Karlsruhe Institute of Technology

aifb

CRITICAL INFORMATION INFRASTRUCTURES RESEARCHGROUP

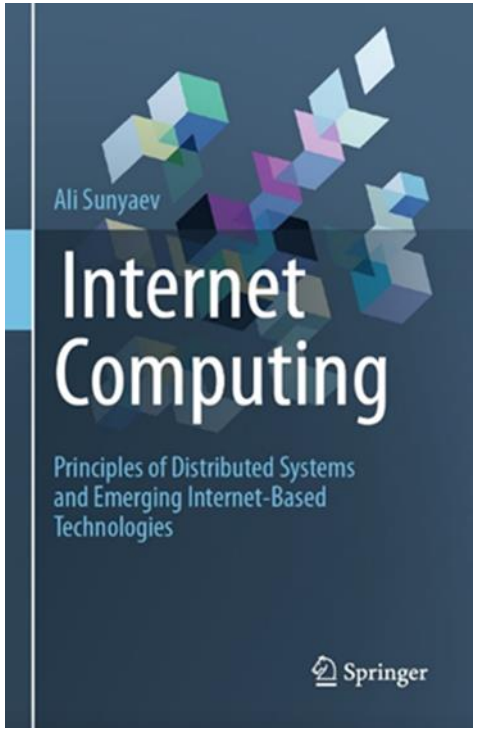
AI: Internet Computing

Lecture 3 — Design of Good Information Systems Architectures

Learning Goals of the Lecture

- To understand what constitutes a good IS architecture
- Being able to distinguish between two different design perspectives
- To understand how to measure and evaluate IS architectures' quality
- To know how the design process needs to be shaped to create an architecture that fulfills its intended purpose

Reference to the Teaching Material Provided



Chapter 3
Design of Good Information Systems Architectures

Abstract
Each information system (IS) has an underlying architecture, although its complexity and scope can vary quite substantially for differ kinds of systems. Since design decisions about the architecture define the very foundation of an IS, the design decisions cannot be easily undone or altered after they were made. If not taken seriously enough, improper IS architecture designs can result in the development of systems that are incapable of adequately meeting user requirements. Understanding the concept of good IS architecture design and taking design decisions diligently is, therefore, highly important for an IS development project's success. In order to answer the question of what constitutes a good IS architecture, this chapter examines the importance of design decisions across a system's lifecycle. In particular, two different perspectives on the concept of good IS architecture design are explicated: (1) design as the process and (2) design as the outcome of a design process. The two perspectives are closely related to each other and generally help explain the more abstract concept of IS architecture design and particularly the characteristics of a good IS architecture.

Learning Objectives of this Chapter
This chapter's main learning objective is to understand what constitutes a good IS architecture. After studying this chapter, the readers will be able to distinguish

Architecture Design

What is determining the architecture?

But:

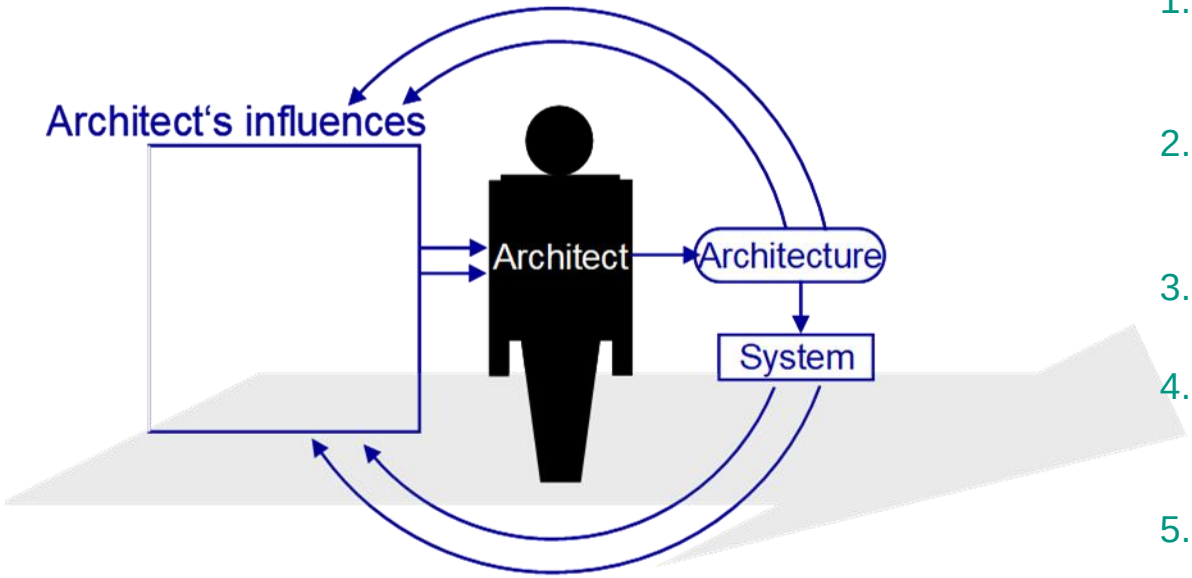
1. Contradicts experience
2. Dangerous

The Sinking of Vasa, 1628

- Extrapolations of the architecture have to be checked
- Architectures have to limit the requirements if necessary

Image source: Original illustrations by Bertil Almqvist for the book ”Sagan om Wasa” (“The Vasa Saga”) [Photograph]. https://www.instagram.com/p/CDtHHF1pE3f/

The Architecture Business Cycle



1. Architecture affects the structure of the developing organization
2. Architecture can affect the enterprise goals of the developing organization
3. Architecture can affect the customer requirements
4. The process of building the system will affect the architect‘s experience
5. A few systems will influence and actually change the software engineering culture and its technical environment

Image source: Figure according to Bass L., Clements P., Kazman R. (2012). *Software Architecture in Practice*. Addison-Wesley, London, UK

Revision:
What is an Information System Architecture?

Definition

Information Systems Architecture:

“Fundamental concepts or properties of an information system in its environment, as embodied in its elements and relationships, and in the principles of its design and evolution.”

ISO/IEC/IEEE, 2011

- System = application(s) + operating system + hardware (+ network)
- System architecture ≠ software architecture
- System engineers/architects are able to choose different system architectures
- This can often be done without changing the software architecture

Source: "ISO/IEC/IEEE Systems and software engineering -- Architecture description," in *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)* , vol., no., pp.1-46, 1 Dec. 2011, doi: 10.1109/IEEESTD.2011.6129467.

Architecture Design

- In the IS architecture context, the term „design“ has a dual meaning:

Definition

Process Perspective

- Architectural design refers to the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of an information system.
  IEEE, 1990

Outcome Perspective

- Architectural design refers to the architectural design process's outcome, that is the collection of hardware and software components and their interfaces, which makes up the framework for the development of an information system.
  IEEE, 1990

Architecture Design

- In the IS architecture context, the term „design“ has a dual meaning:

Definition

Process Perspective

- Architectural design refers to the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of an information system.
  IEEE, 1990

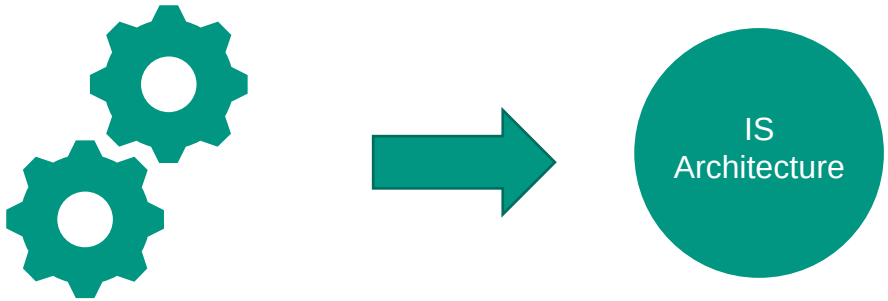
Outcome Perspective

- Architectural design refers to the architectural design process's outcome, that is the collection of hardware and software components and their interfaces, which makes up the framework for the development of an information system.
  IEEE, 1990

Architecture Design

- In the IS architecture context, the term „design“ has a dual meaning:

Process perspective **Outcome perspective**



Note: The IS architecture design is a continuous activity that spans across the system’s entire lifecycle.

Source: "IEEE Standard Glossary of Software Engineering Terminology," in *IEEE Std 610.12-1990* , vol., no., pp.1-84, 31 Dec. 1990, doi: 10.1109/IEEESTD.1990.101064.

„Good“ IS Architecture

- “Good” IS Architecture is fit for purpose (Bass et al., 2012), meaning it:
  - Supports the system’s intended goals
  - Allows for the implementation of required features and behaviours
  - Note: There is no such thing as a system without architecture!

Source: Bass L., Clements P., Kazman R. (2012). *Software Architecture in Practice*. Addison-Wesley, London, UK

„Good“ IS Architecture

- “Good” IS Architecture is fit for purpose (Bass et al., 2012), meaning it:
  - Supports the system’s intended goals
  - Allows for the implementation of required features and behaviours
  - Note: There is no such thing as a system without architecture!

From the outcome perspective

- Designing a system that fulfils intended purpose, its functional and quality specified through requirements.
  → “doing the right things”
- It meets functional and non-functional requirements

Source: Bass L., Clements P., Kazman R. (2012). *Software Architecture in Practice*. Addison-Wesley, London, UK

„Good“ IS Architecture

- “Good” IS Architecture is fit for purpose (Bass et al., 2012), meaning it:
  - Supports the system’s intended goals
  - Allows for the implementation of required features and behaviours
  - Note: There is no such thing as a system without architecture!

From the outcome perspective	From the process perspective
Designing a system that fulfils intended purpose, its functional and quality specified through requirements. → “doing the right things”	Designing by considering how and why design decisions were made throughout the lifecycle of a system. → “doing things right”
It meets functional and non-functional requirements	It is conform to different dimensions (e.g., customer satisfaction, stakeholder’s expectations,…)

Source: Bass L., Clements P., Kazman R. (2012). *Software Architecture in Practice*. Addison-Wesley, London, UK

Why the Need for Well Designed IS Architectures?

- The architecture is the very foundation of the system, meaning:
  - Design decisions cannot be easily undone
  - Creating new architecture or refactoring existing one is not feasible
  - Deficiencies at the beginning can have negative effect on important system behaviours in the future
  - A good IS architecture is critical to meet demands (e.g., performance efficiency)
- Leads more often to higher quality software, since:
  - A good IS architecture “facilitates application systems development, promotes achievements of system’s functional requirements, and supports reconfiguration” while “[…] a bad architecture […] can thwart all three objectives” (Hayes-Roth et al., 1995, p. 288)

Source: Hayes-Roth B., Pfleger K., Lalanda P., Morignot P., Balabanovic M. (1995). *A Domain-Specific Software Architecture for Adaptive Intelligent Systems*. IEEE Transactions on Software Engineering 21 (4):288-301

Drawbacks of IS Architecture

- IS architecture focus can detract from the system‘s more important aspects, such as the functionality and practical business use
- A well-designed IS architecture is not much of use unless it enables the achievement of business goals
- Example: If the architecture is too generic, it can harm the ease of use of the system

Example for the Importance of „good“ Architecture Design

- Apple‘s ecosystem is a careful interplay between the software and hardware components
- A too restrictive architecture gives less incentives for third-party developers to create innovative apps
- A too open and flexible architecture increases complexity and governance efforts, while decreasing the maintainability

iOS

Image source: [Apple Products Unboxed] by O’hayon Julian, August 2nd 2017. Pixabay License.

The Outcome Perspective: IS Architectures‘ Quality

Assess Architecture's Quality

- Identify functional and non-functional IS architecture requirements
- Determine quality attributes for each requirement
- Evaluate the degree to which the IS architecture meets the predetermined quality attributes

Functional Requirements

Definition

Functional requirements define the desired features and functions of a system or one of its components. A functional requirement includes the definition of a functionality and its transformation from an input into a desired output.

Davis

- Strong focus on functionality to serve a particular purpose.
  - Example: user login or item search
- Do not dictate a specific IS architecture design
  - Example: „Storing data in a database“ can be met with different types of databases

Source: Davis AM (1993) Software Requirements: Objects, Functions, and States. PTR Prentice Hall, Upper Saddle River, NJ, USA

Nonfunctional Requirements

Definition

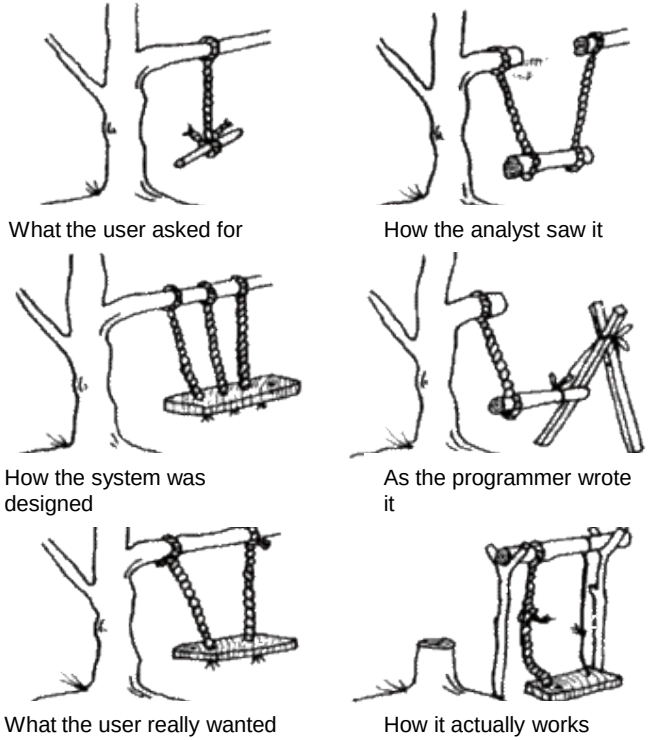
Nonfunctional requirements are requirements that are not specifically concerned with the system's functionality, but rather define general quality attributes and constraints.

Kotonya and Sommerville

- Describes no functionality, but instead the system‘s properties and behaviours
  - Example: „handling a highly fluctuating workload“ translates into either a highly scalable infrastructure or easily migratable to another infrastructure

Source: Kotonya G, Sommerville I (1998) Requirements Engineering: Processes and Techniques. 1 edn. John Wiley & Sons, New York, NY, USA

Challenge



Source: Adapted from Total Quality Management, J Oakland, 1989

Definition: Quality Attributes

- Requirements need to be made assessible:
  - The requirement is represented by a single or combination of several quality attributes (e.g., system's response time in milliseconds)
  - Each quality attribute needs to specify a value or value range (e.g., the system must respond to requests within 0.3 seconds), making it a quality criterion
- IS architecture‘s quality can be assessed through:
  - “The degree to which the architecture's implementation output meets the value range that aligns with its functional and nonfunctional requirements” (IEEE, 1992)

Source: IEEE (1992) Standard for a Software Quality Metrics Methodology. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=237006. Accessed 28 April 2022

Quality Attributes

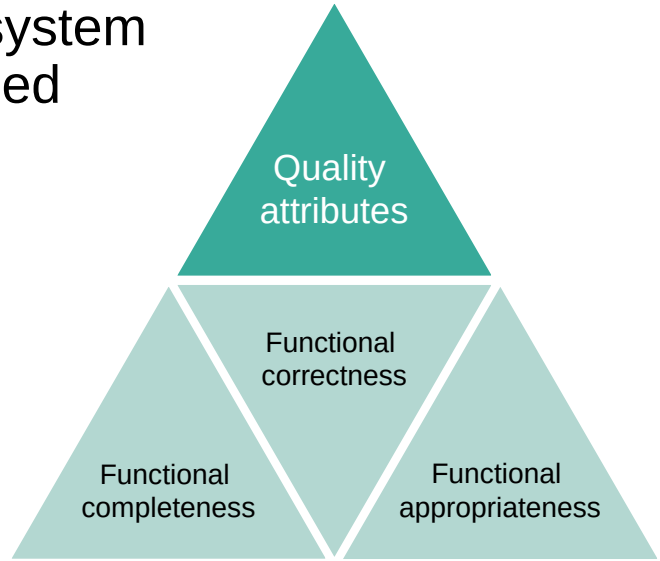
- Framework for software-based systems' quality attributes: ISO/IEC system/software product quality model (ISO/IEC, 2011). This model consists of eight categories of characteristics:

1. Functional suitability
2. Compatibility
3. Maintainability
4. Performance efficiency
5. Reliability
6. Security
7. Portability
8. Usability

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

1. Functional Suitability (1/3)

- Describes the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions (ISO/IEC/IEEE, 2017a)



Source: ISO/IEC/IEEE (2017a) International Standard - Systems and software engineering--Vocabulary. https://standards.ieee.org/standard/24765-2017.html.

1. Functional Suitability (2/3)

- **Functional completeness:**
  - Refers to covering all the defined tasks and user objectives a set of functions provides (ISO/IEC, 2011)
  - Check: Compare the implementation of the functionality in the IS architecture and how the functionality is defined in the specification book

- **Functional correctness:**
  - Expresses the degree to which the IS's set of integrated functions produces the correct results with the required degree of precision (ISO/IEC, 2011)
  - Example: The degree of accuracy of a digital calculator

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

1. Functional Suitability (3/3)

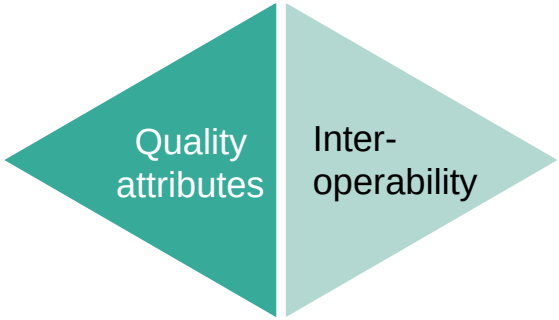
- **Functional appropriateness:**
  - Describes the “degree to which the functions facilitate the accomplishment of specified tasks and objectives” (ISO/IEC, 2011)
  - The manner in which the IS influences its users’ workflow

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

2. Compatibility (1/3)

- Describes the degree to which a product, system, or component can exchange data with other products, systems, or components, and/or perform its required functions, while sharing the same hardware or software environment (ISO/IEC, 2011)

All interacting devices using a particular software



Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

2. Compatibility (2/3)

- Interoperability extends the compatibility between two or more systems, such that, after the data exchange, another system can also interpret the data unambiguously:

Syntactic Interoperability	Semantic Interoperability
Common data model Refers to the structuring and transmission mechanisms for data Separators structure the data Common formats: XML, JSON	Shared meaning Receiving system must know the sending system's vocabulary and grammar

Source: ISO/IEC/IEEE (2017a) International Standard - Systems and software engineering--Vocabulary. https://standards.ieee.org/standard/24765-2017.html.

2. Compatibility (3/3)

JSON Example

```json
{
    "book": {
        "isbn": "978-3-86680-192-9",
        "pages": "483",
        "year": "2019",
        "author": {
            "firstname": "Ali",
            "name": "Sunyaev"
        },
        "appendix": {
            "_format": "pdf",
            "__text": "HGh7Vu2NzW..."
        },
        "_title": "Principles of Internet Computing"
    }
}
```

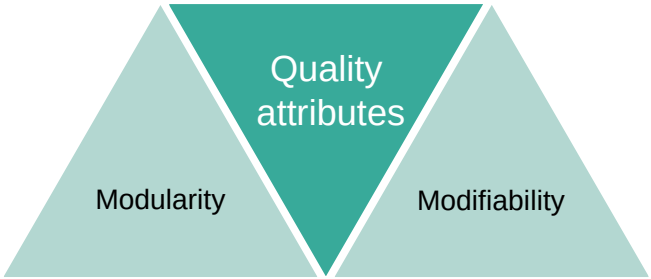
XML Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<book title="Principles of Internet Computing">
    <isbn>978-3-86680-192-9</isbn>
    <pages>483</pages>
    <year>2019</year>
    <author>
        <firstname>Ali</firstname>
        <name>Sunyaev</name>
    </author>
    <!-- here goes a binary file -->
    <appendix format="pdf">
        HGh7Vu2NzW...
    </appendix>
</book>
```

Both are standardized methods with structured attributes and values.

3. Maintainability (1/3)

- Describes the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers (ISO/IEC, 2011)
- The main targets are the program code and the associated documentation



Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

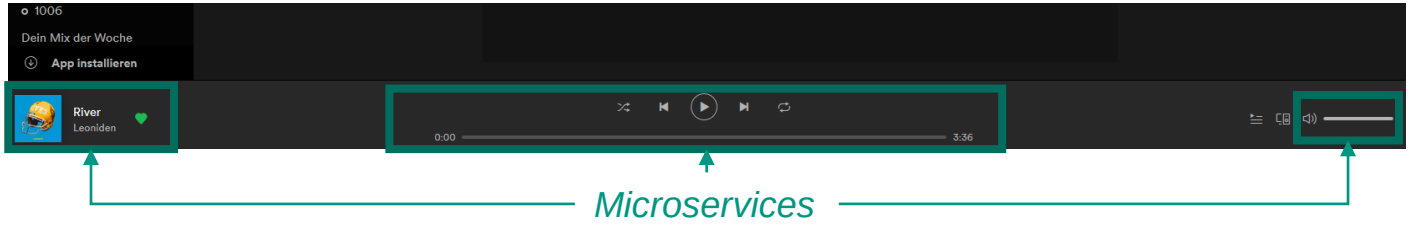
3. Maintainability (2/3)

- The degree of maintainability depends on:
  - Modularity, the degree to which an IS is composed of separated components, such that a change to one component has minimal or no impact on other components (ISO/IEC, 2011)
  - Modifiability, which is the degree to which an IS can be changed effectively and efficiently without introducing defects or decreasing the prevalent product quality (ISO/IEC, 2011)
  - Detailed documentation
  - Dependencies on external services or products

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

3. Maintainability (3/3)

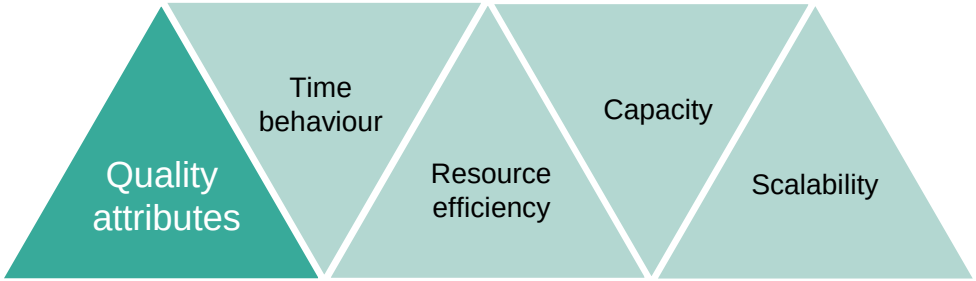
- Spotify is an example for Modularity with the use of microservices
- Microservice architecture:
  - Decomposes the system to smaller components
  - Each component of system, called a microservice, consumes resources independently
  - Microservices are connected to each other using lightweight mechanisms such as http
  - Each microservice could be implemented by independent teams and different technologies, making the system more maintainable (Kargar and Hanifizade, 2018)



Source: Kargar M. J. and Hanifizade A. (2018) Automation of regression test in microservice architecture, 2018 4th International Conference on Web Research (ICWR), Tehran, pp. 133-137, doi: 10.1109/ICWR.2018.8387249

4. Performance Efficiency (1/3)

- Performance in the IS architecture field considers the fulfillment of requirements in terms of time behavior (or latency), resource utilization, and capacity. Performance efficiency compares performance with the amount of resources that are used to achieve a goal under defined conditions (ISO/IEC/IEEE, 2017a)



Source: ISO/IEC/IEEE (2017a) International Standard - Systems and software engineering--Vocabulary. https://standards.ieee.org/standard/24765-2017.html.

4. Performance Efficiency (2/3)

- **Time behavior:**
  - Comprises the duration between an event (e.g., user requests, clock events, or errors), processing times, and the system's resulting response to the event (ISO/IEC/IEEE, 2017a)
  - Example: A system call takes a maximum of 1 second response time

- **Resource-efficiency:**
  - Resource-efficiency is strongly related to resource utilization. Resource utilization is the degree to which the amount and type of resources an IS uses to perform its functions, meet the requirements (ISO/IEC/IEEE, 2017a)
  - Example: A system uses a maximum of 1 GB memory

Source: ISO/IEC/IEEE (2017a) International Standard - Systems and software engineering--Vocabulary. https://standards.ieee.org/standard/24765-2017.html.

4. Performance Efficiency (3/3)

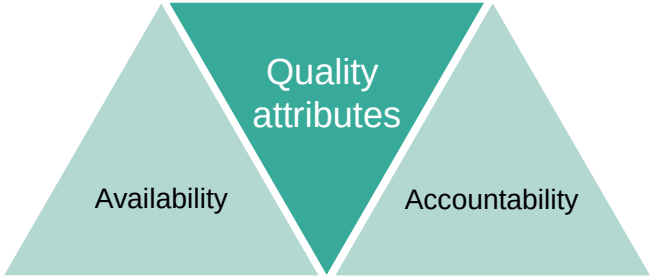
- **Capacity:**
  - The “degree to which the maximum limits of a product or system parameter meet requirements” (ISO/IEC/IEEE, 2017a)

- **Scalability:**
  - The IS architecture's capability to handle decreasing or increasing amounts (e.g., of transactions per second). There are two types of scalability (Bondi 2000): structural scalability and load scalability. To ensure an IS’s load scalability, the system can be scaled horizontally or vertically

Source: ISO/IEC/IEEE (2017a) International Standard - Systems and software engineering--Vocabulary. https://standards.ieee.org/standard/24765-2017.html.
Source: Bondi AB (2000) Characteristics of scalability and their impact on performance. Paper presented at the 2nd international workshop on Software and performance, Ottowa, Ontario, Canada, 17-20 September 2000

5. Reliability (1/2)

- The probability that a system will meet its functional and nonfunctional requirements over a given period of time under given operating conditions (Knight et al., 2003)



Source: Knight JC, Strunk EA, Sullivan KJ (2003) Towards a Rigorous Definition of Information System Survivability. Paper presented at the DARPA Information Survivability Conference and Exposition, Washington, DC, USA, 22-24 April 2003

5. Reliability (2/2)

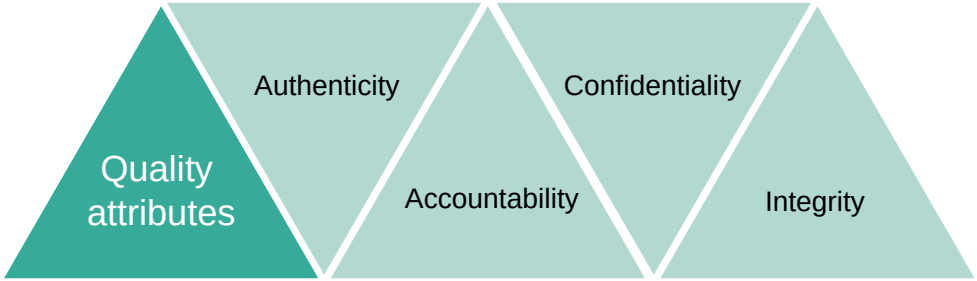
- **Availability:**
  - A statistical probability of an IS being operational at an arbitrary point in time
  - Availability depends on two factors, the mean time between failure (MTBF) and the mean time to repair (MTTR)
  - The availability A can be calculated as follows:
    - A= MTBF/(MTTR+MTBF)
  - Example: The availability of the system should be at 99.9%

- **Accountability:**
  - The “degree to which the actions of an entity can be traced uniquely to the entity” (ISO/IEC, 2011)

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

6. Security (1/3)

- Describes the degree to which a product or system protects information and data such that persons, or other products, or systems have the degree of data access that is appropriate to their types and levels of authorization (ISO/IEC, 2011)



Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

6. Security (2/3)

- **Authenticity:**
  - The “degree to which the identity of a subject or resource can be proved to be the one claimed” (ISO/IEC, 2011)
- **Accountability:**
  - The “degree to which the actions of an entity can be traced uniquely to the entity” (ISO/IEC, 2011)
- **Confidentiality:**
  - Protecting sensitive data, which were collected by the system, against disclosure to unauthorized users or external parties (ISO, 1989)

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.
Source: ISO (1989) Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture. https://www.iso.org/standard/14256.html.

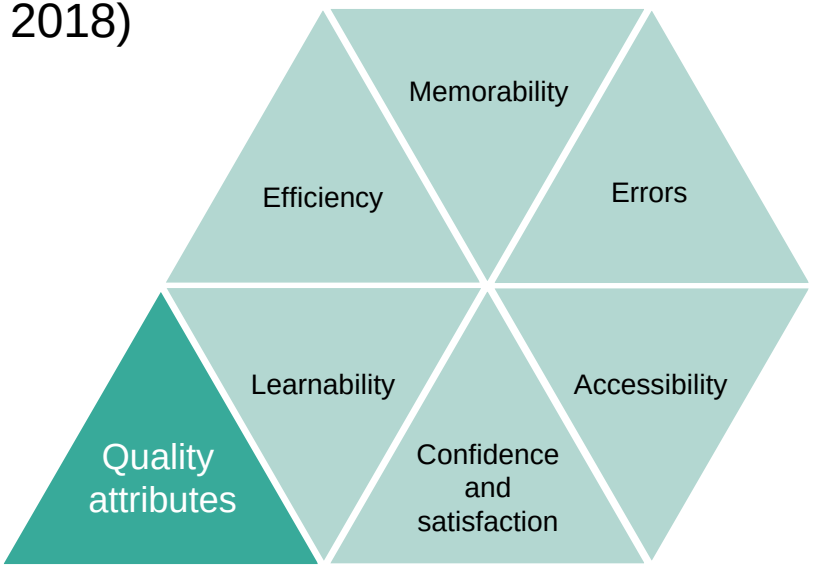
6. Security (3/3)

- **Integrity:**
  - Maintaining and assuring the data's and services' accuracy and consistency over their entire life-cycle (Boritz, 2005)

Source: Boritz JE (2005) IS practitioners' views on core concepts of information integrity. International Journal of Accounting Information Systems 6 (4):260-279

7. Usability (1/4)

- Describes the extent to which specified users can use a product to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use (ISO, 2018)



Source: ISO (2018) Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. https://www.iso.org/standard/63500.html.

7. Usability (2/4)

- **Learnability:**
  - How easy is it for users to accomplish basic tasks the first time they encounter the system?
  - It provides functionality for 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” (ISO/IEC, 2011)
  - Learnability also entails that an IS should provide a high degree of failure prevention to help users learn how to use the system

- **Efficiency:**
  - Once users have learned the system, how quickly can they perform tasks?

- **Memorability:**
  - When users return to the system after a period of not using it, how easily can they re-establish proficiency?

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

7. Usability (3/4)

- **Errors:**
  - How many errors do users make, how severe are these errors, and how easily can users recover from these errors?
- **Confidence and satisfaction:**
  - How pleasant is it to use the system?
- **Accessibility:**
  - Refers to the design of an IS that can also be used by people with disabilities (IEEE, 1992; Nielson, 2012).
  - There are two predominant motivators for designing systems with a high degree of accessibility:
    1. Enlargement of usership
    2. Compliance with legislation, regulations, and standards

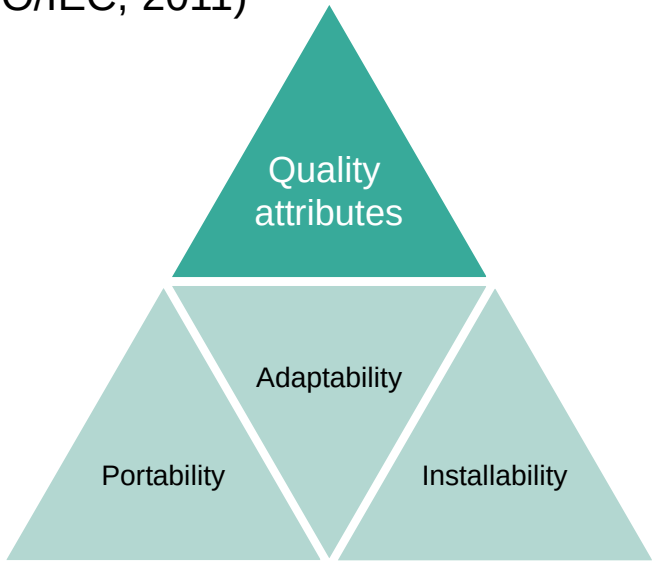
Source: IEEE (1992) Standard for a Software Quality Metrics Methodology. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=237006.
Source: Nielsen J (2012) Usability 101: Introduction to Usability. https://www.nngroup.com/articles/usability-101-introduction-to-usability/. Accessed 28 April 2022.

7. Usability (4/4)

- Approaches to measure an IS's usability:
  - With questionnaires such as the System Usability Scale (SUS) and the USE questionnaire
  - Without questionnaires:
    - Consider the number of errors users make when performing a task
    - The time and effort to accomplish a task
    - The successful operations as a percentage of the total operations
    - The amount of time lost in order to recover from failures

8. Portability (1/2)

- Describes the 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 (ISO/IEC, 2011)



Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

8. Portability (2/2)

- **Portability**
  - Describes the 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 (ISO/IEC, 2011)

- **Adaptability**:
  - The extent to which an IS “can effectively and efficiently be adapted for different or evolving hardware, software, or other operational or usage environments” (ISO/IEC, 2011)

- **Installability:**
  - The degree of effectiveness and efficiency with which an IS “can be successfully installed and/or uninstalled in a specified environment” (ISO/IEC, 2011)

Source: ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.

Limitations

- Often not feasible for the IS architectures to fulfill all the quality attributes to the highest degree
  - Example: Increasing an architecture's availability requires multiple replications running in parallel. Maintaining consistency between replications however becomes a challenge

- A good IS architecture strikes a balance between different quality criteria

The Process Perspective: Design Process

Basic Problems

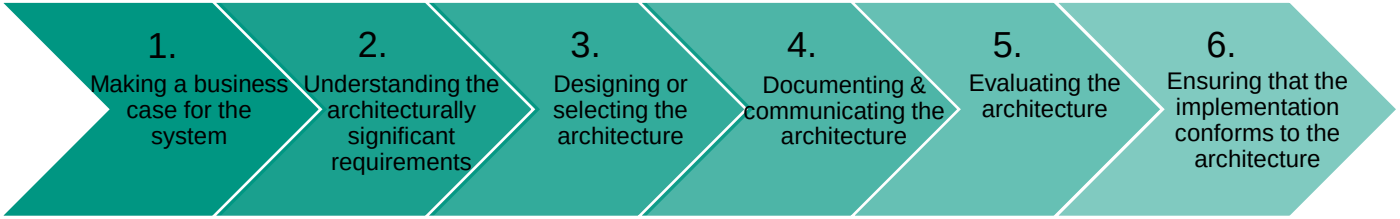
- Maintaining intellectual control over the design of architectures that:
  - Require involvement of and negotiation among multiple stakeholders
  - Developed by large, distributed teams over extended periods of time
  - Must address multiple – possibly conflicting – goals and concerns
  - Must be maintained for a long period of time (Hofmeister et al., 2007)

Source: Hofmeister C, Kruchten P, Nord RL, Obbink H, Ran A, America P (2007) A general model of software architecture design derived from five industrial approaches. Journal of Systems and Software 80 (1):106-126.

Basic Process Activities

- The architecture design process's primary output is an architectural description

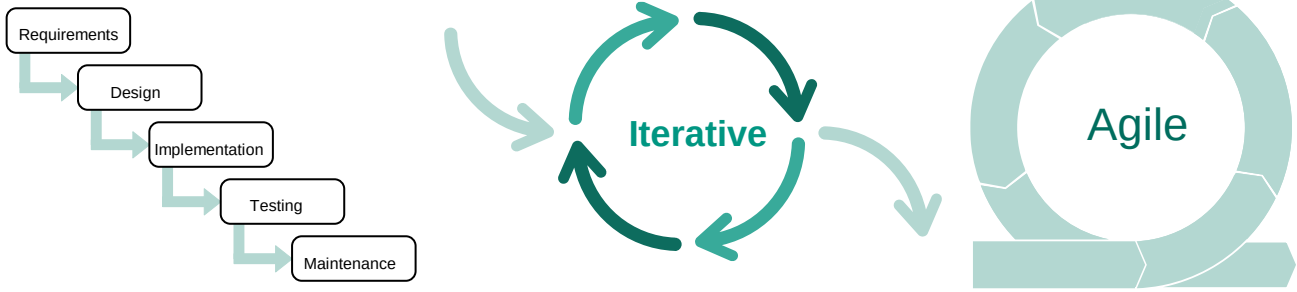
- The basic process activities are:



Order of Activity Execution

- The order of activity execution is not necessarily sequential.

- Depends on activity management strategy (Bass et al., 2012), such as:
  - Waterfall strategy
  - Iterative strategy
  - Agile strategy



Source: Bass L, Clements P, Kazman R (2012) Software Architecture in Practice. Addison-Wesley, London, UK

Waterfall Model

- The process flows in a step-by-step sequence
- The is one direction
- Each phase has entry and exit conditions

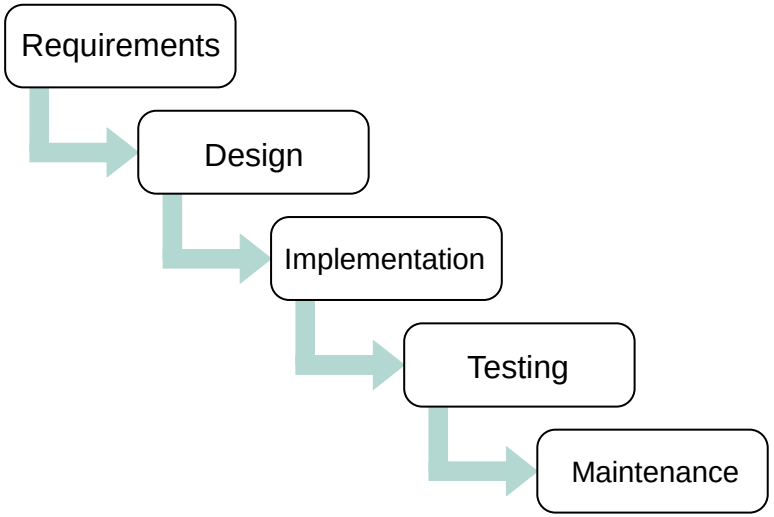


Image source: Adapted from [Waterfall] by Smith Paul, September 17th 2009. Licensed under CC BY-SA 3.0.

Iterative Model

- The process is a series of iterations
- Each iteration starts with objective and requirement identification and alternatives and constraints analysis
- Each iteration ends with an implemented and tested design
- The most known iterative process is Scrum

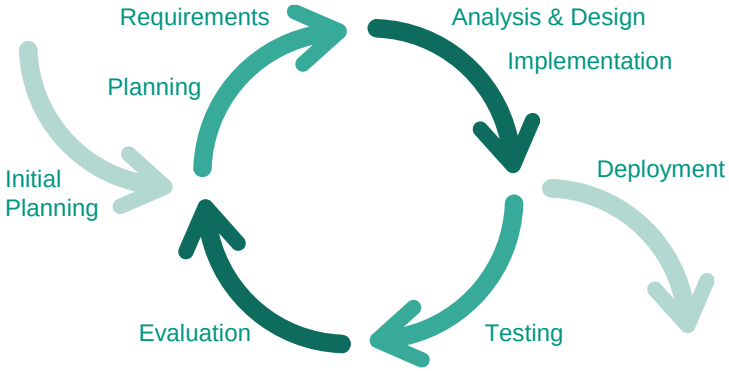
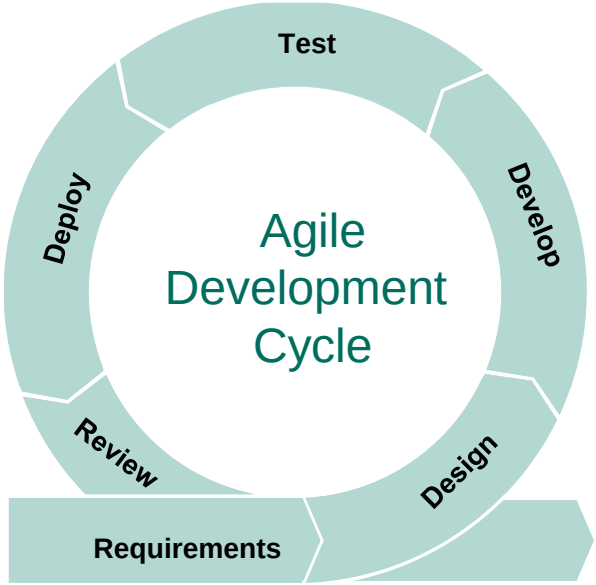


Image source: Adapted from [Iterative development model] by Krupadeluxe, May 5th 2021. Licensed under CC BY-SA 4.0.

Agile Model

- The agile model focuses on early and frequent delivery of working ISs
- There is a close collaboration between developers and customers
- It adapts to changing circumstances



Activity 1:

Making a Business Case for the System

Definition

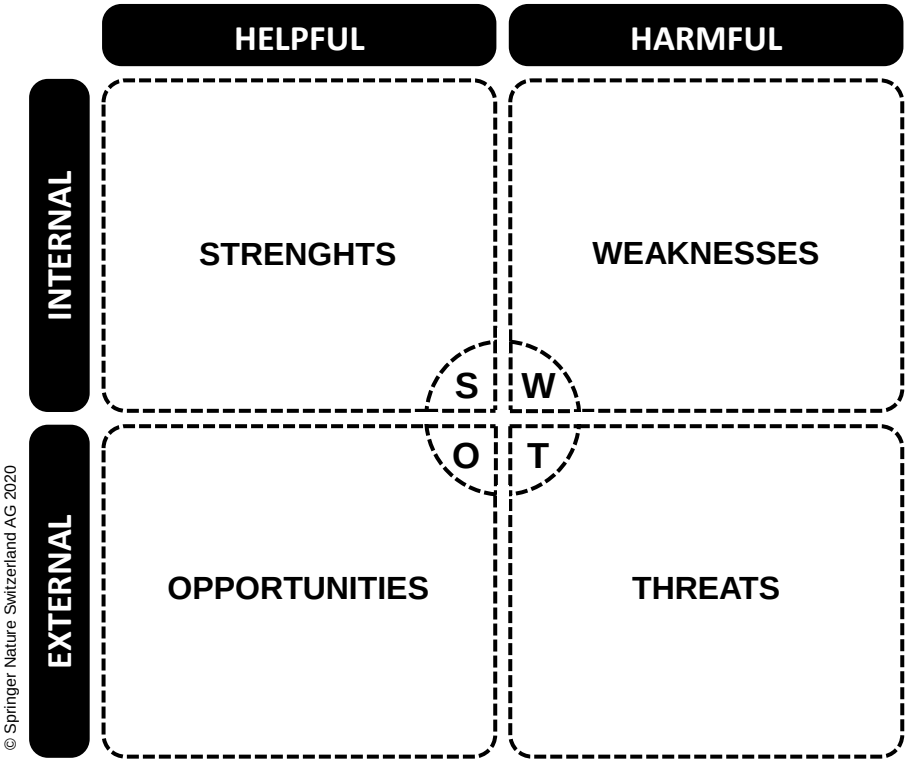
A business case justifies an organizational investment, describing the overall problem that should be solved and the resulting benefits for an organization and its respective stakeholders.

L. Bass, P. Clements, R. Kazman

- It documents the expected costs, benefits and risks
- Architects need to be involved in the creation of the business plan
- To help define business cases, various tools have been developed (e.g., SWOT analysis)

Source: Bass L, Clements P, Kazman R (2012) Software Architecture in Practice. Addison-Wesley, London, UK.

SWOT Matrix



Source: Leigh, D. (2009). SWOT analysis. Handbook of Improving Performance in the Workplace: Volumes 1-3, 115-140.

Activity 2:

Understanding the Architecturally Significant Requirements

Definition

The requirements analysis is the activity of gathering, identifying, and formalizing requirements in order to understand the actual problems for which an architecture, as a solution, is sought, and to learn the purpose and scope of the future system.

S. Albin

- An architect needs to filter out the architecturally significant requirements (ASR)
- The quality attribute workshop is a popular approach for both analysing and eliciting ASRs

Source: Albin S (2003) The Art of Software Architecture: Design Methods and Techniques. John Wiley & Sons, New York, NY, USA

Quality Attribute Workshops

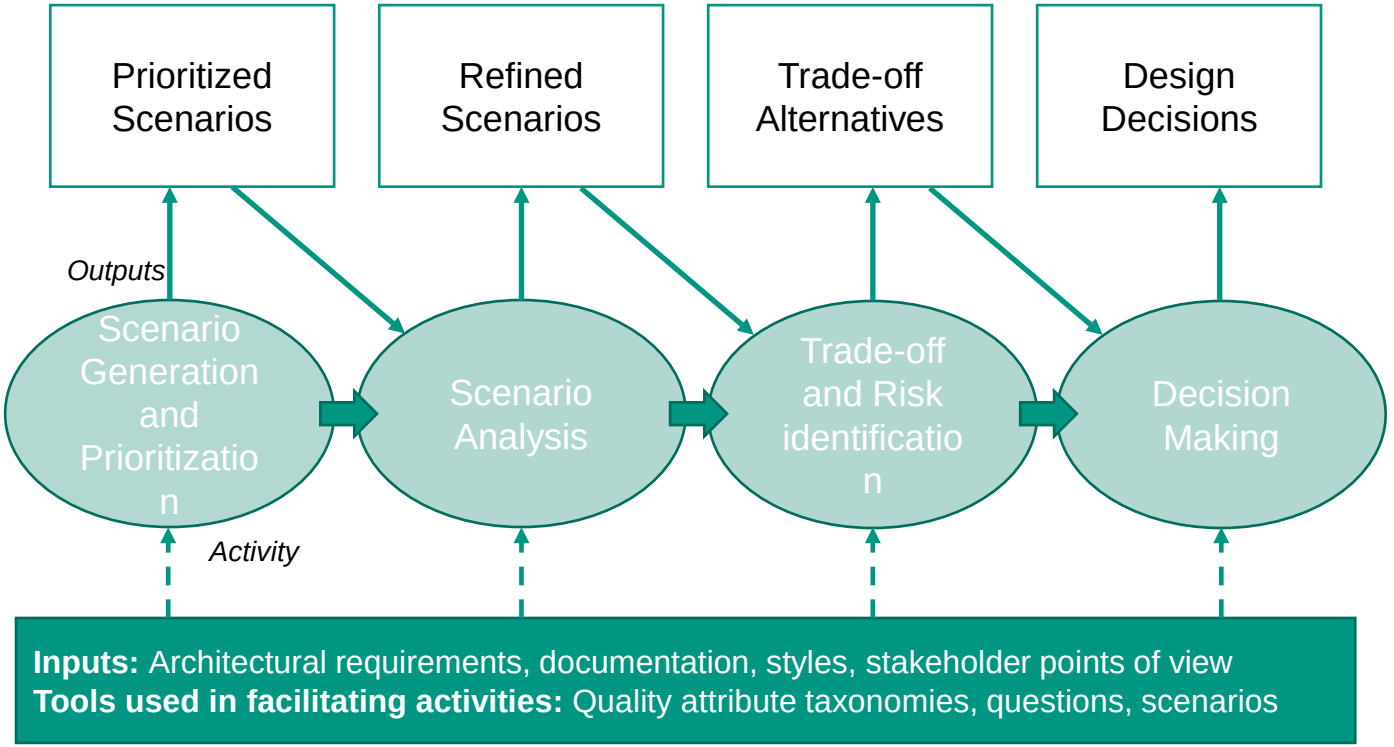


Image source: Adapted from Bergey, J., Barbacci, M., & Wood, W.G. (2000). Using Quality Attribute Workshops to Evaluate Architectural Design Approaches in a Major System Acquisition: A Case Study.

Activity 3:
Designing or Selecting the Architecture

- Choose from the most appropriate architecture design options
- There may be only partial solutions
- Architects need a reliable and rigorous process for selecting candidate architectural solutions to ensure that the decisions mitigate risks and maximize profit
- They can rely on diverse decision-making techniques, such as the quality attributes importance description. It describes to what extent a quality attribute is needed
- The result is a specification of architecture components and information about the design rationale

Activity 4:
Documenting and Communicating the Architecture

- Architecture‘s documentation should be:
  - Informative,
  - Unambiguous
  - Understandable to many people with varied backgrounds

- Various techniques and tools support the documentation and communication of architectural knowledge, such as Knowledge Architect

Architectural Knowledge

Definition

Architectural Knowledge is defined as the knowledge about a software architecture and its environment, such as architectural design and architectural decisions, which shape a software architecture and concepts from architectural design (e.g., components, connectors).

P. Kruchten, P. Laga, H. van Vliet

- Benefits of documenting architectural knowledge:
  - Asynchronous communication
  - Reduced effect of knowledge vaporization
  - Steered and constrained implementation
  - Shaped organizational structure
  - Reuse of architectural knowledge across organizations and projects
  - Training support of new project members

Source: Kruchten P, Lago P, van Vliet H (2006) Building Up and Reasoning About Architectural Knowledge. Paper presented at the International Conference on the Quality of Software Architectures, Västerås, Sweden, 27-29 June 2006

Activity 5:

Evaluating the Architecture

- Determine if the architecture design satisfies the ASRs
- Methods to evaluate the architecture design:
  - Scenario based techniques
  - Architecture trade-off analysis method
  - Unit testing
  - Integration evaluation
- If the design turns out to not satisfy the ASRs, a new design must be created

Architecture Design Evaluation Methods (1/2)

- **Scenario based techniques:**
  - Through using these techniques, an architecture is validated by analyzing how predefined scenarios impact on architectural components (Tekinerdogan, 2004)

- **Architecture tradeoff analysis method:**
  - The most mature methodological approach is found in the architecture tradeoff analysis method, while the economic implications of architectural decisions can also be explored (Bass et al., 2012)

- **Unit testing:**
  - Refers to evaluations focusing on specific pieces of the architecture, such as single components or layers (Bass et al., 2012)

Source: Tekinerdogan B (2004) ASAAM: aspectual software architecture analysis method. Paper presented at the 4th Working IEEE/IFIP Conference on Software Architecture, Oslo, Norway, 12-15 June 2004.
Bass L, Clements P, Kazman R (2012) Software Architecture in Practice. Addison-Wesley, London, UK

Architecture Design Evaluation Methods (2/2)

- **Integration evaluation:**
  - Concentrates on finding problems related to the interfaces between the architectural elements in a design

Source: Tekinerdogan B (2004) ASAAM: aspectual software architecture analysis method. Paper presented at the 4th Working IEEE/IFIP Conference on Software Architecture, Oslo, Norway, 12-15 June 2004.
Bass L, Clements P, Kazman R (2012) Software Architecture in Practice. Addison-Wesley, London, UK

Activity 6:
Ensuring that the Implementation Conforms to the Architecture

- Architects have to ensure that the actual architecture and its implementation fit together throughout the architecture's life cycle
- In the maintenance phase, the architecture still can be changed
- When an architectural change causes the interactions to become more complex, which, in turn, obstructs changes to the system, the architecture is degenerating

Definition

Architectural degeneration is a mismatch between the actual functions of the system and its original design.

B.J. Williams, J.C. Carver

Source: Williams BJ, Carver JC (2010) Characterizing software architecture changes: A systematic review. Information and Software Technology 52 (1):31-51

Steps for Architectural Changes (1/2)

- Prior to making the change an architect should consider the following steps (Williams and Carver, 2010):
  - Change understanding and architecture analysis
    For this task a change analysis tool can be used.
  - Build historical baseline of software change data
    To gain deeper insight into future changes, record information about the change (i.e., change impact, difficulty, and required effort) and its impact on the architecture can be made.
  - Group changes based on impact/difficulty
    Change requests can be grouped based on their characteristics and their similar impact on the system. Heuristics can be developed for each group.

Source: Williams BJ, Carver JC (2010) Characterizing software architecture changes: A systematic review. Information and Software Technology 52 (1):31-51

Steps for Architectural Changes (2/2)

- Facilitate discussion among developers
  A characterization scheme should facilitate consensus, building by providing a list of items for discussion to prevent the change from violating the planned architectural structure.
- Facilitate change difficulty/complexity estimation
  The characterization scheme should allow a developer to determine change complexity as a function of type and size. Also helps facilitate difficulty estimation, because certain types of changes may be more difficult in certain domains than others.

Source: Williams BJ, Carver JC (2010) Characterizing software architecture changes: A systematic review. Information and Software Technology 52 (1):31-51

Example Method for Designing Architectures

- Attribute-Driven Design (ADD) Method follows an iterative Plan, Do, and Check cycle:
  1. **Plan:** Quality attributes and design constraints are considered in order to select which types of elements will be used in the architecture
  2. **Do:** Elements are instantiated to satisfy quality attribute requirements, as well as functional requirements
  3. **Check:** The resulting design is analyzed to determine if the requirements are met

Example Method for Designing Architectures: Attribute-Driven Design (ADD) Method

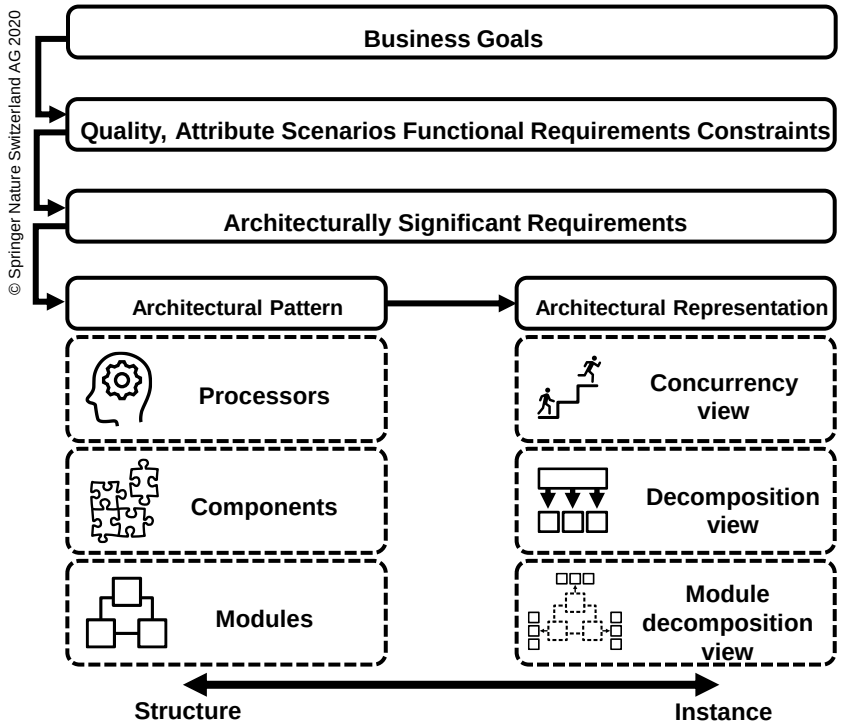
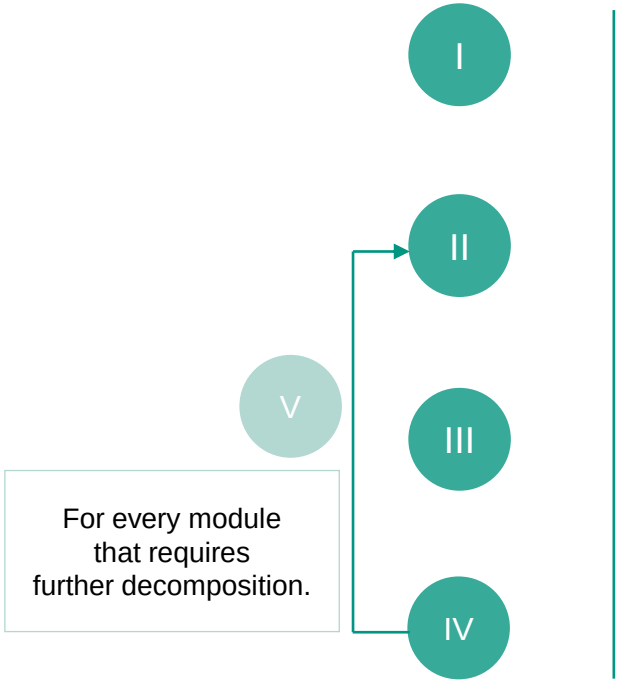


Image source: Adapted from Hofmeister C, Kruchten P, Nord RL, Obbink H, Ran A, America P (2007) A general model of software architecture design derived from five industrial approaches. J Syst Softw 80 (1):106–126

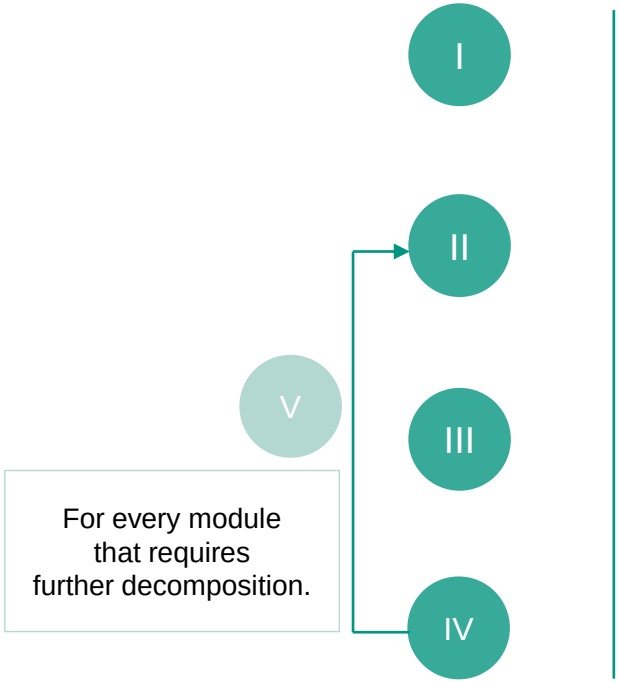
Iterative Steps When Using the ADD Model



I

Confirm that there is sufficient information about the ASR to proceed with the ADD. Architects have to ensure that the stakeholders have stated and prioritized the requirements according to business goals.

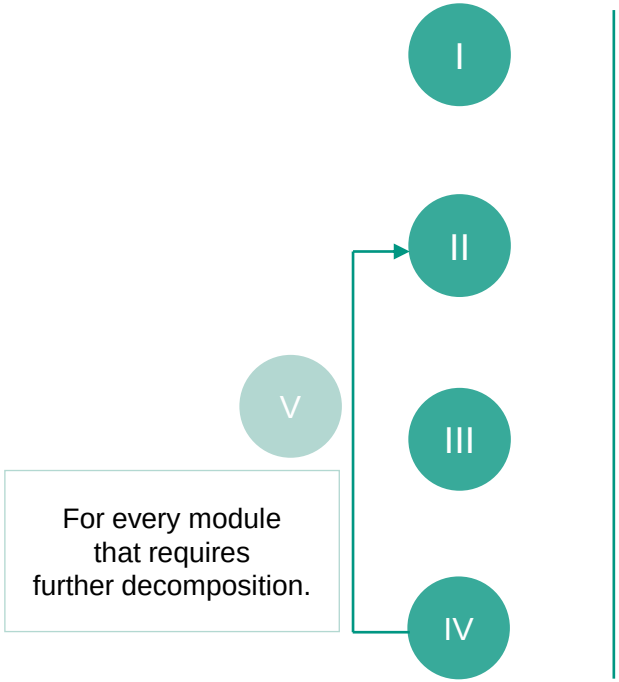
Iterative Steps When Using the ADD Model



II

Choose the module to decompose. The module to start with is usually the whole system. For designs that are already partially completed, i.e., previous iterations via the ADD, the module is an element that is not yet designed. All required inputs for this module should be available (constraints, functional requirements, quality requirements).

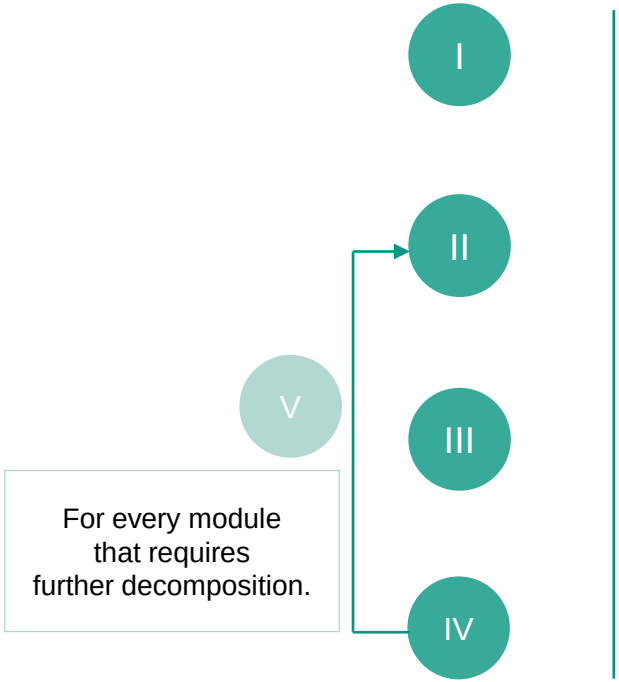
Iterative Steps When Using the ADD Model



III

Identify the ASR for the chosen module from the set of concrete quality scenarios and functional requirements. This step determines what is important for this module.

Iterative Steps When Using the ADD Model



IV

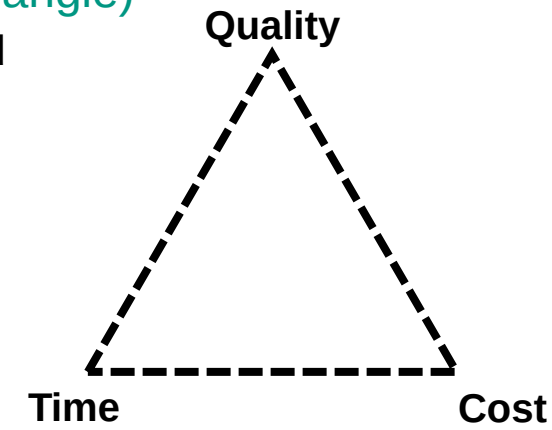
Choose a design concept for the module that satisfies the ASR according to these steps:

- Identify the design concerns.
- For each design concern, create a list of alternative patterns addressing concern.
- Create or select the design pattern based on the tactics that can be used to achieve the ASR. Identify child modules that are required to implement the tactics.
- Instantiate child modules and allocate functionality, as well as responsibilities from use cases, and represent the results using multiple views.
- Define the child modules' interfaces.
- Verify that the modules and child modules, thus far, meet functional requirements, quality attribute requirements, and design constraints.

Success of Architecture Design Process 1/2

- Success remains an ambiguous, inclusive, and multidimensional concept whose definition is bound to a specific context (Ika, 2009)

- Project success has often been known as the ability to stay within time, cost and quality constraints (Iron Triangle)
  - Note: This view is outdated



Source: Ika LA (2009) Project Success as a Topic in Project Management Journals. Project Management Journal 40 (4):6-19

Success of Architecture Design Process 2/2

- Today, project success is measured by considering diverse dimensions in isolation or combination, such as project safety, customer satisfaction, benefits for organizations, and whether the project meets specifications, is free from defects, conforms to stakeholders' expectations, or minimized construction aggravation, disputes, and conflicts (Ika, 2009)
- There is a trade-off between time, quality and cost
- Dimensions should be specified and prioritized in an early design stage and be constantly evaluated

Source: Ika LA (2009) Project Success as a Topic in Project Management Journals. Project Management Journal 40 (4):6-19

References 1/2

- Albin S (2003) The Art of Software Architecture: Design Methods and Techniques. John Wiley & Sons, New York, NY, USA
- Bass L, Clements P, Kazman R (2012) Software Architecture in Practice. Addison-Wesley, London, UK
- Bergey, J., Barbacci, M., & Wood, W.G. (2000). Using Quality Attribute Workshops to Evaluate Architectural Design Approaches in a Major System Acquisition: A Case Study.
- Bondi AB (2000) Characteristics of scalability and their impact on performance. Paper presented at the 2nd international workshop on Software and performance, Ottowa, Ontario, Canada, 17-20 September 2000
- Boritz JE (2005) IS practitioners' views on core concepts of information integrity. International Journal of Accounting Information Systems 6 (4):260-279
- Davis AM (1993) Software Requirements: Objects, Functions, and States. PTR Prentice Hall, Upper Saddle River, NJ, USA
- Hayes-Roth B, Pfleger K, Lalanda P, Morignot P, Balabanovic M (1995) A Domain-Specific Software Architecture for Adaptive Intelligent Systems. IEEE Transactions on Software Engineering 21 (4):288-301
- Hofmeister C, Kruchten P, Nord RL, Obbink H, Ran A, America P (2007) A general model of software architecture design derived from five industrial approaches.Journal of Systems and Software 80 (1):106-126
- IEEE (1990) Standard Glossary of Software Engineering Terminology. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=159342&tag=1.
- IEEE (1992) Standard for a Software Quality Metrics Methodology. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=237006.
- Ika LA (2009) Project Success as a Topic in Project Management Journals. Project Management Journal 40 (4):6-19

References 2/2

- ISO (2018) Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. https://www.iso.org/standard/63500.html.
- ISO/IEC (2011) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html.
- ISO/IEC/IEEE (2017a) International Standard - Systems and software engineering--Vocabulary. https://standards.ieee.org/standard/24765-2017.html.
- Kargar M. J. and Hanifizade A. (2018) Automation of regression test in microservice architecture, *2018 4th International Conference on Web Research (ICWR)*, Tehran, pp. 133-137, doi: 10.1109/ICWR.2018.8387249
- Knight JC, Strunk EA, Sullivan KJ (2003) Towards a Rigorous Definition of Information System Survivability. Paper presented at the DARPA Information Survivability Conference and Exposition, Washington, DC, USA, 22-24 April 2003
- Kotonya G, Sommerville I (1998) Requirements Engineering: Processes and Techniques. 1 edn. John Wiley & Sons, New York, NY, USA
- Kruchten P, Lago P, van Vliet H (2006) Building Up and Reasoning About Architectural Knowledge. Paper presented at the International Conference on the Quality of Software Architectures, Västerås, Sweden, 27-29 June 2006
- Nielsen J (2012) Usability 101: Introduction to Usability. https://www.nngroup.com/articles/usability-101-introduction-to-usability/. Accessed 28 April 2022
- Oakland JS (1989) Total quality management. Butterworth-Heinemann, 1989.
- Tekinerdogan B (2004) ASAAM: aspectual software architecture analysis method. Paper presented at the 4th Working IEEE/IFIP Conference on Software Architecture, Oslo, Norway, 12-15 June 2004
- Williams BJ, Carver JC (2010) Characterizing software architecture changes: A systematic review. Information and Software Technology 52 (1):31-51

Questions

Questions

1. What are the two different perspectives on architecture design?
2. What characterizes a good IS architecture?
3. What types of architectural requirements can be differentiated?
4. Describe three quality attributes of a good IS architecture.
5. What is the role of a business case during the IS architecture design process?
6. Which individuals or groups need to be involved in the IS architecture design process and for what reasons?
7. How can the success of an architecture design process be determined?

Further Reading

- Albin S (2003) The art of software architecture: design methods and techniques. Wiley, New York, NY
- Bass L, Clements P, Kazman R (2012) Software architecture in practice. Addison-Wesley, London
  de Boer RC, Farenhorst R, Lago P, van Vliet H, Jansen AGJ (2007) Architectural knowledge: getting to the core. Paper presented at the 3rd international conference on the quality of software architectures, Medford, MA, 11–13 July 2007
- Gilb T (2005) Competitive engineering: a handbook for systems engineering, requirements engineering, and software engineering using planguage. Elsevier, Butterworth-Heinemann, Oxford
- Hofmeister C, Kruchten P, Nord RL, Obbink H, Ran A, America P (2007) A general model of software architecture design derived from five industrial approaches. J Syst Softw 80(1):106–126
- Taylor RN, Medvidovic N, Dashofy E (2009) Software architecture: foundations, theory, and practice. Wiley, Hoboken, NJ