Quality Requirements, Requirements and Architecture

2017-12-14

Outline

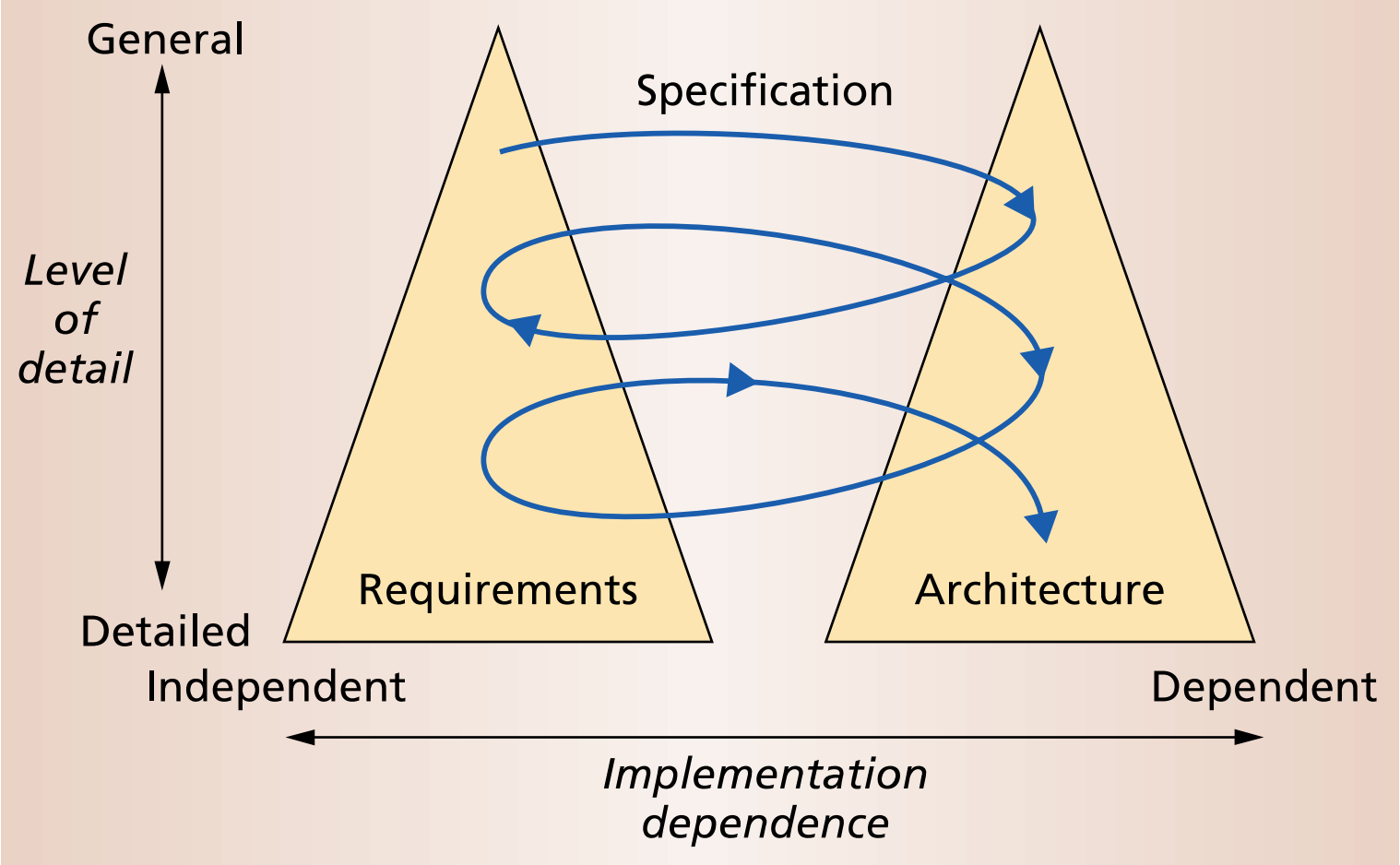
Quality Requirements, Requirements and Architecture

- **Introduction (Requirements vs. Architecture; Definitions)**
- Non-functional Requirements
- Component-Bus-System-Property Approach
- Quality Requirements and Architecture: Quality Attribute Workshop
- Optional: Quality Requirements – A New Look at an Old Problem (Martin Glinz)

- Exercise 4: Architectural Drivers and Requirements

Requirements and Architectures

Twin Peaks Model



Bashar Nuseibeh. "Weaving Together Requirements and Architectures," Computer, vol. 34, no. 3, pp. 115-117, Mar. 2001, doi:10.1109/2.910904

Problem and Solution Structures

- **Development process constructs:**
  - Problem structures (requirements)
  - Solution structures (architecture)
- **For large systems that develop and evolve:**
  - there is often a discontinuity between the two structures ...
  - ... leading to poor traceability of design decisions back to requirements, and inadequate change impact analysis

Requirements vs. Architectures

- **Requirements**
  - Denote stakeholder goals and expectations
  - Expressed in the vocabulary of the problem world
  - Can conflict and change
- **Architectures**
  - Denote systems' structure
  - Expressed in terms of components and inter-connections in the solution world
  - Should be stable and robust

http://prof.so, May 27, 2012 -- *(c) Anthony Finkelstein*

MAY
27

Top 10 Software Engineering Challenges

I believe in the sustained relevance of 'big agenda' in software engineering: providing scientifically well-founded methods and tools to underpin the development of software systems that meet the needs of users, rapidly and at reduced cost. Better, faster, cheaper, in other words. Substantial technical progress has been made but, it could be argued, we are seeing some evidence of receding impact in those areas where that progress has significantly outstripped the capacity of practice to absorb innovation. This seems like an important prompt to review the key challenges we should be addressing and to ensure that they reflect the most outstanding immediate problems, hence this, entirely personal, top 10.

1. *Relating Requirements and Architectures.* It is clear that architectures cannot be directly derived (by refinement or other means) from a requirements specification. On the other hand the identification and elaboration of a suitable architecture are not independent of the requirements it must serve. The relation runs both ways with the architecture acting as a framework for the elicitation of requirements and shaping the space of possibilities that are afforded by the system. This complex, intertwined, co-development of architecture and requirements lies at the very core of software development and though we observe it, we do not understand it.

Traceability Mandated by Standards

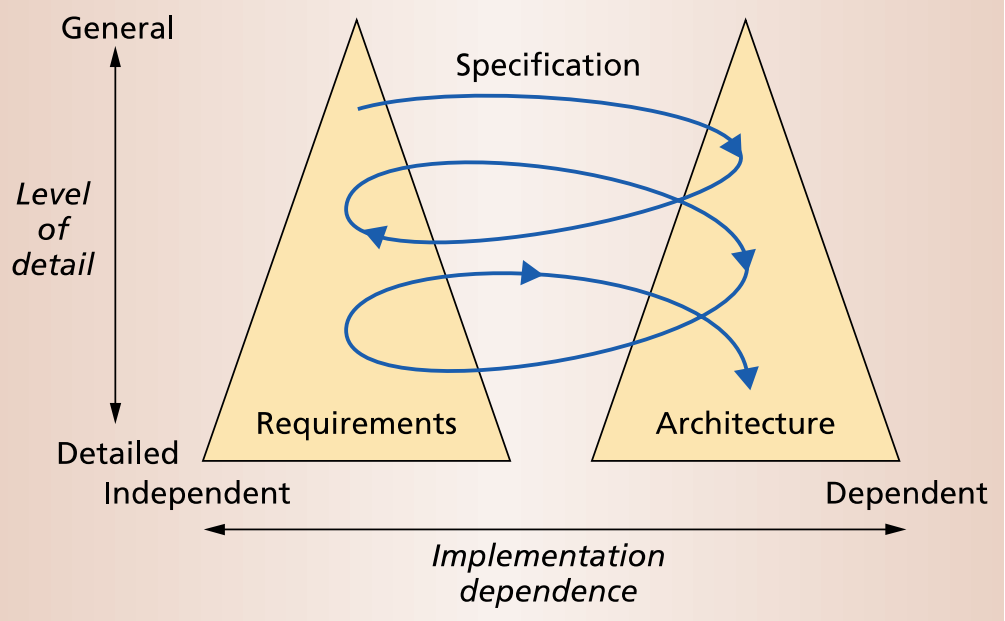
ISO/IEC 15504 Process Assessment („SPICE“)

5.4.3 ENG.3 System architectural design

Process ID	ENG.3
Process Name	System architectural design
Process Purpose	The purpose of the System architectural design process is to identify which system requirements should be allocated to which elements of the system.
Process Outcomes	As a result of successful implementation of System architectural design process: 1) a system architecture design is defined that identifies the elements of the system and meets the defined requirements; 2) the system's functional and non-functional requirements are addressed; 3) the requirements are allocated to the elements of the system; 4) internal and external interfaces of each system element are defined; 5) verification between the system requirements and the system architecture is performed; 6) the requirements allocated to the system elements and their interfaces are traceable to the customer's requirements baseline; 7) consistency and traceability between the system requirements and system architecture design is maintained; 8) the system requirements, the system architecture design, and their relationships are baselined and communicated to all affected parties.

Challenge: Bridging Requirements & Architecture

- How can we refine the requirements into an architecture?
- How can we map new requirements to existing architectural elements?
- How can we deal with both functional and non-functional aspects?
- How can we explore and assess architectural options to provide feedback to requirements?
- How can we find additional requirements?



The Twin Peaks Model

Bashar Nuseibeh, "Weaving together requirements and architecture." IEEE Computer, 34(3), 2001, pages 115–119.

Architecture Definitions (1/2)

- "The *set of structures* needed to reason about the system, which *comprises software elements, relations among them, and properties of both*."
  [Documenting Software Architectures: Views and Beyond (2nd Edition), Clements et al, Addison-Wesley, 2010]

- "The software architecture of a program or computing system is the *structure or structures* of the system, which *comprise software elements, the externally visible properties of those elements, and the relationships* among them."
  [Software Architecture in Practice (2nd edition), Bass, Clements, Kazman; Addison-Wesley, 2003]

- "... the *fundamental organization of a system*, embodied in its components, their relationships to each other and the environment, and *the principles governing its design and evolution*."
  [ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems]

[http://www.sei.cmu.edu/architecture/start/glossary/{classicdefs | moderndefs | community}.cfm]

Architecture Definitions (2/2)

- A software system architecture comprises: (1) A collection of software and system *components, connections, and constraints*. (2) A collection of *system stakeholders' need* statements. (3) A *rationale* which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements.
  [Boehm, et al., 1995]

- "A software architecture is the assignment of specific transformations that are applied to convert a purely logical model of a system that satisfies all functional requirements into *a model of a system that satisfies both functional and non-functional requirements*."
  [Charles Martin,Senior Software Architect, Sun Professional Services, New York, NY]

- "A configurable skeleton of any kind of software beast on which you hang implementation specific muscle to make it live."
  [Adu Matthaeus, Systems Architect, Eikon, Centurion South Africa]

[http://www.sei.cmu.edu/architecture/start/glossary/{classicdefs | moderndefs | community}.cfm]

Non-functional Requirements: Definitions

- "A non-functional requirement is an *attribute of or a constraint on a system*." [Glinz 2007]

- "... global requirements on its development or operational cost, performance, reliability, maintainability, portability, robustness, and the like. (...) *There is not a formal definition or a complete list of nonfunctional requirements*." [Mylopoulos et al. 1992]

- "A description of a *property or characteristic that a software system must exhibit* or a constraint that it must respect, other than an observable system behavior." [Wiegers 2003]

***

"Software quality is the degree to which software possesses a desired combination of attributes (e.g., reliability, interoperability)." [IEEE 061-1992]

Outline

Quality Requirements, Requirements and Architecture

- ✓ Introduction (Requirements vs. Architecture; Definitions)
- **Non-functional Requirements**
- Component-Bus-System-Property Approach
- Quality Requirements and Architecture: Quality Attribute Workshop
- Optional: Quality Requirements – A New Look at an Old Problem (Martin Glinz)

- Exercise 4: Architectural Drivers and Requirements

Definitions

- "A non-functional requirement is an *attribute of or a constraint on a system*."
  [Glinz 2007]

- "... global requirements on its development or operational cost, performance, reliability, maintainability, portability, robustness, and the like. (...) *There is not a formal definition or a complete list of nonfunctional requirements*."
  [Mylopoulos et al. 1992]

- "A description of a *property or characteristic that a software system must exhibit* or a constraint that it must respect, other than an observable system behavior." [Wiegers 2003]

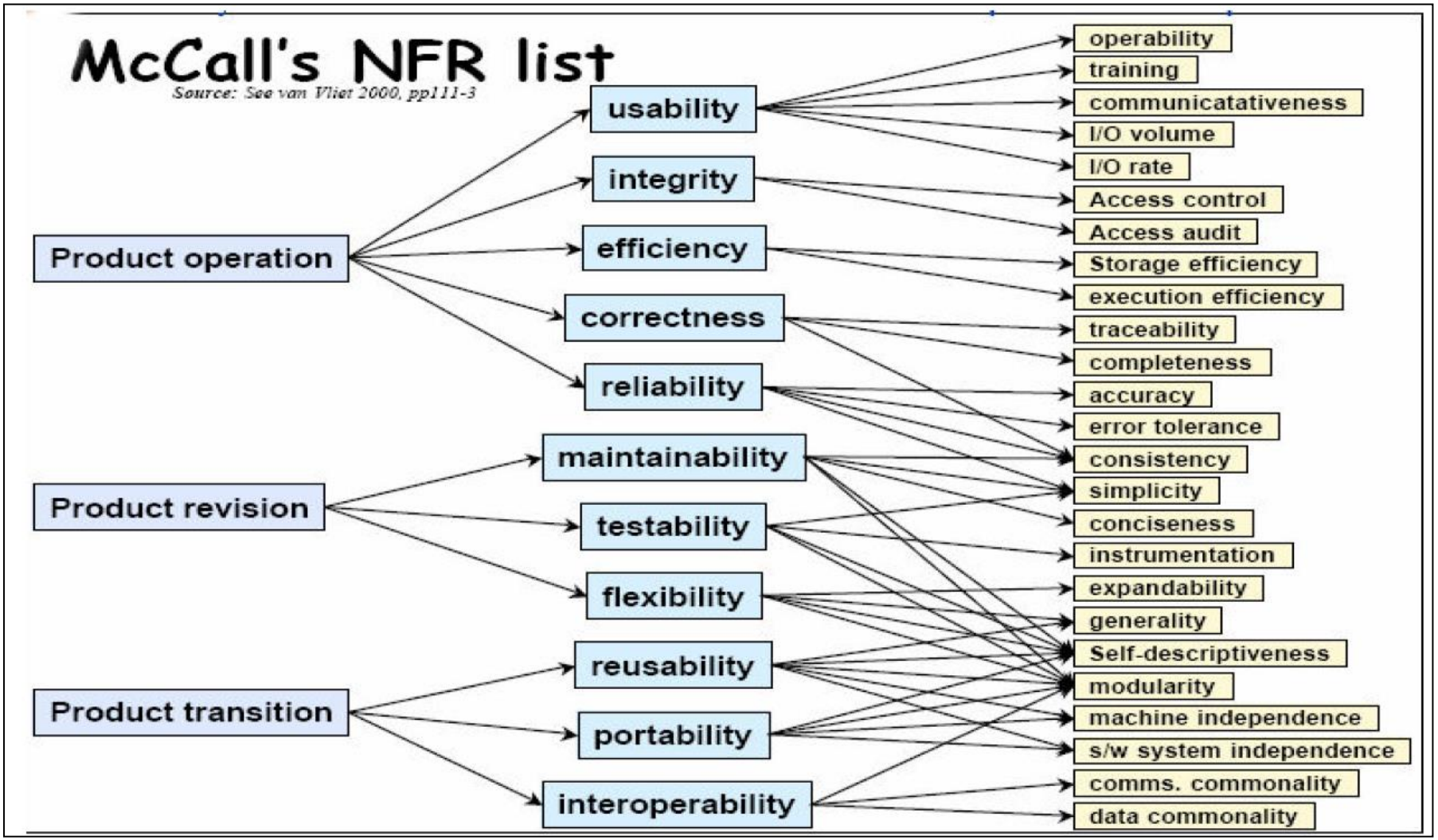
NFR-Categories according Chung

What are Non-Functional Requirements?

- -ilities: understandability, usability, modifiability, inter-operability, reliability, portability, maintainability, scalability, (re-)configurability, customizability, adaptability, variability, volatility, traceability, …
- -ities: security, simplicity, clarity, ubiquity, integrity, modularity, nomadicity, …
- -ness: user-friendliness, robustness, timeliness, responsiveness, correctness, completeness, conciseness, cohesiveness, …
- …and many other things: performance, efficiency, accuracy, precision, cost, development time, low coupling, …

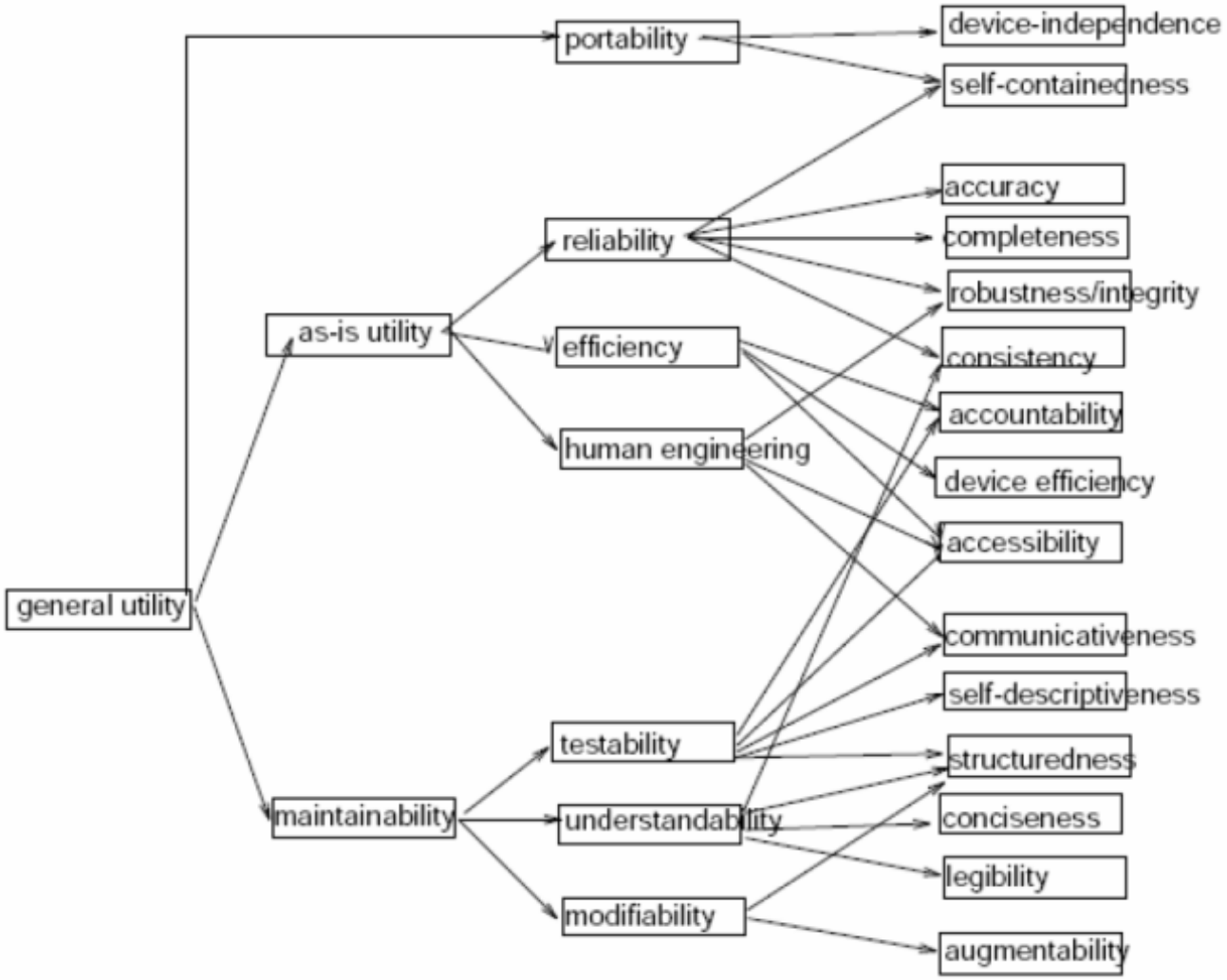
Lawrence Chung

NFR-Categories according McCall



NFR-Categories according Boehm

- Software Quality Tree [Boehm 1976]



NFR-Categories according Grady

- **Dimensions of Quality –Components of FURPS+** [Grady1992]

Functionality	Feature set capabilities, security, generality
Usability	Human factors aesthetics, consistency, documentation
Reliability	Frequency/severity of failure, recoverability, predictability, accuracy, MTBF
Performance	Speed efficiency, resource usage, throughput, response time
Supportability	Testability, Extensibility, Adaptability, Maintainability, Compatibility, Configurability, Serviceability, Installability, Localizability, Robustness

NFR-Types according IEEE-Std 830

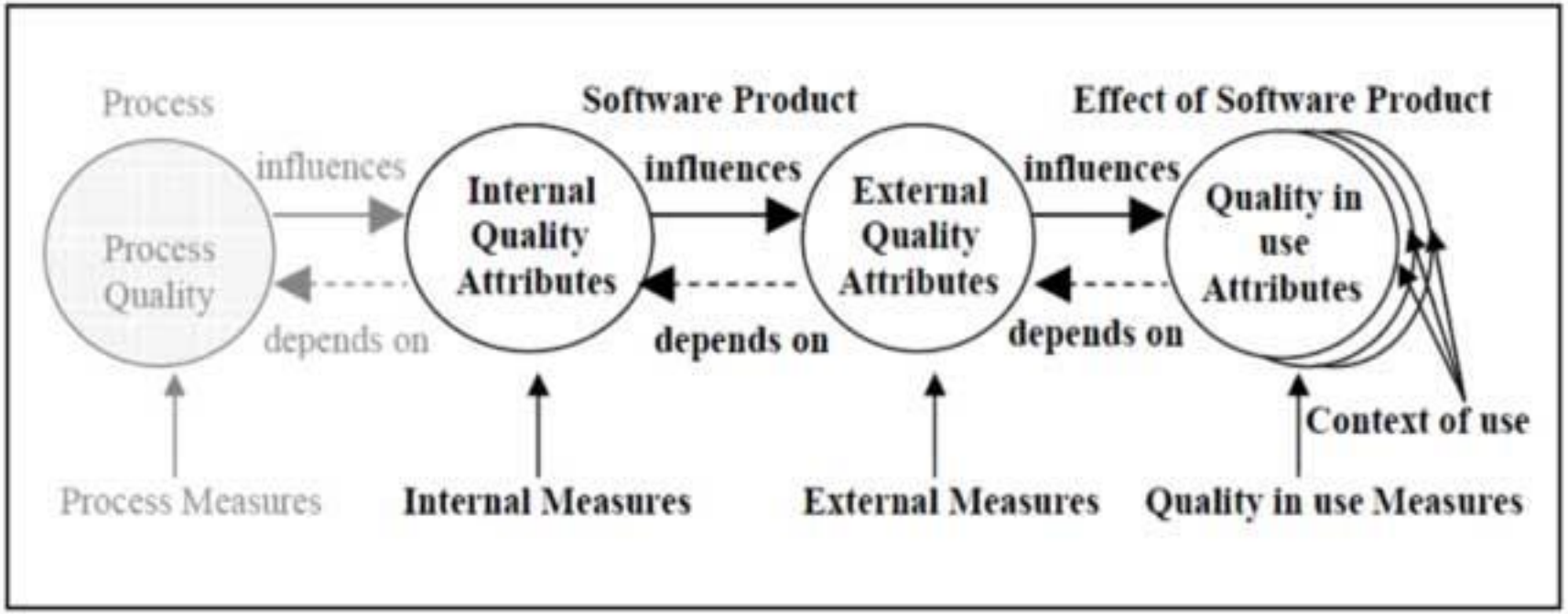
‘IEEE-Std 830 - 1993’ lists 13 non-functional requirements to be included in a *Software Requirements Document*:

- Performance requirements
- Interface requirements
- Operational requirements
- Resource requirements
- Verification requirements
- Acceptance requirements
- Documentation requirements
- Security requirements
- Portability requirements
- Quality requirements
- Reliability requirements
- Maintainability requirements
- Safety requirements

[Cremers and Alda 2006]

ISO/IEC 25010: Product Quality - Overview

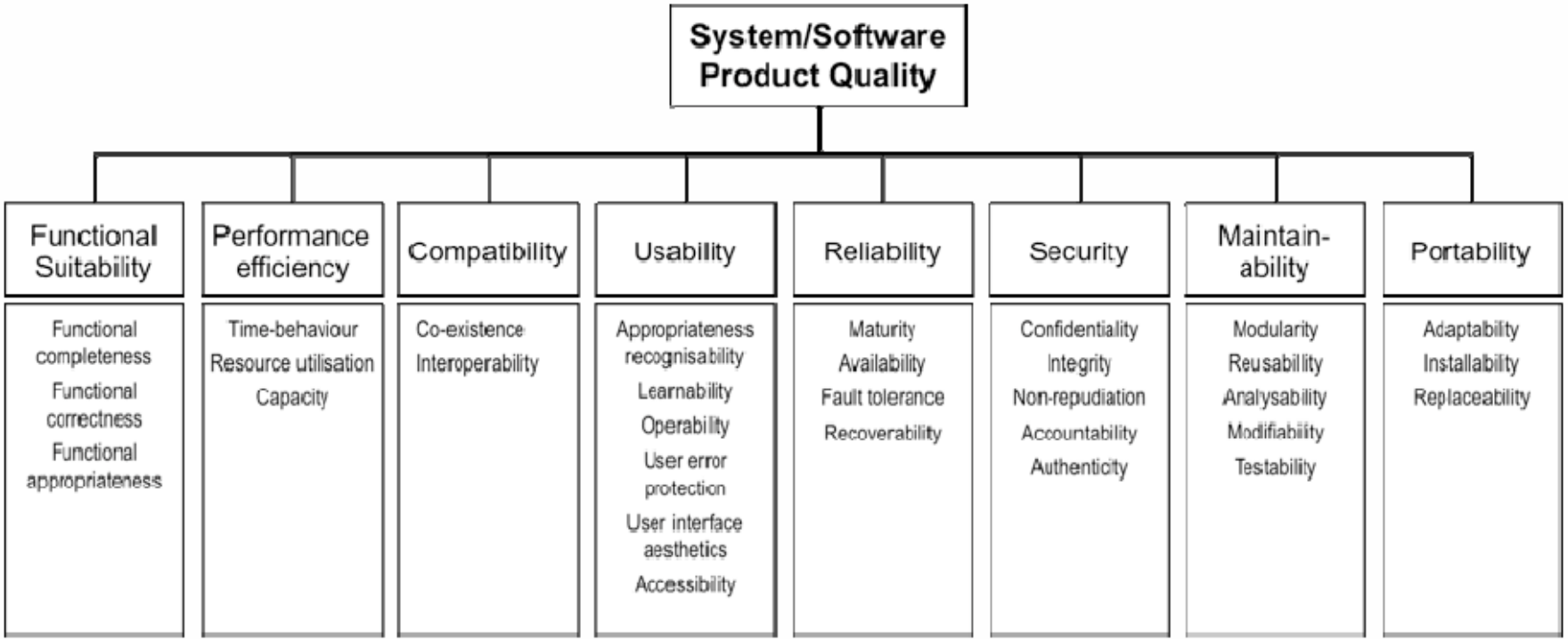
Quality in the Product Life Cycle



[ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Systems and software quality models]

ISO/IEC 25010: System/Software Product Quality

ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Systems and software quality models



ISO/IEC 25010: Product Quality – Def.s (1/4)

- **Functional suitability:** degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions
  - **Functional completeness:** degree to which the set of functions covers all the specified tasks and user objectives
  - **Functional correctness:** degree to which a product or system provides the correct results with the needed degree of precision
  - **Functional appropriateness:** degree to which the functions facilitate the accomplishment of specified tasks and objectives

- **Performance efficiency:** performance relative to the amount of resources used under stated conditions
  - **Time behaviour:** degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements
  - **Resource utilization:** degree to which the amounts and types of resources used by a product or system when performing its functions, meet requirements
  - **Capacity:** degree to which the maximum limits of a product or system parameter meet requirements

ISO/IEC 25010: Product Quality – Def.s (2/4)

- **Compatibility:** degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment
  - **Co-existence:** degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product
  - **Interoperability:** degree to which two or more systems, products or components can exchange information and use the information that has been exchanged
- **Usability:** degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use
  - **Appropriateness recognizability:** degree to which users can recognize whether a product or system is appropriate for their needs
  - **Learnability:** degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use
  - **Operability:** degree to which a product or system has attributes that make it easy to operate and control
  - **User error protection:** degree to which a system protects users against making errors
  - **User interface aesthetics:** degree to which a user interface enables pleasing and satisfying interaction for the user
  - **Accessibility:** degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use

ISO/IEC 25010: Product Quality – Def.s (3/4)

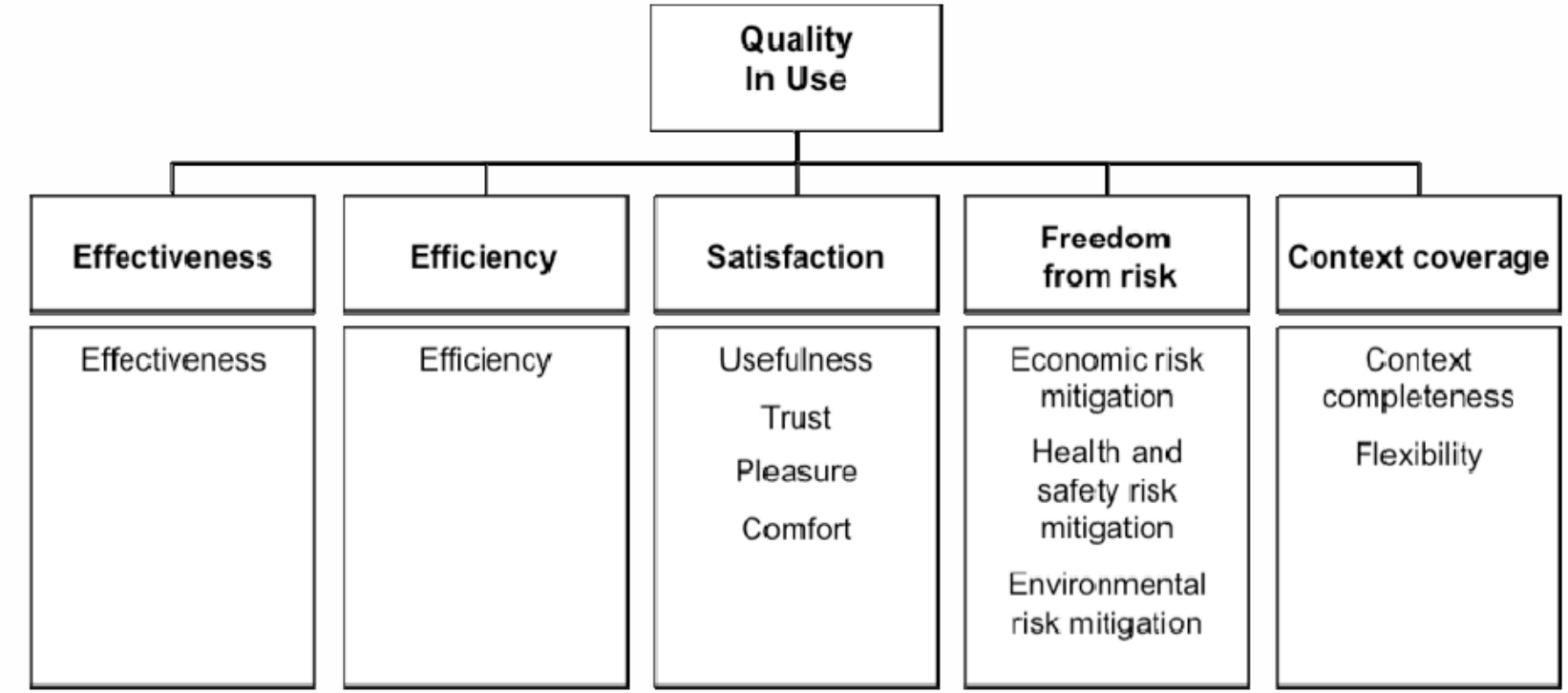
- **Reliability:** degree to which a system, product or component performs specified functions under specified conditions for a specified period of time
  - **Maturity:** degree to which a system meets needs for reliability under normal operation
  - **Availability:** degree to which a system, product or component is operational and accessible when required for use
  - **Fault tolerance:** degree to which a system, product or component operates as intended despite the presence of hardware or software faults
  - **Recoverability:** degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system
- **Security:** degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization
  - **Confidentiality:** degree to which a product or system ensures that data are accessible only to those authorized to have access
  - **Integrity:** degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data
  - **Non-repudiation:** degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later
  - **Accountability:** degree to which the actions of an entity can be traced uniquely to the entity
  - **Authenticity:** degree to which the identity of a subject or resource can be proved to be the one claimed

ISO/IEC 25010: Product Quality – Def.s (4/4)

- **Maintainability:** degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers
  - **Modularity:** degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components
  - **Reusability:** degree to which an asset can be used in more than one system, or in building other assets
  - **Analysability:** degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified
  - **Modifiability:** degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality
  - **Testability:** degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met
- **Portability:** degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another
  - **Adaptability:** degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments
  - **Installability:** degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment
  - **Replaceability:** degree to which a product can be replaced by another specified software product for the same purpose in the same environment

ISO/IEC 25010: Quality in Use

ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Systems and software quality models



ISO/IEC 25010: Quality in Use – Def.s (1/2)

- **Effectiveness:** accuracy and completeness with which users achieve specified goals
- **Efficiency:** resources expended in relation to the accuracy and completeness with which users achieve goals
- **Satisfaction:** degree to which user needs are satisfied when a product or system is used in a specified context of use
  - **Usefulness:** degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use
  - **Trust:** degree to which a user or other stakeholder has confidence that a product or system will behave as intended
  - **Pleasure:** degree to which a user obtains pleasure from fulfilling their personal needs
  - **Comfort:** degree to which the user is satisfied with physical comfort

ISO/IEC 25010: Quality in Use – Def.s (2/2)

- **Freedom from risk:** degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment
  - **Economic risk mitigation:** degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use
  - **Health and safety risk mitigation:** degree to which a product or system mitigates the potential risk to people in the intended contexts of use
  - **Environmental risk mitigation:** degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use
- **Context coverage:** degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified
  - **Context completeness:** degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use
  - **Flexibility** degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements

Outline

Quality Requirements, Requirements and Architecture

- ✓ Introduction (Requirements vs. Architecture; Definitions)
- ✓ Non-functional Requirements
- **Component-Bus-System-Property Approach**
- Quality Requirements and Architecture: Quality Attribute Workshop
- Optional: Quality Requirements – A New Look at an Old Problem (Martin Glinz)

- Exercise 4: Architectural Drivers and Requirements

CBSP Resources

H. Vogl, K. Lehner, P. Grünbacher, A. Egyed, "Reconciling Requirements and Architectures: Using the CBSP approach in an iPhone App Project", In: Proc. 19th IEEE Int'l Requirements Engineering Conference (RE 2011), IEEE Computer Society, Trento, Italy, pp. 273-278, 2011.

P. Grünbacher, A. Egyed, N. Medvidovic, "Reconciling software requirements and architectures with intermediate models", In: Software and Systems Modeling, vol. 3, no. 3, pp. 235-253, 2004.

Example: Catalysts taskmind App Project

Develop *iPhone and iPad client for* existing desktop and web products

- Create and share todo lists with team
- Organize tasks in projects
- Schedule tasks
- Group tasks
- Chat about tasks
- History and Documentation

~7000 users
~3,000 projects
~85,000 tasks and appointments
~230 KLOC

- iOS App and platform extensions
  - ~30,000 Lines of Code
    (Platform: ~18,000, iOS/UI: ~12,000)
- ~300 App Store downloads per week

www.taskmind.net

- taskmind App available for Android and Windows Phone

taskmind Requirements

Examples

- F1. Login with existing account
- F27. View news and daily journal in offline mode
- …
- N7. Synchronization rate 20 seconds
- N8. App startup time maximum 2 seconds
- N15. Minimize amount of data transfer
- …

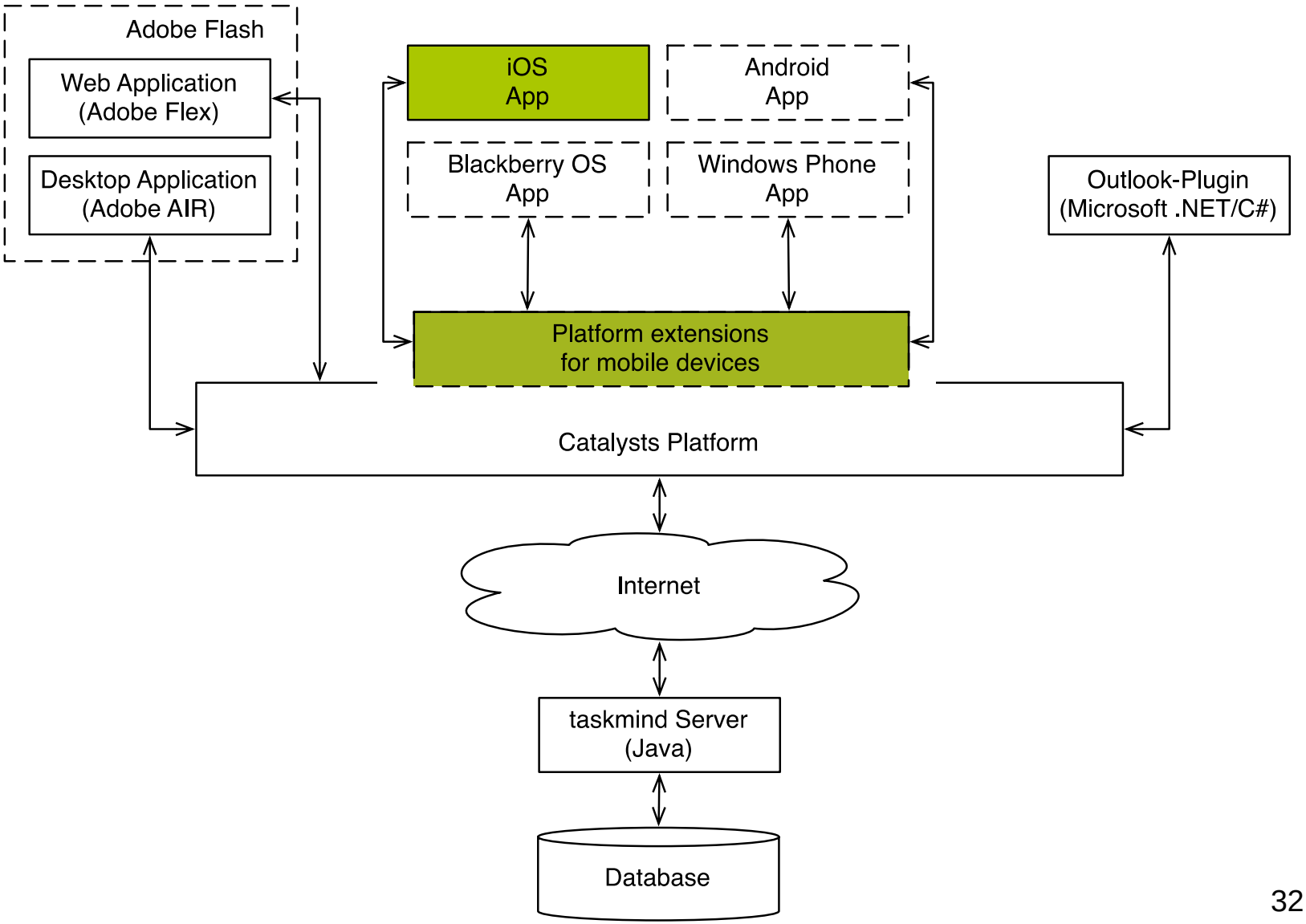
- Elicited 29 (F)unctional and 25 (N)on-functional requirements using Persona Profiles [1] and Scenario-based RE [2]
- Documented using the Volere Template [3]

[1] M. Aoyama. Persona-and-Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products. IEEE Int'l Conference on Requirements Engineering, pages 85–94, 2005.

[2] N. Seyff, N. Maiden, K. Karlsen, J. Lockerbie, P. Grünbacher, F. Graf, and C. Ncube. Exploring how to use scenarios to discover requirements. Requir. Eng., 14:91–111, April 2009.

[3] S. Robertson and J. Robertson. Mastering the Requirements Process (2nd Edition). Addison-Wesley Professional, 2006.

Adaption and Extension of Existing Architecture



Software Architecture Foundations

- **Architectural Elements**
  - processing elements (perform transformation on data elements)
  - data elements (contain the information that is used and transformed)
  - connecting elements (the glue that holds the different pieces of the architecture together; e.g., procedure calls, shared data, messages)
- **Form**
  - Weighted properties and relationships
    - Properties define constraints on the elements
    - Relationships constrain the “placement” of architectural elements
- **Rationale**
  - Motivation for the various choices made

D.E. Perry and A.L. Wolf. Foundations for the Study of Software Architecture. *ACM SIGSOFT Software Engineering Notes*, 17:4 (October 1992).

Rationale and Overall Approach

- A simple process for relating requirements and architectures

- Requirements explicitly or implicitly contain architecturally relevant information
- Use simple taxonomy to bring forth this information
- Classify and refine requirements according to taxonomy

CBSP Elements (1/2)

- **C: Components** (Cd: data comp., Cp: processing comp.)
  - F11: Editing Tasks, F…
    → Cd: Data for tasks
  - F12-13: Different task states
    → Cp: State transition component
- **B: Buses** (i.e., connectors)
  - F1: Login with existing user, F…
    → B: Communication protocol
  - F26: Synchronize offline items when online again
    → B: Connector between offline component and file system
- **S: (Sub-)Systems**
  - F4-6: List tasks per contact, project, …
    → S: Strict separation of data and visualization

CBSP Elements (2/2)

- **CP: Component Property**

  N7: Synchronization rate 20 seconds
  → Cp: Data refresh component
  → CP: Maximum delay of 20 seconds

  N27: Offline task support
  → Cd: Data for tasks
  → CP: persistent

- **BP: Bus Property**

  N15: Minimize amount of data transfer
  → B: communication protocol
  → BP: small amount of data

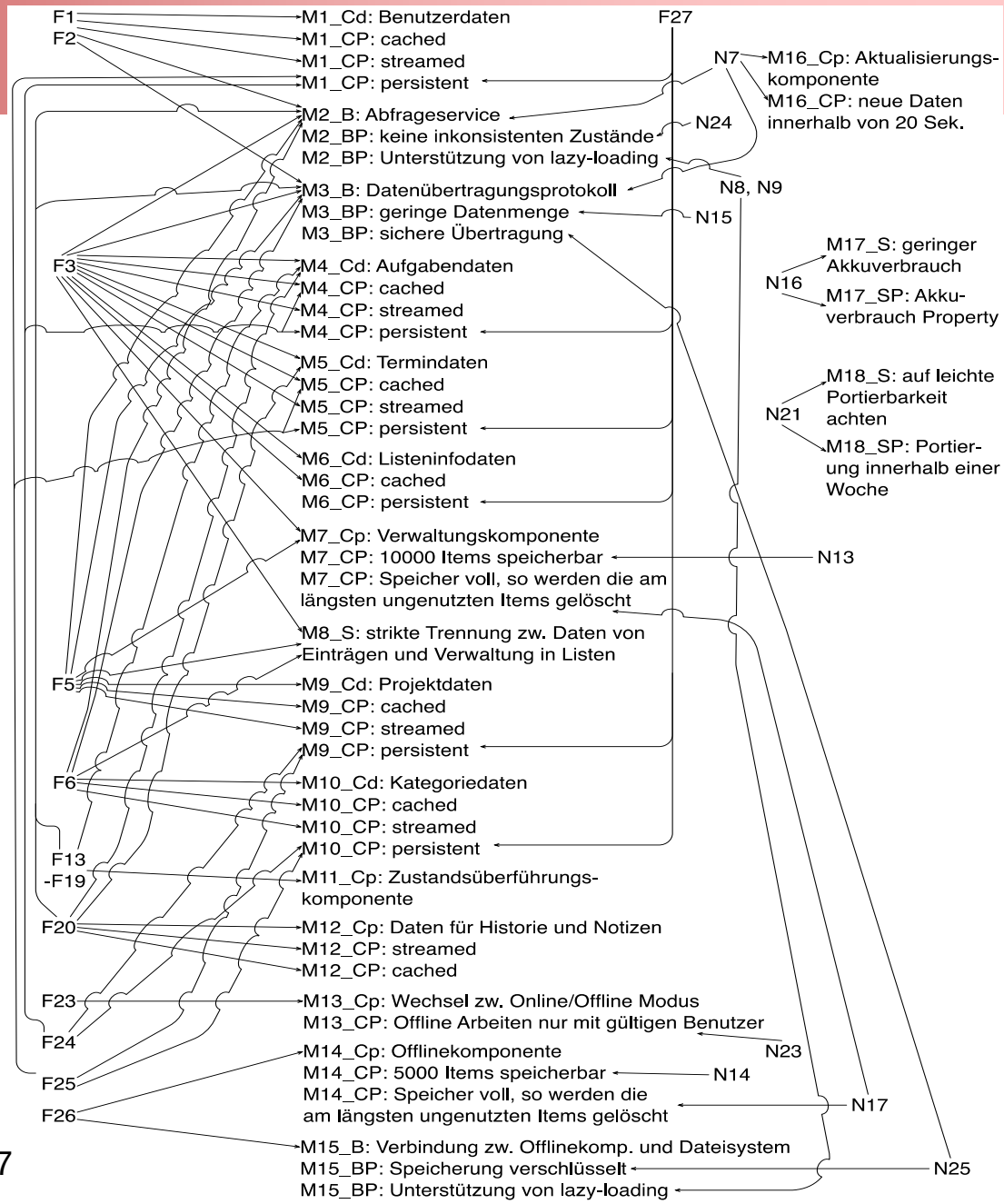
- **SP: System Property**

  N25: Secure transfer and storage
  → SP: data must be securely transferred and stored

taskmind CBSP Model

- **Metrics**
  - 12 C (6 Cd, 6 Cp)
  - 25 CP
  - 3 B
  - 6 BP
  - 3 S
  - 2 SP

- **How can we find adequate architectural styles?**



Assessing Architectural Options

- Ad-hoc definition of pro's and con's of architectural options that fit CBSP elements often too complex

  What is the most adequate architectural option?

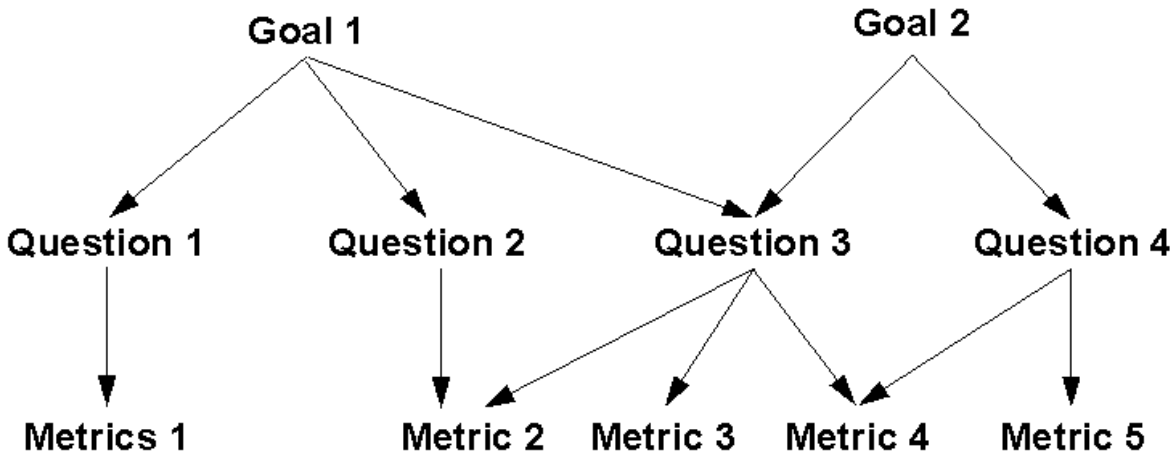
  → Assess important goals and metrics
  e.g. using Goal-Question-Metric Approach (GQM)

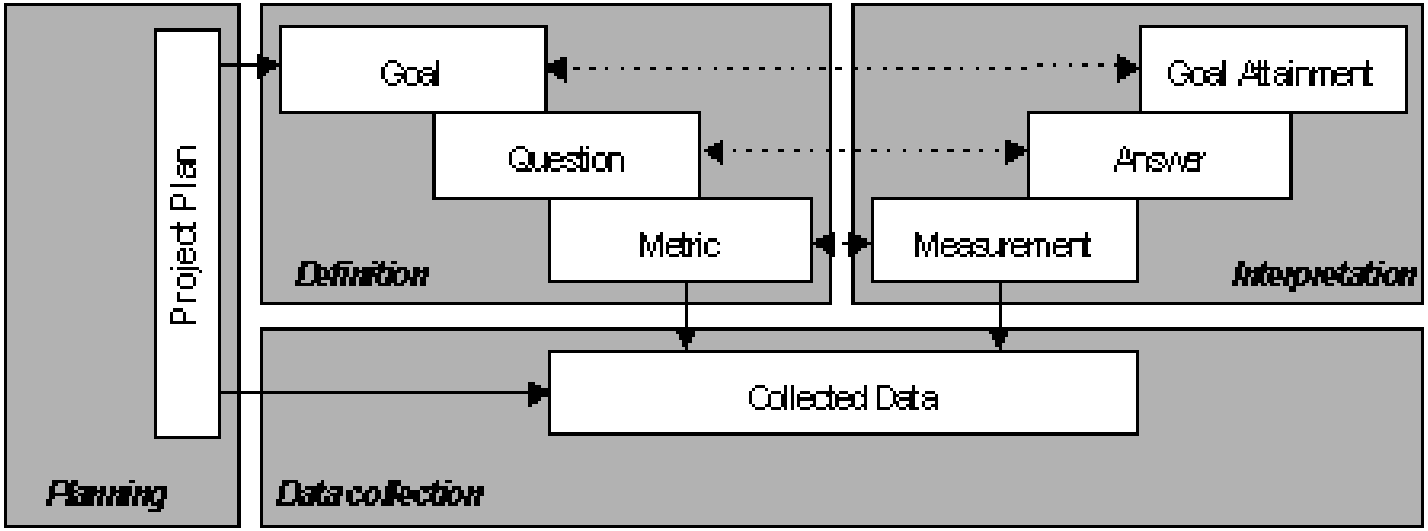
- Example Requirements:
  - Minimize amount of data transfer
  - Startup within 2 seconds

  → B: communication protocol,…

  → BP: small amount of data, fast processing

Goal-Question-Metric Approach (GQM)





Basili V.R., Caldiera G., Rombach H.D., Goal Question Metric Paradigm, In: J. J. Marciniak (ed.), Encyclopedia of Software Engineering 1, New York: John Wiley & Sons, pp. 528-532, 1994.

Goal-Question-Metric Model

Goals	Questions	Metric
Purpose: Compare *Issue:* Amount of data *Object:* Different data formats for persisting of data objects *Viewpoint:* Software architect	Differences with respect to the amount of data?	Data that needs to be transferred (in bytes).
Purpose: Compare *Issue:* Performance *Object:* Different data formats for persisting of data objects *Viewpoint:* Software architect	Differences with respect to serialization and deserialization time?	Time for (de-) serializing objects (in ms).

Assessing Architectural Options: CBSP and GQM

Data format	Bytes transferred	Serialize (ms) (Run 1/2/3)	Deserialize (ms) (Run 1/2/3)
Custom (binary)	360,504	114.8 116.4 135.9	209.5 209.5 184.2
Hessian (binary)	950,696	1171.6 1,150.1 1,158.4	4,284.4 4,280.2 4,254.9
JSON (text)	766,001	2,015.1 1,959.7 1,992.5	4,619.5 4,328.4 4,797.9
XML (text)	1,250,081	3,778.2 3,819.1 3,796.7	6,128.8 6,101.8 6,098.6

N8. App startup time maximum 2 seconds

Feedback to requirements regarding feasibility

Requirement: Minimize amount of data transfer
→ B: communication protocol
→ BP: small amount of data

GQM suggests new architectural component to support custom format

Relating the CBSP Model to the Architecture

CBSP Elements	Architecture Component
M1_Cd: data for users, M4_Cd: data for tasks, M5_Cd: data for appointments, M6_Cd: data for lists, M9_Cd: data for projects, M10_Cd: data for tags, M12_Cd: data for change history and notes	Data Objects
M7_Cp: management component, M8_S: strict separation of data and visualization	Proxies
M11_Cp: task state transition component	State Machine
M2_B: query service	Services
M2_BP: lazy loading, M3_BP: small amount of data	Input Handler, Output Handler
M3_B: communication protocol	Communication Protocol
M13_Cp: Switch between offline and online mode, M14_Cp: Offline component, M15_B: Connection between offline component and file system	Offline Management
M16_Cp: data refresh component	Auto-Update
M17_S: low power consumption, M18_S: attention on simple portability	System wide feature affecting all components

Group Discussion

- Discuss in groups the aspects of using the CBSP approach in practice

- Try to answer the following questions:
  - Do you regard the application of the CBSP approach in practice as:
    - Highly feasible
    - Satisfactorily feasible
    - Rather infeasible
    - Not feasible at all
  - What do you think are the main inhibitors and drawbacks of applying the CBSP approach in practice?

Lessons Learned

- Use of CBSP dimensions
  - Lightweight methodology
  - Improved traceability between requirements and architecture
- CBSP in the presence of an existing architecture
  - Also suitable for evolving an existing architecture
  - Identifying CBSP elements led to identify necessary changes of the original architecture
- Understanding architectural options
  - More CBSP properties than non-functional requirements
  - CBSP and GQM are a good fit
- Flexibility with respect to the choice of requirements elicitation method
  - Personas and scenario walkthroughs provided good input
  - CBSP helps to complete the requirements
- CBSP tool support and visualization
  - Lack of support for visualizing CBSP elements and relationships
  - Maintaining the CBSP model is also cumbersome

Outline

Quality Requirements, Requirements and Architecture

- ✓ Introduction (Requirements vs. Architecture; Definitions)
- ✓ Non-functional Requirements
- ✓ Component-Bus-System-Property Approach
- **Quality Requirements and Architecture: Quality Attribute Workshop**
- Optional: Quality Requirements – A New Look at an Old Problem (Martin Glinz)

- Exercise 4: Architectural Drivers and Requirements

Quality Attribute Workshop (QAW) - Overview

- **Quality Attribute Workshop, 3rd Edition:**
  - "... method that engages system stakeholders early in the life cycle to discover the driving quality attributes of a software-intensive system. The QAW [...] provides a way to identify important quality attributes and clarify system requirements before the software architecture has been created."

- **Scope:**
  - Creation of prioritized and refined scenarios

M. R. Barbacci, R. Ellison, A. J. Lattanze, J. A. Stafford, C. B. Weinstock, W. G. Wood, **Quality Attribute Workshops (QAWs), Third Edition,** *August 2003,* TECHNICAL REPORT, CMU/SEI-2003-TR-016, ESC-TR-2003-016

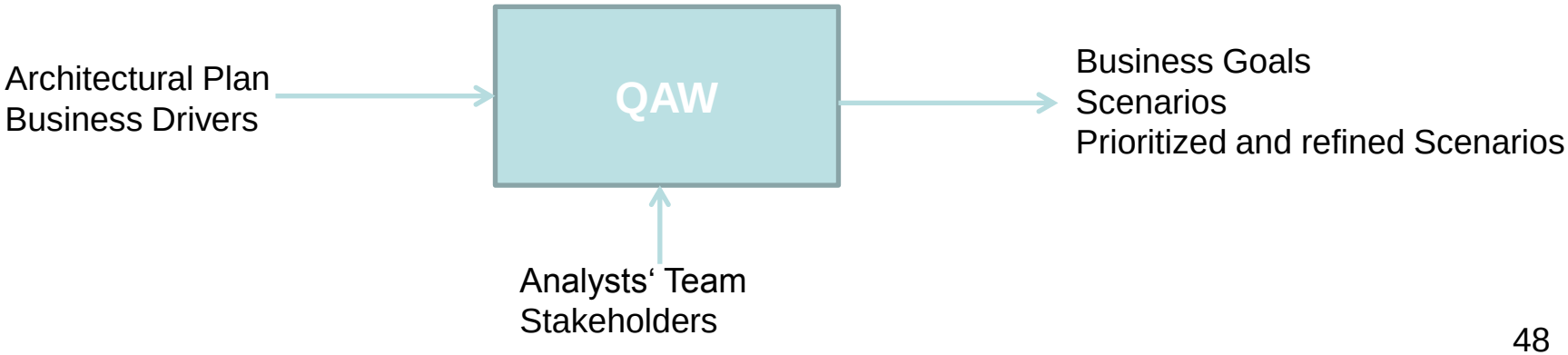
Results and Use of Results

- **QAW Results**
  - List of architectural drivers
  - Prioritized list of raw scenarios
  - Refined scenarios

- **Results can be used to ...**
  - Update architectural vision
  - Refine system and software requirements
  - Guide the development of prototypes
  - Exercise simulations
  - Understand and clarify the system's architectural drivers
  - Influence the order in which the architecture is developed
  - Describe the operation of a system

QAW Method Steps

- Step 1: QAW Presentation and Introductions
- Step 2: Business/Mission Presentation
- Step 3: Architectural Plan Presentation
- Step 4: Identification of Architectural Drivers
- Step 5: Scenario Brainstorming
- Step 6: Scenario Consolidation
- Step 7: Scenario Prioritization
- Step 8: Scenario Refinement



Step 1: QAW Presentation and Introductions

- Moderator / Moderator Team presents motivation for the workshop
- Moderator explains method steps
- Introduction of stakeholders
  - Role within organization
  - Relationship to system under development

- Typical stakeholders of a software systems (Examples):
  - Architect
  - Developer
  - End user
  - Maintainer
  - Administrator
  - Trainer
  - Persons involved in installation, delivery, logistics, planning, acquisition, etc.

Step 1: QAW Presentation and Introductions – Template

Name	Organisation	Represented Role(s)

Step 2: Business/Mission Presentation

- A representative of the stakeholders (typically a manager or management representative) presents the business and/or mission drivers for the system
  - the system's business/mission context
  - high-level functional requirements, constraints, and quality attribute requirements
- During the presentation, the moderators listen carefully and capture any relevant information that may shed light on the quality attribute drivers
- The quality attributes that will be refined in later steps will be derived largely from the business/mission needs presented in this step

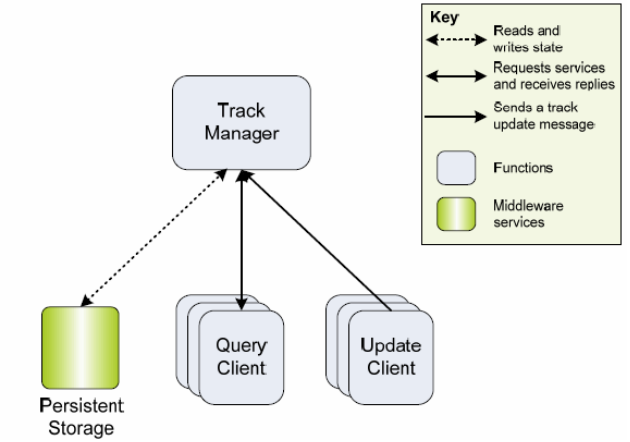
Step 2: Business/Mission Presentation - Details

- Business drivers typically describe
  - Most important functional requirements
  - Technical constraints (e.g. COTS, linkage with other systems, platforms, reuse of legacy)
  - Economic, inner-organizational or political constraints
  - Business goals
  - Business context
  - Most important stakeholders
  - Most important quality attributes influencing architecture

[Kazman et al. 2000]

Step 2: Business/Mission Presentation – Example

- **Funktionale Anforderungen**
  - Der **Track Manager (Fahrweg-/Spur-Manager?)** bietet einen Tracking-Service für zwei Typen von Clients an:
  - **Update client**: Diese Clients senden regelmäßig Track-Updates an den Track-Manager
  - **Query client**: Diese Clients fragen den Track Manager sporadisch ab und bekommen genau eine Antwort auf eine Anfrage.



- **Randbedingungen (Design Constraints)**
  - **Kapazitätseinschränkungen:** Prozessoren sollen 50% Prozessor- und Speicher-Reserven haben bei Auslieferung. LAN soll 50% Durchsatz-Reserven haben. Es gibt 100 Update- und 25 Query-Clients und ca. 100 Updates und 5 Anfragen pro Sekunde.
  - **Persistenter Speicherdienst:** Dienst unterhält eine Statuskopie des Track Managers, welche mindestens einmal pro Sekunde aktualisiert wird. Sollten alle Replikas ausfallen, kann von einem vorherigen Status neu gestartet werden.
  - **Zwei Replikas:** Für Verfügbarkeit und Zuverlässigkeit sollen zwei Replikas des Track Managers unter Normalbedingungen in Betrieb sein.

- **Anforderungen an Qualitätsattribute** (3 wichtige Szenarien schon vorher identifiziert)
  - **"Quick Recovery"** nach Hardware- oder Software-Defekt, zweites Replika des Track-Managers übernimmt
  - **"Slow Recovery"** nach Hardware- oder Software-Defekt; keine Replikas sofort verfügbar, neues Replika des Track-Managers muss erstellt werden
  - **"Re-Start"** unter normalen Betriebsbedingungen arbeitet nur ein Replika und ein Zweites wird hinzugefügt

[Wood 2007]

Step 3: Architectural Plan Presentation

- A technical stakeholder presents the system architectural plans as they stand with respect to these early documents. — Information in this presentation may include
  - plans and strategies for how key business/mission requirements will be satisfied
  - key technical requirements and constraints — such as mandated operating systems, hardware, middleware, and standards — that will drive architectural decisions
  - existing context diagrams, high-level system diagrams, and other written descriptions
- During this time, moderators continue to capture key aspects of the presentation for later reference

Step 3: Architectural Plan Presentation - Details

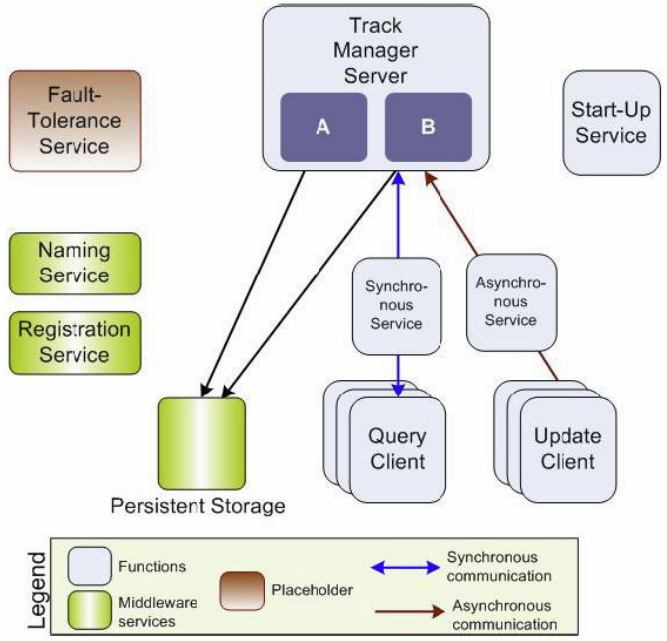
- Important architectural requirements
  - E.g. performance, availability, incl. corresponding measures
  - Existing standards/models/approaches to fulfill requirements
- High-level views on the architecture
  - Functional
  - Modules/Layers/Subsystems
  - Processes, threads and synchronization, dataflows, events
  - Hardware: CPUs, memory, external devices or sensors, networks and communication devices
- Architectural approaches and styles
  - E.g. Client-Server, Blackboard, Pipes and Filters
- Use of COTS, e.g. for reporting, GUI
- 1-3 of the most important Use Case Scenarios
- 1-3 of the most important change scenarios
- Architectural risks and problems

[Kazman et al. 2000]

Step 3: Architectural Plan Presentation – Example

Track Manager wurde in zwei Elemente A und B geteilt
→ erlaubt zwei Strategien:

- Strategie 1: A und B laufen auf einem einzelnen Prozessor, A und B verbrauchen 50% der Prozessorkapazität für 100 Updates und 30 Anfragen, damit werden die Performance-Anforderungen befriedigt
- Strategie 2: A und B laufen auf jeweils einem eigenen Prozessor, gemeinsam können sie 150 Updates und 50 Anfragen bedienen, damit werden die Performance-Anforderungen übererfüllt.



- Kommunikationsmechanismen unterscheiden sich:
  - Update Clients: **asynchrone Kommunikation**
  - Query Clients: **synchrone Kommunikation**

- Elemente A und B enthalten beide Statusdaten, die im Persistenzspeicher gesichert werden müssen.
  Zeitvorgaben für Sicherung und Wiederherstellung des Status: A - 0,8 sek., B - 0,6 sek.
- ...

[Wood 2007]

Step 4: Identification of Architectural Drivers

- The moderators share their list of key architectural drivers and ask the stakeholders for
  - Clarifications
  - Additions
  - Deletions
  - Corrections
- The idea is to reach a consensus on a distilled list of architectural drivers that include high-level requirements, business drivers, constraints, and quality attributes
- The final list of architectural drivers will help focus the stakeholders during scenario brainstorming to ensure that these concerns are represented by the scenarios collected

Step 4: Sample Architectural Drivers

E.g. Quality Model according ISO/IEC 25010

... but also: functional requirements, design constraints, business/mission aspects (e.g. decision for product line approach), other goals or combinations thereof influencing architecture

Step 5: Scenario Brainstorming (1/2)

- The moderators initiate the brainstorming process in which stakeholders generate scenarios
- Each stakeholder expresses a scenario representing his or her concerns with respect to the system in round-robin fashion
- The moderators ensure that representative scenarios exist for each architectural driver listed in Step 4
- The moderators review the parts of a good scenario (stimulus, environment, and response) and ensure that each scenario is well formed during the workshop

Step 5: Scenario Brainstorming (2/2)

- Moderators need to remember that there are three general types of scenarios and to ensure that each type is covered during the QAW:
  - **use case** scenarios - involving anticipated uses of the system
  - **growth** scenarios - involving anticipated changes to the system
  - **exploratory** scenarios - involving **un**anticipated stresses to the system that can include uses and/or changes
- Moderators should note that quality attribute names by themselves are not enough. Rather than say “the system shall be modifiable,” the scenario should describe what it means to be modifiable by providing a specific example of a modification to the system vis-à-vis a scenario

Step 5: Scenario Brainstorming - Scenario Format

Scenario-Format: **Stimulus – Context – Response**

Scenario Information	Exploratory Questions
Non-functional requirement	Which category of non-functional requirement does the scenario come under?
Change (stimulus)	What is the specific event or input that needs to be considered? Who initiates it (e.g., developer, manager, end user, another system)?
Environment (context)	When or where does the change occur (e.g., during normal operation, degraded operation, faulted; at design time, build time, run time)?
Anticipated work (response)	What happens after the change is initiated? What gets acted upon (e.g., architecture, design spec, code, user interface, development schedule)?

Step 5: Scenario Brainstorming – Examples (1/2)

- Sample scenarios: Example: Bank ATM Quality Attribute Workshop
  - **Modifiability** Attribute Scenario I:
    - *A developer wants to add a new auditing business rule at design time in 10 person-days without affecting other functionality*
  - **Modifiability** Attribute Scenario II:
    - *A system administrator wants to employ a new database in 18 person-months without affecting other functionality*

- Enhancement of a requirement towards a scenario:
  - **Requirement**: *"The system shall produce reports for users."*
  - **Scenario**: *"A remote user requests a database report via the Web during peak usage and receives the report within five seconds."*
  - ➢ Scenario sheds more light on the performance aspect of the requirement
  - ➢ Initial requirement has not been lost, but the scenario further explores the performance aspect of this requirement

Step 5: Scenario Brainstorming – Examples (2/2)

- Use case scenario
  - Remote user requests a database report via the Web during a peak period and receives it within 5 seconds.
- Growth scenario
  - Add a new data server to reduce latency in scenario 1 to 2.5 seconds within 1 person-week.
- Exploratory scenario
  - Half of the servers go down during normal operation without affecting overall system availability.

[Gagliardi_and_Wood 2011]

Step 6: Scenario Consolidation

- Similar scenarios are consolidated when reasonable. Moderators ask stakeholders to identify those scenarios that are very similar in content
- Scenarios that are similar are merged, as long as the stakeholders who proposed them agree and feel that their scenarios will not be diluted in the process
- Consolidation is an important step because it helps to prevent a “dilution” of votes during the prioritization of scenarios. Such a dilution occurs when stakeholders split their votes between two very similar scenarios. As a result, neither scenario rises to importance and is therefore never refined
- Moderators should make every attempt to reach a majority consensus with the stakeholders before merging scenarios. Though stakeholders may be tempted to merge scenarios with abandon, they should not do so
- Typically, very few scenarios are merged

Step 7: Scenario Prioritization

- Prioritization of the scenarios is accomplished by allocating each stakeholder a number of votes equal to approx. 30% of the total number of scenarios generated after consolidation
  - For example, if 30 scenarios were generated, each stakeholder gets 30 x 0.3, or 9, votes rounded up to 10

- Voting is done in round-robin fashion, in two passes. During each pass, stakeholders allocate half of their votes. Stakeholders can allocate any number of their votes to any scenario or combination of scenarios. The votes are counted, and the scenarios are prioritized accordingly

Scenario#	Description	Votes
#1		
#2		
#3		
#4		
#5		
.....		

Step 7: Scenario Prioritization - Example

Enhancement to QAW:

- Ranking of scenarios according estimated „Importance“ and „Difficulty“

[Tsakiris et al. 2011]

Scenario Information	Exploratory Questions
Importance	Is the importance of the scenario (relative to other scenarios) to the system's success high, medium, or low? All other things being equal, how likely is this scenario to happen?
Difficulty	What's the relative degree of difficulty (relative to other scenarios) of addressing the scenario – high, medium, or low?

Scenario	Non-functional Requirement	Change	Environment	Anticipated Work	Acceptable Cost	Importance	Difficulty
1	Modifiability	Program Planning wants to change user interface style and format to make interface more attractive to user.	At design time.	None. No side effects on AEPC system - i.e., no changes required to control functions; no variant AEPC system artifacts created. HMI requirements, code, and design verification methods are modified.	Zero time. Zero person-months. Assumption: HMI system is separate from AEPC system.	Low	Low
3	Modifiability	Program Planning moves the system to a platform with a different electrical architecture.	At design time.	New network signals created and existing signals modified.	New network signals created and existing signals modified.	High	High
5	Modifiability	Program Planning wants to change user interface information content.	At design time.	New signals added to AEPC system's interface (outputs). New requirements, design verification methods created. Arbitrator logic revised.	3 calendar-months. 3 person-months. Assumption: HMI system is separate from AEPC system.	Medium	Medium
6	Modifiability	Program Planning changes one or more vehicle components (e.g. motor, engine, battery), i.e. component characteristics change.	At design time.	New AEPC calibration parameters created. Minor modifications to code. New AEPC parameters tested.	2 calendar-months. 2 person-months.	High	Medium
7	Performance	Other development projects require a change of the underlying Vehicle System Control architecture.	At design time.	New strategy developed and tested. Code revised and tested. Requirements, design verification methods, and failure mode effects analysis revised.	3 calendar-months. 3 person-months	Low	Medium
8	Time to delivery	Program Planning moves first delivery 6 months earlier (e.g., targets a different program or different model year).	At design time.	Add new people to project to test and calibrate. Design of system not affected.	0.5 person	High	High

→ Scenarios 3, 6, 8 have highest priority

Step 8: Scenario Refinement

- The top four or five scenarios are refined and documented in more detail
- Further clarify the scenario by clearly describing the following:
  - **stimulus** - the condition that affects the system
  - **response** - the activity that results from the stimulus
  - + source of stimulus - the entity that generated the stimulus
  - **environment** - the condition under which the stimulus occurred
  - + artifact stimulated - the artifact that was stimulated
  - + response measure - the measure by which the system's response will be evaluated
- Describe the **business/mission goals** that are affected by the scenario
- Describe the relevant **quality attributes** associated with the scenario
- Allow the stakeholders to pose questions and raise any issues regarding the scenario (-> quality attribute aspects of the scenario, concerns that the stakeholders might have in achieving the response called for)

Step 8: Scenario Refinement - Template

Scenario Refinement for Scenario N		
Scenario(s):		
Business Goals:		
Relevant Quality Attributes:		
Scenario Components	Stimulus:	
	Stimulus Source:	
	Environment:	
	Artifact (If Known):	
	Response:	
	Response Measure:	
Questions:		
Issues:		

[Barbacci et al. 2003]

Step 8: Scenario Refinement - Example

Scenario Refinement for Scenario N		
Scenario(s):		When a garage door opener senses an object in the door's path, it stops the door in less than one millisecond.
Business Goals:		safest system; feature-rich product
Relevant Quality Attributes:		safety, performance
Scenario Components	Stimulus:	An object is in the path of a garage door.
	Stimulus Source:	object external to system, such as a bicycle
	Environment:	The garage door is in the process of closing.
	Artifact (If Known):	system's motion sensor, motion-control software component
	Response:	The garage door stops moving.
	Response Measure:	one millisecond
Questions:		How large must an object be before it is detected by the system's sensor?
Issues:		May need to train installers to prevent malfunctions and avoid potential legal issues.

[Barbacci et al. 2003]

Real-world Example – Selected Steps' Results

High Performance Automation Domain

Re-Engineering of Shop Floor Management System

- **Step 4:** Identification of Architectural Drivers
  - Goal: Identify ranking of ISO/IEC 25010 quality attributes
  - Three Votings per Stakeholder:
    a) Simple Ranking
    b) Pairwise Comparison
    c) 100-Points-Method
  - Ranking through discussion between stakeholders

- **Step 7:** Scenario Prioritization
  - Goal: Scenario selection for refinement
  - Three Votings pro Stakeholder
    a) Simple Ranking
    b) Allocation of 12 points per stakeholder (total: 33 scenarios)
    c) Estimation of „Importance" and „Difficulty" per scenario
  - Prioritization of Top-5 scenarios through discussion between stakeholders

Real-world Example - Architectural Drivers Ranking

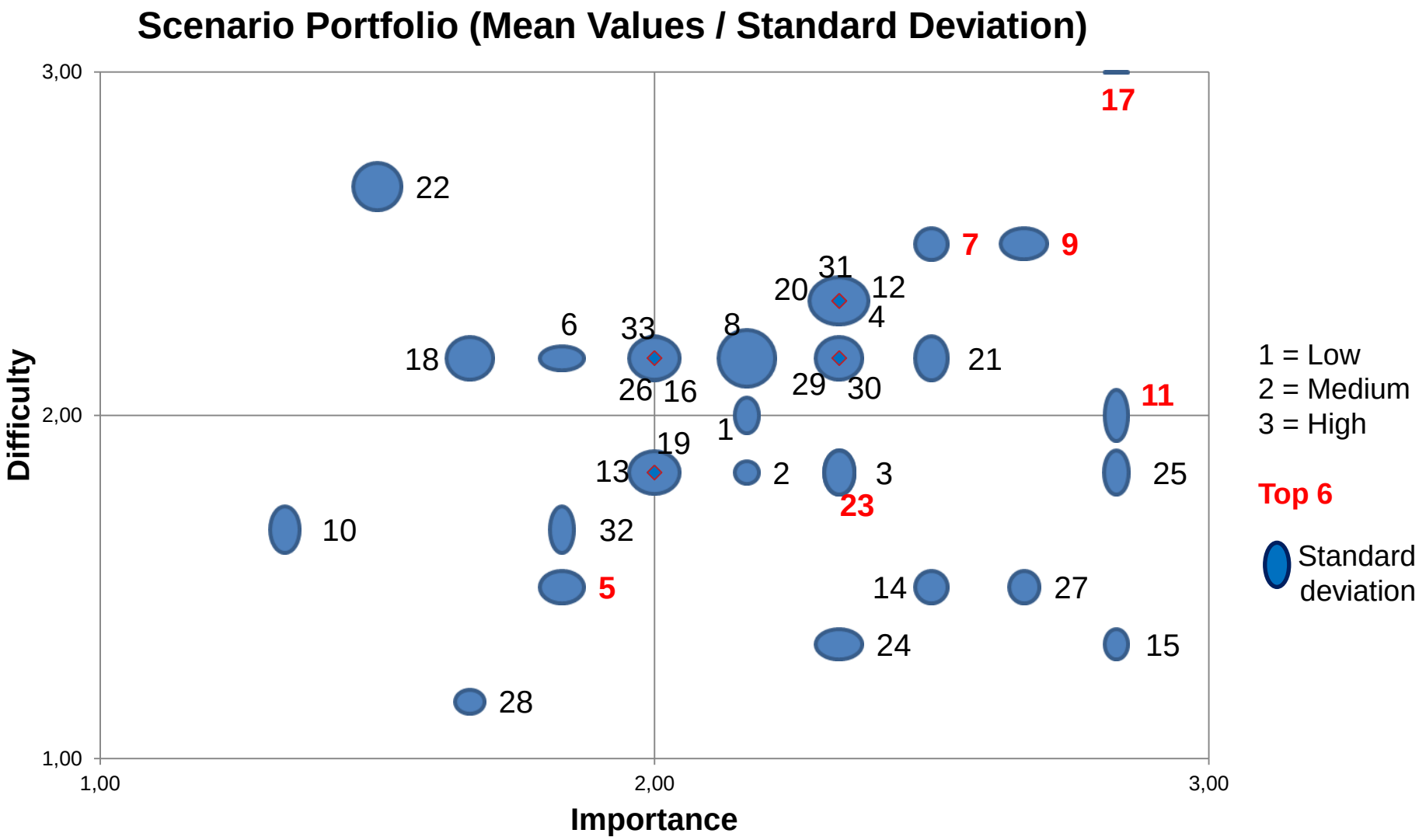
Architectural Driver (Quality Attribute)	Ranking (Stakeholder Consensus)	Simple Ranking (Median)	Pairwise Comparison (Median)	100-Points-Method (Median)
Functional suitability	+	1 (1)	1 (7)	1 (25)
Performance efficiency	-	6 (6)	6 (2)	6 (8)
Compatibility	+	4 (4,5)	4 (3,5)	3 (15)
Usability	+	3 (3,5)	3 (4)	4 (12,5)
Reliability	+	2 (2)	2 (6)	2 (20)
Security	-	7 (6,5)	7 (1,5)	7 (6)
Maintainability	+	4 (4,5)	5 (3)	5 (10)
Portability	-	8 (7)	8 (1)	8 (3,5)

Real-world Example - Consensus on Architectural Drivers

Correlation coefficients between stakeholders: Pairwise Comparison

	SW-Eng.-Mgmt	Prod. mgmt	Appl.-Eng.	QA	Sales	Arch.	Mean	Median
SW-Eng.-Mgmt		0,350	0,153	0,416	0,438	0,482	0,552	0,556
Prod. mgmt	0,350		0,354	0,833	0,500	0,833	0,770	0,806
Appl.-Eng.	0,153	0,354		0,458	0,646	0,354	0,585	0,517
QA	0,416	0,833	0,458		0,646	0,813	0,831	0,842
Sales	0,438	0,500	0,646	0,646		0,479	0,736	0,698
Arch.	0,482	0,833	0,354	0,813	0,479		0,788	0,806

Real-world Example - Scenario Prioritization



Real-world Example – Scenario Prioritization vs. Architectural Drivers Ranking

Architectural Driver (Quality Attribute)	Coverage through Scenarios	Total Points Assigned (via scenarios)	Simple Ranking (Median)	Pairwise Comparison (Median)	100-Points-Method (Median)
Functional suitability	6	7	1	1	1
Performance efficiency	3	9	6	6	6
Compatibility	3	7	4	4	3
Usability	2	4	3	3	4
Reliability	3	9	2	2	2
Security	1	1	7	7	7
Maintainability	14	33	4	5	5
Portability	1	2	8	8	8

Real-world Example - Consensus on Scenario Priorities (1/2)

Correlation coefficients between stakeholders

Method: Allocation of 12 Points

	SW-Eng.-Mgmt	Prod. mgmt	Appl.-Eng.	QA	Sales	Arch.	Mean	Median
SW-Eng.-Mgmt		0,037	0,037	0,449	0,477	0,220	0,716	0,743
Prod.mgmt	0,037		-0,147	0,046	0,109	0,032	0,338	0,183
Appl.-Eng.	0,037	-0,147		-0,026	-0,091	-0,170	0,177	0,083
QA	0,449	0,046	-0,026		0,126	0,505	0,671	0,505
Sales	0,477	0,109	-0,091	0,126		-0,020	0,491	0,629
Arch.	0,220	0,032	-0,170	0,505	-0,020		0,553	0,395

Real-world Example - Consensus on Scenario Priorities (2/2)

Correlation coefficients between stakeholders

Method: Allocation of 12 Points **(Sub-Set: High Priority Scenarios)**

	SW-Eng.-Mgmt	Prod. mgmt	Appl.-Eng.	QA	Sales	Arch.	Mean	Median
SW-Eng.-Mgmt		-0,066	-0,044	0,000	0,589	-0,179	0,729	0,787
Prod.mgmt	-0,066		-0,281	-0,456	0,373	-0,316	-0,132	0,050
Appl.-Eng.	-0,044	-0,281		-0,154	-0,251	-0,488	-0,267	-0,168
QA	0,000	-0,456	-0,154		0,000	0,248	0,361	0,136
Sales	0,589	0,373	-0,251	0,000		-0,404	0,589	0,668
Arch.	0,179	-0,316	-0,488	0,248	-0,404		0,000	-0,189

Outline

Quality Requirements, Requirements and Architecture

- ✓ Introduction (Requirements vs. Architecture; Definitions)
- ✓ Non-functional Requirements
- ✓ Component-Bus-System-Property Approach
- ✓ Quality Requirements and Architecture: Quality Attribute Workshop
- **Optional: Quality Requirements – A New Look at an Old Problem (Martin Glinz)**
- Exercise 4: Architectural Drivers and Requirements

Outline

Quality Requirements, Requirements and Architecture

- ✓ Introduction (Requirements vs. Architecture; Definitions)
- ✓ Non-functional Requirements
- ✓ Component-Bus-System-Property Approach
- ✓ Quality Requirements and Architecture: Quality Attribute Workshop
- ✓ Optional: Quality Requirements – A New Look at an Old Problem (Martin Glinz)

- **Exercise 4: Architectural Drivers and Requirements**

Exercise 4: Architectural Drivers and Requirements

Elicit, discuss, analyze, and prioritize the driving quality attributes and important system requirements for the example project „Web-based PISA Assessment“ as a major input for sub-sequent (fictitious) system implementation, in particular architecture alternatives elaboration and selection.

Build on the results of Exercise 1 (WinWin Negotiation) and Excercise 2 (Use Case Analysis) and use a role play approach to perform a **Quality Attribute Workshop** (QAW, 3rd edition) in order to identify architectural drivers, identify and select architecture relevant requirements and corresponding scenarios, and prioritize and refine scenarios.

Members of your team are expected to role-play key project and business stakeholders. Mandatory roles comprise business management (cf. step 2), software engineering/architecture engineering (cf. step 3), and a moderator role.

Perform the following steps:

1) QAW Presentation and Introductions
2) Business/Mission Presentation
3) Architectural Plan Presentation
4) Identification of Architectural Drivers
5) Scenario Brainstorming
6) Scenario Consolidation
7) Scenario Prioritization
8) Scenario Refinement

Document the steps, decisions and results of your role play and submit a report via TUWEL. Each group is expected to submit a pdf document named Exercise4Team<YourTeamNumber>. Please list all team members on the front page of your report.

Outline

Quality Requirements, Requirements and Architecture

- ✓ Introduction (Requirements vs. Architecture; Definitions)
- ✓ Non-functional Requirements
- ✓ Component-Bus-System-Property Approach
- ✓ Quality Requirements and Architecture: Quality Attribute Workshop
- ✓ Optional: Quality Requirements – A New Look at an Old Problem (Martin Glinz)

- ✓ Exercise 4: Architectural Drivers and Requirements