Software architecture

• The design process for identifying the sub-systems making up a system and the framework for sub-system control and communication is architectural design.
• The output of this design process is a description of the software architecture.

Architectural design

• An early stage of the system design process.
• Represents the link between specification and design processes.
• Often carried out in parallel with some specification activities.
• It involves identifying major system components and their communications.
The architecture of a packing robot control system

Advantages of explicit architecture

- Stakeholder communication
  - Architecture may be used as a focus of discussion by system stakeholders.
- System analysis
  - Means that analysis of whether the system can meet its non-functional requirements is possible.
- Large-scale reuse
  - The architecture may be reusable across a range of systems
  - Product-line architectures may be developed.

Architectural design decisions

- Architectural design is a creative process so the process differs depending on the type of system being developed.
- However, a number of common decisions span all design processes and these decisions affect the non-functional characteristics of the system.

Architectural design decisions

- Is there a generic application architecture that can be used?
- How will the system be distributed?
- What architectural styles are appropriate?
- How will the system be decomposed into modules?
- What control strategy should be used?
- How will the architectural design be evaluated?
- How should the architecture be documented?
Architecture and system characteristics

- **Performance**
  - Localise critical operations and minimise communications. Use large rather than fine-grain components.

- **Security**
  - Use a layered architecture with critical assets in the inner layers.

- **Availability**
  - Include redundant components and mechanisms for fault tolerance.

- **Maintainability**
  - Use fine-grain, replaceable components.

Architectural views

- What views or perspectives are useful when designing and documenting a system’s architecture?
- What notations should be used for describing architectural models?
- Each architectural model only shows one view or perspective of the system.

4 + 1 view model of software architecture

- A logical view, which shows the key abstractions in the system as objects or object classes.
- A process view, which shows how, at run-time, the system is composed of interacting processes.
- A development view, which shows how the software is decomposed for development.
- A physical view, which shows the system hardware and how software components are distributed across the processors in the system.
- Related using use cases or scenarios (+1)
What is an Architectural Style?

• An architectural style, sometimes called an architectural pattern, is a set of principles—a coarse grained pattern that provides an abstract framework for a family of systems.
• The most important benefit is that they provide a common language.

Client/Server Architectural Style

• The major components of this model are:
  – A set of servers that offer services to other components.
  – A set of clients that call on the services offered by servers.
  – A network that allows the clients to access these services.

Client/Server Architectural Style

• Clients know servers.
• Servers need not to be aware of clients.
• Benefits:
  – Sharing resources.
  – Separation between clients and servers.
  – Security.
• Disadvantages:
  – Single point of failure.
  – DoS.
  – Server failure.
  – Performance prediction is difficult.
  – Server belongs to a different organization.

Layered Architecture Style

Layered architecture focuses on the grouping of related functionality within an application into distinct layers that are stacked vertically on top of each other.
Layered architecture

• Advantages
  – Allows replacement of entire layers so long as the interface is maintained
• Disadvantages
  – Performance
Pipe and Filter Architecture Style

- This is a model of the run-time organization of a system where functional transformations process their inputs and produce outputs.
- Data flows from one to another and is transformed as it moves through the sequence.

Object-Oriented Architectural Style

- An object-oriented design views a system as a series of cooperating objects, instead of a set of routines or procedural instructions.
- Objects are discrete, independent, and loosely coupled; they communicate through interfaces, by calling methods or accessing properties in other objects, and by sending and receiving messages.

Main benefits:
- **Understandable.** It maps the application more closely to the real world objects, making it more understandable.
- **Reusable.** It provides for reusability through polymorphism and abstraction.
- **Testable.** It provides for improved testability through encapsulation.
- **Extensible.** Encapsulation, polymorphism, and abstraction ensure that a change in the representation of data does not affect the interfaces that the object exposes, which would limit the capability to communicate and interact with other objects.
- **Highly Cohesive.** By locating only related methods and features in an object, and using different objects for different sets of features, you can achieve a high level of cohesion.
**Component-Based Architecture Style**

- Focuses on the decomposition of the design into individual functional or logical components that expose well-defined communication interfaces containing methods, events, and properties.
- Higher level of abstraction than object-oriented design principles.

**Main benefits**
- Ease of deployment
- Reduced cost
- Ease of development
- Reusable

**OMA**
- Application Objects
- Vertical CORBA Facilities
- Horizontal CORBA Facilities
- Object Request Broker
- Naming, Trading, Event, etc.
- CORBA Services
- User Interface, System management, Task management, etc.
- Banking, Healthcare, etc.

**CORBA**
- Object Implementation
- Dynamic Interface
- IDL Stub
- ORB Interface
- ORB Interface
- ORB
- Static IDL Skeleton
- Dynamic Skeleton
- Object Adaptor
- GIOP
Message Bus Architectural Style

- It is a style for designing applications where interaction between applications is accomplished by passing messages (usually asynchronously) over a common bus
- Examples
  - Enterprise Service Bus
  - Internet Service Bus
Message Bus Architectural Style

• Main benefits
  – Extensibility. Applications can be added to or removed from the bus without having an impact on the existing applications
  – Low complexity. Application complexity is reduced because each application only needs to know how to communicate with the bus
  – Flexibility. The set of applications that make up a complex process, or the communication patterns between applications, can be changed easily

Service-Oriented Architectural Style

• Service-oriented architecture (SOA) enables application functionality to be provided as a set of services, and the creation of applications that make use of software services.
• Services are loosely coupled because they use standards-based interfaces that can be invoked, published, and discovered
Service-Oriented Architectural Style

- Main benefits
  - **Domain alignment.** Reuse of common services with standard interfaces increases business and technology opportunities and reduces cost.
  - **Abstraction.** Services are autonomous and accessed through a formal contract, which provides loose coupling and abstraction.
  - **Discoverability.** Services can expose descriptions that allow other applications and services to locate them and automatically determine the interface.
  - **Interoperability.** Because the protocols and data formats are based on industry standards, the provider and consumer of the service can be built and deployed on different platforms.

Summary

- Software architecture and Architectural design
- Common Architectural Styles
  - Client/Server
  - Layered
  - Pipe and filter
  - Object-oriented
  - Component-based
  - Message-bus
  - SOA

References

- An Introduction to Software Architecture, David Garlan and Mary Shaw.
- Software Engineering, Ian Sommerville, 9th Edition