Object oriented analysis and modeling
Software architecture

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

For more details I recommend

Microsoft Application Architecture Guide

Basic questions

- How will the users be using the application?
- How will the application be deployed into production and managed?
- What are the quality attribute requirements for the application, such as security, performance, concurrency, internationalization, and configuration?
- How can the application be designed to be flexible and maintainable over time?
- What are the architectural trends that might impact your application now or after it has been deployed?

Software architecture

Architecture should:

- Expose the structure of the system but hide the implementation details.
- Realize all of the use cases and scenarios.
- Try to address the requirements of various stakeholders.
- Handle both functional and quality requirements.

The Architectural Landscape

Consider the environment of the product.

- User empowerment
  Allow the user to define how they interact with your application, but keep the key scenarios as simple as possible.
- Market maturity.
  Use existing platform and technology options.
- Flexible design.
- Future trends.

More questions to consider

- What are the foundational parts of the architecture that represent the greatest risk if you get them wrong?
- What are the parts of the architecture that are most likely to change, or whose design you can delay until later with little impact?
- What are your key assumptions, and how will you test them?
- What conditions may require you to refactor the design?

Software architecture principles

- Build to change instead of building to last.
- Model to analyze and reduce risk.
- Use models and visualizations as a communication and collaboration tool.
- Identify key engineering decisions.

Software architecture principles

- Do not try to get it all right the first time
- Design just as much as you can in order to start testing the design.
- Iteratively add details to the design over multiple passes to make sure that you get the big decisions right first, and then focus on the details.

Testing the architecture

- What assumptions have I made in this architecture?
- What explicit or implied requirements is this architecture meeting?
- What are the key risks with this architectural approach?
- What countermeasures are in place to mitigate key risks?
- In what ways is this architecture an improvement over the baseline or the last candidate architecture?

How to find the right architecture

- Determine the Application Type
- Determine the Deployment Strategy
- Determine the Appropriate Technologies
- Determine the Quality Attributes
- Determine the Crosscutting Concerns (authentication, logging, chaching, . . .)

You cannot focus on everything.

Key scenario.

- It represents an issue—a significant unknown area or an area of significant risk.
- It refers to an architecturally significant use case.
    - Business Critical.
    - High Impact.
- It represents the intersection of quality attributes with functionality.
- It represents a tradeoff between quality attributes.

How to find the right architecture

Break stuff into layers and components

- Keep in mind ordinary design principles
    - separation of concerns, SRP, Principle of least knowledge, DRY, Composition over inheritance ...
- Do not mix different stuff into layers (design patterns, component types, data types).
- Be explicit about how layers communicate with each other, use abstraction.
- Define a clear contract for components.

Architectural styles

- Provide abstract framework for a family of systems
- Help communication
- We can combine styles

Key architectural styles

- Client/Server
  Segregates the system into two applications, where the client makes requests to the server. In many cases, the server is a database with application logic represented as stored procedures.
- Component-Based Architecture
  Decomposes application design into reusable functional or logical components that expose well-defined communication interfaces.
- Domain Driven Design
  An object-oriented architectural style focused on modeling a business domain and defining business objects based on entities within the business domain.

Key architectural styles

- Layered Architecture
  Partitions the concerns of the application into stacked groups (layers).
- Message Bus
  An architecture style that prescribes 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.
- N-Tier / 3-Tier
  Segregates functionality into separate segments in much the same way as the layered style, but with each segment being a tier located on a physically separate computer.

Key architectural styles

- Object-Oriented
  A design paradigm based on division of responsibilities for an application or system into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object.
- Service-Oriented Architecture (SOA)
  Refers to applications that expose and consume functionality as a service using contracts and messages.

Key architectural styles

Benefits of respective styles

Representing and Communicating Your Architecture Design

Cover various views:

- Use case view
- Static view
- Dynamic view
- Physical view
- Development view

Visualizing architecture

- To capture architecture an high level view of the system - UML Model diagram
- To capture physical architecture - UML Deployment diagram
- To capture components and interfaces - UML component diagrams
- High level development view - UML package diagram