (Rf > 0.50) | Likely to cause significant serious disruption of schedule, increase in cost, or degradation of performance even with special contractor emphasis and close government monitoring. |
| MODERATE (0.25 ≤ Rf ≤ 0.5) | Can potentially cause some disruption of schedule, increase in cost, or degradation of performance. However, special contractor emphasis and close government monitoring will probably be able to overcome difficulties. |
| LOW (Rf < 0.25) | Has little potential to cause disruption of schedule, increase in cost, or disruption of performance. Normal contractor effort and normal government monitoring will probably be able to overcome difficulties. |
Rf Action Guidelines Table

<table>
<thead>
<tr>
<th>#</th>
<th>Pf</th>
<th>Cf</th>
<th>Rf</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Do Nothing</td>
</tr>
<tr>
<td>2</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
<td>Do Nothing</td>
</tr>
<tr>
<td>3</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>Reduce likelihood or establish contingency</td>
</tr>
<tr>
<td>4</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Monitor, take action if needed</td>
</tr>
<tr>
<td>5</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Take action if needed</td>
</tr>
<tr>
<td>6</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Take action to reduce likelihood</td>
</tr>
<tr>
<td>7</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>Develop cost effective mitigation plan</td>
</tr>
<tr>
<td>8</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Take action</td>
</tr>
<tr>
<td>9</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Take action</td>
</tr>
</tbody>
</table>

![Probability and Impact Matrix](chart.png)
Evaluating using Risk Analysis

- Risk analysis should clarify the possible outcomes and assign values to the probabilities and impacts

<table>
<thead>
<tr>
<th>Risk area</th>
<th>Risk</th>
<th>Prob†</th>
<th>Impact†</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rqmt Stability</td>
<td>If requirements change, then architecture design will slip; fixed need date prevents meeting users’ need</td>
<td>High</td>
<td>Med</td>
<td>Med</td>
</tr>
<tr>
<td>Design Perform</td>
<td>If throughput rqmts are not achievable with COTS S/W; then schedule will slip</td>
<td>Low</td>
<td>Med</td>
<td>Med</td>
</tr>
<tr>
<td>Rqmt Scale</td>
<td>If effort is larger than expected, then will not be able to staff causing extensive slips</td>
<td>Med</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Risk list

Ranked risk list

†Prob – High: 1>P>0.7, Med: 0.7>P>0.4, Low: 0.4>P>0.1, None: 0.1>P
Impact – High: >$1M or slip>3 months, Med: ...
Using Tactics To Reduce Risk

Tactics

Quality Attribute Analysis

• Performance
• Modifiability
• Availability
• Security
• Testability
• Usability

Architecture Risk Analysis

Critical Parameters

Gaps

Technical Risks

Simulation/Modeling/Prototyping Activities

Lower Risks

Modeled CP

DFSS

Software Architecture Specification

Software Development

• ML Designer
• Monte Carlo Schedule Analysis
• MS Threat Modeling Tool
• Prototyping (e.g. RUP:Initiation)
• Customer Demos
Kinds of Tactics

- **Performance**
  - Resource Demand
  - Resource Management
  - Resource Arbitration

- **Security**
  - Resisting Attacks
  - Detecting Attacks
  - Recovering from an Attack

- **Usability**
  - Separate User Interface
  - Support User Initiative
  - Support System Initiative

- **Availability**
  - Fault Detection
  - Recovery: Preparation and Repair
  - Recovery: Reintroduction
  - Prevention

- **Modifiability**
  - Localize Changes
  - Prevention of Ripple Effect
  - Defer Binding Time

- **Testability**
  - Manage Input/Output
  - Internal Monitoring
Examples of Tactics

- **Performance**
  - FIFO
  - Leaky Bucket

- **Security**
  - Trusted Computing Base
  - Authenticate Users
  - Authorize Users

- **Usability**
  - Parameter Hiding
  - Undo
  - Clearly Marked Exits

- **Availability**
  - Trusted Computing Base
  - FIFO
  - Leaky Bucket
  - Garbage Collection

- **Modifiability**
  - Abstract Common Services
  - Anticipate Expected Changes
  - Runtime Registration

- **Testability**
  - Record/Playback
  - Built-in Monitors
Let’s say we’re developing an architecture for a phone.

The gap analysis of the quality attributes revealed that the security response of the phone is a critical parameter
- Desired Response: Keep Mean-Time-To-Detect within 5 minutes (i.e. to detect an attack)

What design decisions can we make to achieve this?
- Leverage a tactic called “Trusted Computing Base”

Design decision:
- Choose an architectural boundary, within which, the data is trusted.
- Note that this decision could result in a degradation of performance response. Tactics will often support one attribute, but at the expense of another.
Let’s say we’re developing an architecture for an infrastructure product that handles call processing.

The gap analysis of the quality attributes revealed that the performance response is a critical parameter – Desired Response: Maintain response/turnaround time of 5 ms for 95% of inbound events.

What design decisions can we make to achieve this?
– Leverage a tactic called “First In, First Out (FIFO)”

Design decision:
– Queue events for service in First-in, first-served manner.
– Note that this decision could result in a degradation of availability, since an individual user that gets buried in the queue will perceive a fault…
Example: Anticipate Expected Changes

- The gap analysis of the quality attributes revealed that the **modifiability** response is a critical parameter.
  - Desired Response: Changing the audio transport model from circuit-switch to packet-based routing some time in the future should cost no more than 5 SM (Staff Months).

- What design decisions can we make to achieve this?
  - Leverage tactics called “Anticipate Expected Changes”

- Design decision:
  - Keep the transport-specific details isolated in an audio programming layer, distinct from the call processing control logic.
Example: Parameter Hiding

- The gap analysis of the quality attributes revealed that the **usability** response is a critical parameter
  - Desired Response: Keep the number of configurable parameters below a threshold value (e.g. 20 fields).

- What design decisions can we make to achieve this?
  - Leverage a tactic called “Parameter Hiding”

- Design decision:
  - Make a wider set of configurable parameters available to a select user base (e.g. beta/field personnel) for performance tuning, but not to the general users.
Risk Assessment Summary

- Disciplined risk management can be applied to all business aspects
- Use the process correctly.
  - Don’t skip steps in an ill-founded attempt to speed up the process.
- Train key business personnel as coaches/facilitators
- Hold regular risk reviews
  - Work the process through the program, not just at the Risk Assessment Session.
- Manage available information to:
  - Use what we know
  - Understand what we don’t know
  - Minimize what we don’t know we don’t know!
- Share lessons learned across businesses
Evaluating Architecture Using DFSS Techniques
1. Quality Tradeoffs
   - Changing an existing product’s architecture is risky…

2. Design* Tradeoffs
   - Prioritization Matrix
   - Pugh Matrix
   - Monte Carlo analysis

3. Economic Tradeoffs
   - Cost Benefit Analysis Model (CBAM)
   - Developing architecture roadmaps for a product

* In this context, “Design” refers to making choices when architecting…
Quality Tradeoffs

- Changing a legacy architecture is one of the riskiest development activities that a team can undertake.

- The keys are to:
  - Keep the effort under control by making changes in small steps
  - Understand the risks that derive from how well you know your architecture, requirements, and current implementation:
    - Risk of insufficient documentation of legacy products
    - Risk of insufficient domain experience/understanding
    - Risk of insufficient architecture skills on team
  - Make the investment that is needed (i.e. long term architecture phase)
The ability to make decisions is a vital skill, both in our personal and our professional lives.

- What am I going to have for lunch?
- What kind of computer should I buy next?
- What display and keypad should Motorola use to meet a customer’s need for a robust design?

To make a decision you need to know:

- What are the possible choices?
- What criteria distinguish the choices?

If only one criteria is important, the decision is easy:

- I want the quickest lunch.
- I want the cheapest computer.
List Architecture Needs

<table>
<thead>
<tr>
<th>Architecture Needs</th>
<th>QA</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>User needs a response to an inbound call request within 400 ms.</td>
<td>Performance</td>
<td>Use Case/AR</td>
</tr>
<tr>
<td>Denial of service attacks must be identified and thwarted in 95% of the cases.</td>
<td>Security</td>
<td>Use Case/AR</td>
</tr>
<tr>
<td>The system must deliver a mean-time-to-failure of 24 hours or better.</td>
<td>Availability</td>
<td>Use Case/AR</td>
</tr>
<tr>
<td>The cost to add new system objects to the NM interface must be less than 5 SM.</td>
<td>Modifiability</td>
<td>Use Case/AR</td>
</tr>
<tr>
<td>User must be able to configure a new site subsystem within 5 commands.</td>
<td>Usability</td>
<td>Use Case/AR</td>
</tr>
<tr>
<td>System will need to be able to add Phase 2 security standards within 18 months.</td>
<td>Security</td>
<td>Change/Growth</td>
</tr>
<tr>
<td>System must remain operational with traffic bursts of up to 4M users per hour.</td>
<td>Performance</td>
<td>Extreme/Stress</td>
</tr>
</tbody>
</table>
## Prioritize Needs

<table>
<thead>
<tr>
<th>Architecture Needs</th>
<th>QA</th>
<th>Type</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>User needs a response to an inbound call request within 400 ms.</td>
<td>Performance</td>
<td>Use Case/AR</td>
<td>90</td>
</tr>
<tr>
<td>Denial of service attacks must be identified and thwarted in 95% of the cases.</td>
<td>Security</td>
<td>Use Case/AR</td>
<td>60</td>
</tr>
<tr>
<td>The system must deliver a mean-time-to-failure of 24 hours or better.</td>
<td>Availability</td>
<td>Use Case/AR</td>
<td>70</td>
</tr>
<tr>
<td>The cost to add new system objects to the NM interface must be less than 5 SM.</td>
<td>Modifiability</td>
<td>Use Case/AR</td>
<td>30</td>
</tr>
<tr>
<td>User must be able to configure a new site subsystem within 5 commands.</td>
<td>Usability</td>
<td>Use Case/AR</td>
<td>40</td>
</tr>
<tr>
<td>System will need to be able to add Phase 2 security standards within 18 months.</td>
<td>Security</td>
<td>Change/ Growth</td>
<td>10</td>
</tr>
<tr>
<td>System must remain operational with traffic bursts of up to 4M users per hour.</td>
<td>Performance</td>
<td>Extreme/ Stress</td>
<td>80</td>
</tr>
</tbody>
</table>
Introduction to Design Tradeoff Analysis

- Usually, there are many choices and many selection criteria. To select a restaurant for lunch, you might consider:
  * distance
  * price
  * variety of entrees
  * cuisine
  * service
  * previous experience

- For a computer, price would seldom be the only criteria:
  * display
  * package deal
  * amount of RAM
  * intended use
  * size
  * mobile technology

- Note that selection criteria seldom point to a perfect choice, because the perfect fit may not exist. To get the perfect cuisine, for example, you might have to drive further.
Design Tradeoff Analysis Tools & Methods

- Common steps in Design Tradeoff Tools
  1. List the choices, options, or alternatives.
  2. List the selection criteria (e.g. critical parameters).
  3. Score each choice against each criteria.
  4. Score the choices against each other.
  5. Document which choice scores the highest.

- Design Tradeoff Tools & Methods to be presented:
  - Prioritization Matrix
  - Pugh Matrix
  - Cost Benefit Analysis Method (CBAM)

Design Tradeoff tools help you select & document the best choice.
DFSS Tradeoff Tools

What do we need in a tradeoff tool to help us make architecture decisions?
- Analysis support to help us sort through the many choices and conflicting criteria
- A disciplined approach that helps remove emotion and politics from the decision-making process

Consider these examples of architecture decision-making:
1. Choosing between a few alternatives when designing the primary components (architecture elements) in the product architecture
2. Selecting one among many 3rd party products
3. Choosing between two algorithms for a queuing model
4. Creating an architecture roadmap for a product that prioritizes a set of architecture changes by economic value
Tradeoff Example 1

- Choosing between a few design alternatives:
  - Captured as an open issue
    - with a few alternative solutions
    - with pros and cons for each alternative

- Use a Prioritization Matrix when you need to:
  - accommodate both qualitative and quantitative data
  - apply criteria weighting
Prioritization Matrix

- Prioritization Matrix = a weighted, subjective analysis
  - Multiple alternatives
  - Success criteria
  - Subjective weightings of the criteria

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Criteria Weightings</th>
<th>Alternatives</th>
<th>Positive correlation of alternative to criteria</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Prioritization Matrix

1. List the alternatives
2. Establish the criteria
3. Weight the criteria (on a 1-10 scale)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Performance</th>
<th>Development Cost</th>
<th>Future Modifiability</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criteria Weightings</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Alternatives</td>
<td>Positive correlation of alternative to criteria</td>
<td>Score</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a new mobility handler</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extend the current mobility tracker</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use classes in ACE framework</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reuse the iDEN mobility handler</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Note:** The table above shows the criteria weightings and alternatives along with their respective scores. The criteria include Performance, Development Cost, Future Modifiability, and Security.

---

**Raytheon**

Integrated Defense Systems
Prioritization Matrix

4. Rate each alternative against each criteria (1-10 scale)
5. Rank the alternatives by score
6. If no clear winner emerges, consider adding criteria

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Performance</th>
<th>Development Cost</th>
<th>Future Modifiability</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criteria Weightings</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Alternatives</td>
<td>Positive correlation of alternative to criteria</td>
<td>Score</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a new mobility handler</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Extend the current mobility tracker</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Use classes in ACE framework</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Reuse the iDEN mobility handler</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
DFSS Tradeoff Example 2

- Selecting one among many 3rd party products:
  - Captured as a list of 3rd party offerings
    - with costs and benefits for each offering

- Use a Pugh Matrix when you need to:
  - do a quick qualitative assessment
  - determine if there is an obvious winning concept
  - get a quick feel for concept comparison
  - down-select from many concepts to the most promising few
Summary: compares and selects best ideas & concepts using a simple system of “better than”, “worse than”, and “same” scoring. Identifies best features from each concept and creates hybridized solutions.

<table>
<thead>
<tr>
<th>Criteria / Concept</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of achieving 105-125 DbA</td>
<td>S</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>Ease of achieving 2000-5000Hz</td>
<td>S</td>
<td>S</td>
<td>N</td>
<td>S</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Resistance to corrosion, erosion &amp; water</td>
<td>-</td>
<td>-</td>
<td>O</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>Resistance to vibration, shock &amp; acceleration</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistance to temperature</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response time</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complexity: number of stages</td>
<td>U</td>
<td>-</td>
<td>+</td>
<td>E</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Power consumption</td>
<td>M</td>
<td>-</td>
<td>V</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ease of maintenance</td>
<td>S</td>
<td>+</td>
<td>A</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Weight</td>
<td>-</td>
<td>-</td>
<td>L</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Size</td>
<td>-</td>
<td>-</td>
<td>U</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Number of parts</td>
<td>S</td>
<td>S</td>
<td>A</td>
<td>+</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Life in service</td>
<td>S</td>
<td>-</td>
<td>T</td>
<td>+</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Manufacturing cost</td>
<td>-</td>
<td>S</td>
<td>E</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ease of installation</td>
<td>S</td>
<td>S</td>
<td>D</td>
<td>S</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Shelf life</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

\[ \Sigma^+ \]
\[ \Sigma^- \]
\[ \Sigma^S \]

Benefit:
- Identifies the superior concepts
- Aids in hybridization of concepts by identifying strengths & weakness of each concept
1. Prepare all concepts to similar level of detail
2. Define concept evaluation criteria
3. Construct evaluation matrix

4. Select datum to evaluate concepts against
5. Compare each concept against the datum

6. Analyze concepts and create new superior hybrid concepts
7. Eliminate weak concepts from evaluation matrix

8. Confirm superior concept
Select the Datum – Step 1

1. Choose one of the alternatives to be the straw man, or “datum”
   - Rather than comparing each concept against the others, all concepts are compared against the datum
     - The datum “takes the heat” thus rendering a focus for all other concepts to beat
   - Datum for Initial Evaluation
     - Benchmarked best-in-class design in the context of your product requirements and competitive environment
     - If no best-in-class design exists, the datum is the concept that the team believes is the strongest among the alternatives
   - Datum for Subsequent Evaluation Iterations
     - Select the strongest concept that is present in the matrix (either the initial datum or a concept that has emerged as stronger)
Select the Datum – Step 2 & 3

2. Systematically compare each concept against the datum
   - For each evaluation criteria, rate the concept as:
     + Better than, less than, less prone to, easier than, etc., relative to the datum
     - Worst than, more expensive than, more complex than, more prone to than, etc., relative to the datum
     S Same as the datum

3. For each concept, sum the +, - and s ratings
**Compare Each Concept Against The Datum**

<table>
<thead>
<tr>
<th>Evaluation Criteria #1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>N</th>
<th>Product Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s</td>
<td>-</td>
<td>+</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation Criteria #2</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation Criteria #3</td>
<td>D</td>
<td>+</td>
<td>+</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>s</td>
<td>-</td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>T</td>
<td>-</td>
<td>-</td>
<td>s</td>
<td>-</td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>U</td>
</tr>
<tr>
<td>Evaluation Criteria #N</td>
<td>M</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td></td>
<td>U</td>
</tr>
<tr>
<td>∑+</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>∑-</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>∑s</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DFSS Tradeoff Example 3

- Choosing between two algorithms for a queuing model:
  - Two alternative approaches to handling inter-process communications within the product’s concurrency model
    - each algorithm has different advantages and weaknesses
    - there is variability on the traffic patterns
    - the requirements are stated as a range of acceptable utilization %

- Use Monte Carlo Analysis* when you need to:
  - handle variability in the inputs
  - do sophisticated statistical modeling
  - determine whether the results fall within acceptable ranges

* [Refer to “Applying DFSS to Software and Hardware Systems, Maass & McNair, Prentice-Hall, 2009]
Summary of Architecture Evaluation Techniques

- A comprehensive set of scenarios was presented to assist in a complete evaluation of the architecture.

- Methods were defined to help measure the architecture so an evaluation could be performed. Using capability flow up, flow down along with simulation or models provides a very powerful metrics based evaluation technique.
Patricia McNair  
IDS Deputy Director of Six Sigma  
Raytheon Co.  
978-858-5456  
Email: Patricia.D.Mcnair@raytheon.com

Liz Markewicz  
Risk Manager  
Raytheon Co.  
978.470.9923  
Email:mmarkewl@raytheon.com