• Introduction
• Rationale
• Attributes
• Use Cases
• Next Steps
JPL is part of NASA and Caltech

- Federally-funded (NASA-owned) Research and Development Center (FFRDC)
- University Operated (Caltech)
- $1.73B Business Base
- 5,000 Employees
- Founded in the 1930s
- Focus on robotic missions for solar system exploration
JPL Spacecraft and Instruments
Across the Solar System

- 50+ Years of Space Exploration
- 24 missions and 12 instruments active
- Current programs and projects include:
  - **Planetary Missions**
    - Mars Rovers and Orbiters (MER, MRO, MO, MSL)
    - Outer Planets (Voyager 1&2, Cassini, Juno)
    - Other Bodies (Dawn, EPOXI)
  - **Earth and Moon Observations**
    - Climate (ACRIMSAT, CloudSat, MLS)
    - Earth Observations (ASTER, AIRS, MISR)
    - Oceans (Aquarius, GRACE, Jason, OSTM, QuickScat)
      - Moon (GRAIL, Diviner)
  - **Astrophysics**
    - Universe (WISE, Spitzer, GALEX, Herschel, Keck)
    - Exoplanets (Kepler)
  - **Deep Space Network**
Continuous Robotic Presence on and in-orbit around Mars

2001 Mars Odyssey

Mars Reconnaissance Orbiter

Mars Express (ESA)

Opportunity

Curiosity

“Do not go where the path may lead, go instead where there is no path and leave a trail”

--- Ralph Waldo Emerson

Opportunity’s tracks

Meridiani Planum
Mission & Charter of NASA’s SARB*

Mission:
Manage flight software complexity through better software architecture

Background
• Established in 2009 based on recommendation from Flight Software Complexity study to NASA Chief Engineer
• Targets projects in Formulation Phase to maximize impact

Charter
• Provide constructive feedback to flight projects in the formative stages of software architecting
• Focus on architectural improvements to reduce and/or better manage complexity in requirements, analysis, design, implementation, verification, and operations
• Spread best architectural practices, principles, and patterns across flight software centers
• Contribute to NASA Lessons Learned

* SARB = Software Architecture Review Board
Quality Attributes have a significant impact on the system design, software architecture and cost

- Requirement for software portability (e.g., Consider abstraction layers)
- Requirement for software decoupling (e.g., Consider a message passing interface, aka Software Bus)

It’s uncommon to see Quality Attribute requirements at the mission level

- Quality Attribute requirements tend to be derived requirements
- Software architects and engineers need to do a little “selling” to convince project management it’s in the project or organization’s best interest
  - Organizations tend to think across missions and will more readily consider cross-cutting requirements

Quality Attributes and associated priorities should be traded, documented, and reviewed early by all stakeholders
Creating the Quality Attribute Table

What problem were we trying to solve?

- A method to objectively evaluate an architecture in the domain of space mission flight software
  - Space Universal M0dular Architecture (SUMO) architecture survey
  - NASA’s Software Architecture Review Board (SARB)

- Many of the surveyed software architecture description documents had a list of quality attributes, but:
  - Attributes were inconsistent
  - Attribute definitions were inconsistent
  - Attribute lists were incomplete
  - Available architecture documents outside the domain
  - Missing objective evaluation criteria
• SUMO started this effort to evaluate several software architectures at NASA, DoD, and in industry in the hope of establishing a level of commonality that could be exploited to reduce cost and expand markets

• NASA’s SARB picked up this effort to provide more objective evaluation criteria for use during architecture reviews

• The authors worked with software architects across NASA centers and DoD
  – Most notably ARC, JPL, JSC, GSFC, AFRL, APL, and NAVAIR

• Reviewed documents available on the internet

• Created an initial list and refined it over several months
Quality Attribute Table Format

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Aspects of Requirement</th>
<th>Requirement</th>
<th>Rationale</th>
<th>Evidence of verification</th>
<th>Tactic to achieve</th>
<th>Project specified</th>
<th>Project Prioritization (NA, Low, Med, Hi)</th>
<th>Project intended variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>A design and implementation property of the architecture and applications supporting their use on systems other than the initial target system.</td>
<td>Real-time and non-real-time</td>
<td>The architecture shall ...</td>
<td>1) Supports both...</td>
<td>Demonstrate execution ...</td>
<td>Application logic is ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Operating systems</td>
<td>The architecture shall...</td>
<td>Operation system selection is...</td>
<td>Demonstrate execution on multiple operating systems ...</td>
<td>Standards and abstractions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor / platform</td>
<td>The architecture shall...</td>
<td>Processors and platforms are typical variation points...</td>
<td>Is the architecture Processor/Platform interface abstraction sufficient ...</td>
<td>Standards and abstractions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Intended to help software teams think carefully about quality attributes
  - What do you mean by “portability,” “availability,” “safety,” etc.?
  - What are the different aspects of an attribute?
  - What requirements should it impose?
  - How does it add value to the project architecture?
  - What evidence will demonstrate that the QA is achieved?
  - What tactic(s) will be used to achieve it?
  - What is its priority?

This table is not comprehensive and is intended to be extended
Quality Attributes in the QA Table

- **Column A: The Quality Attributes**
  - The first column in each row is the quality attribute to be addressed. This column contains the chosen term indicating the non-functional requirement or property of the architecture to be implemented or reviewed. The term was selected through consensus by the SARB members, since different perspectives led to differing opinions as to which terms best fit the desired property.
  - Quality Attributes are:
    - Portability
    - Interoperability
    - Modifiability
    - Performance
    - Availability
    - Reusability
    - Predictability
    - Usability
    - Scalability
    - Verifiability
    - Manage complexity
    - Security
    - Safety
    - Openness

These are important in mission-critical real-time embedded systems.
QA Description and Aspects

- **Column B: Description of the QA and other terms used to describe the quality**
  - Each Quality Attribute identified in Column A is defined in Column B to help the user understand what is meant by the term. For example, “Portability” is defined as “A design and implementation property of the architecture and applications supporting their use on systems other than the initial target system.” Numerous references were used to define each.

- **Column C: Aspects of the QA**
  - The term “Aspect of” is intended to define a context for the attribute. The “State/behavior” aspect of the QA “Predictability” can be rephrased as “the predictability of the state/behavior of the architecture.” The QA “Portability” has numerous entries for “Aspect of” that help provide context; they allow the architect or evaluator to individually specify whether the application or system is portable across real-time/non-real-time implementations, across operating systems, across avionics platforms, or across any combination.
### QA Requirements, Rationale, Evidence

**Column D: Requirements**
- Column D contains sample requirements that the architecture must satisfy to claim support of a quality attribute. These requirements are verifiable statements, and are specific to each “Aspect of” row, as they need to be associated with a specific QA context.

**Column E: Rationale**
- The “Rationale” column documents how each QA requirement adds value to an architecture for a project or projects. The team did not list all possible rationale, but focused on the one or two considered most important. For example, a project may have a requirement that the “architecture shall support application execution in real-time and non-real-time environments” allowing deployment on flight and development/test (e.g. desktop) run-time environments.

**Column F: Evidence of/Verification**
- Provides evidence that the requirement has been verified, or how it will be verified. For example, one aspect of portability is OS portability, and the associated requirement (Column D) is: “The architecture shall support application execution on a range of operating systems without modification of the application.” This requirement would be convincingly met if the project “demonstrates execution on multiple operating systems with no changes to the application.”
Tactics to Achieve, Project Prioritization

• **Column G: Tactic to Achieve**
  
  – A tactic is a design decision that influences the control of a quality attribute response [Bass et al, 2003]. Thus, Column G is where the project identifies design decisions to be used in meeting the requirements in Column D. Explicitly identifying such decisions enables experienced reviewers to challenge a decision if they feel the tactic is inadequate or insufficiency described. For example, in the aspect of Portability related to operating systems, the QA table provides “standards and abstractions” as general tactics.

• **Columns H-I: Project Prioritization and Project Intended Variation**
  
  – Each row of the table has two columns for use by project software architects, implementers, and reviewers. “Project Prioritization” and “Project intended variation” are to be completed by project personnel in the very early stage of development concurrently with the system requirements. All QAs should be reviewed to decide/establish the priority of each (Not Applicable, Low, Medium or High) in Column H.
• Review the Quality Attribute list for things to consider early in the architecture formulation phase

• Assign the Quality Attribute priorities
  – Complete the table even if the QA is not applicable

• Create the variation points

• Create the architectural trades

• Get stakeholder input

• Develop and document additional tactics to achieve
Developers’ Usage

• Become a stakeholder in the architecture
  – Provide inputs to the trades
  – Help the architects understand any implementation, maintenance, or performance impacts for the QAs being considered

• Use the “Tactic to achieve” for guidance on design and implementation
  – Help improve and expand the tactics

• Document the “Evidence of/verification”

• Include QAs requirements in design and code reviews
Reviewers’ Usage

- Examine the alignment of Project Prioritization and driving system requirements
- Are the tactics to achieve valid for intended attribute
- Are the trades sufficiently documented and contain valid rationale
- Is the evidence included in the architecture description document
- Have all the stakeholders been considered and their interests addressed
Next Steps

• Use the QA table to evaluate an architecture
  – NASA’s core Flight System (cFS) software product line
  – Potential to evaluate JPL’s Core software product line

• Use the QA table for the next SARB review

• Expand QA table based on reviews

• Post table on FSW Workshop website at
  http://flightsoftware.jhuapl.edu

• For further information, ref 2016 IEEE paper “Quality Attributes for Mission Flight Software: A Reference for Architects”
Questions?
From Caltech students testing rockets to exploring the planets in our lifetime.

- Caltech students (1936)
- Missiles (1940s)
- Explorer 1 (1958)
- Mars Exploration Rovers (2004–present)
- Spitzer Space Telescope (2004–present)
- Earth Science (1978–now)
JPL’s mission for NASA is **robotic** space exploration

- Mars
- Solar system
- Exoplanets
- Astrophysics
- Earth Science
- Interplanetary network

Of the 73 known planetary/moon exploration missions that the U.S. has launched to date, JPL has managed 52 of them.
NASA’s Mars Rovers – A Family Portrait

- Curiosity, 2012
- Spirit/Opportunity, 2004
- Sojourner, 1997