National University
Of Computer and Emerging Sciences

Software Design and Architecture

Designing for Non-Functional Requirements (Properties)

Sajid Anwer

Department of Software Engineering,
FAST-NUCES, CFD Campus

Lecture Outline

- Fundamentals of Non-functional properties

- Different types of NFP
  - » Efficiency
  - » Complexity
  - » Scalability
  - » Adaptability

Lecture Material

- Software Architecture, Foundation, Theory, and Practice (Ch#12)

Fundamentals of NFP

- A software system's *non-functional property (NFP)* is a constraint on the manner in which the system implements and delivers its functionality

- Providing the desired functionality is often quite challenging
  - » Market demands
  - » Competition
  - » Strict deadlines
  - » Limited budgets

- However, the system's success will ultimately rest on its NFPs
  - » "This system is too slow!"
  - » "It keeps crashing!"
  - » "It has so many security holes!"
  - » "Every time I change this feature I have to reboot!"

Fundamentals of NFP

- A Challenges of designing for NFP's
  - » Only partially understood in many domains
    - E.g., MS Windows and security
  - » Qualitative vs. quantitative
  - » Frequently multi-dimensional
  - » Non-technical pressures
    - E.g., time-to-market or functional features

Types of NFP -- Efficiency

- Efficiency is a quality that reflects a software system's ability to *meet its performance* requirements while minimizing its usage of the resources in its computing environment
  - » Efficiency is a measure of a system's resource *usage economy*

- What can software architecture say about efficiency?

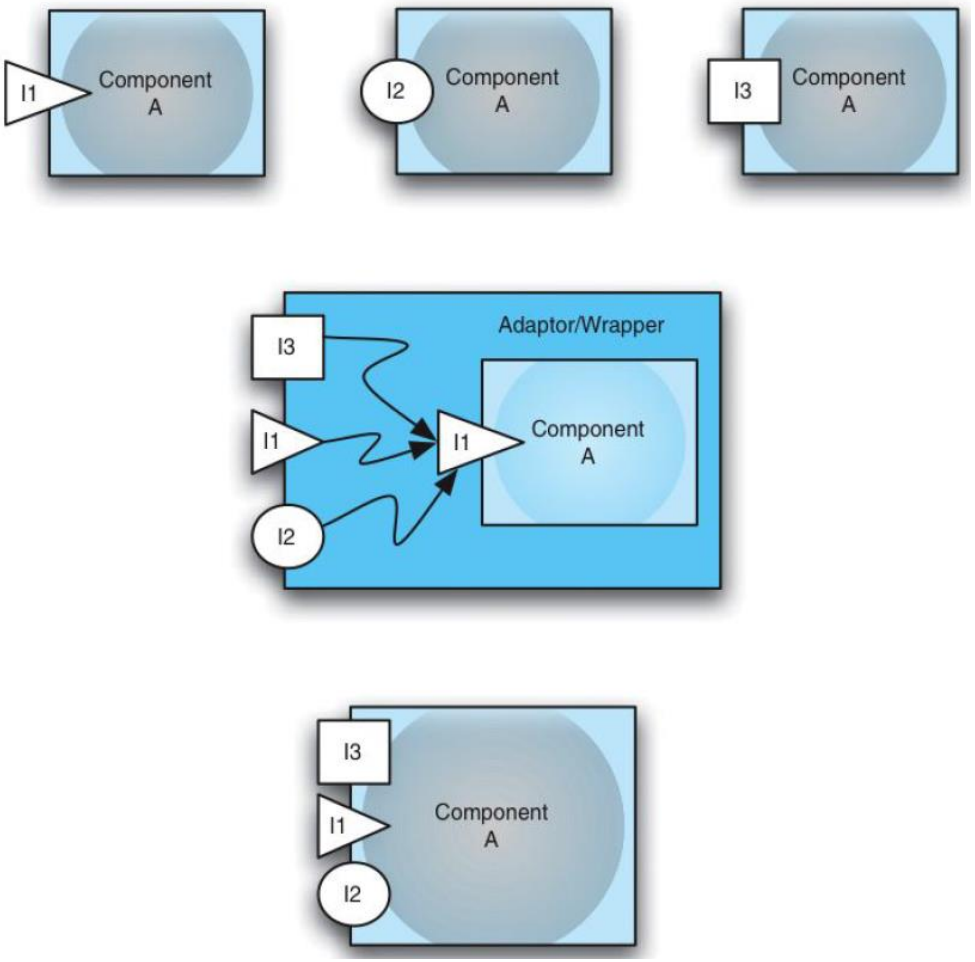
- Efficiency starts at the *architectural level*!

Types of NFP -- Efficiency

- Software Components and Efficiency
  - » Keep the components "*small*" whenever possible
  - » Keep component *interfaces simple and compact*
  - » Allow *multiple interfaces* to the same functionality
  - » Separate data components from processing components
  - » Separate data from meta-data

Types of NFP -- Efficiency

- Software Components and Efficiency



Types of NFP -- Efficiency

- Software Connectors and Efficiency
  - » Carefully select connectors
    - Broadcast
    - Direct
    - Security
  - » Use broadcast connectors with caution
  - » Asynchronous communication

Types of NFP -- Efficiency

- Architectural Configuration and Efficiency
  - » Keep frequently interacting components together
    - Cache
    - Hoarding
    - Prefetch
  - » Carefully select and place connectors in the architecture

Types of NFP -- Efficiency

- Architectural Styles not fit for good Efficiency
  - » Asynchronous communication
    - Real-time systems
  - » Large repository-based systems
  - » Data need to be delivered incrementally

Types of NFP -- Complexity

- IEEE Definition
  - » Complexity is the degree to which a software system or one of its components has a design or implementation that is *difficult to understand and verify*

- Complexity is a software system's a property that is directly proportional to the *size of the system*, number of its *constituent elements*, their *internal structure*, and the number and *nature of their interdependencies*

Types of NFP -- Complexity

- Software Components and Complexity
  - » Separate *concerns* into different components
  - » Keep only the functionality inside components
    - Interaction goes inside connectors
  - » Keep components *cohesive*
  - » Be aware of the impact of *off-the-shelf* components on complexity
  - » Insulate processing components from changes in data format

Types of NFP -- Complexity

- Software Connectors and Complexity
  - » Treat connectors explicitly
  - » Keep only interaction facilities inside connectors
  - » Separate interaction concerns into different connectors
  - » Restrict interactions facilitated by each connector
  - » Be aware of the impact of off-the-shelf connectors on complexity

Types of NFP -- Complexity

- Architectural Configurations and Complexity
  - » Eliminate unnecessary dependencies
  - » Manage all dependencies explicitly
  - » Use hierarchical (de)composition

Types of NFP -- Scalability

- Scalability is the capability of a software system to be adapted to meet *new requirements of size and scope*

- Portability is a software system's ability to execute on multiple platforms with minimal modifications and without significant degradation in functional or non-functional characteristics

Types of NFP -- Scalability

- Software Components and scalability
  - » Give each component a single, clearly defined purpose
  - » Define each component to have a simple, understandable interface
  - » Do not burden components with interaction responsibilities
  - » Avoid unnecessary heterogeneity
    - Results in architectural mismatch
  - » Distribute the data sources
  - » Replicate data when necessary

Types of NFP -- Scalability

- Software Connectors and Scalability
  - » Use explicit connectors
  - » Give each connector a clearly defined responsibility
  - » Choose the simplest connector suited for the task
  - » Be aware of differences between direct and indirect dependencies
  - » Avoid placing application functionality inside connectors
    - Application functionality goes inside components
  - » Leverage explicit connectors to support data scalability

Types of NFP -- Scalability

- Architectural Configuration and Scalability
  - » Avoid system bottlenecks
  - » Make use of parallel processing capabilities
  - » Place the data sources close to the data consumers
  - » Try to make distribution transparent
  - » Use appropriate architectural styles

Types of NFP -- Adaptability

- Adaptability is a software system's ability to *satisfy new requirements* and adjust to new operating conditions during its lifetime

- Software Components and Adaptability
  - » Give each component a single, clearly defined purpose
  - » Minimize component interdependencies
  - » Avoid burdening components with interaction responsibilities
  - » Separate processing from data
  - » Separate data from metadata

Types of NFP -- Adaptability

- Software Connectors and Adaptability
  - » Give each connector a clearly defined responsibility
  - » Make the connectors flexible
  - » Support connector composability
- Architectural Configuration and Adaptability
  - » Leverage explicit connectors
  - » Try to make distribution transparent
  - » Use appropriate architectural styles

Types of NFP

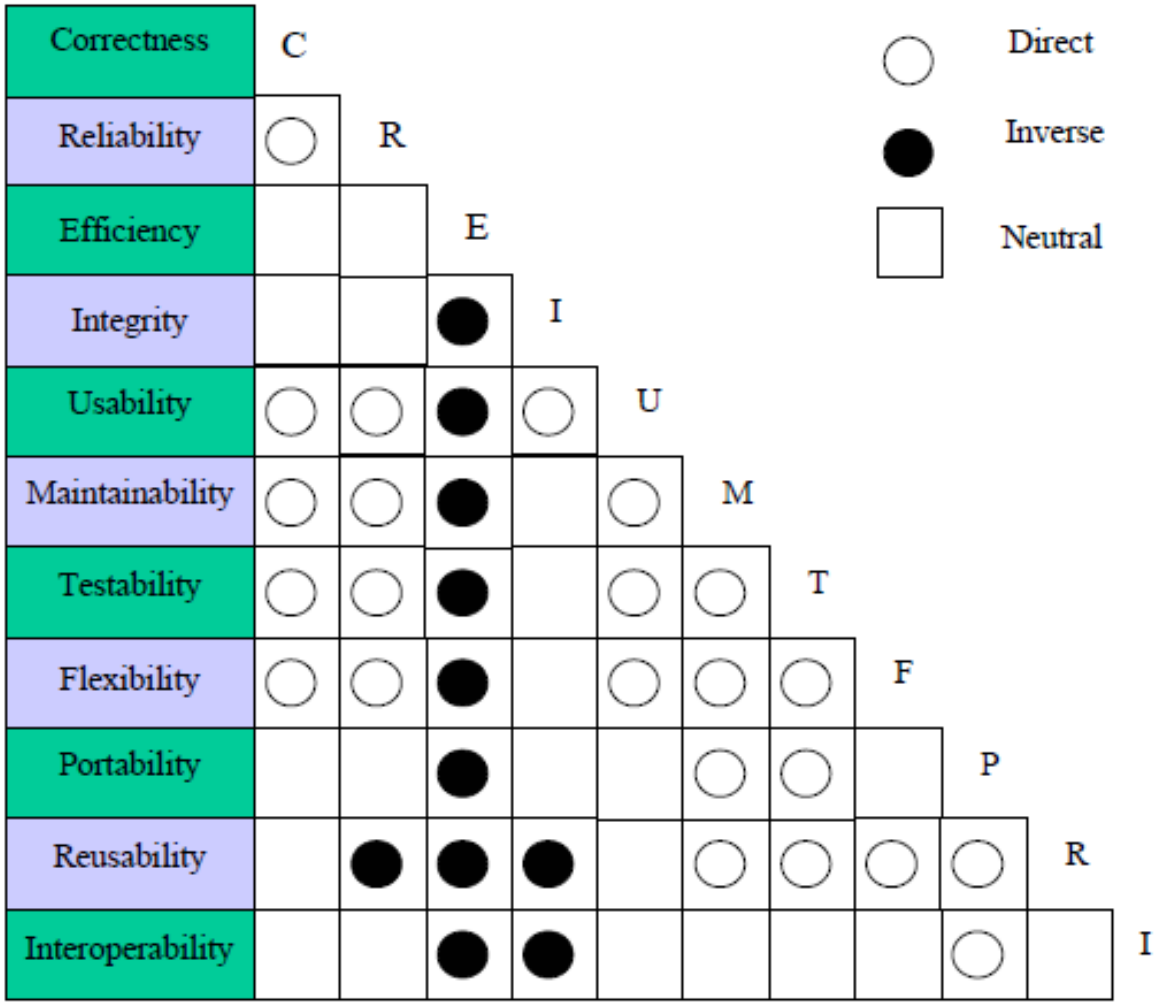


Figure 2.2 Perry's relational model of software quality

Further Reading

- Architectural Views (Ch#6) (Included in exam)
  - » Logical
  - » Physical
  - » Deployment
  - » Concurrency
  - » Behavioral
- Service-Oriented Architecture (Ch#11)
- Architecture Description Language (Ch#6)
- Reference Architecture