# Introduction to Software Architecture

目录

[Introduction to Software Architecture 1](#_Toc133953569)

[1.1. main concepts 2](#_Toc133953570)

[1.1.1. definition 2](#_Toc133953571)

[1.1.2. Example 2](#_Toc133953572)

[1.1.3. more details of Software Architecture 3](#_Toc133953573)

[1.1.4. structure and views 3](#_Toc133953574)

[1.1.5. Architectural Patterns 4](#_Toc133953575)

[1.1.6. make a good architecture 4](#_Toc133953576)

[1.2. Importance 5](#_Toc133953577)

[1.2.1. Inhibiting or Enabling System’s Quality Attributes 5](#_Toc133953578)

[1.2.2. Reasoning About and Managing Change 5](#_Toc133953579)

[1.2.3. Predicting System Qualities 5](#_Toc133953580)

[1.2.4. Enhancing Communication Among Stakeholders 5](#_Toc133953581)

[1.2.5. Earliest Design Decisions 5](#_Toc133953582)

[1.2.6. Defining Constraints on an Implementation 5](#_Toc133953583)

[1.2.7. Influencing the Organizational Structure 5](#_Toc133953584)

[1.2.8. Enabling Evolutionary Prototyping 5](#_Toc133953585)

[1.2.9. Improving Cost and Schedule Estimates 6](#_Toc133953586)

[1.2.10. Transferable, Reusable Model 6](#_Toc133953587)

[1.2.11. Using Independently Developed Components 6](#_Toc133953588)

[1.2.12. Restricting Design Vocabulary 6](#_Toc133953589)

[1.2.13. Basis for Training 6](#_Toc133953590)

[1.3. many contexts 6](#_Toc133953591)

[1.3.1. How is Architecture Influenced? 6](#_Toc133953592)

[1.3.2. Contexts of Software Architecture 6](#_Toc133953593)

[1.3.3. Stakeholders 7](#_Toc133953594)

[1.3.4. Architecture Influence Cycle 7](#_Toc133953595)

[1.3.5. What Do Architectures Influence? 8](#_Toc133953596)

[1.4. summary 8](#_Toc133953597)

## main concepts

### definition

#### Architecture

The process of planning, designing and constructing structures.

* Building
* Business
* Cognitive
* Computer
* Software
* …

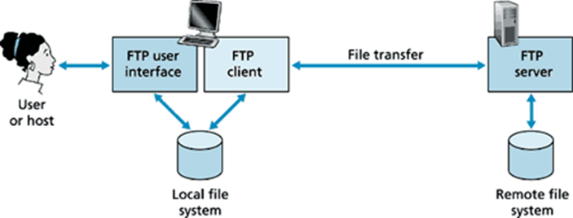
#### Software Architecture

definition from our textbook

The 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.

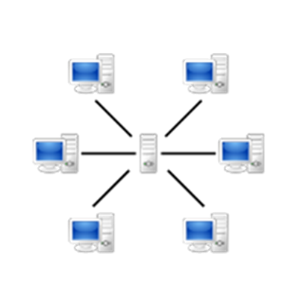
### Example

1. You work as a software developer and your client asked you to develop **a file sharing application** for them.
   * What kind of questions do you need to ask your client before you start your work? What information do you need to know?
2. Perhaps you ask your client what functions the software needs to have.
3. The client gave you these simple functional requirements.
4. The client gave you these simple functional requirements.
   * The application should enable sharing files with other **users all around the world**.
   * Users need to be able to **upload** the files for sharing.
   * Users need to be able to **download** the files they are interested in.
5. ***How will you design this software?***
6. A simple solution



* + You decide to use a client-server model.
  + There will be 2 main components.
    - Client - clients initiate the transfer.
    - Server – servers satisfy the request.
  + In your design, the server is centralized and there can be many clients requesting the files from the server.

1. You develop the software based on this design, deploy it, and receive feedback from the client that they are satisfied with the software.
2. After a while, another client asks you to develop a software for them. After initial consultation, you find out that the functional requirements are almost the same.
   * The application should enable sharing files with other users all around the world 🡪 *the same.*
   * Users need to be able to upload the files for sharing 🡪 *the same.*
   * Users need to be able to download the files they are interested in 🡪*the same.*
   * **Users need to be able to delete the uploaded files** 🡪 *new function.*
3. Since the functional requirements are the same, and the first client was happy with the software, you decide to use a client-server model for this project as well.
4. You also add additional functions for the users to meet the new functional requirements.
5. You develop the software based on this design, deploy it, and receive a feedback from the client that they are **NOT** satisfied with the software.
   * *What went wrong?*
6. After discussion with the client, you find out that the server crashes often and none of the users are able to download the file at this time.
7. You investigate further and find out that while the first client had less than a hundred of users of the software, the second client had thousands of users that made requests from the same server at the same time.



* + The server became overloaded and this resulted in crashes or slowing down of the connection.

1. So what went wrong?
   * Although the two clients had the same functional requirements, but their quality requirements were different.
     + The quality requirements were not discussed at the early stages of software design.
     + **The choice of software architecture was not guided by the quality requirements.**
2. How to avoid this problem?
   * Select the software architecture that will be more fit for the quality requirements in the second scenario.
   * For example, for the second file sharing system, to avoid the bottleneck caused by the central server in client-server model, you can choose **P2P architecture.**

电脑截图

描述已自动生成

1. **Conclusions**
   * Two systems with the same functionality can have different architectures.
     + Architectural decisions are NOT made based on functional requirements.
     + Architecture is driven by quality requirements.
   * Two systems with different functionality can have the same architectures
   * Example🡪 client-server model can be used in
     + File sharing systems
     + Online banking
     + Email
     + Network printing
     + Ecommerce
     + …

### more details of Software Architecture

#### Software Architecture = Structures of a software system

* Each structure compromises
  + Software elements
  + Relations among the elements
  + Properties of elements
  + Properties of relations

#### Software Architecture = the discipline of creating such structures and systems

Blueprint for the system and the developing project, laying out the task necessary to be executed by the design teams.

#### Architecture Is a Set of Software Structures

* A structure is a set of elements held together by a relation.
* Software systems are composed of many structures, and no single structure holds claim to being the architecture.
* There are three important categories of architectural structures.
  + Module
  + Component and Connector
  + Allocation

#### Architecture vs Design

* The architecture - the selection of architectural elements, their interaction and the restrictions on those elements and their interactions.
* The design - modularization and the detailed interfaces of the elements of the system, their algorithms and procedures, and the kinds of data needed to support the architecture and satisfy the requirements.

#### Architecture is an Abstraction

* An architecture specifically omits certain information about elements that is **not useful for reasoning about the system**.
* The architectural abstraction lets us **look at the system in terms of its elements, how they are arranged, how they interact, how they are composed**, and so forth.
* This abstraction is essential to taming the complexity of an architecture.

### structure and views

* A view is **a representation** of a coherent set of architectural elements, as written by and read by system stakeholders.
* A structure is **the set of elements itself**, as they exist in software or hardware.
* In short, a view is a representation of a structure.
  + For example, a module structure is the set of the system’s modules and their organization.
  + A module view is the representation of that structure, documented according to a template in a chosen notation, and used by some system stakeholders.
* Architects design structures. They document views of those structures.
* Example of views of a file sharing system

图示

描述已自动生成

### Architectural Patterns

#### Pattern

* a model or design used as a guide.
* an example for others to follow.

#### Architectural Patterns

* Architectural elements can be composed in ways that solve particular problems.
  + The compositions have been found useful over time, and over many different domains.
  + They have been documented and disseminated.
  + These compositions of architectural elements, called architectural patterns.
  + Patterns provide packaged strategies for solving some of the problems facing a system.
* An architectural pattern delineates the element types and their forms of interaction used in solving the problem.

#### Architectural patterns vs Design patterns

* Architectural patterns are **broader in scope** than Design patterns.
  + Architectural pattern focuses more on the abstract view of idea while Design pattern focuses on the implementation view of idea.
  + Design patterns provide very specific software related tasks whereas Architectural pattern are solutions for business problems
* One Architectural pattern can be implemented by using many design patterns.
  + There is one to many relationships between architectural pattern and design pattern.

### make a good architecture

#### Every System has a Software Architecture importance

* But the architecture may not be known to anyone.
  + Perhaps all of the people who designed the system are long gone
  + Perhaps the documentation has vanished (or was never produced)
  + Perhaps the source code has been lost (or was never delivered)
* An architecture can exist independently of its description or specification

#### What Makes a “Good” Architecture?

* Can we compare two architectures and say that one is better than the other?
* When can we say that the architecture is good?
* What can we do to produce a good architecture?
* There is no such thing as an inherently good or bad architecture.
* Architectures can be evaluated but only in the context of specific stated goals.
* Architectures are either more or less fit for some purpose.
* **There are, however, good rules of thumb.**
  + Process “Rules of Thumb”: how the work should be organized to produce a “good” architecture
  + Structural “Rules of Thumb”: how a “good” architecture should look like
* **Process “Rules of Thumb”**
  + The architecture should be the product of a single architect or a small group of architects with an identified technical leader.
  + The architect (or architecture team) should base the architecture on a prioritized list of well-specified quality attribute requirements.
  + The architecture should be documented using views.
  + The architecture should be evaluated for its ability to deliver the system’s important quality attributes.
  + The architecture should lend itself to incremental implementation.
* **Structural “Rules of Thumb”**
  + The architecture should feature well-defined modules.
  + The architecture should never depend on a particular version of a commercial product or tool.
  + Modules that produce data should be separate from modules that consume data.
  + This tends to increase modifiability.
  + Don’t expect a one-to-one correspondence between modules and components.
  + Every process should be written so that its assignment to a specific processor can be easily changed, perhaps even at runtime.
  + The architecture should feature a small number of ways for components to interact.
  + The system should do the same things in the same way throughout.

## Importance

### Inhibiting or Enabling System’s Quality Attributes

* Whether a system will be able to exhibit its desired (or required) quality attributes is substantially determined by its architecture.
  + Performance
  + Modifiability
  + Security
  + Scalability
  + Reusability
  + …

### Reasoning About and Managing Change

* About 80 percent of a typical software system’s total cost occurs after initial deployment
  + accommodate new features
  + adapt to new environments
  + fix bugs, and so forth
* Every architecture partitions possible changes into three categories
  + A *local* change can be accomplished by modifying a single element.
  + A *nonlocal* change requires multiple element modifications but leaves the underlying architectural approach intact.
  + An *architectural* change affects the fundamental ways in which the elements interact with each other and will require changes all over the system.
* Obviously, local changes are the most desirable.
* A good architecture is one in which the most common changes are local, and hence easy to make.

### Predicting System Qualities

* When we examine an architecture we can confidently predict that the architecture will exhibit the associated qualities.
* The earlier you can find a problem in your design, the cheaper, easier, and less disruptive it will be to fix.

### Enhancing Communication Among Stakeholders

* The architecture—or at least parts of it—is sufficiently abstract that most nontechnical people can understand it.
* Most of the system’s stakeholders can use as a basis for creating mutual understanding, negotiating, forming consensus, and communicating with each other.
* Each stakeholder of a software system is concerned with different characteristics of the system.
  + Users, client, manager, architect

### Earliest Design Decisions

* Software architecture is a manifestation of the earliest design decisions about a system.
* These early decisions affect the system’s remaining development, its deployment, and its maintenance life.
* What are these early design decisions?
  + Will the system run on one processor or be distributed across multiple processors?
  + Will the software be layered? If so, how many layers will there be? What will each one do?
  + Will components communicate synchronously or asynchronously?
  + What communication protocol will we choose?
  + Will the system depend on specific features of the operating system or hardware?
  + Will the information that flows through the system be encrypted or not?

### Defining Constraints on an Implementation

* An implementation exhibits an architecture if it conforms to the design decisions prescribed by the architecture.
  + The implementation must be implemented as the set of prescribed elements
  + These elements must interact with each other in the prescribed fashion
* Each of these prescriptions is a constraint on the implementer.

### Influencing the Organizational Structure

* Architecture prescribes the structure of the system being developed.
* The architecture is typically used as the basis for the work-breakdown structure.
* The work-breakdown structure in turn dictates
  + units of planning, scheduling, and budget
  + interteam communication channels
  + configuration and file-system organization
  + integration and test plans and procedures
  + the maintenance activity

### Enabling Evolutionary Prototyping

* Once an architecture has been defined, it can be prototyped as a skeletal system.
  + A skeletal system is one in which at least some of the infrastructure is built before much of the system’s functionality has been created.
* The fidelity of the system increases as prototype parts are replaced with complete versions of these parts.
* This approach aids the development process because the system is executable early in the product’s life cycle.
* This approach allows potential performance problems to be identified early in the product’s life cycle.
* These benefits reduce the potential risk in the project.

### Improving Cost and Schedule Estimates

* Architecture is used to help the project manager create cost and schedule estimates early in the project life cycle.
* Top-down estimates are useful for setting goals and apportioning budgets.
* Bottom-up understanding of the system’s pieces are typically more accurate than those that are based purely on top-down system knowledge.
* The best cost and schedule estimates will typically emerge from a consensus between the top-down estimates (created by the architect and project manager) and the bottom-up estimates (created by the developers).

### Transferable, Reusable Model

* Reuse of architectures provides tremendous benefits for systems with similar requirements.
  + Not only can code be reused, but so can the requirements that led to the architecture in the first place
  + When architectural decisions can be reused across multiple systems, all of the early-decision consequences are also transferred

### Using Independently Developed Components

* Architecture-based development often focuses on components that are likely to have been developed separately, even independently.
* Commercial off-the-shelf components, open-source software, publicly available apps, and networked services are example of interchangeable software components.
* The payoff can be
  + Decreased time to market
  + Increased reliability
  + Lower cost
  + Flexibility (if the component you want to buy is not terribly special purpose, it’s likely to be available from several sources)

### Restricting Design Vocabulary

* As useful architectural patterns are collected, we see the benefit in restricting ourselves to a relatively small number of choices of elements and their interactions.
  + We minimize the design complexity of the system we are building.
  + Enhanced reuse
  + More regular and simpler designs that are more easily understood and communicated.
  + More capable analysis
  + Shorter selection time
  + Greater interoperability

### Basis for Training

* The architecture can serve as the first introduction to the system for new project members.
* Module views show someone the structure of a project.
  + Who does what, which teams are assigned to which parts of the system, and so forth.
* Component-and-connector explains how the system is expected to work and accomplish its job.

## many contexts

### How is Architecture Influenced?

archinf.tiff

### Contexts of Software Architecture

We put software architecture in its place relative to four contexts:

* Technical. What technical role does the software architecture play in the system?
* Project life cycle. How does a software architecture relate to the other phases of a software development life cycle?
* Business. How does the presence of a software architecture affect an organization’s business environment?
* Professional. What is the role of a software architect in an organization or a development project?

#### Technical Context

* The most important technical context factor is the set of quality attributes that the architecture can help to achieve.
* The architecture’s current technical environment is also an important factor.
  + Standard industry practices.
  + Software engineering techniques prevalent in the architect’s professional community.

#### Project Life-cycle Context

* Software development processes are standard approaches for developing software systems.
* They tell the members of the team what to do next.
* There are four dominant software development processes:
  + Waterfall
  + Iterative
  + Agile
  + Model-driven development
* Architecture Activities
  + Architecture is a special kind of design, so architecture finds a home in each process of software development.
  + There are activities involved in creating a software architecture, using it to realize a complete design, and then implementing.
    - Understanding the architecturally significant requirements
    - Creating or selecting the architecture
    - Documenting and communicating the architecture
    - Analyzing or evaluating the architecture
    - Implementing and testing the system based on the architecture
    - Ensuring that the implementation conforms to the architecture

#### **Business** Context

* Systems are created to satisfy the business goals of one or more organizations.
  + **Development organizations**: e.g., make a profit, or capture market, or help their customers do their jobs better, or keep their staff employed, or make their stockholders happy.
  + **Customers** have their own goals: e.g., make their lives easier or more productive.
  + **Other organizations**, such as subcontractors or government regulatory agencies, have their own goals.
* Architects need to understand the goals.
* Many of these goals will influence the architecture.
* Architecture and business goals

BG.tiff

#### Professional Context

* Software architect is a career choice.
* Architects need more than just technical skills.
  + Architects need diplomatic, negotiation, and communication skills.
  + Architects need the ability to communicate ideas clearly.
* Architects need up-to-date knowledge.
  + Know about (for example) patterns, or database platforms, or web services standards.
  + Know about business considerations.

### Stakeholders

* A stakeholder is anyone who has a stake in the success of the system.
* Stakeholders typically have different specific concerns on the system.
* Early engagement of stakeholders to understand the constraints of the task, manage expectations, negotiate priorities, and make tradeoffs.

SH.tiff

### Architecture Influence Cycle

* Contexts the architecture is influenced by
  + Business
  + Technical
  + Project
  + Professional
* Contexts the architecture influences
  + Business
  + Technical
  + Project
  + Professional

### What Do Architectures Influence?

AIC.tiff

* Influence Technical context
* The architecture can affect stakeholder’s requirements for the next system.
* A customer may relax some of their requirements.
* Shrinkwrapped software has affected people’s requirements, as it is inexpensive and of high quality.
* Influence Project context
  + The architecture affects the structure of the developing organization.
  + An architecture prescribes the units of software to be implemented and integrated to the system.
  + These units are the basis for the development project’s structure.
  + the development, test, and integration activities all revolve around the units.
* Influence Business context
  + The architecture can affect the business goals of the developing organization.
  + The architecture can provide opportunities for the efficient production and deployment of similar systems, and the organization may adjust its goals to take advantage of its newfound expertise.
* Influence Professional context
  + The process of system building will affect the architect’s experience.
  + A system that was successfully built will make the architect more inclined to build systems using the same approach in the future.
  + Architectures that fail are less likely to be chosen for future projects.

## summary

* ***The 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.
* An architecture has an impact on the architect, the organization, and, potentially, the industry.