

Software Architecture Evaluation Framework The Aerospace Corporation

Software Architecture Evaluation

- Software architecture is a key part of many of our largest programs
 - Primary carrier of system qualities, such as performance, modifiability, and security
 - Problems expensive to fix in time and money, especially if caught late

Project Goals

- Development of a framework of evaluation dimensions tailored for Space and Space – related programs
- Document methodologies successfully used across acquisition phases
- Provide guidance/training for teams conducting software architecture evaluation
- Develop a tool that supports workflow for performing evaluations
- Keep the emphasis on important features to evaluate, not evaluation process
- Output of an evaluation :
 - Identification of specific strengths and weaknesses
 - Actionable recommendations



Why a Framework?

- Ensure comprehensive coverage
 - Standardize assessment dimensions
 - Ensure attention to all key areas of importance
 - Guide partitioning of evaluation work
 - Avoid reinventing the wheel
- Maintain Objectivity
 - Make evaluation repeatable across teams and evaluations
- Leverage Aerospace corporate experience
 - Communicate software architecture expertise
 - Communicate an (increasing amount) of domain-specific guidance
- Maintain Evaluation Focus
 - Support drill-down to specific issues while maintaining "big picture" context

What is the Framework and How is it Applied?

- At its Core: A structured collection of 1000+ questions about an Space/Space -related program's software architecture
 - Organized as follows:
 - Four top-level categories
 - Each category contains multiple dimensions (concerns)
 - Each dimension has a set of evaluation questions
- Evaluators tailor framework for the target program and evaluation goals
 - Evaluators select a subset of dimensions/questions
 - Questions are tailored for the target program/evaluation
 - Can include deleting, modifying, and/or adding new questions
- Method agnostic and complementary with scenario based evaluations



Framework: Categories and Dimensions

Architecture Description

Architecture **Scope**Documentation

Architecture **Tradeoffs**Documentation

Use of Viewpoints and Views

Scope of Documentation

Consistency Across Views

Use of Description Notations

Organization and Traceability of Documentation

Architecture Scope, Priorities, Tradeoffs

Architectural **Scope**Software Quality **Attributes**and **Tradeoffs**

Architectural Satisfaction of Functionality/Qualities

Architectural **Decisions** and

Tradeoffs Reflect Program Drivers

Flowdown of

Enterprise/System
Architecture to Software
Architecture

Allocation of Requirements to Software Architecture

Software Architecture

Consistency with User Interface

Software Architecture and Data Architecture are Integrated

COTS/GOTS

Appropriateness

Reuse Appropriateness

Integration of Reuse into Software Architecture

Openness and Platform independence

External Interfaces

Modularity and Layered

Architecture

Scalability

Flexibility

Timeliness and Performance

Reliability and Availability

Security / Information Assurance

Manageability

Technology Readiness

Usability

Safety

Extendibility/Extensibility

Survivability

Architecture Development/ Evolution Methodology

Software Architecture
Process

Personnel and Skill Mix

Communication and Interaction

Tools

Note: Many of these dimensions drill down into specific domains

These questions are tailored into program-specific questions to address key requirements and important characteristics the architecture needs to deliver

Alan.D.Unell@aero.org
Software Engineering Subdivision

Example Questions

Sample Requirement Addressed:

"The system shall be modifiable and flexible and expandable"

Category: "Arch. Satisfaction of Functionality/Qualities"

Dimension: "Modularity and Layered Architecture"

- 1. Is there a clear and reasonable separation of concerns (for example, application from infrastructure, user interface details from application behavior, hardware/operating system dependencies, middleware & commercial software dependencies)?
- 2. Are modular design principles (high cohesion among components, weak coupling & well-defined interfaces between components) incorporated to allow software to be functionally partitioned into scalable components?
- 3. What is the adopted layering model?
 - Are there any layer violations? Are the risks of these violations and adequate mitigations plans identified?
- 4. Is a layering model used consistently throughout the architecture (an example of inconsistency: some permit a component to use services at any lower layer, some permit use of services only at the next layer)?

Using the Framework in a Software Architecture Evaluation

- The framework does <u>not</u> define an evaluation process
 - Though usage guidance is provided based on experience
- Tailor the framework to account for:
 - Different evaluation goals
 - Number of evaluators & their expertise levels
 - Specific domains to be investigated
 - Space? Ground? Launch?
 - Program lifecycle stage
 - Architectural information available
 - Documents only? Access to architects and system experts?



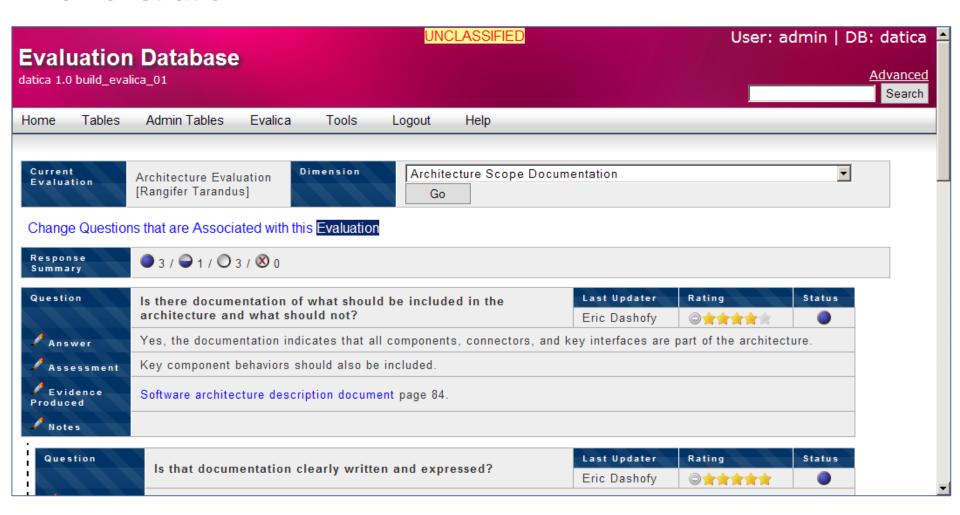
Tool Support: "Evalica" Evaluation Tool *Motivation*

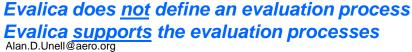
- Organize a growing list of 1000+ questions
 - With complex relationships:
 - Parent-child: Parents set context for children
 - **Dimensions:** Each question in exactly one dimension (i.e., concern)
 - Multi-domain: Some questions pertain to one or more NSS-related domains
 - Permit users to rapidly subset questions by multiple criteria
- Provide a clearinghouse for evaluators to
 - Tailor the questions to their evaluation
 - Capture answers and track the progress of an evaluation
- Generic enough to be used for other types of evaluations



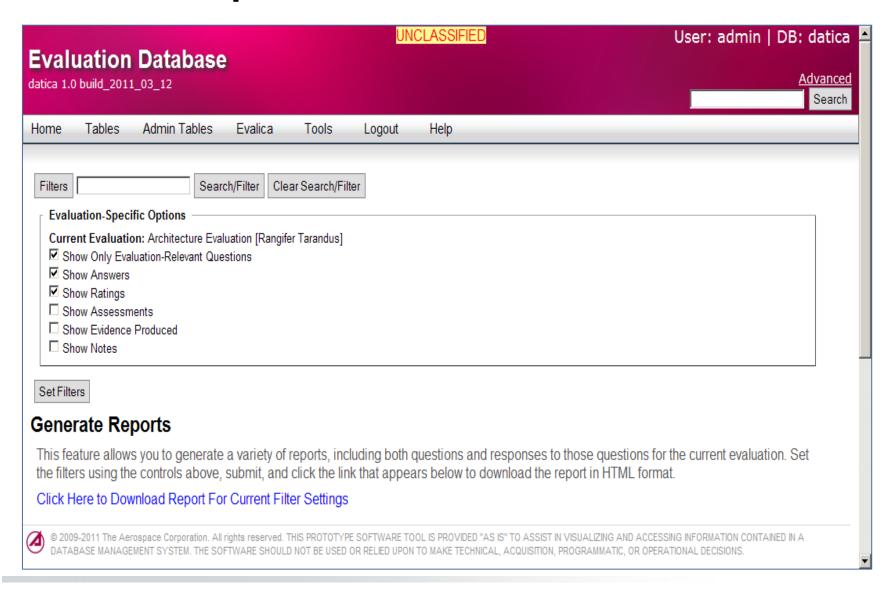
Tool Support: "Evalica" Evaluation Tool

Demonstration





Generate Reports





Notes

All trademarks, tradenames, and service marks are the property of their respective owners."

