Introduction to Architecture

Content

- What is architecture?
- Motivation for architecture
- Non-functional requirements

What is architecture?

- The system's skeleton.
- What is left when you have removed everything possible and still are able to tell how the system works.
- Tells *which* problems are solved and *where* they are solved.

What is architecture? (cont)

- Shall guarantee that all requirements, both functional and non-functional, can be solved.
- Non-functional requirements are a very important part of the architecture.
- Involves choosing what hardware and software to use, which means that the architecture is very platform dependant.

What is architecture? (cont)

- What is a good architecture depends very much on the purpose of the application.
  - Architecture is not as generic as design.

What is architecture? (cont)

- This course presents good architectures for distributed applications consisting of view, business logic and data.
  - Focus on web based view.
  - Assumes Java EE is used.
  - The course does not give you knowledge to chose between different architectures for other kinds of applications.

Why Architecture?

- Organization
  - Reflect, decide, document
  - Do not make decisions by coincidence
- Conflicting requirements from users, customers, system administrators and developers.
- Balance functionality, cost, quality and time.

Benefits of a Good Architecture

- Understanding of the system's structure and its limitations.
- Common vocabulary
- Reuse
  - Code: objects and components
  - Code style: code conventions, patterns, documentation etc

Benefits of a Good Architecture (cont)

- Better organization of the project.
  - Divide the resources after components, subsystems, functionality etc
- Makes it easier to divide the application and maintain low coupling and high cohesion.

Non-Functional Requirements

- All requirements that do not concern *what* the program should do, but *how* it should work.
  - performance, reliability, maintainability, security etc
- Very important to specify before development starts.
- Very important to implement from the start.
  - *DO NOT try to add them when the program works*.

Non-Functional Requirements (cont)

- Be realistic, do not write a wish list.
- It must be possible to verify that they are fulfilled.
  - Do not write *fast enough*, but rather *first visible sign of response within 1 second in 99% of the calls measured from the outer firewall*.

Non-Functional Requirements (cont)

- Availability
  - The application can be reached often enough by as many users as required, for example *available 98% of the time* and/or *unavailable at most four minutes per week*.
- Reliability
  - To what extent the application shall behave as expected, for example *at most 1 of 1000 sessions fails*.

Non-Functional Requirements (cont)

- Response time
  - for example *first visible sign of response within 1 second in 99% of the calls measured from the outer firewall.*
- Capacity
  - Measures how much load can be handled concurrently, for example *10 concurrent transactions while maintaining other non-functional requirements.*

Non-Functional Requirements (cont)

- Scalability
  - To what extent the application can maintain its performance with higher load if additional hardware is added. Vertical scaling means to add hardware (cpu, memory) to the servers. Horizontal scaling means to add more servers. For example *Maintain all non-functional requirements at 90% load increase if the number of servers are doubled.*

Non-Functional Requirements (cont)

- Manageability
  - How complex the system administrator's work may be, for example *at most two work hours a month may be spent to install upgrades*.
- Configurability
  - For example which changes can be made to the system without having to recompile anything.

Non-Functional Requirements (cont)

- Packaging
  - In what format should the product be delivered.
- Standards
  - What standards must the application follow
- Requirement to use certain software and/or hardware.

Non-Functional Requirements (cont)

- Usability
  - *The screen should be visible at a distance of one meter* or *eight out of ten persons chosen by the customer should consider the application easy to use*.
- Security
  - There are lots of different security aspects. Criteria could be *30 minutes using the best known techniques to change the balance of an account* or *three failed log in attempts in a row should be logged.*

Non-Functional Requirements (cont)

- Security Requirements:
  - authentication, to prove ones identity.
  - authorization, which rights do different users have?
  - non-repudiation, users can not deny what they have done.
  - integrity, data can not be modified in any unallowed manner.
  - privacy, data can not be read in any unallowed manner.
  - logging, what should be logged and where?

Non-Functional Requirements (cont)

- Design goals
  - Non-functional requirements that are not really observable to the users, but rather to the developers.
  - Affects the possibility (and cost) to maintain functional and non-functional requirements during the application's life time.
  - Hard to measure.
  - Can be evaluated with code reviews or by performing the task that should be made easy (for example to add or change functionality).

Non-Functional Requirements (cont)

- Some design goals:
  - *Adaptability* or *extensibility*, how easy it is to extend or change the application.
  - *Portability*, how much work is required to adapt the application to other hardware, operating system, user interface etc.
  - *Interoperability*, how easy it is for other applications to communicate with this application.

Non-Functional Requirements (cont)

- *Testability*, to what extent the system facilitates testing, for example by giving informative error messages or being able to unit test.
- *Reusability*, both to use existing components in this application and to use newly developed components in other applications.