Architecting Systems to Meet Expectations - Managing Quality Characteristics To Reduce Risk

Paul R. Croll  
CSC  
pcroll@csc.com

Chair, NDIA Software Industry Experts Panel

Co-Chair, DoD Software in Acquisition Working Group on Software Quality Attributes
Outline

- The Systems Quality Challenge
- Architecture And Quality Defined
- Quality Attribute-Based Approaches To Architecting Systems
- Making The Case For Architectural Quality
- Customer Implications Of Quality-Attribute-Based Architectural Approaches
- Process Maturity And Product Quality
- A Current Concern: Architecting For System Assurance
- Summary
It’s About The Architecture . . .

One of the top ten emerging systemic issues, from fifty-two in-depth program reviews since March 2004, was inadequate software architectures.

It’s Also About Quality . . .

The NDIA Top Software Issues Workshop examined the current most critical issues in software engineering that impact the acquisition and successful deployment of software-intensive systems.

Two issues emerged that were focused specifically on the relationship between software quality and architecture:

- Ensure defined quality attributes . . . are addressed in requirements, architecture, and design.
- Define software assurance quality attributes that can be addressed during architectural trade-offs.

The Systems Quality Challenge

If we are successful in managing risk for the systems we build, and meet stakeholder expectations, we must:

– Start as early as possible in the design process to understand the extent to which those expectations might be achieved
– Develop candidate system architectures and perform architecture trade-offs
– Define and use a set of quantifiable system attributes tied to stakeholder expectations, against which we can measure success
The Systems Quality Challenge Is A Software Quality Challenge

- Most systems we encounter today contain software elements and most depend upon those software elements for a good portion of their functionality
- Modern systems architecture issues cannot be adequately addressed without considering the implications of software architecture
Architecture Defined

The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.


The set of all of the most important, pervasive, higher-level, strategic decisions, inventions, engineering trade-offs, assumptions, and their associated rationales concerning how the system meets its allocated and derived product and process requirements.

Quality Defined

Software quality: The degree to which software possesses a desired combination of attributes.


Software product quality: The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs.

Developing systematic ways to relate the software quality attributes of a system to the system’s architecture provides a sound basis for making objective decisions about design tradeoffs and enables engineers to make reasonably accurate predictions about a system’s attributes that are free from bias and hidden assumptions. The ultimate goal is the ability to quantitatively evaluate and trade off multiple software quality attributes to arrive at a better overall system.

Relationships Between Attributes

- **Collaboration**
  - Increasing the degree to which one attribute is realized increases the realization of another

- **Damage**
  - Increasing the degree to which one attribute is realized decreases the realization of another

- **Dependency**
  - The degree to which one attribute is realized, is dependent upon the realization of at least some sub-characteristics of another

Optimization Among Quality Attributes

Example: A large telecommunication application

- Good optimization (Collaboration)
  - balance among multiple quality attributes, such as maintainability, performance and availability

- Poor optimization (Damage)
  - Focusing solely on maintainability often results in poor system performance
  - Focusing on performance and availability alone may result in poor maintainability

Explicit architectural decisions can facilitate optimization among quality attributes

Understanding Quality In The Context Of Architectural Structures

Structures for describing architectures
- *Functional structure* is the decomposition of the functionality that the system needs to support
- *Code structure* is the code abstractions from which the system is built
- *Concurrency structure* is the representation of logical concurrency among the components of the system
- *Physical structure* is just that, the structure of the physical components of the system
- *Developmental structure* is the structure of the files and the directories identifying the system configuration as the system evolves

Using architectural structures to understand quality
- *Concurrency and Physical* structures are useful in understanding system Performance
- *Concurrency and Code* structures are useful in understanding system Security
- *Functional, Code, and Developmental* structures are useful in understanding system Maintainability

Attribute-Driven Design (ADD) produces an initial software architecture description from a set of design decisions that show:

- Partitioning of the system into major computational and developmental elements
- What elements will be part of the different system structures, their type, and the properties and structural relations they possess
- What interactions will occur among elements, the properties of those interactions, and the mechanisms by which they take place

In the very first step in ADD, quality attributes requirements are expressed as the system’s desired measurable quality attribute response to a specific stimulus.

Knowing these requirements for each quality attribute supports the selection of design patterns and tactics to achieve those requirements.

Understanding The Consequences Of Architectural Decisions With Respect To Quality Attributes

- The Architecture Tradeoff Analysis Method\(^{SM}\) (ATAM\(^{SM}\)) is dependent upon quality attribute characterizations, like those produced through ADD, that provide the following information about each attribute:
  - The stimuli to which the architecture must respond
  - How the quality attribute will be measured or observed to determine how well it has been achieved
  - The key architectural decisions that impact achieving the attribute requirement

ATAM takes proposed architectural approaches and analyzes them based on quality attributes
  - generally specified in terms of scenarios addressing stimuli and responses
    - Use case scenarios, describing typical uses of the system
    - Growth scenarios, addressing planned changes to the system
    - Exploratory scenarios, addressing any possible extreme changes that would stress the system

ATAM also identifies sensitivity points and tradeoff points

Some Real World Architecture Review Issues

- Results from four AT&T companies
  Between 1989 and 2000
- More than 1,000 issues
- Six classes of issues
  - Product architecture and design, 29–49%
  - Management controls, 14–26%
  - Problem definition, 10–18%
  - Process, 4–19%
  - Technology, 3–14%
  - Domain knowledge, 2–5%

Making The Case For Architectural Quality

The Quality Case
- The set of claims, supporting arguments, and supporting evidence that provide confidence that the system will in fact demonstrate its expected quality characteristics
- Common types of quality cases include:
  - safety cases
  - security cases
  - assurance cases

The Architectural Quality Case
- The architectural claims, supporting arguments, including architectural decisions and tradeoffs, architectural representations, and demonstrations that the architecture will exhibit its expected quality characteristics

Risk Management Implications Of Quality-Attribute-Based Architectural Approaches

- Stakeholder *quality requirements* will have been distilled into *architectural drivers* which will have shaped the system architecture.
- Tradeoffs will have been made to optimize the realization of important *quality characteristics*, in concert with stakeholder expectations.
- The level of confidence that the resultant architecture will meet those expectations will be known.
- Stakeholders will be knowledgeable of any *residual risk* they are accepting by accepting the delivered system.

Process Maturity Does Not Guarantee Product Quality

The CMMI® embodies the process management premise that, *the quality of a system or product is highly influenced by the quality of the process used to develop and maintain it.*

However:

*Several recent program failures from organizations claiming high maturity levels have caused some to doubt whether CMMI® improves the chances of a successful project.*

Source: R. Hefner. **CMMI Horror Stories: When Good Projects Go Bad.** SEPG Conference, March 2006
Process maturity can in many cases improve project performance, but special attention to the engineering processes is required to ensure that stakeholder quality expectations are realized in resultant products.
A Current Concern: Architecting For System Assurance

The challenge:

Integrating a heterogeneous set of globally engineered and supplied proprietary, open-source, and other software; hardware; and firmware; as well as legacy systems; to create well-engineered integrated, interoperable, and extendable systems whose security, safety, and other risks are acceptable – or at least tolerable.


The vision:

The requirements for assurance are allocated among the right systems and their critical components, and such systems are designed and sustained at a known level of assurance.

Architectural Principles For Assurance

- Isolate critical components from less-critical components
- Make critical components easier to assure by making them smaller and less complex
- Separate data and limit data and control flows
- Include defensive components whose job is to protect other components from each other and/or the surrounding environment
- Understanding the interrelationships between components and their linkages
- Use defense-in-depth measures where appropriate
- Beware of maximizing performance to the detriment of assurance

Summary

If we are to be successful in managing risk for the systems we build, and meet stakeholder expectations, we must:

- Start as early as possible in the design process to understand the extent to which those expectations might be achieved
- Define a set of *quantifiable quality attributes* tied to stakeholder expectations, against which we can measure success and understand the residual risk stakeholders are being asked to accept
- Develop candidate system architectures and perform architecture trade-offs using those attributes
For More Information . . .

Paul R. Croll  
CSC  
10721 Combs Drive  
King George, VA  22485-5824  

Phone:  +1 540.644.6224  
Fax:    +1 540.663.0276  
e-mail:  pcroll@csc.com