Software Architecture in Practice

Chapter 4: Software Qualities

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Sponsored by the U.S. Department of Defense

Lecture Objectives

This lecture will enable students to
- **be familiar with four classes of system qualities and examples from each class**
- **describe the general way that qualities are achieved by architectural means**

Quality Attributes

Long ago, people like Dijkstra and Parnas realized that in software, *structure matters.*

If the only criterion for acceptance was getting the right answer, we would not need architecture: Unstructured, monolithic systems would suffice.

But other things also matter.
- **modifiability**
- **time of development**
- **performance**
- **coordination of work teams**

Carnegie Mellon University
Software Engineering Institute

System Qualities

Quality is the fitness of a product for its intended use.

There are four classes of system qualities.

1. **Runtime qualities: those that can be measured as the system executes (perhaps an infinite number of times)**
2. **Non-runtime qualities: those that cannot be measured as the system executes**
3. **Business qualities**
4. **Architecture qualities**

Carnegie Mellon University
Software Engineering Institute

System Qualities Measurable by Observing the System

Does it do what was expected?

Is it easy to use?

Does it compute the right answer?

How long does it take?

Can an unauthorized user hack in to it?

How often does it crash or stop working?

Carnegie Mellon University
Software Engineering Institute

System Qualities Not Measurable by Observing the System

How long does it take to build?

How long does it take to test?

How long does it take to integrate?

How much did it cost?

How easy will it be to modify?

System Qualities

System qualities are largely dependent on architectural decisions.

Knowing something about one quality provides no information about other qualities.

For example
- How fast will a modifiable system run?
- How modifiable will a secure system be?

A change in structure that improves one quality often affects the other qualities.

Architectural Effects on System Qualities -1

Architecture affects most qualities, but not all aspects of all qualities.

Consider usability.
- Choice of radio buttons, dialog boxes, or command line input affects usability, but these decisions are not architectural.
- Isolating those decisions so that they are changeable is architectural, but that affects modifiability, not usability.

Architectural Effects on System Qualities -2

Architecture can only permit, not guarantee, any quality attribute: “What the architecture giveth, the implementation can taketh away.”

System Qualities

→ ***Runtime qualities***

Non-runtime qualities

Business qualities

Architecture qualities

Runtime System Qualities: *Functionality*

Definition: the ability of the system to do the work for which it was intended

It requires components to interact, cooperate, and synchronize correctly.

It is largely orthogonal to the structure: the A-7E system showed this. (Before A-7E, some people said avionics programs *could not* be built using information-hiding modules.)

Runtime System Qualities: *Performance* -1

Definition: the response time, utilization, and throughput behavior of the system

Architectural mechanisms
- **partitionability of functionality**
- **flexibility of allocation to hardware**
- **monitoring mechanisms (built into the final product) and tools (external to the product)**
- **tuning mechanisms (built into the final product) and tools (external to the product)**
- **real time scheduling and monitoring mechanisms**
- **consistent way of getting and relinquishing control**
- **packaged facilities for modeling performance**

Runtime System Qualities: *Performance* -2

Measurement
- estimating environments and resource usages of various system components
- using analytic models, queuing theory, and/or simulation to determine latency, bottlenecks, and other measures

Example measurement techniques
- rate monotonic analysis (RMA)
- software performance evaluation (SPE)

Runtime System Qualities: *Security* -1

Definition: a measure of a system's ability to resist unauthorized attempts at usage or behavior modification, while still providing service to legitimate users

Architectural mechanisms
- activity monitoring mechanisms for intrusion detection
- encryption and decryption
- minimization of number of points of entry
- security kernels and shells

Runtime System Qualities: *Security* -2

Measurement: inspection or estimation methods (such as CERT's ISRE method).

Using these techniques, one would try to ascertain how the system would respond to particular attack or misuse scenarios.

Runtime System Qualities: *Availability* -1

Definition: the measure of the time that the system is up and running correctly; the length of time between failures and the length of time needed to resume operation after a failure

Architectural mechanisms
- system-wide error recovery strategies
- provisions for managing unreliable transports
- redundancy of critical components
- redundancy of critical communication paths

Runtime System Qualities: *Availability* -2

Architectural mechanisms (continued)
- live switching capability
- monitoring of liveness and reporting failures
- templates for fault raising, catching, and propagating
- quick recovery or startup capability

Measurement: Markov modeling. The mean time to failure and mean time to repair of critical components must be identified (typically measured) or estimated.

Runtime System Qualities: *Usability* - 1

Definition: the ease of use and of training the end users of the system. Usability has five aspects.
- learnability: how is the software to learn; how easy is the documentation to use?
- efficiency: does the software keep pace with the user’s tasks?
- affect: does the user feel good about using the system?
- helpfulness: does the system communicate well and assist the user in resolving problems?
- control: does the user feel that the software is responding normally and consistently?

Runtime System Qualities: *Usability* -2

Architectural mechanisms

- existence of mock-up tools and environment
- common user interface (UI) look and feel, and common UI toolkit
- common style guide and supporting framework

Measurements

- performing cognitive walk throughs and prototypes of the UI
- doing user testing of prototypes or fully functioning system

Runtime System Qualities: *Interoperability* -1

Definition: the ability of two or more systems to cooperate *at runtime*

Architectural mechanisms

- minimizing the external complexity of the components (i.e., interfaces, environmental, assumptions, pre- and post-conditions)
- limited set of interaction mechanisms and protocols
- existence of a universal naming scheme
- name service facilities
- careful attention to component interfaces

Runtime System Qualities: *Interoperability* -2

Measurement: software architecture analysis method (SAAM). Try to assess:

- How does a developer identify system elements?
- How does a user identify system elements?

System Qualities

Runtime qualities

→ *Non-runtime qualities*

Business qualities

Architecture qualities

Non-Runtime System Qualities: *Modifiability* -1

Definition: the ease with which a software system can accommodate changes to its software. Modifiability is perhaps most closely aligned to the architecture because of the locality principle: c*hanges confined to a single component are usually easier than changes to many components.*

Since architecture defines the components, architecture defines which changes are local.

Non-Runtime System Qualities: *Modifiability* -2

Architectural mechanisms
- consistent functional decomposition and allocation of functionality
- use of (a small number of) patterns
- consistent packaging of functionality
- information hiding and abstraction
- layering
- decoupling the producers and consumers of data

Measurement
- SAAM
- metrics

Non-Runtime System Qualities: *Portability* -1

Definition: the ability of the system to run under different computing environments. These environments can be hardware, software, or a combination of the two.

Non-Runtime System Qualities: *Portability* -2

Architectural mechanisms
- **platform/network independent layer(s)**
- **use of interface standards**

Measurement
- **SAAM**
- **inspection**

Non-Runtime System Qualities: *Reusability* -1

Definition: the degree to which existing components can be reused in new applications. Reusing components reduces time to market and cost of future systems.

Non-Runtime System Qualities: *Reusability* -2

Architectural mechanisms
- regularity and minimization of patterns
- creating an application framework
- creating a product line architecture

Measurement
- SAAM
- inspection

Non-Runtime System Qualities: *Integrability*

Definition: the ability to make the separately developed components of the system work correctly together

Architectural mechanisms
- minimization of interface assumptions
- limited set of interaction mechanisms/protocols

Measurement
- SAAM
- inspection

Non-Runtime System Qualities: *Testability* -1

Definition: the ease with which software can be made to demonstrate its faults

Architectural mechanisms
- self-monitoring, capture, playback, and reporting mechanisms
- specialized environment for execution (e.g., for embedded systems)
- packaged simulation tools
- testing tools (that help in assessing code coverage and condition coverage)

Non-Runtime System Qualities: *Testability* -2

Architectural mechanisms (continued)
- framework for formal specification of behavior
- facilities for monitoring and debugging
- consistent error handling scheme

Measurements
- walk throughs
- reviews
- inspections

System Qualities

Runtime qualities

Non-runtime qualities

→ ***Business qualities***

Architecture qualities

Business Qualities -1

Cost and schedule
- cost
- time to market
- projected lifetime
- utilization of legacy systems

Business Qualities -2

Marketability

- **size of market targeted by the architecture**
- **capability of included functions**
- **marketing plans (e.g., incremental release versus total release)**

Business Qualities -3

Appropriateness for organization

- **availability of workforce**
- **allocation of expertise**
- **alignment of team structure and software structure**

System Qualities

Runtime qualities

Non-runtime qualities

Business qualities

→ ***Architecture qualities***

Architecture Qualities - 1

Conceptual integrity: A system is constructed from a small number of architectural structures that interact in a small number of ways.

Brooks argues that this is the most important aspect of a system. [Brooks 95]

Conceptual integrity is typically achieved via a single architect, or a small, well-coordinated architecture team.

Carnegie Mellon University
Software Engineering Institute

Architecture Qualities - 2

Correctness/completeness: Architecture should allow satisfaction of all the (behavioral and quality) requirements.

"Buildability:" The system must be buildable with given resources.

Carnegie Mellon University
Software Engineering Institute

Architecture and Quality Attributes

Attributes of *large* software systems are principally determined by their architecture.

We should analyze these attributes *before* we build.

Analyses are typically performed in isolation. However, the attributes of a system interact.

***All* design involves trade-offs.**

We should *reason* about these trade-offs.

Lecture Summary - 1

There are four classes of system qualities, applying to

- the executing system (runtime)
- development or maintenance efforts (non-runtime)
- the business environment
- the architecture itself

Architecture is the key to achieving qualities in all of the classes.

Lecture Summary - 2

Lessons

- Plan for qualities ahead of time. Build them into the architecture.
- It is important to understand the architectural aspects of qualities.
- Keep tight reins on the implementation to make sure that the contributions of the architecture are not undermined.

Discussion Question

1. How many other qualities of software can you name that were not covered in this lecture? For each, when is it observable? How is it measured? How is it analyzed? With which other qualities does it most often interact?

2. Brooks argues that conceptual integrity is a key to successful systems. Do you agree? Can you think of successful systems that have not had this property? If so, what factors made those systems successful anyway? How would you go about measuring a system to see if it meets Brooks' prescription?