Проектиране на софтуерни системи. Дизайн и архитектура

Software architecture

- To create a reliable, secure and efficient product, you need to pay attention to architectural design which includes:
  - its overall organization,
  - how the software is decomposed into components,
  - the server organization
  - the technologies that you use to build the software. The architecture of a software product affects its performance, usability, security, reliability and maintainability.
- There are many different interpretations of the term 'software architecture'.
  - Some focus on 'architecture' as a noun - the structure of a system and others consider 'architecture' to be a verb - the process of defining these structures.

The IEEE definition of software architecture

Architecture is the fundamental organization of a software system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution.

Software architecture and components

- A component is an element that implements a coherent set of functionality or features.
- Software component can be considered as a collection of one or more services that may be used by other components.
- When designing software architecture, you don't have to decide how an architectural element or component is to be implemented.
- Rather, you design the component interface and leave the implementation of that interface to a later stage of the development process.

Why is architecture important?

- Architecture is important because the architecture of a system has a fundamental influence on the non-functional system properties
- Architectural design involves understanding the issues that affect the architecture of your product and creating an architectural description that shows the critical components and their relationships.
- Minimizing complexity should be an important goal for architectural designers.
  - The more complex a system, the more difficult and expensive it is to understand and change.
  - Programmers are more likely to make mistakes and introduce bugs and security vulnerabilities when they are modifying or extending a complex system..

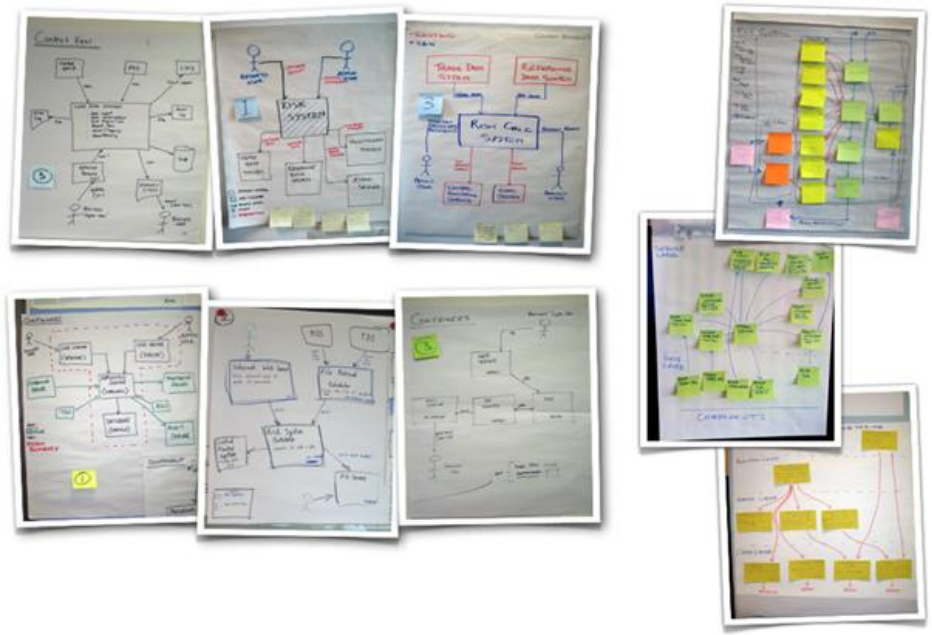
Software Architecture

- Architecture is important because the architecture of a system *has a fundamental influence on the non-functional system properties.*
- *… eases the communication among stakeholders.*
- *… defines constraints on implementation.*
- *… makes it easier to reason about and manage change.*
- *… enables more accurate cost and schedule estimates.*
- **…**

Perspectives & Views

- Due to the high complexity of software systems, there are different *perspectives* defined in software architecture.

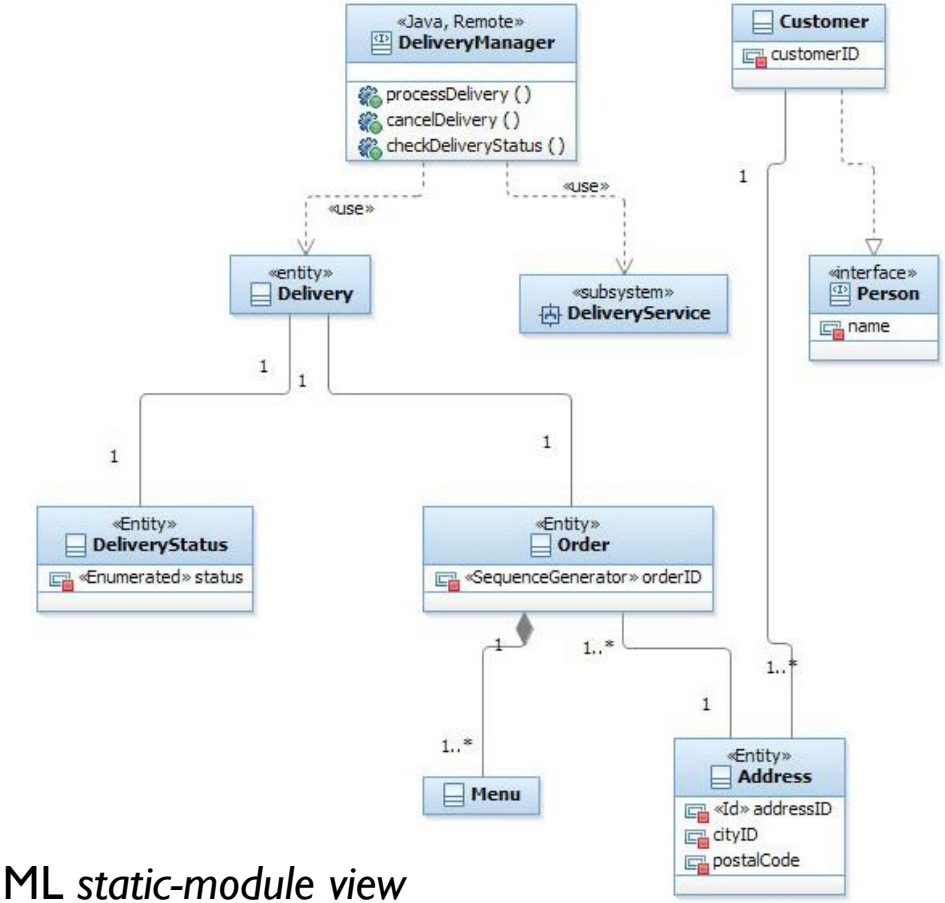
- The most important are the *static*, the *dynamic* and the *allocation* perspective, each one with a number of *views*.



Perspectives & Views

- *How is it structured as a set of code units module views ?*

- **Module Views - Static**

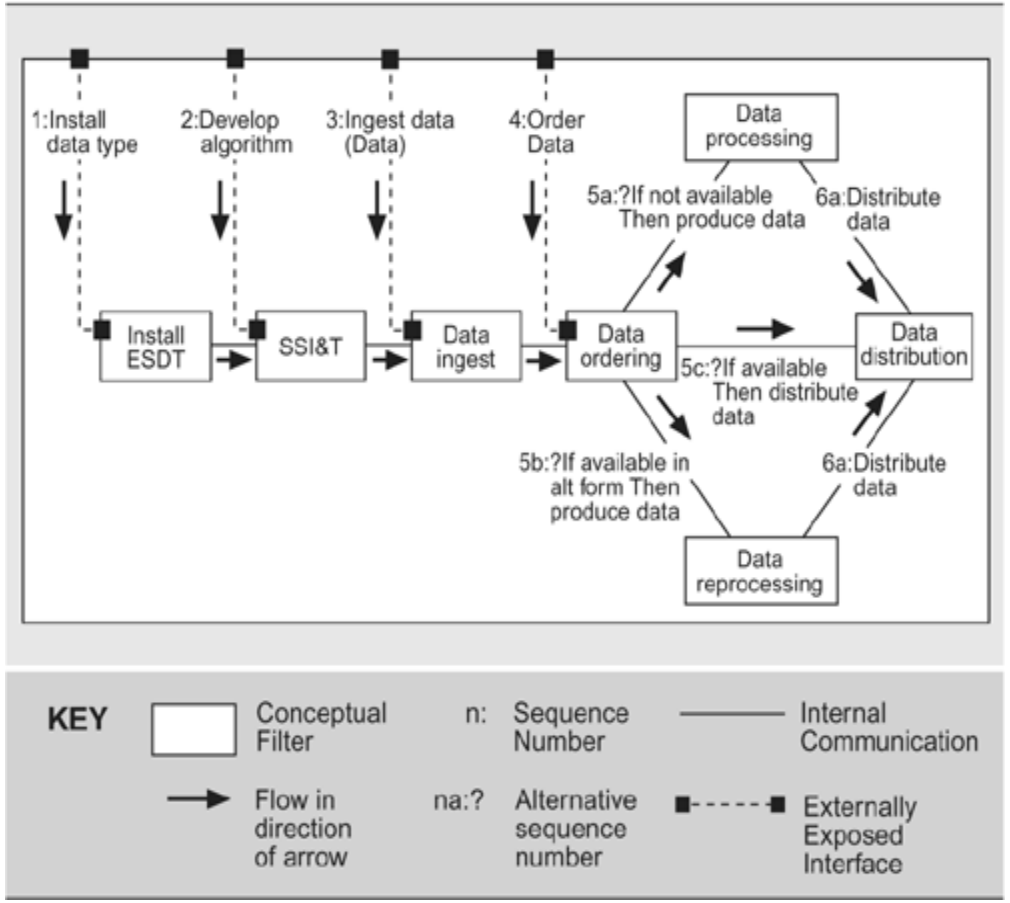


UML *static-module view*

Perspectives & Views

- *How is it structured as a set of elements that have run-time behavior and interactions between them ?*

- **Component & Connector Views**
  Dynamic

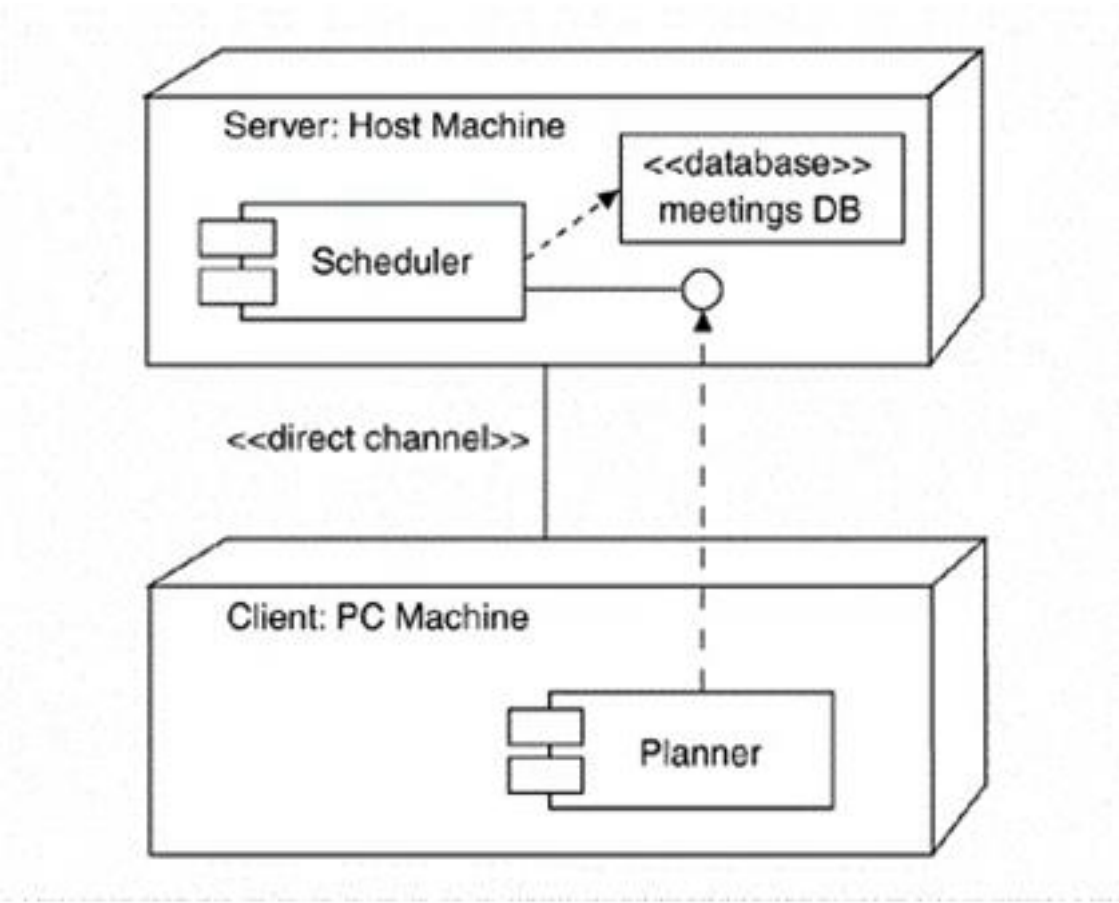


Dynamic perspective C&C view

Perspectives & Views

- *How does it relate to non-software structures in its environment ?*

- **Allocation Views - Deployment**



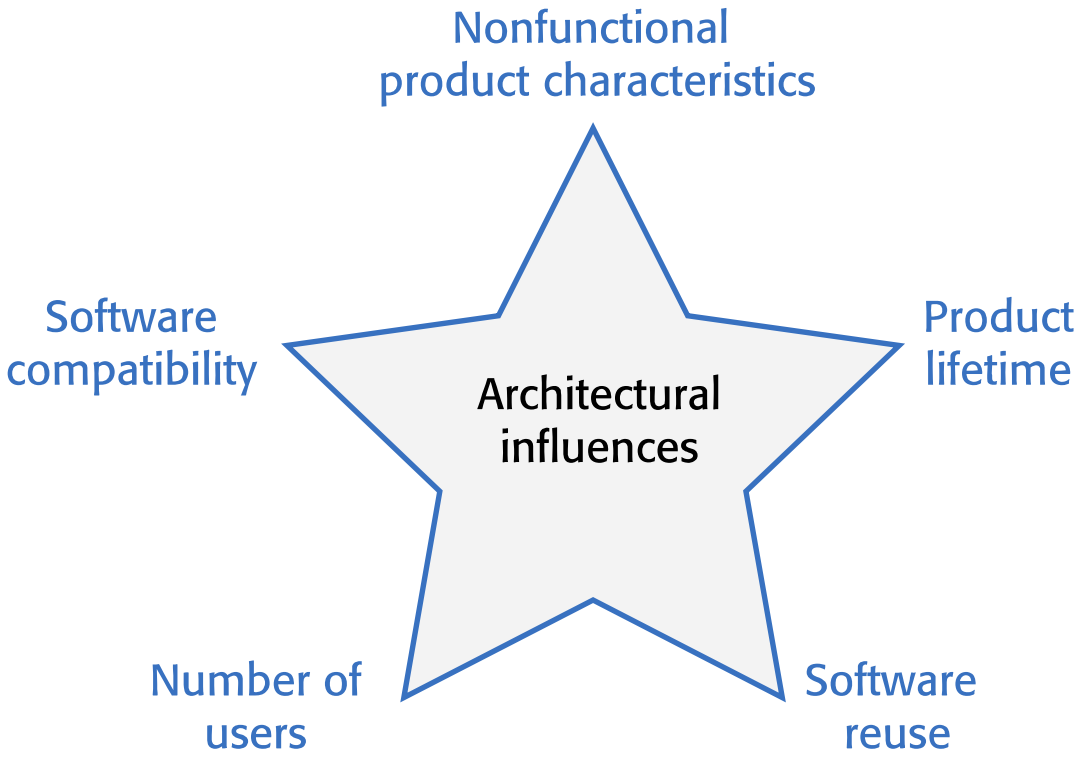
Non-functional system quality attributes

- *Responsiveness*
  Does the system return results to users in a reasonable time?
- *Reliability*
  Do the system features behave as expected by both developers and users?
- *Availability*
  Can the system deliver its services when requested by users?
- *Security*
  Does the system protect itself and users' data from unauthorized attacks and intrusions?
- *Usability*
  Can system users access the features that they need and use them quickly and without errors?
- *Maintainability*
  Can the system be readily updated and new features added without undue costs?
- *Resilience*
  Can the system continue to deliver user services in the event of partial failure or external attack?

Architecture Drivers

- Architectural drivers are the design forces that will influence the early design decisions the architects make.
- Architectural drivers are not all of the requirements for a system, but they are an early attempt to identify and capture those requirements, that are most influential.
- Architectural drivers can be one of the following :
  - ***Constraints***
    - ***Technical Constraints***
    - ***Business Constraints***
  - ***Quality Attributes***
  - ***High-Level Functional Requirements***

Issues that influence architectural decisions



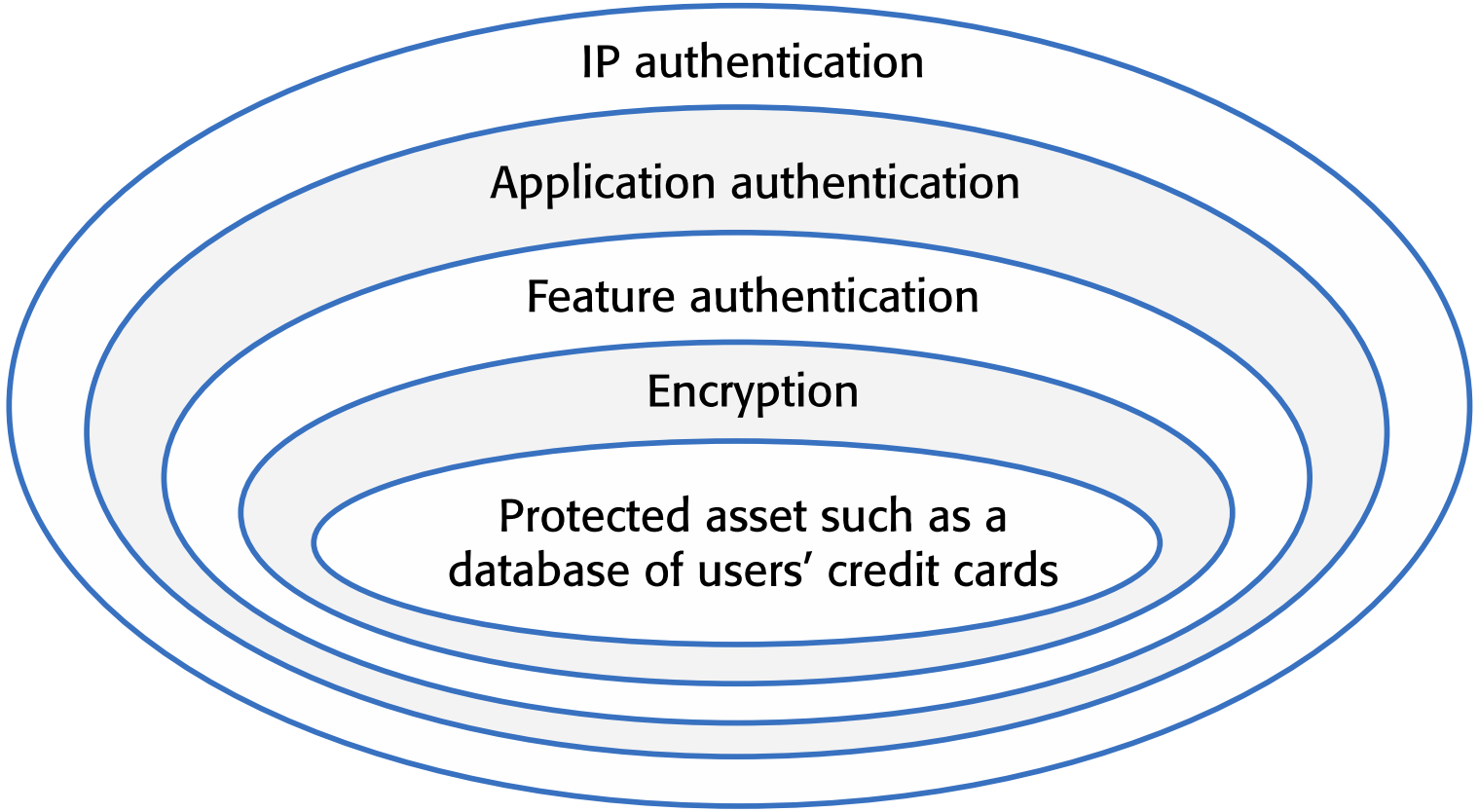
Trade off: Maintainability vs performance

- System maintainability is an attribute that reflects how difficult and expensive it is to make changes to a system after it has been released to customers.
  - You improve maintainability by building a system from small self-contained parts, each of which can be replaced or enhanced if changes are required.
- In architectural terms, this means that the system should be decomposed into fine-grain components, each of which does one thing and one thing only.
  - However, it takes time for components to communicate with each other. Consequently, if many components are involved in implementing a product feature, the software will be slower.

Trade off: Security vs usability

- You can achieve security by designing the system protection as a series of layers An attacker has to penetrate all of those layers before the system is compromised.
- Layers might include system authentication layers, a separate critical feature authentication layer, an encryption layer and so on.
- Architecturally, you can implement each of these layers as separate components so that if one of these components is compromised by an attacker, then the other layers remain intact.

Authentication layers



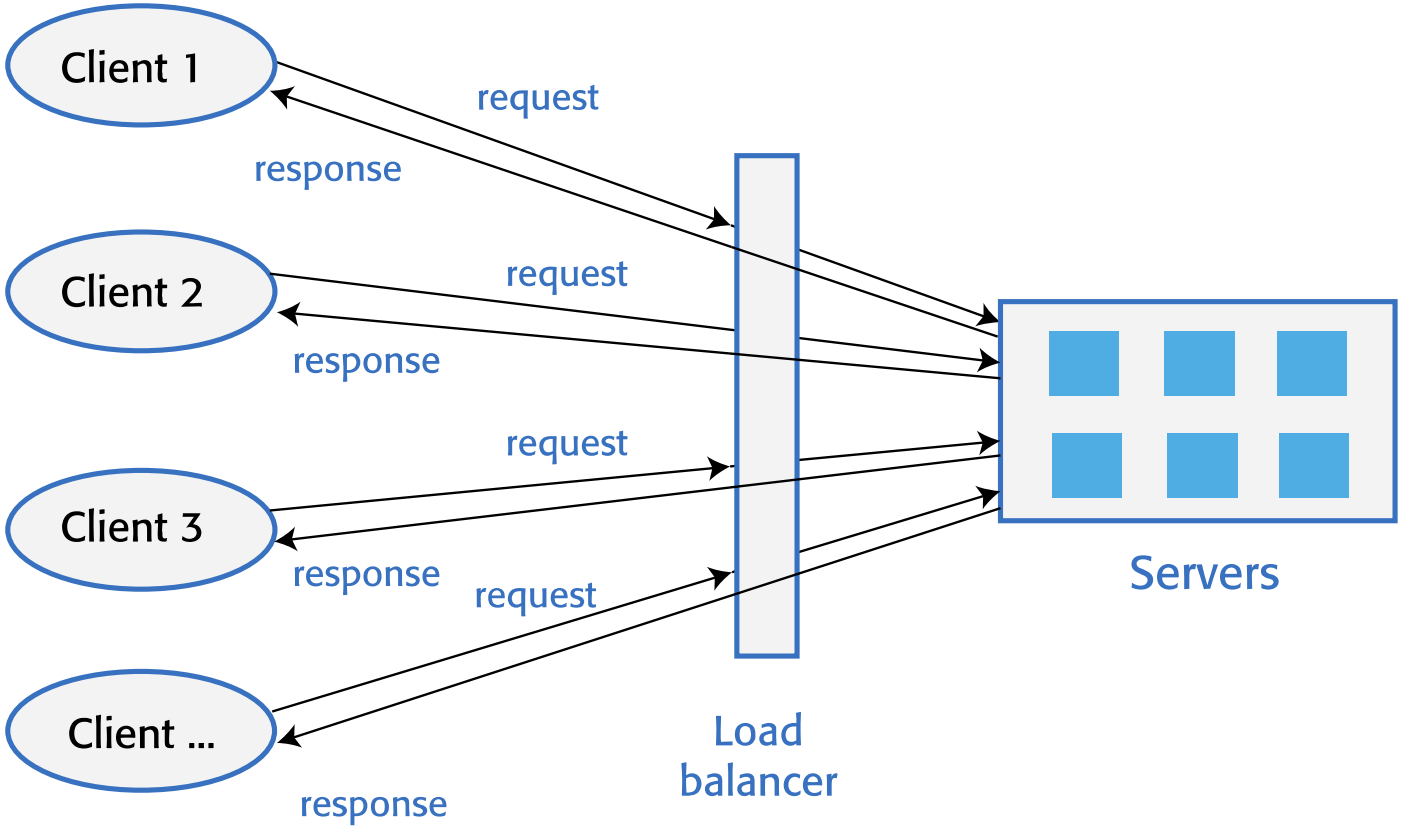
Usability issues

- A layered approach to security affects the usability of the software.
  - Users have to remember information, like passwords, that is needed to penetrate a security layer. Their interaction with the system is inevitably slowed down by its security features.
  - Many users find this irritating and often look for work-arounds so that they do not have to re-authenticate to access system features or data.
- To avoid this, you need an architecture:
  - that doesn't have too many security layers,
  - that doesn't enforce unnecessary security,
  - that provides helper components that reduce the load on users.

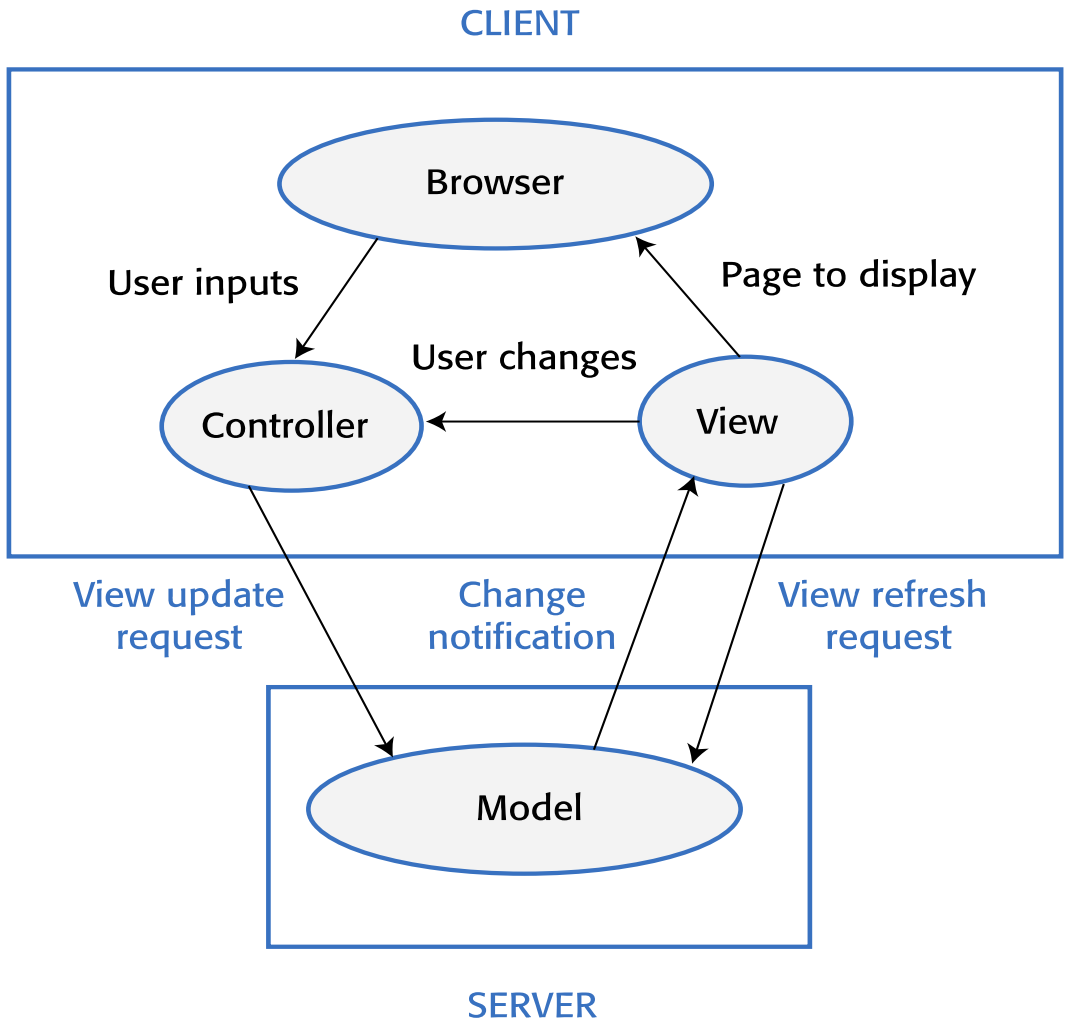
Trade off: Availability vs time-to-market

- Availability is particularly important in enterprise products, such as products for the finance industry, where 24/7 operation is expected.
- The availability of a system is a measure of the amount of 'uptime' of that system.
  - Availability is normally expressed as a percentage of the time that a system is available to deliver user services.
- Architecturally, you achieve availability by having redundant components in a system.
  - To make use of redundancy, you include sensor components that detect failure, and switching components that switch operation to a redundant component when a failure is detected.
- Implementing extra components takes time and increases the cost of system development. It adds complexity to the system and therefore increases the chances of introducing bugs and vulnerabilities.

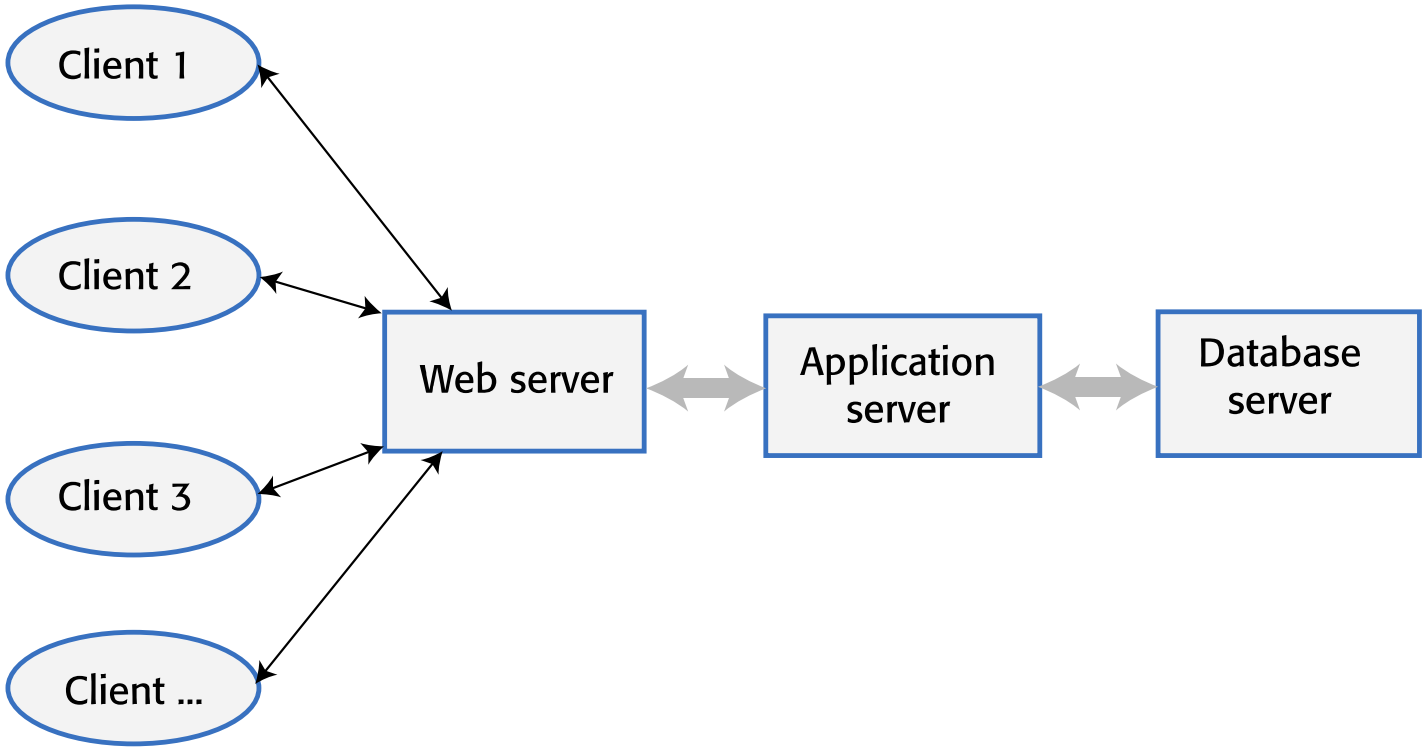
Client-server architecture



The model-view-controller pattern



Multi-tier client-server architecture



Key points

- Software architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
- The architecture of a software system has a significant influence on non-functional system properties such as reliability, efficiency and security.
- Architectural design involves understanding the issues that are critical for your product and creating system descriptions that shows components and their relationships.
- The principal role of architectural descriptions is to provide a basis for the development team to discuss the system organization. Informal architectural diagrams are effective in architectural description because they are fast and easy to draw and share.
- System decomposition involves analyzing architectural components and representing them as a set of finer-grain components.

Key points

- To minimize complexity, you should separate concerns, avoid functional duplication and focus on component interfaces.
- Web-based systems often have a common layered structure including user interface layers, application-specific layers and a database layer.
- The distribution architecture in a system defines the organization of the servers in that system and the allocation of components to these servers.
- Multi-tier client-server and service-oriented architectures are the most commonly used architectures for web-based systems.
- Making decisions on technologies such as database and cloud technologies are an important part of the architectural design process.

Процес на проектиране на софтуерната архитектура

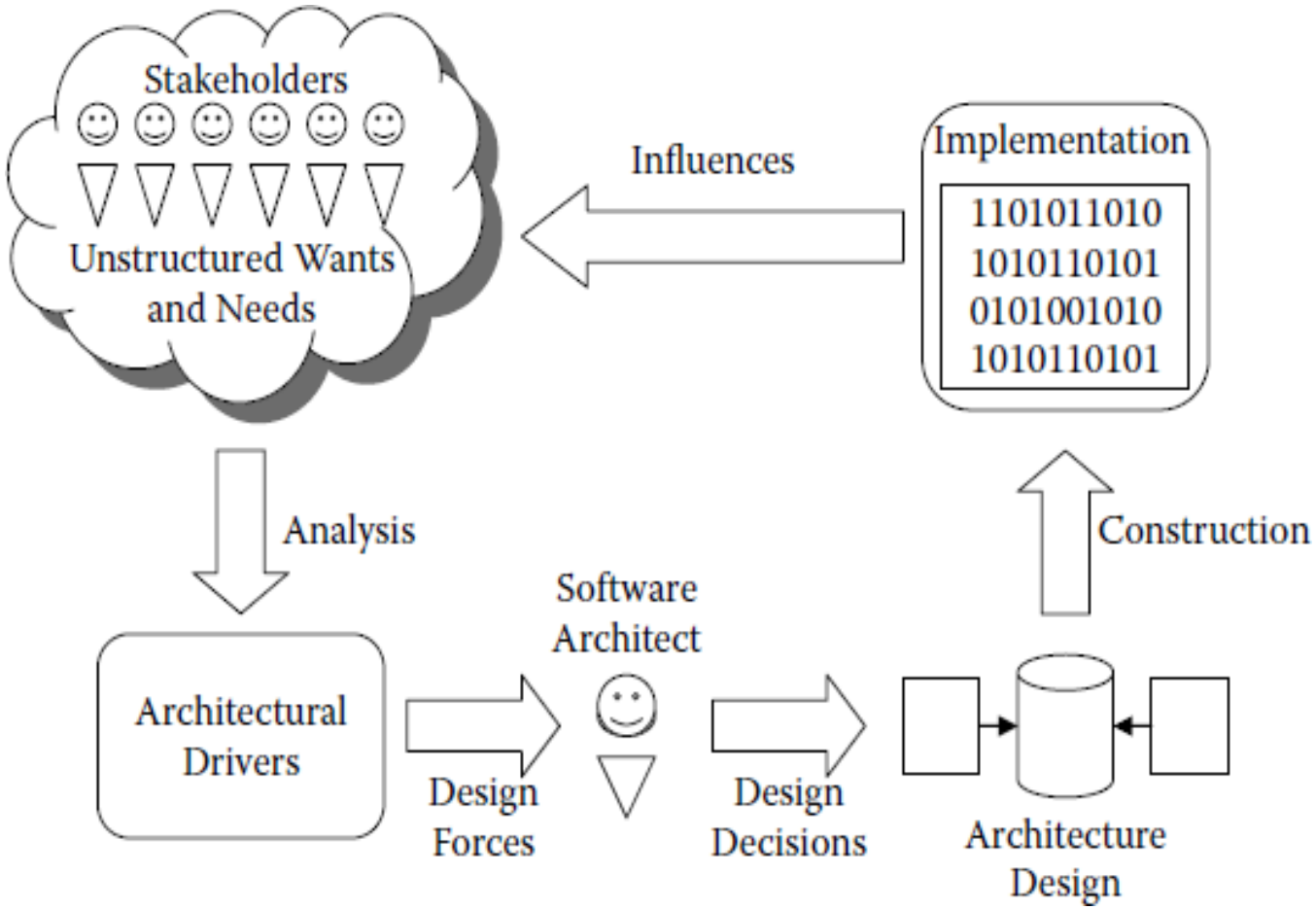
Как започва проектирането на архитектурата?

- Проектирането започва при наличието на изисквания. От друга страна, не се изисква наличието на много изисквания, за да започне проектирането

- Архитектурата се оформя от няколко (десетина) основополагащи функционални, качествени и бизнес изисквания – т.н. архитектурни драйвери

- Ще разгледаме т.нар. процес Architecture Driven Design (ADD)

Цикличен процес на създаване на архитектурата



Как се избират драйверите?

- Идентифицират се тези цели на системата, които са с най-висок приоритет
- Тези цели се превръщат в сценарии за употреба или за постигане на качествено свойство
- От тях се пробират не повече от 10 – тези, които имат най- голямо влияние върху архитектурата
- След като драйверите бъдат избрани, започва проектирането. Започва и задаването на въпроси, което може да доведе до промяна в изискванията, което пък може да доведе до промяна в драйверите и т.н.

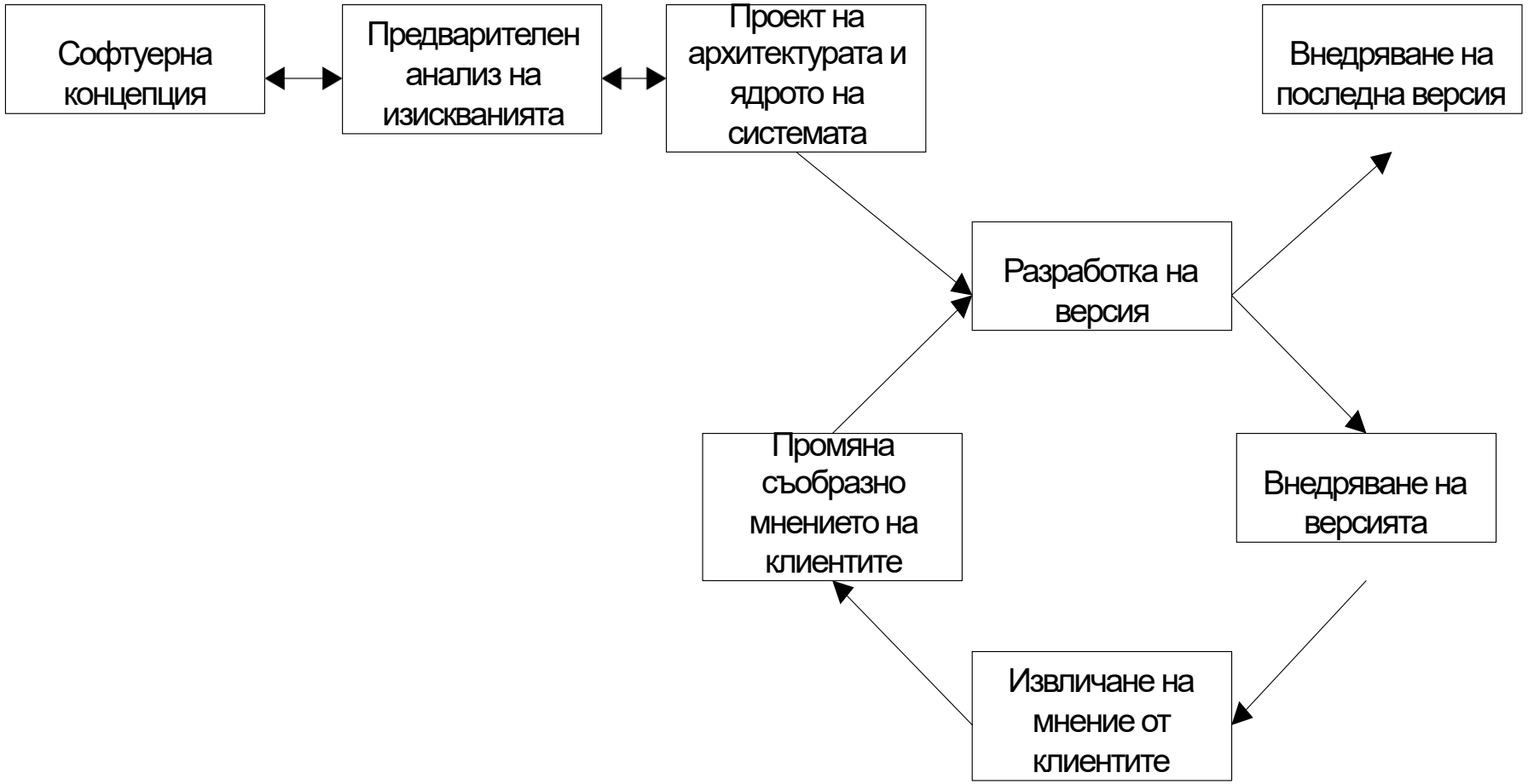
ADD – Attribute Driven Design

- ADD е подход за проектиране, в който основна роля играят качествените свойства (атрибути)
- Това е рекурсивен процес на дефиниране на архитектурата, като на всяка стъпка се използват тактики и архитектурни модели за постигане на желаните качествени свойства
- В следствие на приложението на ADD се получават първите няколко нива на модулната декомпозиция и на други структури, в зависимост от случая
- Това е първото проявление на архитектурата и като такова е на достатъчно високо ниво, без излишни детайли

Стъпки на ADD

- Избира се модул (с известни изисквания), който ще се декомпозира.
- Модулът се детайлизира:
  - Избират се архитектурните драйвери (най-важните изисквания за този етап)
  - Избира се архитектурен модел, който удовлетворява драйверите. Модела се избира или създава въз основа на тактиките за постигане на избраните свойства. Идентифицират се типовете под-модули, необходими за постигането на тактиките
  - Създават се под-модули от идентифицираните типове и им се приписва функционалност съгласно сценариите за употреба. Създават се всички необходими структури.
  - Дефинират се интерфейсите към и от под-модулите
  - Проверяват се и се детайлизират изискванията, като същевременно се поставят ограничения върху под-модулите. На тази стъпка се проверява дали всичко съществено е налично и се подготвят под-модулите за по-нататъшна декомпозиция
- Това се повтарят за всички модули, които се нуждаят от по-нататъшна декомпозиция

Цикличен процес на създаване на архитектурата



Формиране на екипи

- След като се идентифицират първите няколко нива на декомпозицията, могат да се формират екипи, които да работят по съответните модули
- Структурата на екипите обикновено отговаря на структурата на декомпозицията
- Екипа представлява обособена екосистема, със собствени правила и експертиза
- Комуникацията в рамките на екипа е различна от комуникацията между екипите

Създаване на скелетна система

- Когато архитектурата е готова донякъде и има сформирани екипи може да се започне работа по системата
- По време на разработката обикновено се използват стъбове, за да могат модулите да се разработват и тестват поотделно
- Но с кои модули да започне създаването на скелетна система?

Последователност на реализацията

- Първо се реализират компонентите, свързани с изпълнението на и взаимодействието между архитектурните компоненти (middleware)
- След това се реализират някои прости функционалности
- Последователността по нататък може да се диктува от:
  - Намаляване на риска – първо най-проблематичните;
  - В зависимост от наличния персонал и квалификацията му;
  - Бързото създаване на нещо, което се продава;
- След това на база структурата на употребата се определят следващите функционалности

Въпроси ?

Допълнителни материали

- *I. Sommerville, Engineering Software Products,* ***Chapter 4***

- *Attribute-Driven Design Method Collection, Software Engineering Institute, CMU -* https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=484077