Fachhochschule Nordwestschweiz
Hochschule für Technik

Software Architecture

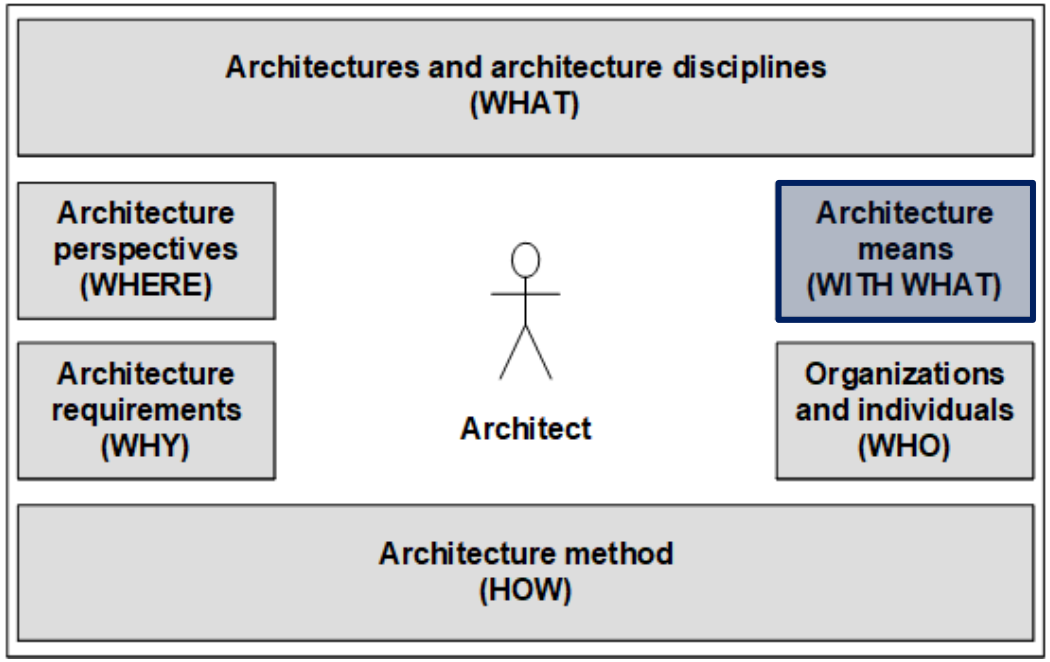
Architecture Means

BSc

Ingo Arnold

Lecture Opening
Architecture Orientation Framework

Architecture **WITH WHAT discusses the broad range of architectural means architects use to plan, develop, and operate architectures**. Thus what we consider here is the architect's toolbox, if you wish. [Vogel, Arnold et al 2011].



Lecture Opening
Motivation

The software architect's toolbox is huge and offers a broad variety of tools. While no architect is an expert of all tools in that box, every architect should at least have an overview of its drawers, sub-drawers and sub-sub-drawers.

We will not be able to look at the entire toolbox in detail. In this deck I will only give a rough overview of the toolbox.

In the following units, we will then delve further into selected topics.

Lecture Opening
Learning Objectives

You …

- gained an overview of the architect's toolbox.
- understand the rough classification of the toolbox as well as its overall structure.
- know some of the architectural tools in the toolbox, so that you can orientate yourself in our following in-depth lectures.

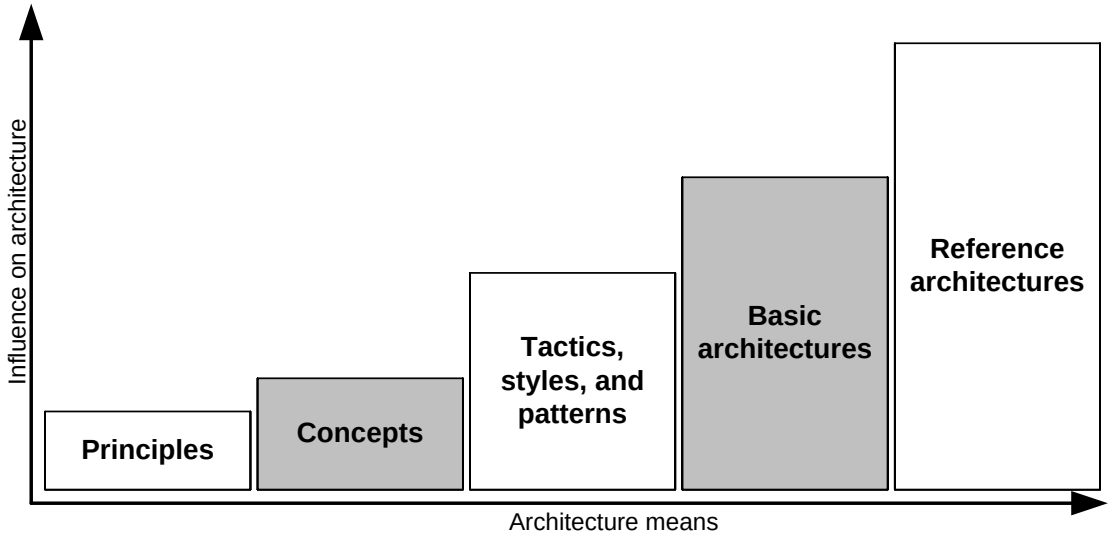
Lecture Agenda

- Architecture Means
- Architecture Principles
- Basic Architecture Concepts
- Architecture Tactics, Styles, and Patterns
- Basic Architectures
- Reference Architectures

Architecture Means
Overview

Different types of architecture means influence the architecture under construction in different ways. The influence increases from the architecture principles right up to the reference architecture.

We will only dive deeper into a few selected principles, basic concepts, etc. in addition to the overview provided in this deck. For further classification, my book is recommended accordingly.



Lecture Agenda

- Architecture Means
- Architecture Principles
- Basic Architecture Concepts
- Architecture Tactics, Styles, and Patterns
- Basic Architectures
- Reference Architectures

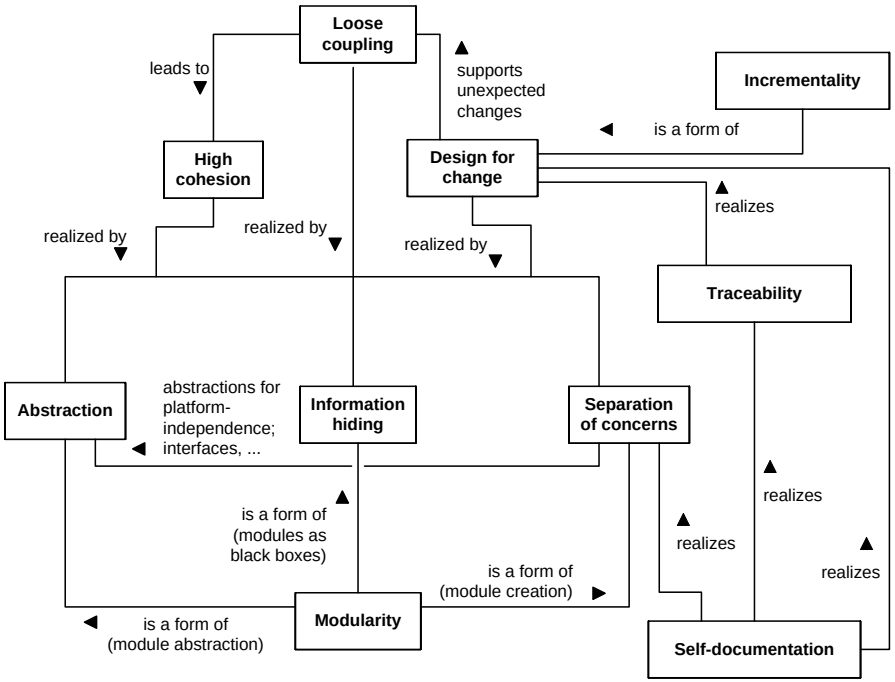
Architecture Principles
Overview

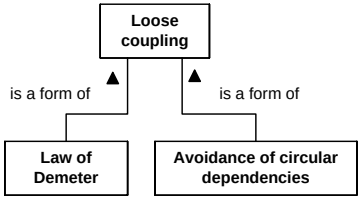
Architecture principles are proven guidelines that you should apply when developing or modifying an architecture. However, they say nothing about how you should apply these principles in concrete cases.

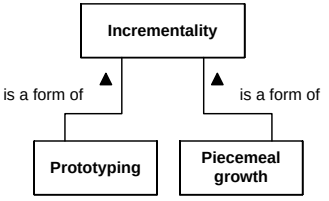
It is difficult to say that an architecture is good or bad in itself (it merely fulfils its conditions more or less well). However, if you take general principles into account when designing a software architecture, you tend to get a good architecture.

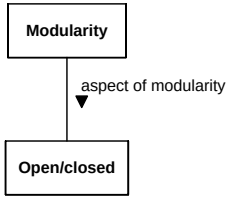
Architecture Principles
Overview

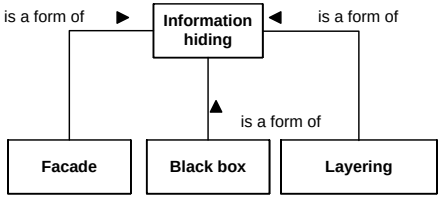
Selected principles and their relationships.

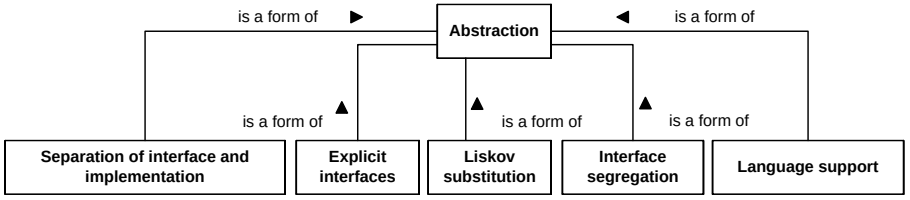






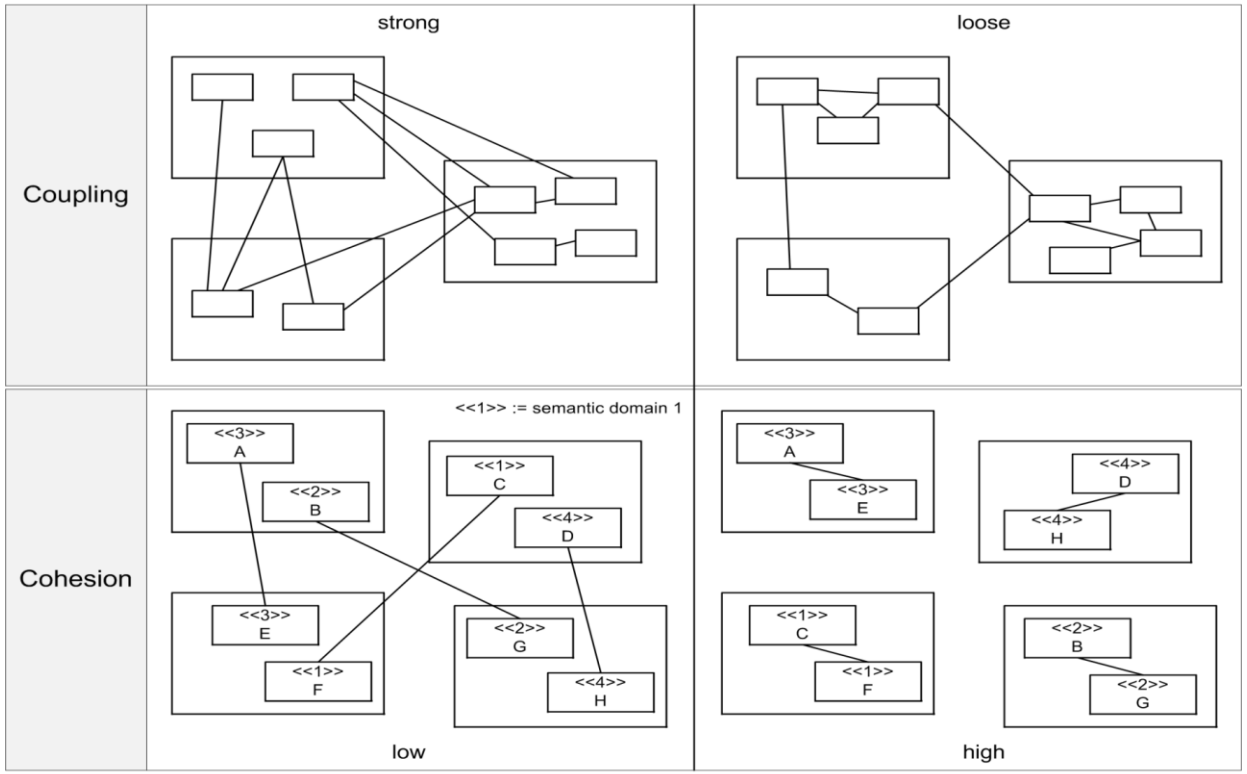






Architecture Principles
Coupling

Mutual dependencies exist between many architectural principles, as illustrated here using *coupling* and *cohesion* as examples.



Architecture Principles
Coupling

Consider this program and describe what Client binds to in terms of the Database, DatabaseConnection, Query, and ResultSet classes?

```
public class Client {
	public static void main(String[] args) {
	    DatabaseConnection dbCon = Database.newConnection("protocol:server:port:db:connection-details");
	    dbCon.open();

	    // a thousand things are happening in between

	    Query query = dbCon.createQuery("Select * from Users");
	    ResultSet rs = query.perform();

	    while(! rs.eof() ) {
	        String name = rs.getColumn("name");
	        String firstName = rs.getColumn("firstName");
	        String city = rs.getColumn("city");
	        System.out.println(firstName + " " + name + " " + city);

	        String record = rs.getRecord();
	        System.out.println(record);

	        rs.next();
	    }

	    // a thousand things are happening in between

	    dbCon.close();
	}
}
```

Architecture Principles
Coupling

Consider this program and describe what Client binds to in terms of the Database, DatabaseConnection, Query, and ResultSet classes?

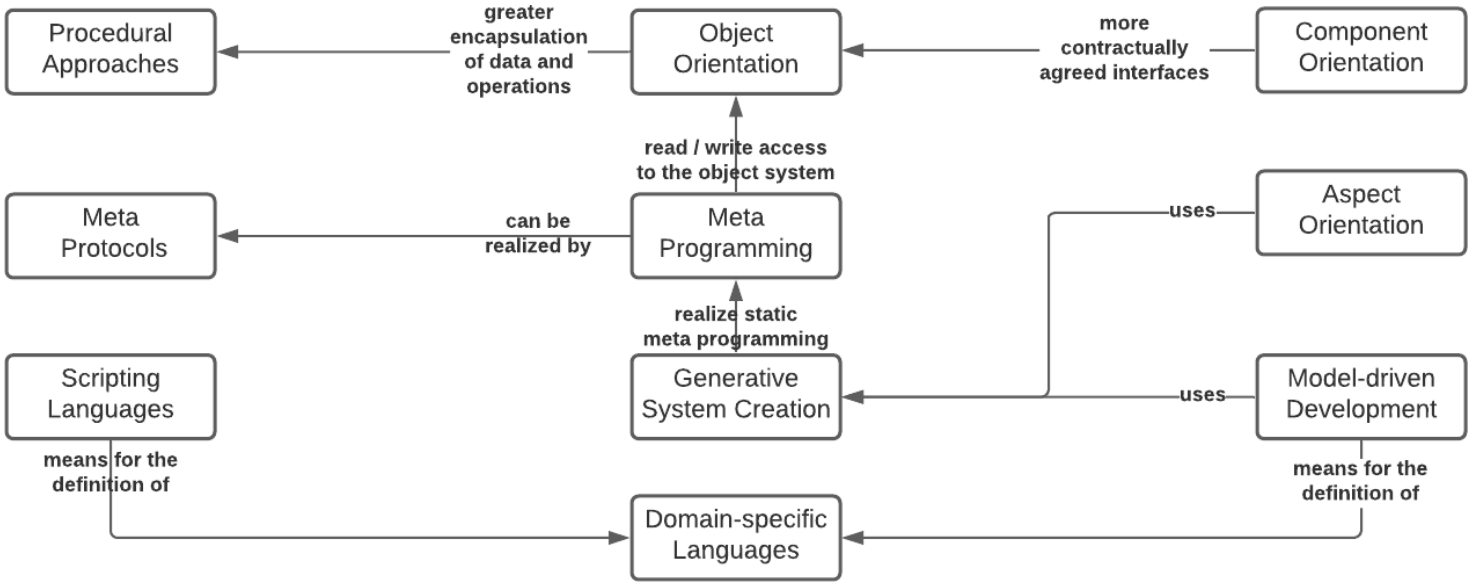
Lecture Agenda

- Architecture Means
- Architecture Principles
- Basic Architecture Concepts
- Architecture Tactics, Styles, and Patterns
- Basic Architectures
- Reference Architectures

Basic Architecture Concepts
Overview

Basic Concepts is what the name implies: essential paradigms and concepts of architecture.

In my book, this section starts with simple procedural approaches and then gradually moves to more comprehensive approaches such as aspect orientation, component orientation, and model-driven architecture.



Basic Architecture Concepts
Overview

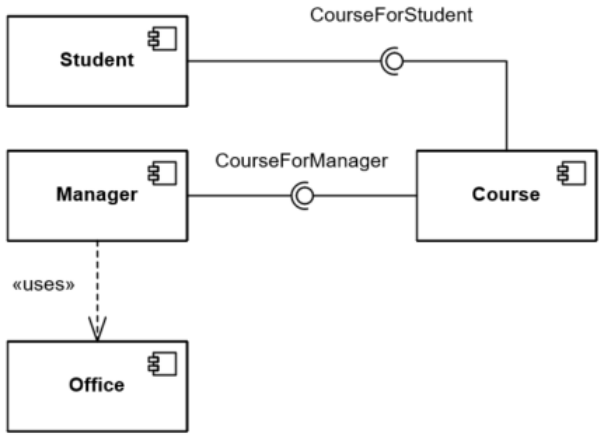
Brief overview of basic architecture concepts.

Procedural Approaches	Procedures are a traditional and widely used means of structuring architectures. They allow the decomposition of a complex algorithm into repeatable individual algorithms. Procedures implement the principle of separation of concerns, since they make it possible to decompose a complex algorithmic problem into simpler subproblems
Object Orientation	Object orientation is based on the idea of bundling data processed by a set of related procedures together with these procedures. The procedures – called operations or methods in object orientation – can process their data exclusively. In this way, object orientation attempts to directly implement the principle of information hiding and the principle of modularity

Basic Architecture Concepts
Overview

Brief overview of basic architecture concepts.

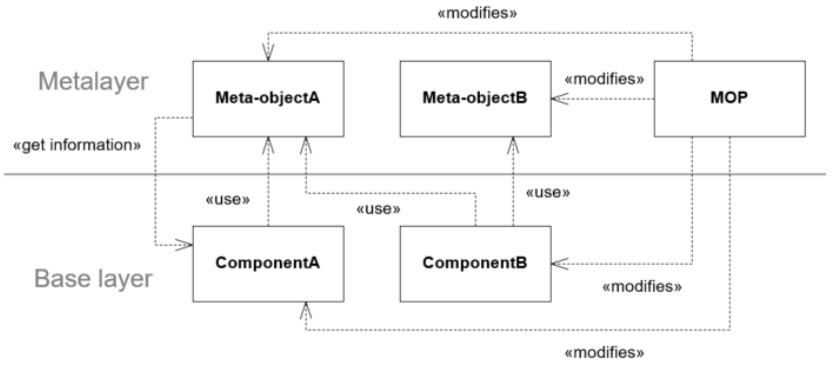
Component Orientation	Components are reusable, self-contained building blocks of an architecture. The component orientation arose from the problem that objects implement the modularity principle, but are often too small to be used as reusable units. In addition one desires from a reusable system building block often further characteristics, like the implementation of non-functional requirements, or the simultaneous provision of several identical component instances for increasing the scalability. Component platform examples are DLLs, JavaBeans, ActiveX, JEE components like EJBs, .NET components, CORBA components



Basic Architecture Concepts
Overview

Brief overview of basic architecture concepts.

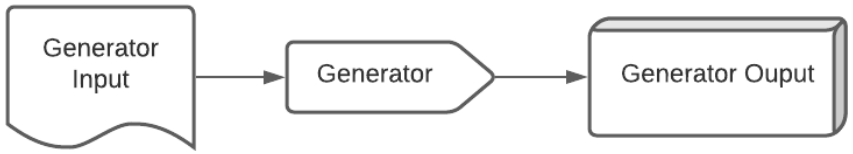
Meta Programming	Many programming languages distinguish between the program as a set of executable instructions and the data that the program works with. Actually, the programs themselves are also data. So in metaprogramming, programs are treated as data. Programs that use other programs (or themselves) as data are usually called metaprograms. Meta programming techniques and technologies are reflection or introspection capabilities (e.g., java reflection API), macro mechanisms (e.g., C++ pre-processor changes the program before it is sent to the C++ compiler)



Basic Architecture Concepts
Overview

Brief overview of basic architecture concepts.

Generative System Creation	If we look beyond software development, we see that in other engineering disciplines, processes are always automated when certain recurring tasks have to be performed in a similar way. The use of generative technologies in software development aims to adapt the degree of automation in the creation of software to that of other engineering disciplines. Examples of generators are XSLT, annotation-based Javadoc generators, IDL generators, “inline” keyword in C/C++



Basic Architecture Concepts
Overview

Brief overview of basic architecture concepts.

Model-Driven Development	Model-driven software development (MDD) is when models are not only used for documentation purposes, but are central artifacts of an executable system. A special case of the generative system creation discussed above. Generator input is a system model while the generator output is the physical equivalent of the initially modelled system.
Aspect Orientation	Aspect orientation avoids spreading so-called crosscutting concerns across the code or design. These are concerns that are general or go beyond the application logic. Security is an example of such a concern.

Basic Architecture Concepts
Overview

Brief overview of basic architecture concepts.

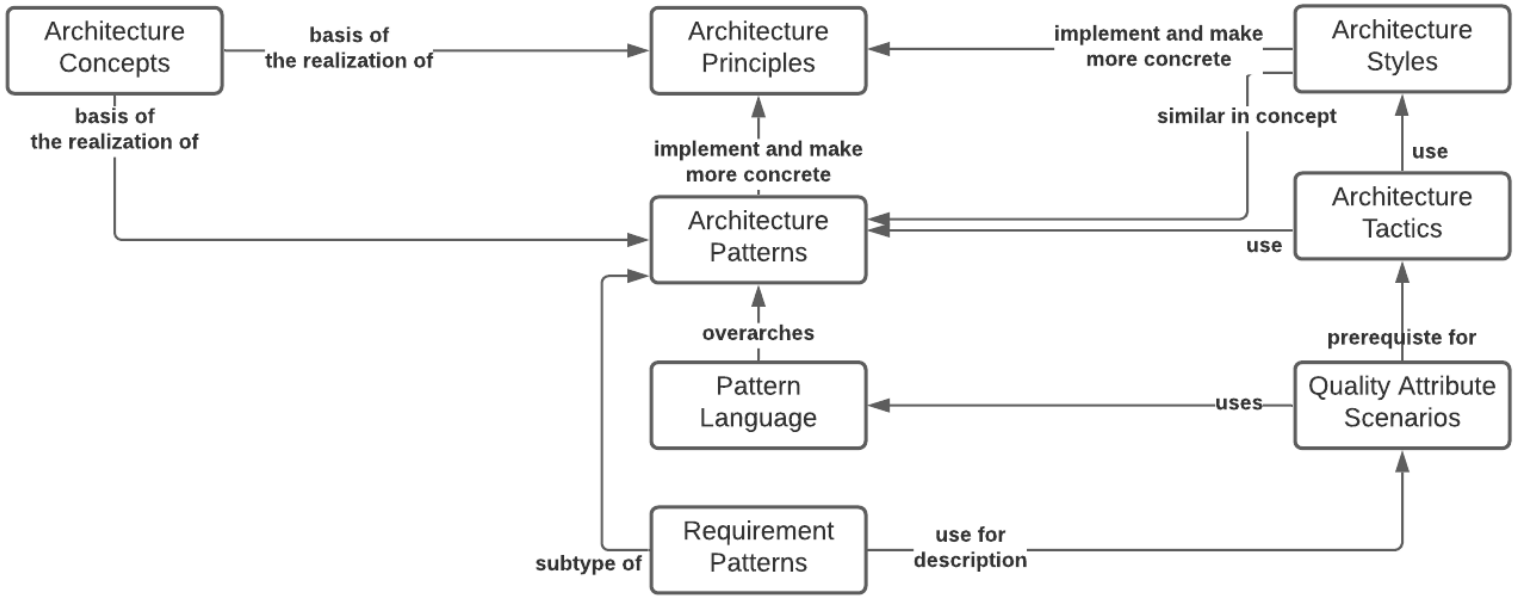
Scripting Languages and Platform Object Models	Scripting languages are originally programming languages for controlling software systems. They typically use a two-language approach. The core system is implemented in a programming language other than the scripting language, and the scripting language only handles the composition of the system components in an executable system. Examples are JavaScript and the HTML DOM, or ABAP in SAP systems.

Lecture Agenda

- Architecture Means
- Architecture Principles
- Basic Architecture Concepts
- Architecture Tactics, Styles, and Patterns
- Basic Architectures
- Reference Architectures

Architecture Tactics, Styles, and Patterns
Overview

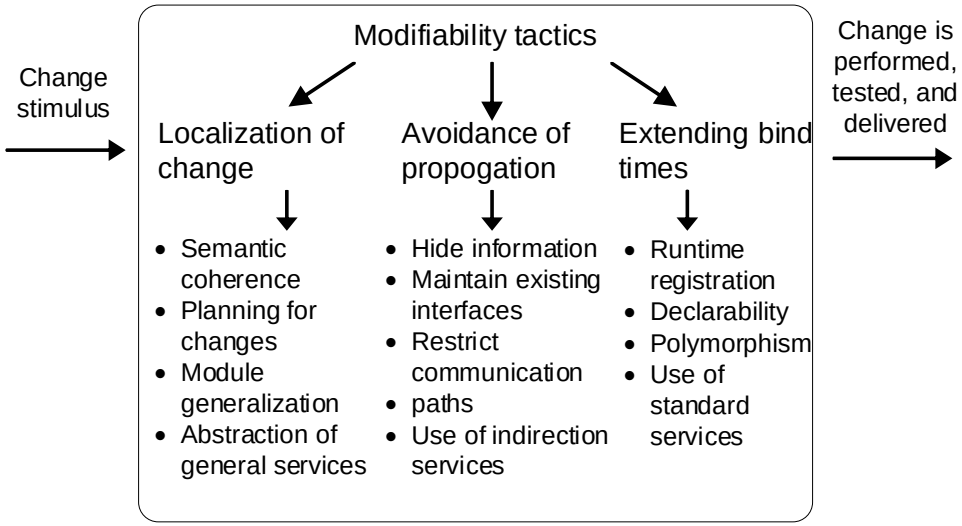
In my book this section copes with architectural tactics, styles, and patterns. These three means have in common that they describe principle solutions to certain recurring problems in architectural design.



Architecture Tactics, Styles, and Patterns
Overview

Brief overview of tactics, styles, and patterns.

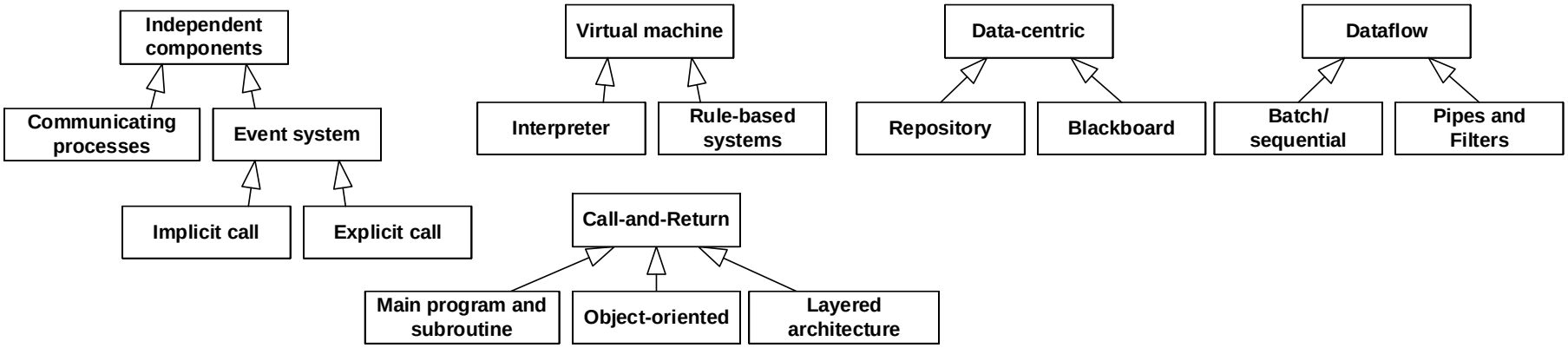
Architecture Tactic	Architectural tactics provide guidelines for implementing a wide variety of quality features of the system being designed and its architecture. So, in principle, an architectural tactic helps you to get a first idea about a design problem. You can then develop this idea further and, for example, also use styles and patterns as further tools.



Architecture Tactics, Styles, and Patterns
Overview

Brief overview of tactics, styles, and patterns.

Architecture Style	An architectural style is a pattern of structural organization of a family of systems. For them, an architectural style consists of the following elements: a set of building blocks that perform certain functions at runtime. A topological arrangement of these building blocks. A set of connectors that govern communication and coordination among the building blocks. A set of semantic restrictions that specify how building blocks and connectors can be connected to each other. An architecture style primarily reflects the fundamental structure of a software system and its properties. You can therefore use a style to categorize architectures.



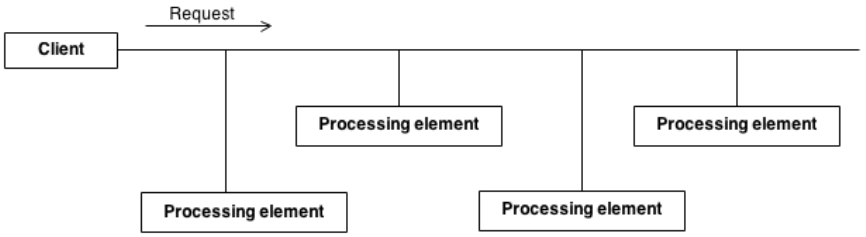
Architecture Tactics, Styles, and Patterns
Overview

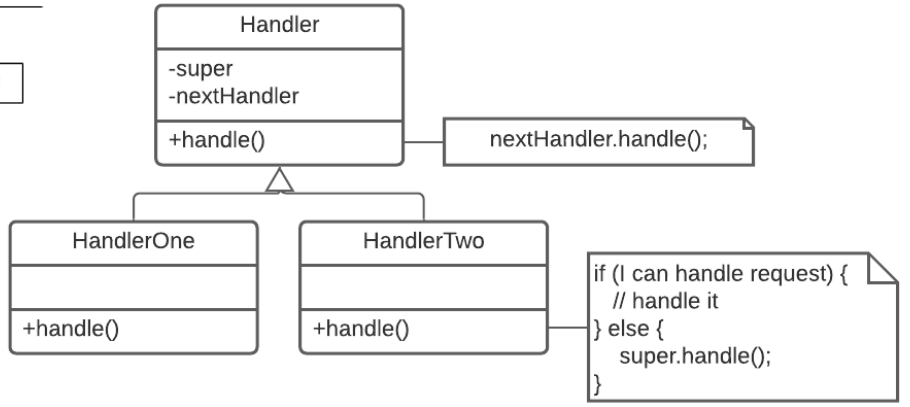
Brief overview of tactics, styles, and patterns.

Architecture Pattern

In the context of software architecture, both design patterns and architecture patterns are very important. They have in common that they usually represent structural, technical solutions. In general, design patterns describe more specific design solutions that have a local impact, while architecture patterns describe more system structures that have an impact on the entire system.

Example **chain of responsibility** pattern



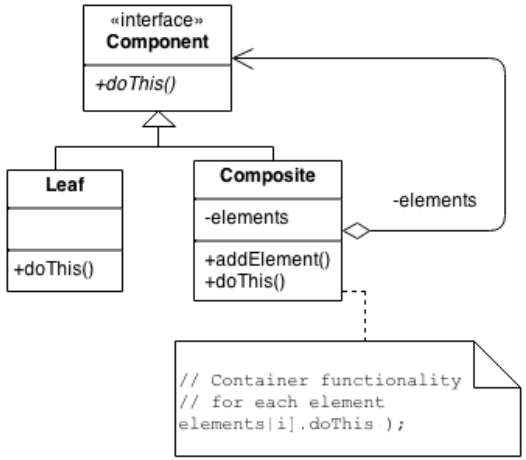


Architecture Tactics, Styles, and Patterns
Overview

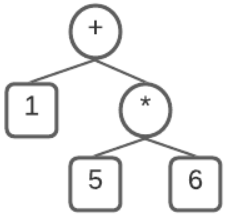
Brief overview of tactics, styles, and patterns.

Architecture Pattern

Example **composite** pattern



Apply the composite pattern for a simple expression interpreter



Architecture Tactics, Styles, and Patterns
Overview

Brief overview of tactics, styles, and patterns.

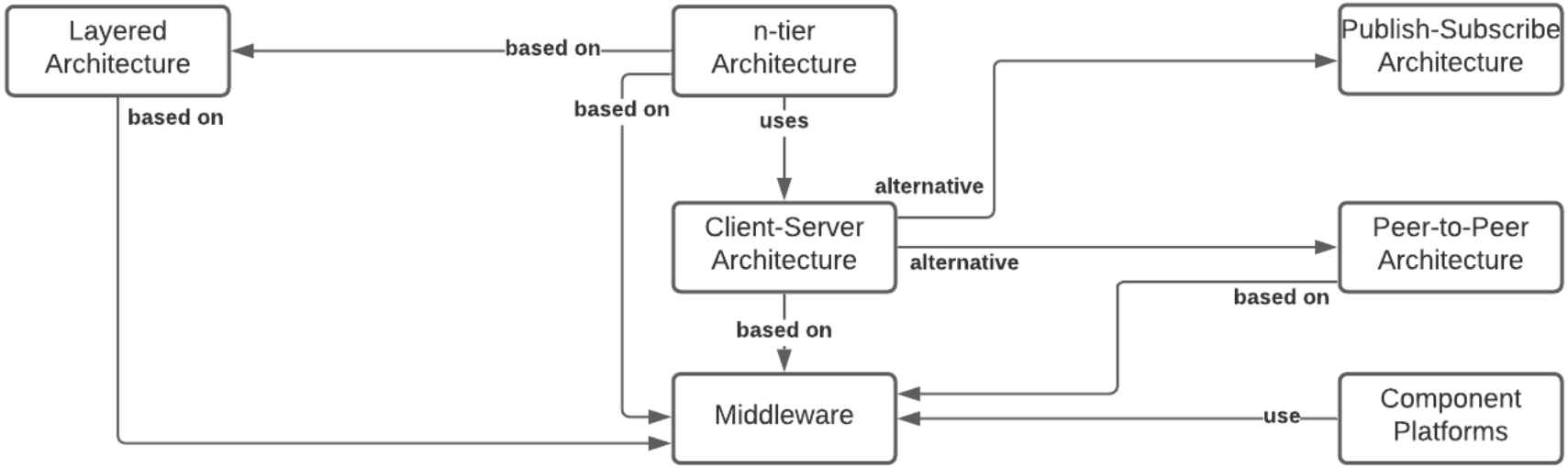
Architecture Pattern Language	A pattern language is a collection of semantically related patterns that provide solution principles for problems in a particular context. The focus of pattern languages is particularly on the relationships of the patterns in the language. Specifically, this means that the pattern descriptions are highly integrated – for example, in that the context of one pattern picks up another pattern and that particular emphasis is placed on describing the pattern interactions
Requirements Pattern	A requirements pattern is a methodical tool that you can use to systematically develop "good" requirements. Like architecture patterns, requirements patterns refer to visibly recurring issues for which generally valid solution proposals can be formulated. A requirements pattern can thus take the form of a template for concrete requirements of a specific requirements category.

Lecture Agenda

- Architecture Means
- Architecture Principles
- Basic Architecture Concepts
- Architecture Tactics, Styles, and Patterns
- Basic Architectures
- Reference Architectures

Basic Architectures
Overview

This section of my book discusses foundational architecture approaches that are adopted in many systems. Basic architectures use the various architectural means discussed earlier. They therefore represent a more concrete architectural guideline that you can use to structure entire systems.



Basic Architectures
Overview

Brief overview of basic architectures.

Layered Architecture	A fundamental issue of architectures that affects both functional and non-functional requirements is the partitioning of a system into groups of similar functions or responsibilities. The layers style describe a typical solution. One applies layers in situations where one group of building blocks depends on another group of building blocks to perform its function, and that group in turn depends on another group of building blocks, and so on. The layered pattern states that each layer groups a number of building blocks and the layer above it provides services through interfaces. In each layer, the building blocks of that layer can freely interact with each other. Between two layers, communication may only take place via the specified interfaces.
N-tier Architecture	A tier is a means of structuring the distribution of software building blocks on hardware building blocks. A tier contains interrelated system building blocks (software and hardware building blocks) that are interconnected via a network. For example, the classic client-server model based on a two-tier architecture.

Basic Architectures
Overview

Brief overview of basic architectures.

Client-Server Architecture	In modern software architecture, client-server operation is of great importance. Here, the user runs application programs (clients) on his computer and these programs access the resources of the server. The resources are centrally managed, shared and made available. The client-server model is based on a simple request-response scheme. This means that files no longer have to be transferred as a whole, but can be requested specifically.
Publish Subscribe Architecture	Publish-subscribe addresses the problem of informing a set of clients about runtime events. Publish-subscribe allows the event consumer to register for specific event types. When such an event occurs, the event producer informs all registered consumers - for example, via a publish/subscribe system. The publish-subscribe system thus decouples the producer and the consumer of events.

Basic Architectures
Overview

Brief overview of basic architectures.

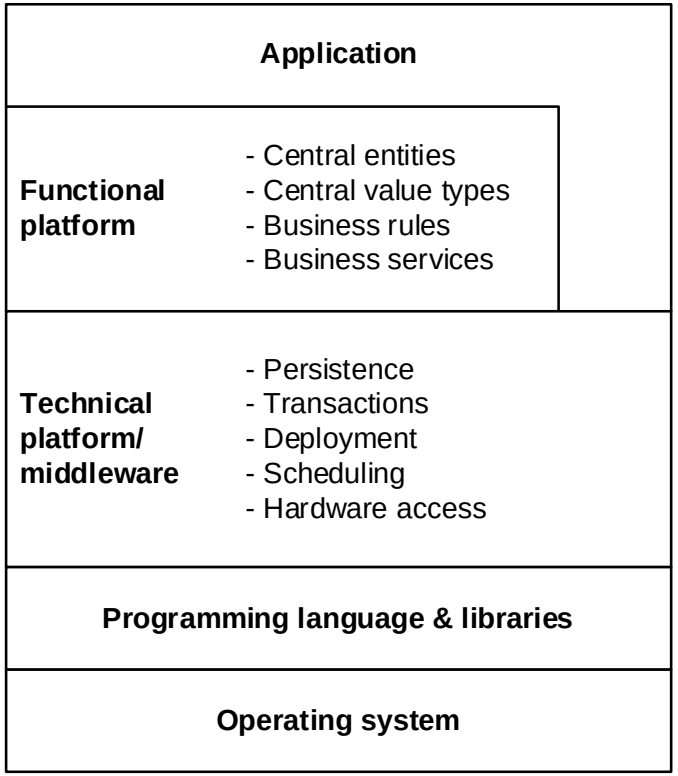
Peer-2-Peer Architecture	Peer-to-peer (P2P) uses a number of equal peers. In the pure P2P architecture structure, there is no central server. Each peer can offer and take advantage of services on the network. The entire status of the system is distributed among the peers. Internally, P2P systems are partly implemented with standard middleware, but the user of a P2P system is not aware of this. In the P2P architecture, one can add or remove a service at any time. Clients must therefore find out which services are currently available before using a service.
Middleware	Middleware establishes a platform that provides application services for all aspects of distribution, such as distributed invocation, efficient access to the network, transactions, messaging and directory services, time services, security services, and more. Middleware establishes a type of network operating system (NOS) that makes the complexities of a distributed environment accessible to software developers efficiently and effectively.

Basic Architectures
Overview

Brief overview of basic architectures.

Component Platforms

Typical examples of component platforms in the enterprise environment are JEE or .NET. These component platforms are based on the separation of technical concerns and functional concerns for an information system. Examples of technical concerns in the enterprise environment include distribution, security, persistence, transactions, concurrency, and resource management.
In other environments, such as embedded systems, other technical concerns may be important.

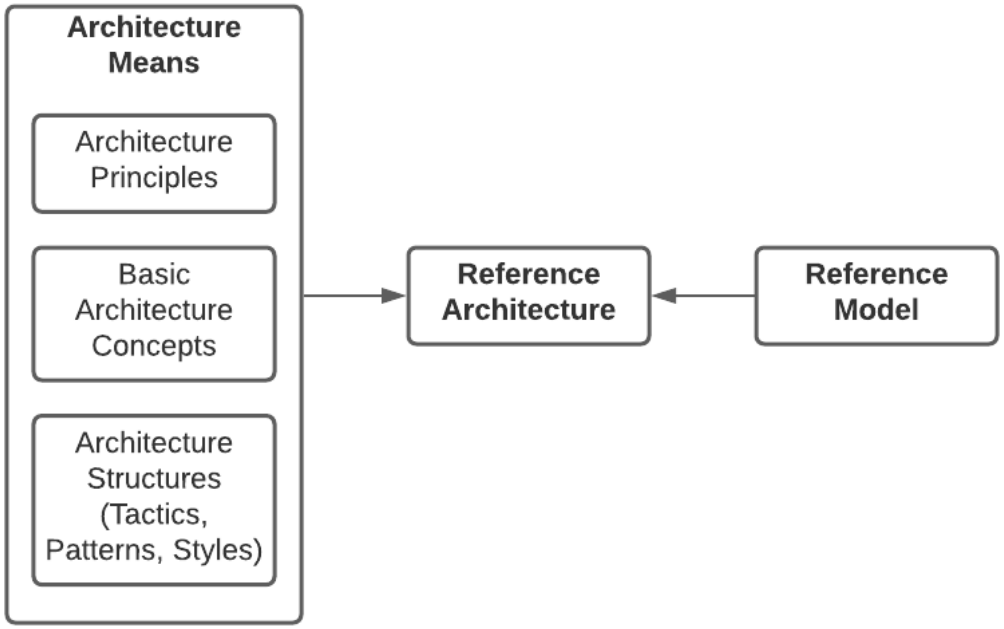


Lecture Agenda

- Architecture Means
- Architecture Principles
- Basic Architecture Concepts
- Architecture Tactics, Styles, and Patterns
- Basic Architectures
- Reference Architectures

Reference Architectures
Overview

Reference architectures combine general architectural knowledge and experience with specific requirements for a coherent architectural solution for a particular problem domain. They document the structures of the system, the essential system building blocks, their responsibilities and their interaction.



Reference Architectures
IBM reference architecture examples

IBM offers a range of reference architectures for selected topics.

Data fabric

Unlock the value of all of your accessible data to gain valuable insights. Discover, govern, and secure your data.

Business automation

Automate your business processes to increase productivity and performance. Incorporate AI models to create new workflows that take advantage of your data.

Observability

Bring your operational information into one management platform to proactively predict problems and build automated responses to keep downtime to a minimum.

Security

Build the best possible defense to keep your enterprise safe and secure. Protect your enterprise from perpetrators and recover rapidly from security incidents.

Regulated workloads

Adopt a cloud platform that complies with industry regulations while delivering the benefits of cloud native applications and development.

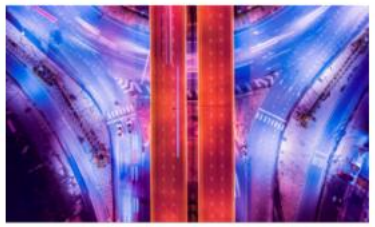
IBM Z

Integrate your IBM Z systems of record into your hybrid cloud transformation strategy to get the best of both worlds.

https://www.ibm.com/cloud/architecture/

Reference Architectures
IBM reference architecture examples

One of these reference architectures IBM calls **business automation**.



Workflow management

Automate your business workflows to increase agility, visibility, and consistency across your business processes.

Content services

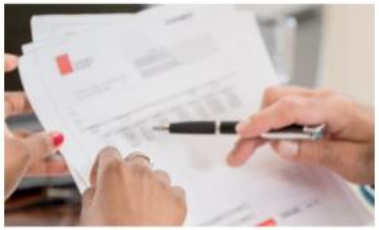
Share and govern unstructured content so that you can better engage customers, automate business processes, and enhance collaboration.

Decision management

Automate repeatable decisions and decisions that are based on policies and business rules to help an organization meet its goals.

Robotic process automation

Use robotic process automation (RPA) and digital workers to automate mundane tasks so that employees can focus on higher value work.



Document processing

Capture and extract data from documents that you can use to optimize your business processes.

Operational intelligence

Use operational intelligence to analyze data and gather insights that you can use to improve your business

Integration

Integrate cloud applications across hybrid and multicloud environments.

Event driven

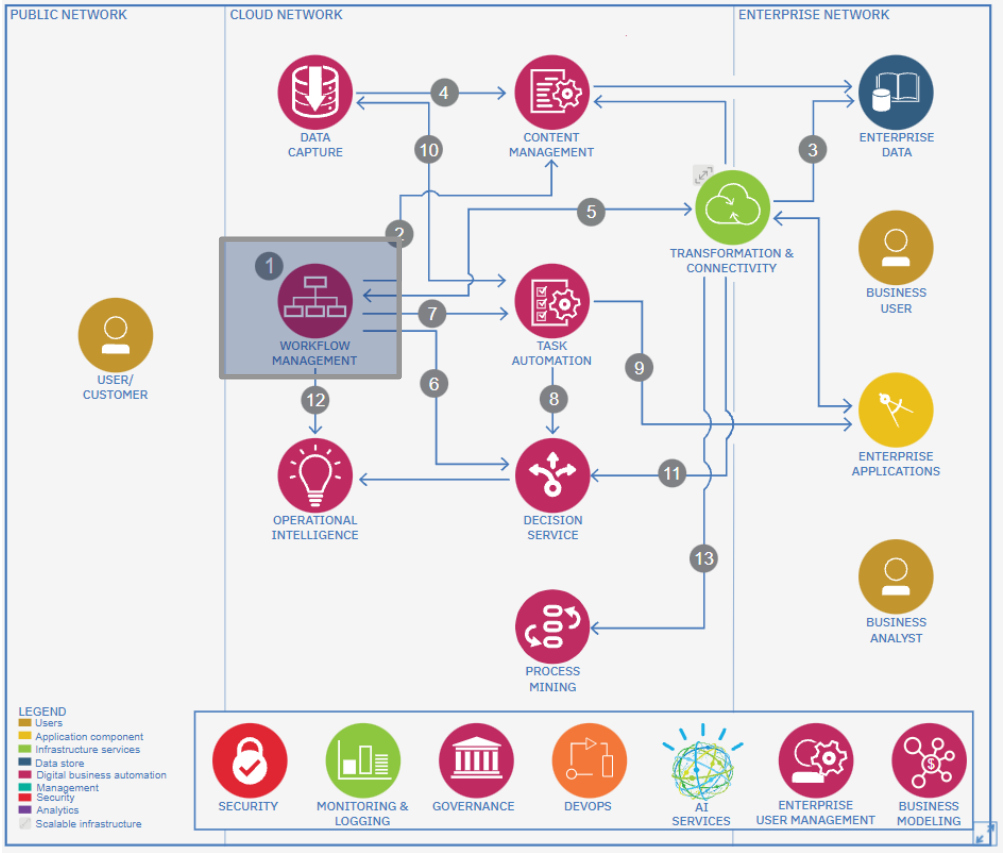
Develop and extend reactive applications by using CQRS and event sourcing.

https://www.ibm.com/cloud/architecture/technical-decision-points/business-automation

Reference Architectures

IBM reference architecture examples

The **business automation** reference architecture

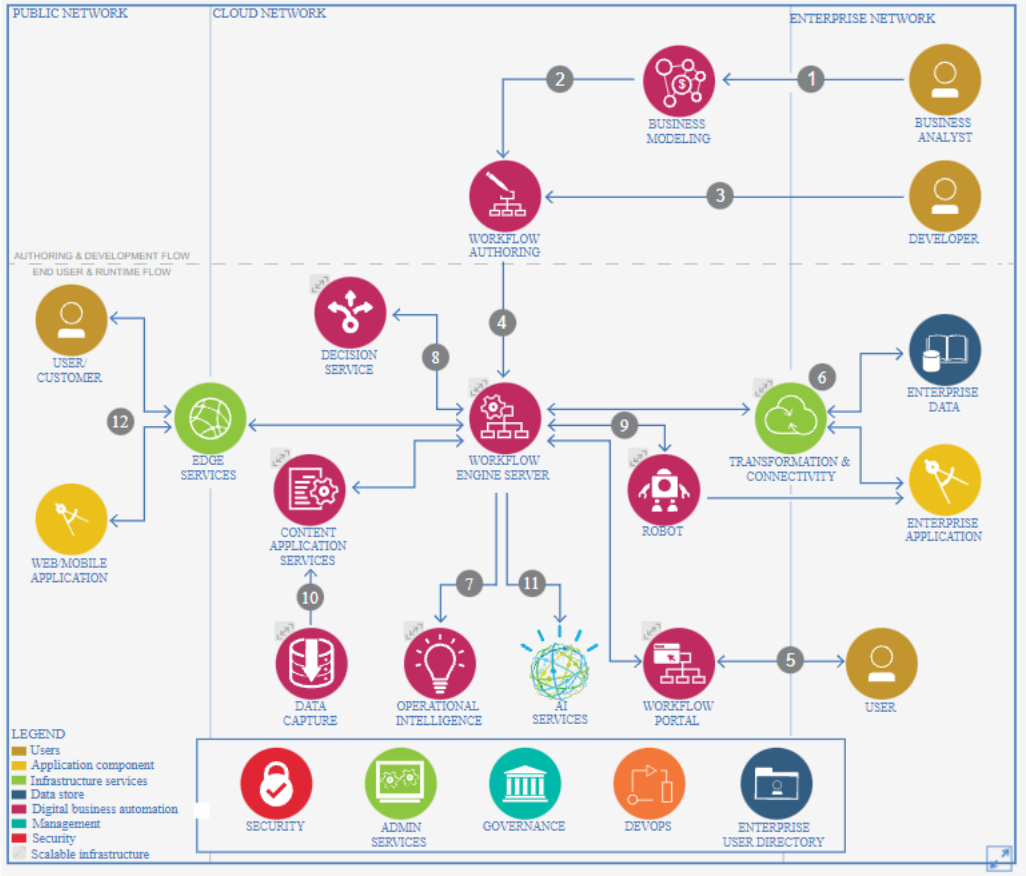


https://www.ibm.com/cloud/architecture/architectures/dba-architecture/reference-architecture

Reference Architectures

IBM reference architecture examples

Workflow management reference architecture



https://www.ibm.com/cloud/architecture/architectures/workflowDomain/reference-architecture

Bibliography
Lecture

[Vogel, Arnold et al 2011]
Vogel, Oliver; Arnold, Ingo; Chugtai, Arif; Kehrer, Timo, *Software Architecture: A Comprehensive Framework and Guide for Practitioners*, Springer Science and Business Media, Berlin Heidelberg, 2011

Questions