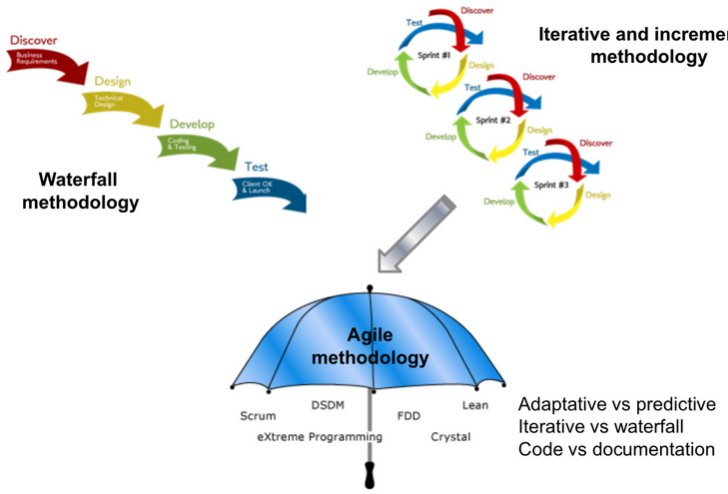
AS - Teoria 1

Unit 1. Introduction to Software Architecture and Design

Software Development Methodologies

Def. Software development methodologies are a specific collection of principles and/or practices applied to develop a software system.

Methodologies may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.



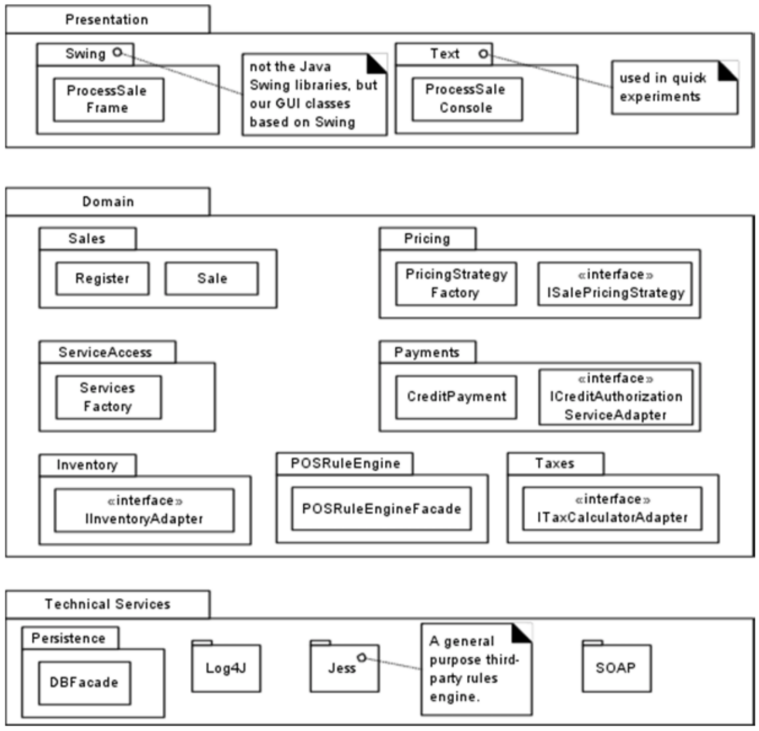
Software Architecture and Design

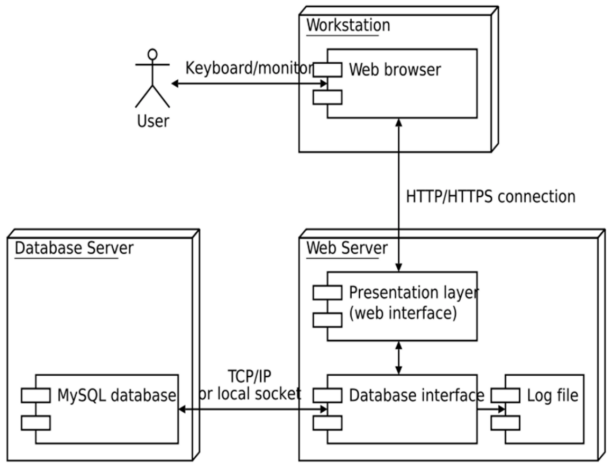
Def Design. Software *design* is the activity of applying different techniques and principles in order to define a system up to the level of detail needed to physically build it (i.e., implement it). One of the outputs of the software design is the software architecture.

Def Arch. Software *architecture* of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

Logical Architecture and Physical Architecture

Def logic. Logical architecture is the large-scale organization of the software components into packages, subsystems and layers that logically separate the functionality of a software system. It is called logical architecture because there is no decision about how these elements are deployed across different computational nodes.





Def phy. Physical architecture is the organization and distribution of the logical architecture across different computational nodes in a network.

Role of Design Patterns in Software Design

Def. Design patterns are general reusable solutions to commonly occurring problems within a given context in software design.

- Design patterns may be used in traditional and agile methodologies.
- Two types of patterns used at the design phase:
  - Architectural patterns
  - Design patterns

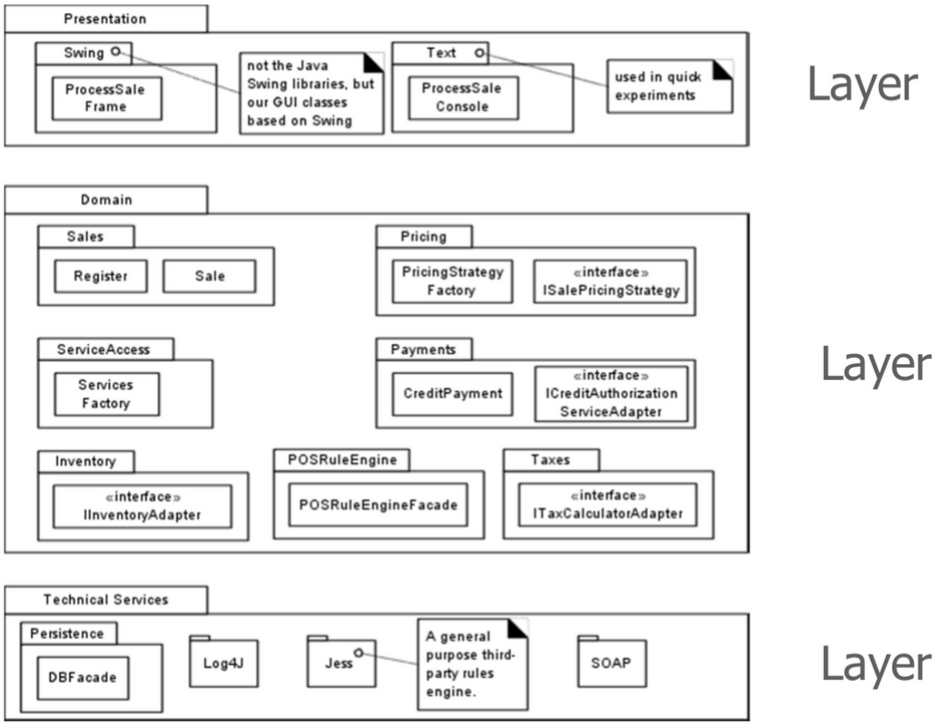
Architectural patterns

Def. An architectural pattern expresses a fundamental structural organization or schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.

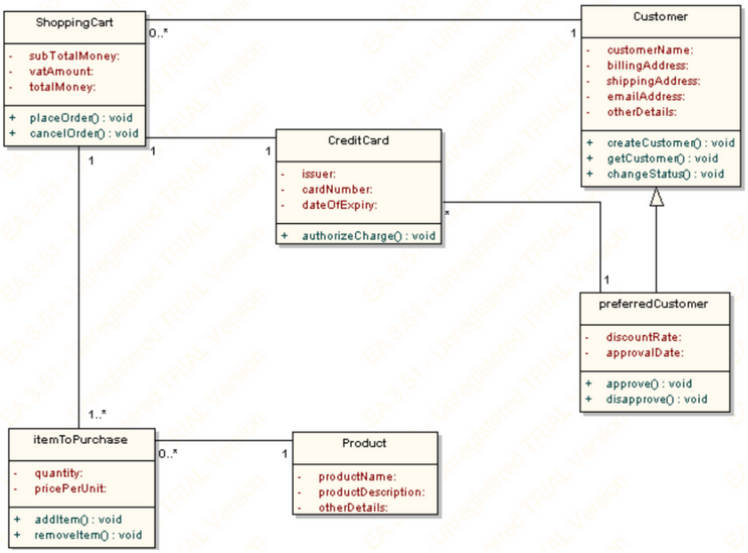
Architectural patterns describe different aspects of applications (deployment aspects, structure and design issues and others communication factors). A typical software system will use a combination of more than one architectural patterns.

Logic Types:

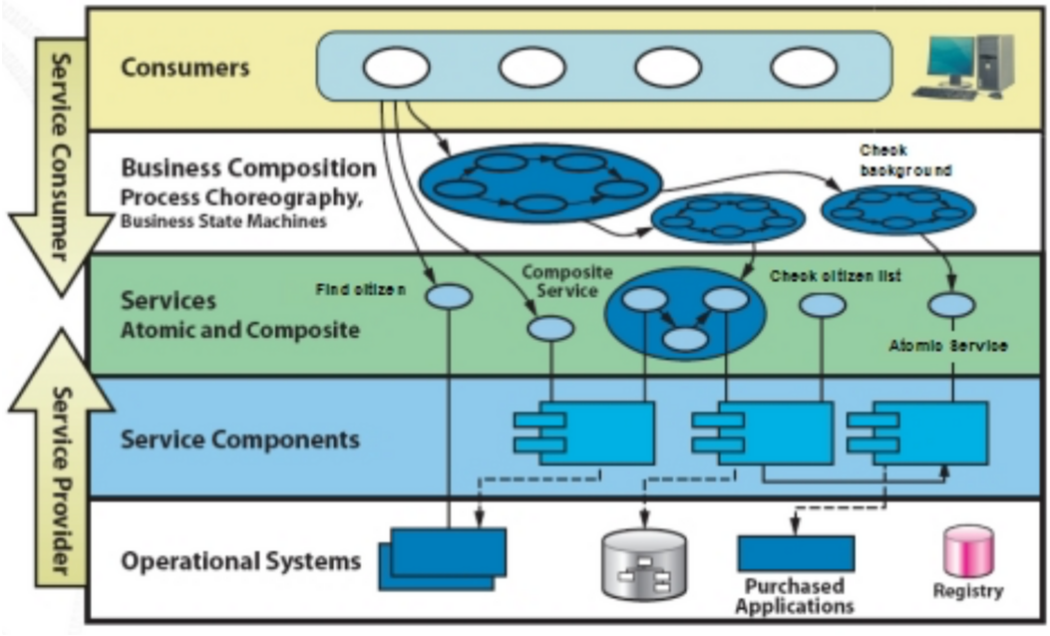
- **Layers**: Layered architecture partitions the concerns of the application into stacked groups (layers).



- **Object**: Object-Oriented divides the responsibilities of a system into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object.

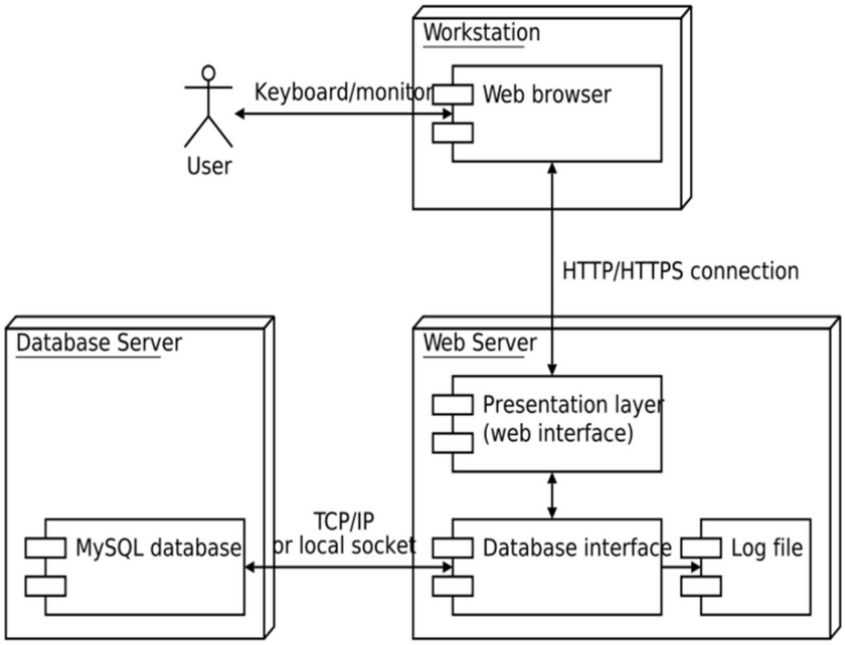


- **SOA**: Service-Oriented Architecture refers to systems that expose and consume functionality as a service using contracts and messages.

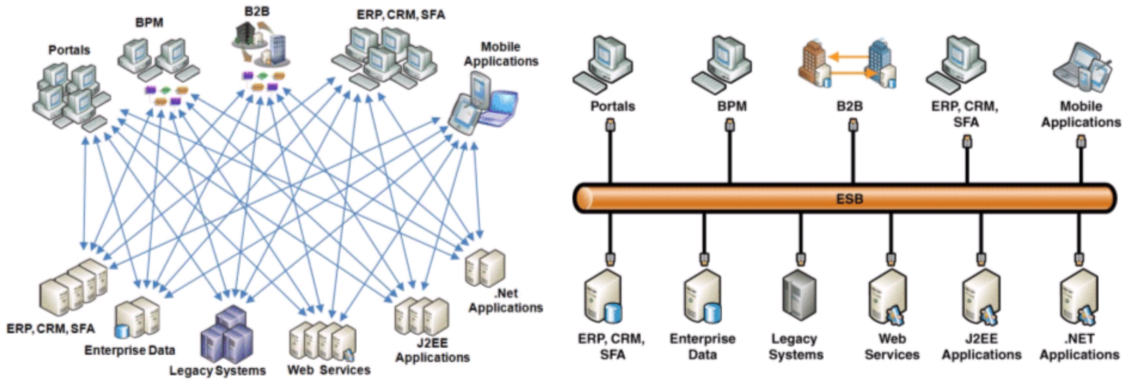


Physical types:

- **N-Tier/3-Tier** segregates functionality into separate segments in much the same way as the layered pattern, but with each segment being a tier located on a physically separate computer.



- **Message Bus** prescribes the use of a software system that can receive and send messages using one or more communication channels, so that applications can interact without needing to know specific details about each other.



Design patterns

Def. A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly recurring structure of communicating components that solves a general design problem within a particula context.

- Some design patterns: State, Expert, Controller, ...

Software Architecture and Design in Traditional and Agile Methodologies

Agile	**Waterfall**
Architecture is informal and incremental	Architecture is very well documented and completed before coding starts
Developers share ownership of code	Each developer is responsible for one area
Continuous integration	Integration performed at the end or after milestones
Focus on completing stories (functionality) in short iterations	Focus on completing modules (parts of the architecture) at different large milestones
Relies on engineering practices (TDD, refactoring, design patterns...)	Doesn't necessarily rely on engineering practices.
Light process and documentation	Heavy process and documentation
Requires cross-trained developers, knowledgeable in all required technologies	Relies on a small group of architects/designers to overview the complete code, the rest of the team can be very specialized.
Main roles: Developer	Main roles: Architect, Developer
Open door policy. Developers are encouraged to talk directly with business, QA and management at any time. Everyone's point of view is considered.	Only a few developers, and some architects can contact business people. Communication mainly only happens at the beginning of the project and at milestones.