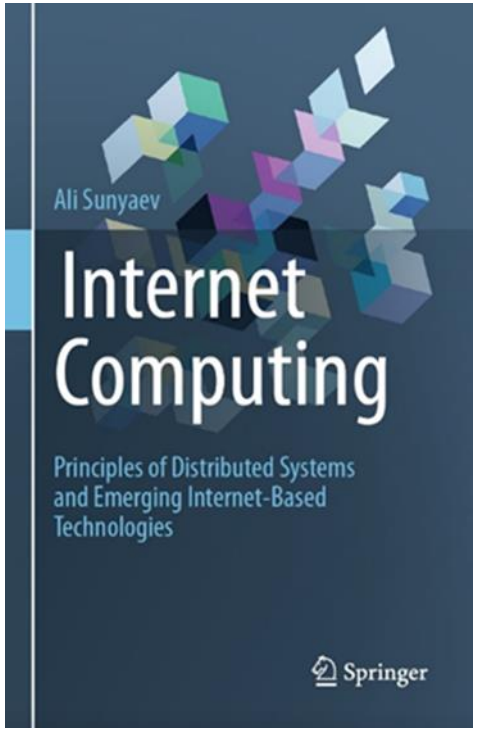
AI: Internet Computing

Lecture 2 — Information Systems Architecture

Learning Goals of the Lecture

- Understand the concept of IS architecture and its roles and purposes
- Understand how properties of IS architectures can influence system features and behaviors
- Get to know the most important and widespread architectural patterns

Reference to the Teaching Material Provided



Chapter 2
Information Systems Architecture

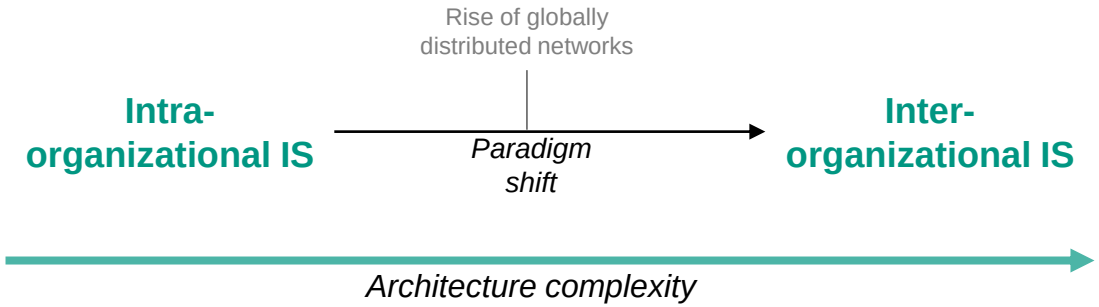
Abstract
Information systems (IS) are composed of different, interrelated elements that aim to fulfill desired features. This chapter introduces the concept of IS architecture as a way to meaningfully describe and design the underlying structure of an IS. It presents nine basic principles of IS architectures and explains them using the example of the cloud-based storage service Dropbox. This introduction to IS architecture also introduces the most common architectural patterns (i.e. the client-server architecture, tier architectures, peer-to-peer architecture, model view controller, and service-oriented architecture) in the realm of Internet computing and briefly discusses their strengths and weaknesses.

Learning Objectives of this Chapter
The main learning objective of this chapter is to understand the concept of IS architecture and its roles in the development of distributed IS in the realm of Internet computing. Having read this chapter, readers will understand the purposes of IS architectures and why architectural models can look different although they represent the same system. By getting to know the basic principles of IS architecture, readers will be able to understand how properties of IS architectures can influence system features and behaviors. Finally, readers will get to know the most important

Defining Information Systems Architecture

Information System Architecture

- Information system (IS) context: Ambiguous use of the term **„architecture“**—different levels of abstraction *(Lankhorst, 2017)*
  - **Enterprise architectures** describe structures, business processes, and infrastructures in complex organizations
  - **Hardware architectures** describe technical components of systems
  - **Software architectures** describe software systems of different sizes and scopes
  - …



Information Systems 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)

You are super rich and order an extraordinary car. What would you demand?

Image source: Bugatti (2021). *Bugatti Chiron Pur Sport* [Photograph]. Instagram. https://www.instagram.com/p/CKMMj3-nrhz/

You turn the ignition key. No reaction. What would you expect?

Image source: [Cars covered in snow] by Socha Arek, November 10th 2016. Pixabay License.

You turn the ignition key. The headlights are shining, but the engine does not start. What would you expect?

Image source: [Car in snow] by Svenska, July 6th 2015. Pixabay License.

The analysis model is a functional model of the user (e.g., stimulus-response model)

- Links events with input and output
- Data flows, memory, objects, events and states are only used to describe external features (e.g., user behavior)
  - They don‘t really need to be verifiable in the system
- Even essential memories do not need to be realized as isolated elements within the real system
- **Outer, external, or phenomenological model**

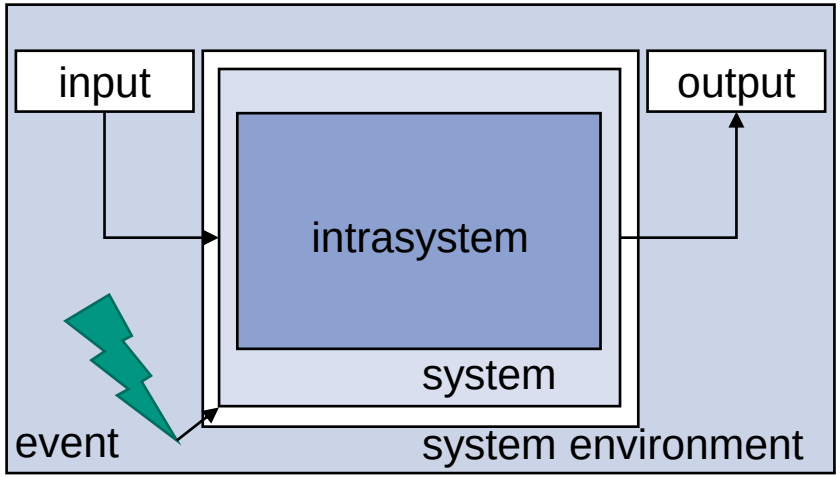


Figure according to Bachmann, F., Klein, M., Bass, L., Clements, P., & Kazman, R. (2003). Understanding quality attributes. Len Bass, Paul Clements, Rick Kazman (alle Hrsg.). Software Architecture in Practice. Addison-Wesley Professional, 2, 71-98.

System architecture is an explanatory system model

- Links events with input and output; also, links internal components with their external features and relationships between these features
- Components, their features, and relationships are “real”
- Internal model or model of internal structure
- But: **Not every internal model is an architecture** (e.g., an architecture is independent of the implementation)

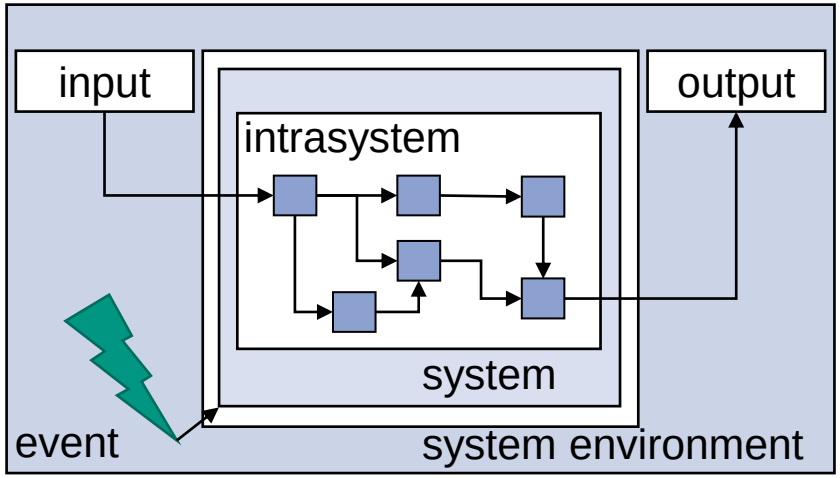
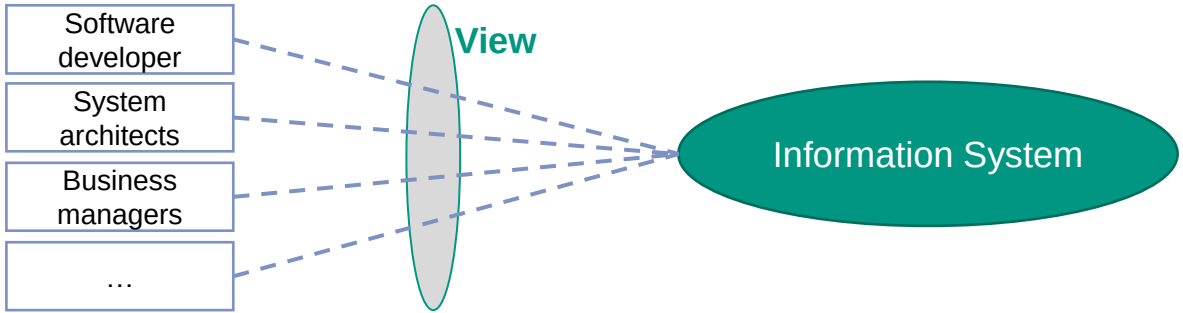


Figure according to Bachmann, F., Klein, M., Bass, L., Clements, P., & Kazman, R. (2003). Understanding quality attributes. Len Bass, Paul Clements, Rick Kazman (alle Hrsg.). Software Architecture in Practice. Addison-Wesley Professional, 2, 71-98.

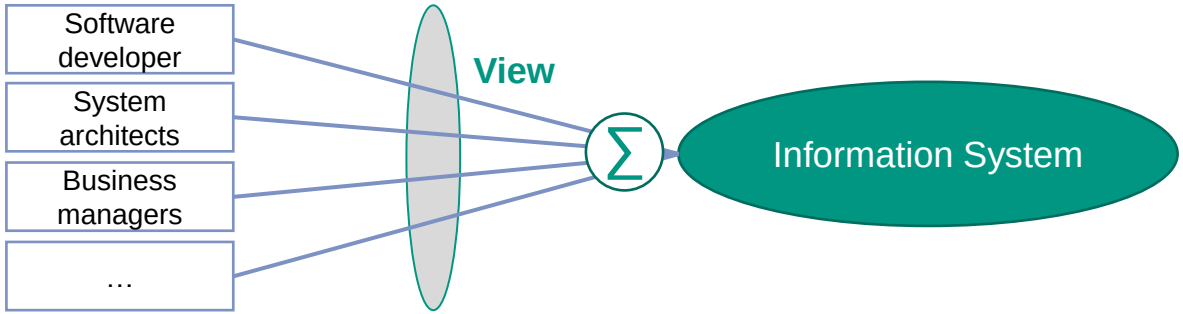
Different Views on Information System Architecture

- Architectural considerations involve different stakeholders with fundamentally different needs



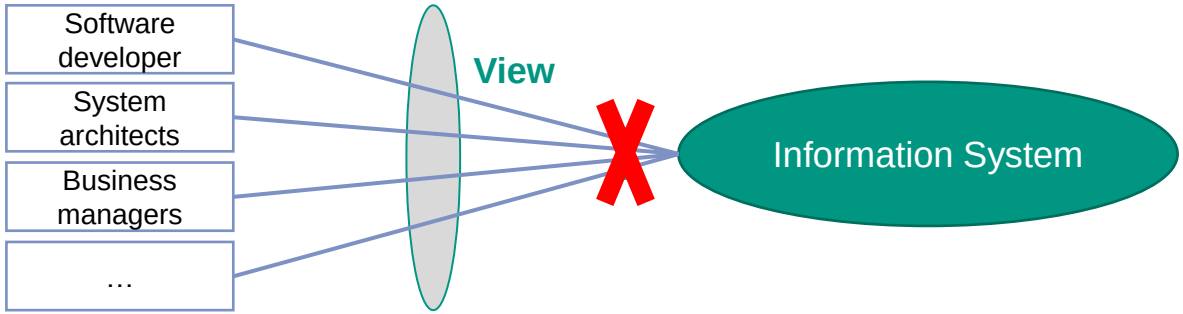
- Each **view** is an analytical description of an IS—considering multiple views allows to compensate for weaknesses within single views *(Golden, 2013)*

Different Views on Information System Architecture



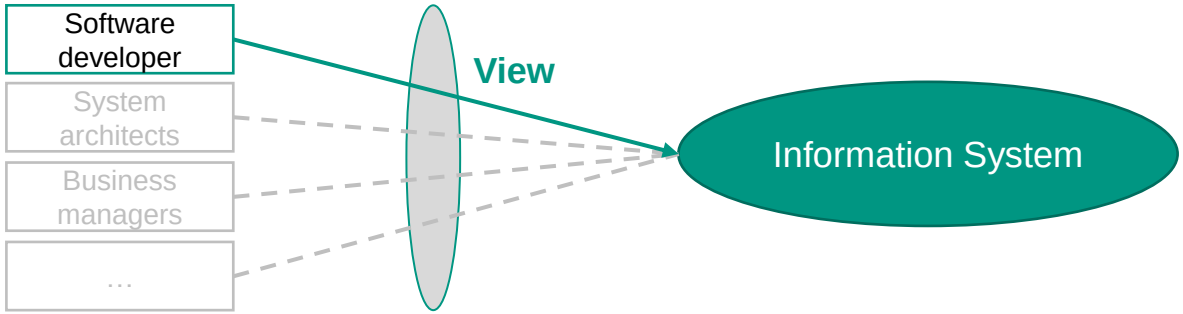
- Intuitively, the best definition of an information system is the set of all different views

Different Views on Information System Architecture



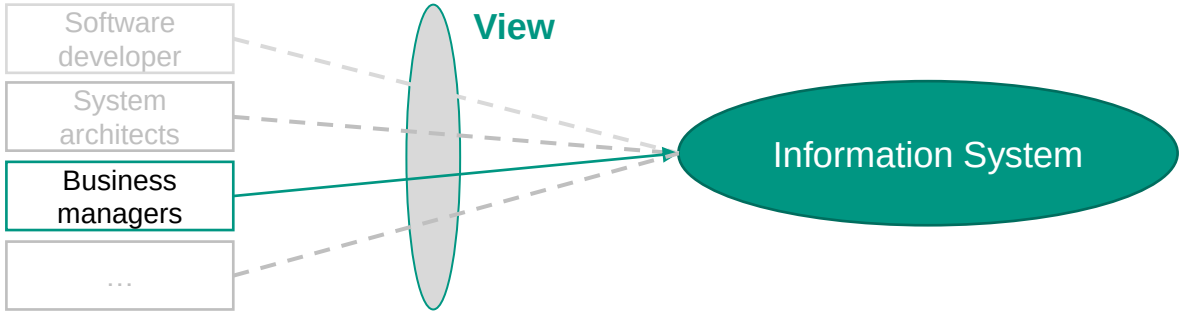
- Intuitively, the best definition of an information system is the set of all different views
- However: *Objective, unified* and *exhaustive* definition of an IS often proves impossible

Different Views on Information System Architecture



- Intuitively, the best definition of an information system is the set of all different views
- However: *Objective, unified* and *exhaustive* definition of an IS often proves impossible
- Instead: Different views to consider different stakeholder needs
  - e.g., software developers need to understand software's underlying architecture and interfaces to adjacent systems

Different Views on Information System Architecture



- Intuitively, the best definition of an information system is the set of all different views
- However: *Objective, unified* and *exhaustive* definition of an IS often proves impossible
- Instead: Different views to consider different stakeholder needs
  - e.g., business managers require an overview of a firm's application landscape to do investment decisions

Architectural Models

- To make the architecture of an IS tangible and communicate it with other stakeholders, IS architects use **architectural models** (Zachman, 1987)

Definition

Architectural Model:
An illustration, created using available standards, in which the primary concern is to represent the architecture of an IS from a specific perspective and for a specific purpose.

Purposes of Architectural Models

- In IS, **architectural models** can serve various purposes (Sawyer, 1997; Castro et al., 2002):
  - Enable IS architects to think deeply about IS architecture and design an IS that meets existing requirements
  - Serve as a tool to document components and interrelationships of existing IS
  - Facilitate an understanding of an IS and its evolution
  - Provide a common language to enable all stakeholders to reason about an IS' structural properties
  - May provide best practices and lessons learned that can be transferred to different IS

The Principles of Information System Architecture

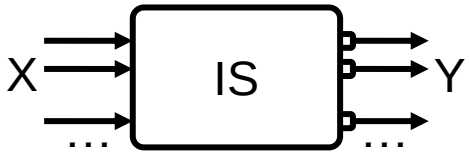
The Principles of Information System Architecture

- Certain principles can be applied to every IS architecture
- In the following, the nine principles of system architecture by Golden (2013) are presented

Running example:

Principle 1: Architecture Models Information System Boundaries, Inputs, and Outputs

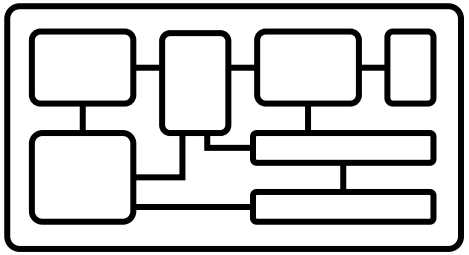
- IS under consideration is modeled as a system that performs certain functions
- IS needs to be defined by…
  - … its perimeters / boundaries
  - … certain inputs and outputs
  - … its internal state (functions)



- *Input: Data files, user configurations, …*
- *Functions: Ability to access files, from anywhere / every device*
- *Output: Customized access to files stored in Dropbox*

Principle 2: An Information System Can Be Broken Down Into a Set of Smaller Subsystems

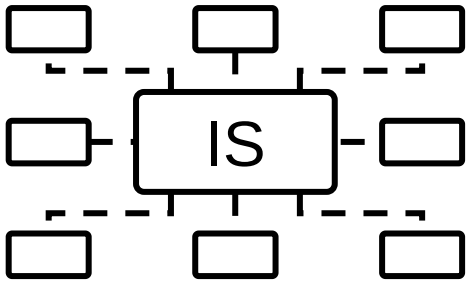
- Every IS can be broken down into a set of interconnected subsystems—each could be subject to its own architectural examination
- However: IS too complex to be understood solely by is entire set of subsystems
- Relationships between different subsystems imply collective behavior of an IS



- *Subsystems: Authentication system, encryption system, storage server system, … (Dropbox, 2019)*
- *Solely the sum of subsystems does not comprehensively explain Dropbox's overall behavior -> relationships necessary!*

Principle 3: An Information System Can Be Considered in Interaction with Other Systems

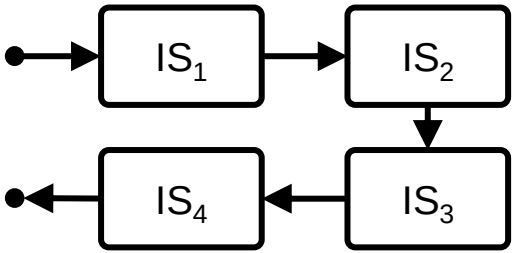
- Every IS can be considered in interaction with other IS—no IS exists in a void!
- Individual IS, taken together, may form higher-level IS
- Primary interest of architects: How does an IS connect to other IS in its intermediate environment?



- *Dropbox interacts with many end users simultaneously—highly heterogeneous systems (operating systems, devices, screen sizes, …)*
- *Requirements of different users need to be taken into account by IS architects!*

Principle 4: An Information System Can Be Considered Through Its Entire Lifecycle

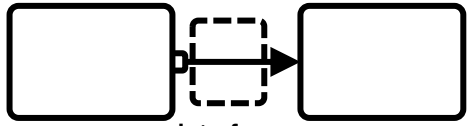
- An IS typically goes through various stages in its lifetime
  - Design phase, development phase, test phase, and operation phase
- Different requirements for each phase
- Architects can consider an IS through its entire lifecycle



- *Architecture of cloud computing services such as Dropbox changes continually to meet new requirements (Lins et al., 2015)*
- *Example: Infrastructure move from Amazon S3 to in-house solution Magic Pocket necessitated far-reaching architecture changes*

Principle 5: An Information System Can Be Linked to Another Information System via an Interface

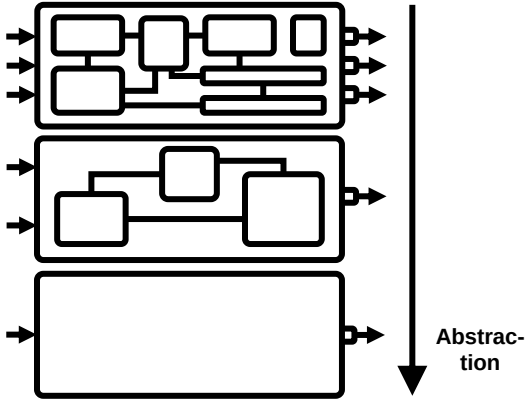
- Remember Principle 3: No IS exists in isolation!
- Connections between IS can be modeled via interfaces
- Interfaces explain mechanisms how two IS interact



- *Dropbox must enable users to access its services in a convenient, reliable, and secure way via interfaces*
- *For private users: Client applications for common operating systems*
- *For commercial users: API called DBX Platform*

Principle 6: An Information Can Be Modeled at Various Different Abstraction Levels

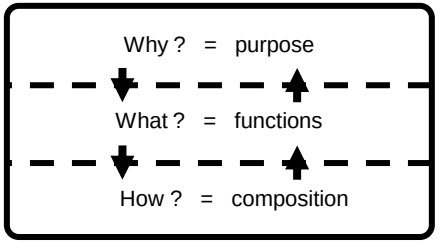
- An IS can be considered at different abstraction levels
- Appropriate abstraction level strongly depends on IS's properties of interest
  - Interest in generated output of an IS
    -> high abstraction level
  - Interest in altering certain functionalities of an IS
    -> low abstraction level



- *Dropbox can be considered as a holistic cloud computing service, as a set of interconnected hardware devices, or as a set of software codes*
  - *All realistic and valid abstraction levels*
  - *Relevance depends on architecture modeling's purpose*

Principle 7: An Information System Can Be Viewed Along Several Layers

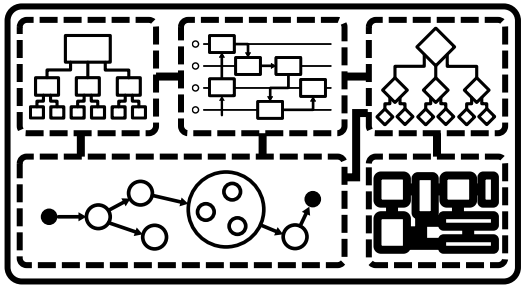
- An IS architecture can be viewed along several layers
- Allows stakeholders to reason in an isolated way about specific aspects
- Most common approach in IS:
  Divide IS into logical layers that describe a grouping of its functions
- Golden (2013) proposes three layers: *purpose, functions, composition*



- *Dropbox is cloud computing service that seeks to enable users to store and receive data files AND a set of functions that serve this purpose*
- *All these functions are implemented via physical and non-physical components*

Principle 8: An Information System Can Be Described Through Interrelated Models with Given Semantics

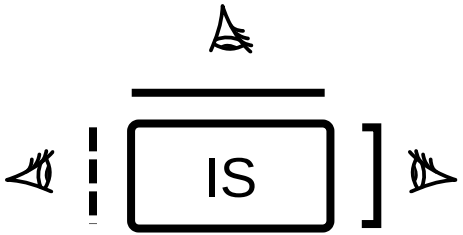
- An IS's behavior is not static, but can depend on a variety of aspects
  - e.g., user input, IS's state
- *Example*: Firm shuts down IS during weekend for maintenance
- IS must be designed in a way that it only performs a reduced set of functionalities and potentially display errors messages



- *Dropbox offers possibility to share data with non-registered users*
- *IS must treat users differently depending on whether they are logged in or just received permission to access data*

Principle 9: An Information System Can Be Described Through Different Perspectives

- An IS can be described through various perspectives corresponding to different stakeholders interacting with the IS
- Different requirements -> heterogeneous perceptions of an IS



- *Private user: Most interested in provided functionality of Dropbox, not in technical details or underlying hardware components*
- *Hardware supplier: Need for a more detailed model of Dropbox's technical architecture*

Principles of Information Systems Architecture: Recap

- Principles that can be applied to every IS architecture:
  1. *Architecture Models Information System Boundaries, Inputs, and Outputs*
  2. *An Information System Can Be Broken down into a Set of Smaller Subsystems*
  3. *An Information System Can Be Considered in Interaction with Other Systems*
  4. *An Information System Can Be Considered Through Its Entire Lifecycle*
  5. *An Information System Can Be Linked to Another Information System via an Interface*
  6. *An Information System Can Be Modeled at Various Abstraction Levels*
  7. *An Information System Can Be Viewed Along Several Layers*
  8. *An Information System Can Be Described Through Interrelated Models with Given Semantics*
  9. *An Information System Can Be Described Through Different Perspectives*

- Principles help to understand why representations of IS architecture can look very different although they represent the same system

Architectural Views

Architectural Views: Motivation

- Each architectural model is a representational abstraction of an IS's architecture
- Impossible to capture all the information relevant to all potential stakeholders in one single model

- Instead: Multiple models of an IS from different perspectives

The Logical View	The Process View	The Development View	The Physical View

Famous 4+1 architectural view model by Kruchten (1995)

Architectural Views

The Logical View	The Process View	The Development View	The Physical View

- Decomposition of IS into a set of key abstractions taken from the problem domain, represented as objects *(Object-oriented decomposition)*

- Purpose:
  - Realization of functional requirements (i.e., what the system should provide to its users)
  - Identification of common mechanisms and design elements across different parts of the IS

Architectural Views

The Logical View	**The Process View**	The Development View	The Physical View

- Representation of dynamic aspects of an IS by explaining its runtime behavior, processes, and the interaction between processes
- A process is a group of tasks that form an executable unit
- Addresses issues of *concurrency, distribution, IS integrity, fault tolerance*, and how main abstractions from logical view fit process architecture

Architectural Views

The Logical View	The Process View	**The Development View**	The Physical View

- IS's software is packed into small chunks (*subsystems*) that can be developed by a small number of developers *(Subsystem Decomposition)*
- Subsystems are organized in a hierarchy of layers, each providing an interface to layers above it

- Serves as a basis for
  - … requirements allocation
  - … allocation of work to teams
  - … cost evaluation and planning
  - … monitoring project progress
  - … reasoning about code re-use, portability, and security

Architectural Views

The Logical View	The Process View	The Development View	**The Physical View**

- Depicts IS from the perspective of an IS engineer
- Concerned with topology of IS components on physical layer, as well as physical interconnections
- Used to ensure suitable mapping of software components to hardware components

Architectural Patterns

Architectural Patterns: Motivation

- In IS design, developers often encounter recurring problems
- To avoid having to redesign an IS architecture from scratch every time, certain best practices have emerged, called architectural patterns *(Avgeriou and Zdun, 2005)*

Architectural Patterns: Definition

Definition

Architectural Pattern:
An abstract description of a recommended architectural approach that has been tested and proven in different information systems and environments.

- Abstractly describes the possibility to structure an IS's architecture in ways that were successful in previous IS
- An architectural patterns should include information about…
  - … circumstances in which the pattern should be used
  - … its strengths and weaknesses
- Note: architectural pattern ≠ architecture
  - Various architectures may implement same pattern

Architectural Patterns

- There are many different architectural patterns

- In the following: Description of several relevant architectural patterns
  - Client-server architectures
  - Tier architectures
  - Peer-to-peer architectures
  - Model-view-controller architectures
  - Service-oriented architectures

Client-Server Architectures: Basics

- Basic idea: Distribute tasks or workloads between the providers of a resource or service (servers) and service requesters (clients)—communication via network
- A server runs programs that share the server's resources with clients, while a client requests a server's content or service function
- A computer can be a client, a server, or both, determined by the nature of the application that requires its service functions

Client-Server Architectures: Discussion

+ Distributed nature makes it easy to integrate new clients or upgrade servers without affecting the IS's other parts
- Every server constitutes a single point of failure

- IS based on client-server architectures have led to various advances in information technology *(Alonso et al., 2004)*
  - *Remote procedure call* (see lecture about Middleware)
  - Application Programming Interfaces (APIs)
  - …

Client-Server Architecture: Task Distribution

- In a client-server architecture, tasks, workloads, and capabilities can be distributed along the IS's participants to different degrees—different client types

Thin Client

- Lightweight computer that is optimized to establish a remote connection to the network, capturing user input, and displaying output
- Server does most of the work
- Client is easier to install and maintain

Fat Client

- Provides rich functionality independent of the servers—characterized by the ability to perform many functions without connection
- Still requires periodic connection to network and servers

Tier Architectures: Motivation

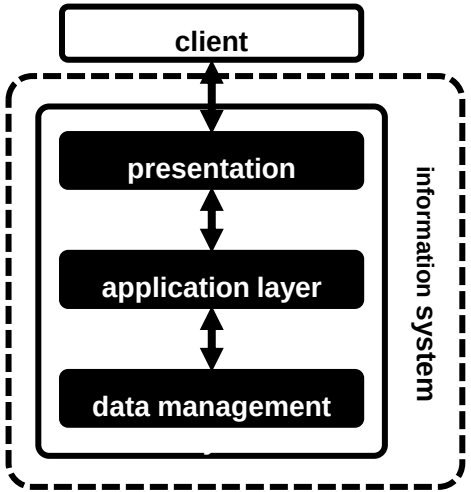
- IS architects often decompose IS into layers that represent a logical grouping of functions
- Basic idea: *Minimize dependencies* between layers, to allow modification of specific layers, instead of reworking the entire IS
- A layer is a *logical* structuring mechanism for the elements that make up the software solution

≠

- A tier is a *physical* structuring mechanism for the IS infrastructure

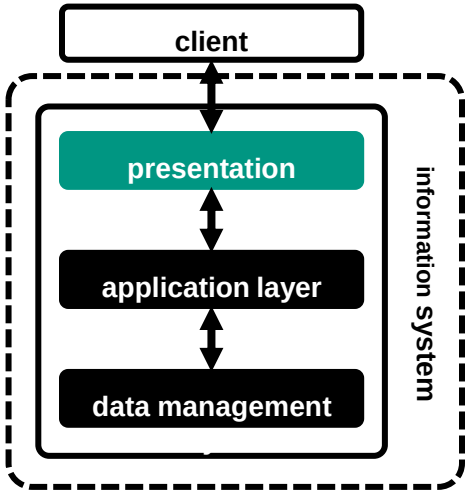
Tier Architectures: Motivation

- In general: A layer may only send a request to a layer below it!
- The most common classification has three layers *(Alonso et al., 2004)*:
  - The presentation layer
  - The application layer
  - The data management layer
- Several different types of tier architectures:
  - One-tier architectures
  - Two-tier architectures
  - Three-tier architectures
  - Multi-tier architectures



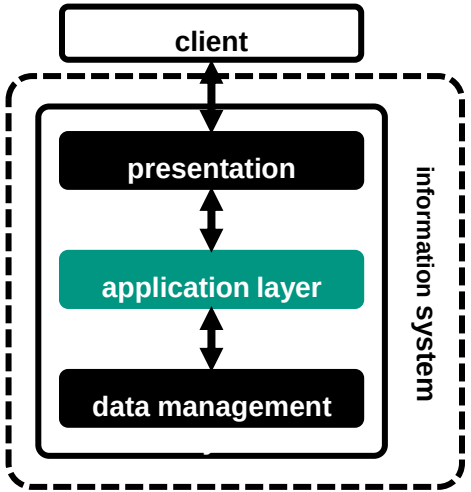
Tier Architectures: The Presentation Layer

- Every internet-based application communicates with other entities
- Information must be presented in some form to external entities to allow interaction
- Components of an IS concerned with these task form the presentation layer
- **Presentation layer ≠ IS's client**
  - Client can be completely external and not be part of IS. Example: Systems accessed through Web browser using plain HTML documents



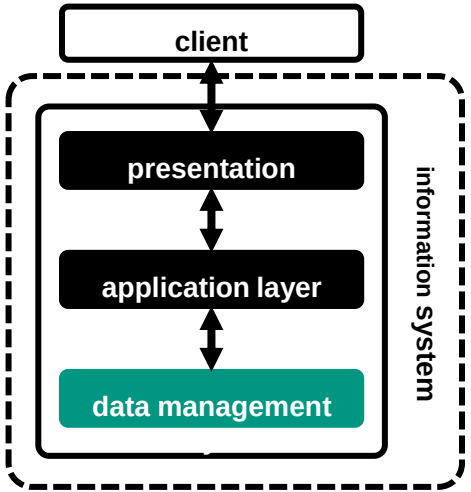
Tier Architectures: The Application Layer

- Usually, some sort of data processing happens in an IS to be able to deliver and present desired results to external entities
- *Programs* that implement the de facto operations necessary to fulfill clients' requests through the presentation layer form the application layer
- *Example:* Program that implements the operations necessary for a withdrawal request from a bank account
  - Operations: Take request, verify sufficient funds, check withdrawal limit, create log entry, update account balance, give approval



Tier Architectures: The Data Management Layer

- Most IS need some form of data to work with (e.g., stored in data repositories like databases or file systems)
- The data management layer is comprised of all of an IS's components that contribute in some way to the ongoing storage of the necessary data
- *Banking example*: Bank's account database
- Many architectures include external systems into the data management layer, who have data management layers themselves—this gives possibility of *recursive design* of an IS



Tier Architectures: Architecture Types

- Remember: A layer may only send a request to a layer below it!
- Two main architecture types that build on this

Strict Layering

- A layer is only allowed to request functionality from the layer immediately below

+ Only adjacent layers affected by change—high flexibility
+ Easier testability and maintainability
- Inefficient

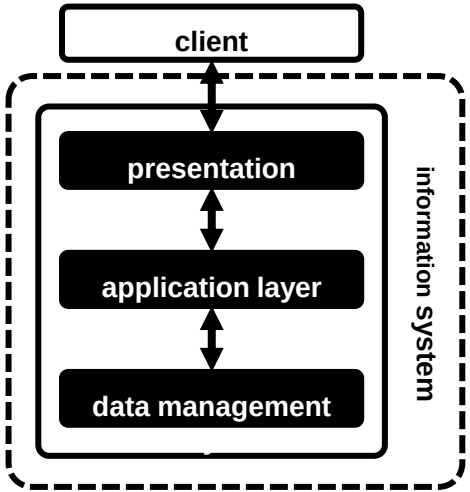
Loose Layering

- A layer is allowed to request functionality from any layer below

+ High efficiency—no need to pass data through intermediate layers
- Higher interdependencies

Tier Architectures: One-Tier Architectures

- One-tier architectures emerged from computer architectures used several decades ago
- All layers were merged into a single tier, because architects had no choice because of limited computing power
- Interaction with ISs via terminals
- Entire presentation layer resides in server—controls every aspect of interaction with client

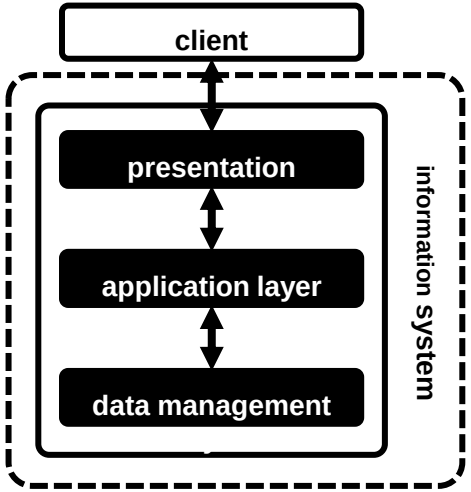


Tier Architectures: One-Tier Architectures

+ Architects are free to merge layers as much as necessary
+ Liberal use of assembly code and low-level optimizations possible to increase throughput and reduce response time

- Difficult and expensive to maintain
- Difficult to modify, because of a lack of architectural understanding, insufficient documentation, and lack of qualified programmers

- Today: Development of efficient one-tier architectures technically possible—however, mostly relevant to those who deal with old mainframe legacy systems

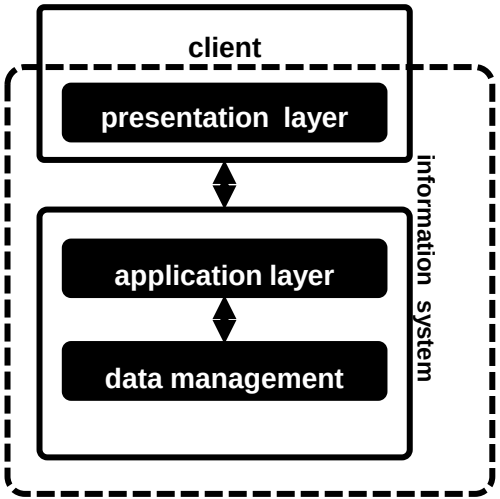


Tier Architectures: Two-Tier Architectures

- Possibility of moving the presentation layer to the IS's clients (i.e., to the PCs) led to the emergence of two-tier architectures

+ More available resources for application and data management layers on server
+ Tailoring of presentation layer for different clients possible without increasing complexity

- *Legacy problem*: If client integrates services from different servers, it has many additional requirements, greatly increasing complexity, and resulting in another application layer in the client

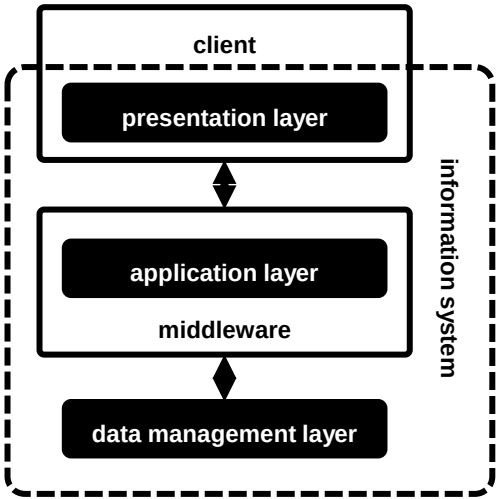


Tier Architectures: Three-Tier Architectures

- Three-tier architectures introduce an additional tier between clients and servers in order to solve the problem of highly complex clients

- Usually based on a clear separation of layers:
  - Presentation layer: client
  - Application layer: middleware
  - Data management layer: all servers

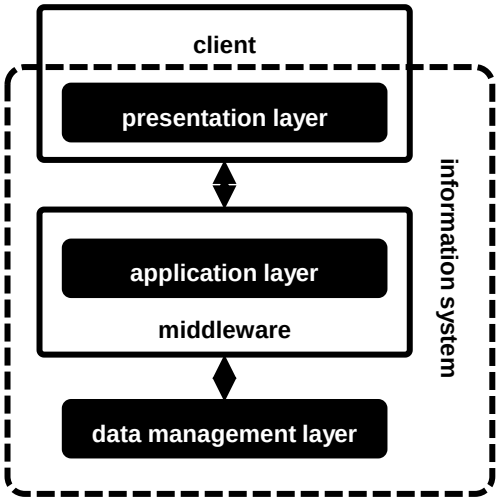
- Middleware: Infrastructures that support the development of the application logic *(focus of later lecture)*



Tier Architectures: Three-Tier Architectures

+ Scalability by running each layer on a different server
+ Opportunity to write application logic that is less tied to underlying data management

- More communication effort between data management layer and application layer
- Legacy problem, when integration has to happen via the Internet



Tier Architectures: Multi-Tier Architectures

- Increased relevance of the Internet as an access channel led to multi-tier architectures
  - Incorporation of web servers as part of the presentation layer
  - Treated as an additional tier due to complexity

- Very complex architecture of IS that can encompass many different tiers through successive integration efforts

- Often contains middleware with redundant functionality
- Efforts and costs strongly increase with number of tiers

Tier Architectures: Multi-Tier Architectures

- Many multi-tier IS today encompass a large collection of networks, single computing devices, clusters, and links to other IS
  - Difficult to identify where one system ends and the next begins

- Remote clients of all types access the system via the Internet
- Requests are forwarded to a cluster of machines that comprise the web server
- Internally, there may be additional clients across the organization who also use the IS's services
- Commonly, the application logic is also distributed across a cluster of machines
  - Possibly even middleware platforms for different applications and functionalities coexisting in the same IS
- Backend: data management layer

➢ Highly complex!

Scalability of Distributed Systems

- Q: How can we **increase the number of supported clients**?
- **Vertically scale up** the server
  - add more resources to existing machines
  - more memory, faster processor, more processor cores
- **Horizontally scale out** the server
  - add more physical machines
- Q: What about vertical/horizontal scalability in **one-tier architectures**?

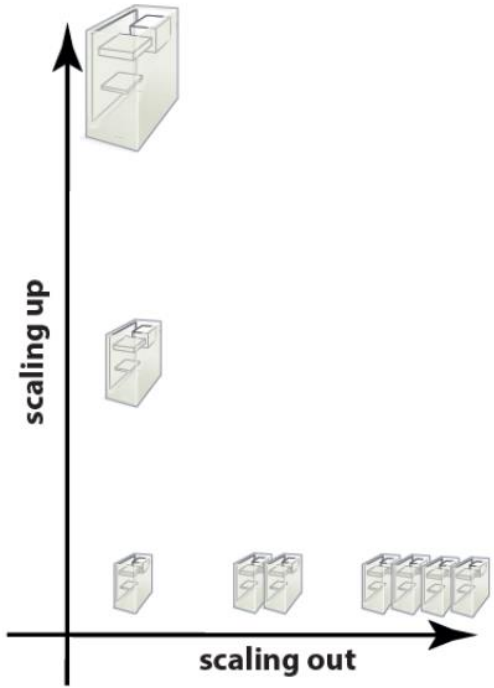
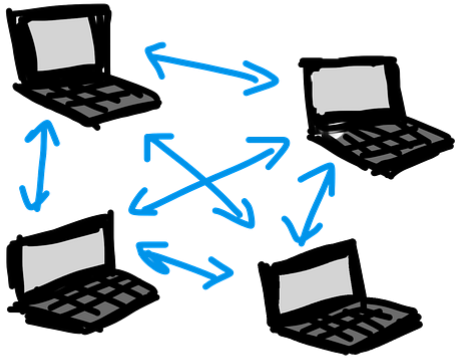


Figure according to Turowski, M., & Lenk, A. (2015). Vertical scaling capability of OpenStack. In Service-Oriented Computing-ICSOC 2014 Workshops (pp. 351-362). Springer, Cham.

Peer-to-Peer Architectures: Basics

- Basic thought: Every participant of a network (also called peer or node) has the same capabilities and responsibilities *(Steinmetz and Wehrle, 2004)*
- Peers make a portion of their resources available to other peers without need for central coordination
  - Processing power, storage, network bandwidth, ...
- Popularized in 1999 by file-sharing system *Napster (Hess et al., 2002)*
- Today: Division of peers into groups, depending on qualifications, which take on specific tasks
  - Internal overlay network consisting of the most suitable computers that takes over the organization of the other computers



napster

Image source: [Computer Network] by Schoen Sandra, May 10th 2014. Pixabay License.

Peer-to-Peer Architectures: Types

- Two main types of peer-to-peer architectures *(Schollmeier, 2001)*

Unstructured peer-to-peer networks	Structured peer-to-peer networks
Formed by nodes that randomly form connections to one another—do not impose a particular structure *(Lv et al., 2002)*	Overlay is organized into a specific topology—protocol ensures that any node can efficiently search the network for a file or resources, even if this resource is extremely rare *(Dabek et al., 2003)*

- Main application purpose of peer-to-peer architectures: Content distribution
  - Software publication and distribution (e.g., file sharing), content delivery networks (e.g., for server load balancing), streaming (e.g., peer casting), …
  - Other application domains: Science, search, and communication networks

Peer-to-Peer Architectures: Discussion

- Peer-to-peer architectures are a key issue in the controversy about network neutrality
  - *No restrictions on Internet content, formats, technologies, equipment or modes of communication*

- Supporters of peer-to-peer architecture argue that governments and large ISPs can control Internet content by directing network structure toward a client-server architecture
  - Financial barriers for individuals and small publishers; inefficiencies in sharing large files

Peer-to-Peer Architectures: Discussion

- Assessing advantages and disadvantages of peer-to-peer architectures involves comparisons with client-server architectures

- Each new node adds demand to the system:
  - *Peer-to-peer architecture*:
    - New nodes also add capacity, because new nodes are also required to share resources
    - Enormous increase in system security and file verification mechanisms → High resistance against attacks
  - *Client-server architecture*
    - Share of demand, but not of resources
    - Given number of resources must be shared between increasing number of clients

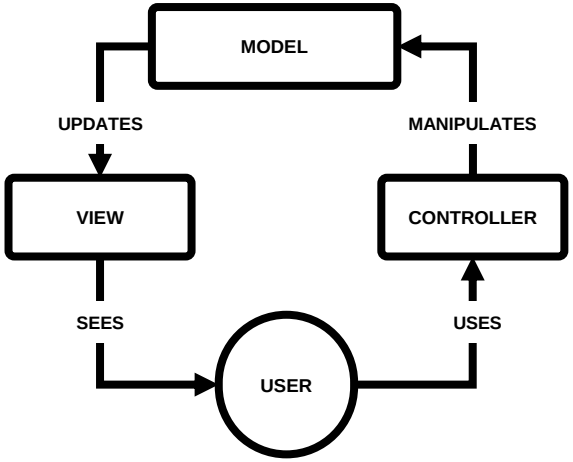
Model-View-Controller Architectures: Basics

- The Model-View-Controller (MVC) architectural pattern divides an IS into three subcomponents:
  - The model contains data represented by the associated presentation and operations
  - The presentation (view) is responsible for the representation of the model data and the realization of user interactions
  - The controller manages the presentation and the model

- Main objective: Flexible program design that facilitates later modification, as well as reusability of individual components *(Krasner and Pope, 1988)*
  - *Example:* Writing an application using the same model and making it available to Windows, Mac, and Linux—only controller and view must be reimplemented

Model-View-Controller Architectures: Interactions

- The three components in an MVC interact with each other in several ways
  - View knows the model whose data it represents—is informed about changes to the data and can update the representation accordingly
  - Controller manages both the view and the model
  - View informs controller about user interactions, it evaluates these, and adjusts the view and changes the data in the model



Service-Oriented Architecture: Basics

- Basic Goal: Increase re-usability of business processes by encapsulating them into services *(Alonso et al., 2004; Papazoglou, 2012)*
- A service is considered as a *“representation of a repeatable business activity that has a specified outcome, that is self-contained, that may be composed of other services, and that is a ‘black box’ to consumers of the service” (The Open Group 2009)*
  - Not restricted in their granularity—may perform a simple business function or a highly complex process
- Service-oriented architecture (SOA) describes an abstract IS architecture that delivers services to clients via published and discoverable interfaces

Service-Oriented Architecture: Components

- At a high abstraction level, SOA consists of three main components
  - The service provider hosts services and provides interfaces that allow a service requester to access these services
  - The service brokers are the registries, where all means to access the service are published (mainly the *interface*)
  - The service requester can query the broker's repository to find a service offering with matching characteristics
- Once a matching service is found, the service broker passes the required information to the service requester, who can then bind the service interface and invoke the underlying service
  - Multiple providers can offer the same service without affecting the service requester
- Dynamic find-bind triangle
  - Allows for construction of highly flexible and reliable IT infrastructures *(Georgakopoulos and Papazoglou, 2006)*
  - Lowers the effort required in provision / consumption of services via the internet

Summary

- IS are complex systems—to handle this complexity, developers utilize representations of IS architectures
  - Performing an architectural analysis can yield many benefits (*documentation, communication* , …)
- Different architectural views can address differing needs of stakeholders (developers, business managers, …)
  - *Nine principles of IS architectures*
  - Kruchten's 4+1 architectural view model (1995): *Logical view, process view, development view, physical view*
- IS architects aim to capture best practices in architectural patterns
  - Client-server architectures as a fundamental pattern of modern applications
  - Tier architectures: Layering of systems into presentation / application / data management layer; one-/two-/three-/multi-tier architectures
  - Peer-to-peer architectures: Built around the concept of peers
  - Model-view-controller (MVC): Division of system into model, view, and controller
  - Service-oriented architectures (SOA): Built around the idea of services

References 1/2

Alonso G, Casati F, Kuno H, Machiraju V (2004) Web services. In: Alonso G, Casati F, Kuno H, Machiraju V (eds) Web services: concepts, architectures and applications. Data-centric systems and applications, 1st edn. Springer, Berlin, pp 123–149

Avgeriou P, Zdun U (2005) Architectural patterns revisited: a pattern language. Paper presented at the 10th European conference on pattern languages of programs, Irsee, 6–10 July 2005

Boh WF, Yellin D (2006) Using enterprise architecture standards in managing information technology. J Manag Inf Syst 23(3):163–207

Buschmann F, Henney K, Schmidt DC (2007) On patterns and pattern languages. Pattern-oriented software architecture, vol 5. Wiley, Chichester

Castro J, Kolp M, Mylopoulos J (2002) Towards requirements-driven information systems engineering: the Tropos project. Inf Syst 27(6):365–389

Dabek F, Zhao B, Druschel P, Kubiatowicz J, Stoica I (2003) Towards a common API for structured peer-to-peer overlays. Paper presented at the international workshop on peer-to-peer systems, Berkeley, CA, 20–21 Feb 2003

Dropbox (2019) Under the hood: architecture overview. https://www.dropbox.com/business/trust/ security/architecture. Accessed 29 May 2019

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable objectoriented software. Addison-Wesley professional computing series. Addison-Wesley, Boston, MA

Georgakopoulos D, Papazoglou MP (2006) Overview of service-oriented computing. Paper presented at the 4th international conference on service oriented computing, Chicago, IL, 4–7 Dec 2006

Golden B (2013) A unified formalism for complex systems architecture. Ecole Polytechnique, Palaiseau

Hess T, Anding M, Schreiber M (2002) Napster in der Videobranche? Erste Überlegungen zu Peerto-Peer-Anwendungen für Videoinhalte. In: Schoder D, Fischbach K, Teichmann R (eds) Peerto-peer. Xpert.press. Springer, Berlin, pp 25–40 Huhns MN, Singh MP (2005) Service-oriented computing: key concepts and principles. IEEE Internet Comput 9(1):75–81

ISO/IEC (2005) Information technology – open distributed processing – unified modeling language (UML) version 1.4.2. https://www.iso.org/standard/32620.html. Accessed 15 Sept 2019

ISO/IEC/IEEE (2011) Systems and software engineering – architecture description. https://www. iso.org/standard/50508.html. Accessed 15 Sept 2019

Kircher M, Jain P (2013) Pattern-oriented software architecture, patterns for resource management. Wiley software patterns series, vol 3. Wiley, Chichester

References 2/2

Krasner GE, Pope ST (1988) A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. J Object-Oriented Program 1(3):26–49

Kruchten P (1995) The 4+1 view model of architecture. IEEE Softw 12(6):42–50 Lankhorst M (2017) Enterprise architecture at work. The enterprise engineering series, 4th edn. Springer, Berlin

Lins S, Thiebes S, Schneider S, Sunyaev A (2015) What is really going on at your cloud service provider? Creating trustworthy certifications by continuous auditing. Paper presented at the 48th Hawaii international conference on system sciences, Kauai, Hawaii, 5–8 Jan 2015

Lins S, Grochol P, Schneider S, Sunyaev A (2016a) Dynamic certification of cloud services: trust, but verify! IEEE Secur Priv 14(2):66–71

Lins S, Schneider S, Sunyaev A (2016b) Trust is good, control is better: creating secure clouds by continuous auditing. IEEE Trans Cloud Comput 6(3):890–903

Lv Q, Cao P, Cohen E, Li K, Shenker S (2002) Search and replication in unstructured peer-to-peer networks. Paper presented at the 16th international conference on supercomputing, New York, NY, 22–26 June 2002

Papazoglou MP (2012) Web services and SOA: principles and technology, 2nd edn. Pearson, Harlow

Reenskaug T (1979) Thing-model-view-editor: an example from a planning system. http://heim.ifi. uio.no/~trygver/1979/mvc-1/1979-05-MVC.pdf. Accessed 15 Sept 2019

Schmidt DC, Stal M, Rohnert H, Buschmann F (2013) Pattern-oriented software architecture, patterns for concurrent and networked objects. Wiley software pattern series, vol 2. Wiley, Chichester

Schollmeier R (2001) A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. Paper presented at the 1st international conference on peer-topeer computing, Linköping, 27–29 Aug 2001

Sommerville I, Sawyer P (1997) Requirements engineering: a good practice guide. Wiley, New York

Steinmetz R, Wehrle K (2004) Peer-to-peer-networking & computing. Informatik-Spektrum 27 (1):51–54

The Open Group (2009) SOA source book. Van Haren Publishing, Zaltbommel

Winter R, Fischer R (2006) Essential layers, artifacts, and dependencies of enterprise architecture. Paper presented at the 10th IEEE international enterprise distributed object computing conference workshops, Hong Kong, 16–20 Oct 2006

Zachman JA (1987) A framework for information systems architecture. IBM Syst J 26(3):276–292

Questions

Questions

1. What are the main purposes of IS architecture analysis?
2. What are the nine principles of IS architecture?
3. What are architectural views and why is it important to perform architectural analysis from different perspectives?
4. How are tasks and workload distributed in a client-server architecture?
5. What is the main difference between a client-server architecture and a peer-to-peer architecture?
6. What is the fundamental concept behind service-oriented architecture?

Further Reading

- Alonso G, Casati F, Kuno H, Machiraju V (2004) Web services. In: Alonso G, Casati F, Kuno H, Machiraju V (eds) Web services: concepts, architectures and applications. Data-centric systems and applications, 1st edn. Springer, Berlin, pp 123–149
- Sommerville I, Sawyer P (1997) Requirements engineering: a good practice guide. Wiley, New York