Jarosław Kuchta

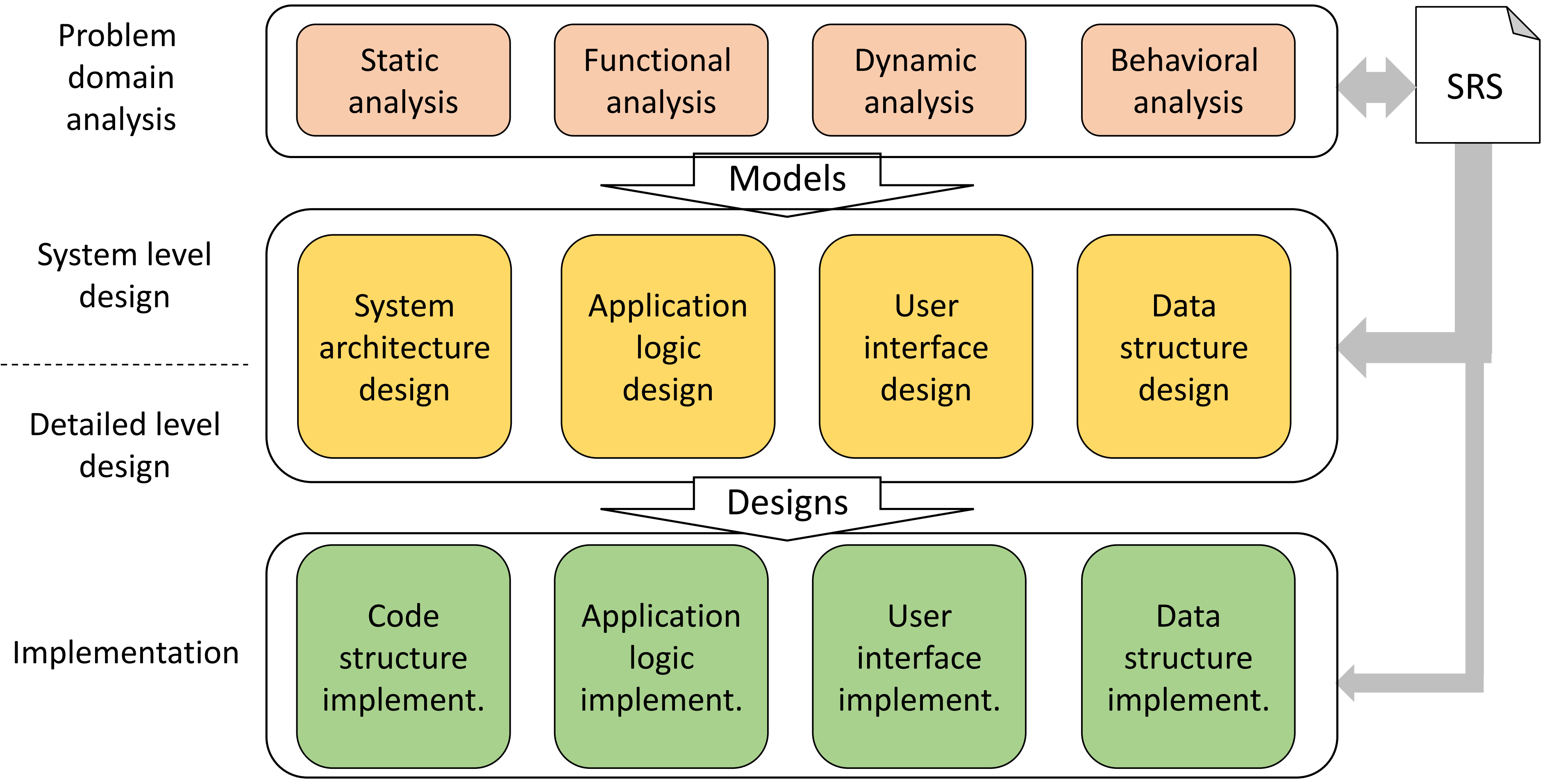
Web Applications Design

Introduction

Agenda

- The role of design in the web applications development (classic approach, agile approach)
- Analytic models and design models
- Multilayer design – frontend/backend development vs. full stack development
- Modern approaches to the application development: RAD, modeling in the IDE, frameworks, design patterns
- Specific problems in the web applications: portability, performance, scaling, security, safety, availability, globalization

The role of design in the web applications development



Analysis – design – implementation (classic approach)

	Analysis	Design	Implementation
Goal	Customer requests understanding	Developing a solution concept	Realization of a developed concept
Questions	What to do? What for?	What to make? How?	How to realize?
Sources	Requirements specification, domain publications, documents from the customer	Requirements specification, analytic models, domain patterns, other projects documentation	Design documents, Requirements specification frameworks documents &how-to
Means	Interviews with the customer UML modeling	Requirements considering design construction	Design considering, code generation & hand-coding, code run, debugging
Effects	Requirements specification, use-case model, class model,	System architecture design, application logic design, user interface design, data structure design	Deployed application, source code, code documentation

Aspects of analysis and design

Design Analysis	System architecture	Application logic	User interface	Data structure
Static	System components	Classes, properties	UI scheme, forms	Database scheme (tables)
Functional	Features roles (rights)	Interfaces, interactions	Menu, commands, navigation	Queries, stored procedures
Behavioral	Events, states	Events, actions, activities	Event handling, error handling	Walidacja danych, kopie zapasowe
Dynamic	Performance, availability	Asynchronous activities	Responsiveness, client logic	Synchronization, d.b. migration

Different concepts

- **Aspect** a separate set of system features that are being considered when other features are omitted
- **Model** a description of the actual existing or projected system showing its selected aspect
- **Diagram** – graphical representation of the model in the form of a drawing whose elements have a certain meaning

Note that:

- Diagram elements are mapped to the model elements.
- ***A diagram is not a model***:
  - Not all model elements are represented on the diagram.
  - There may be different representations of the model (e.g. CRC for model classes).
  - One model can be presented in several diagrams.
- Model elements are mapped to the system elements.
- ***Model is not a system***:
  - The model describes the system only in the selected aspect.
  - Not all system elements are modeled.
  - Should the model fully describe all the elements of the system, it would be equivalent to the system itself.

Models and diagrams

Model	Diagram	Aspect	Analysis	Design
Use case model	Use case diagram, class diagram	Functional	System Capability Modeling	Modelling capabilities (permissions) for users
Class model, object model	Class diagram, Object diagram	Static	Modeling concepts of "business" and their relationships	Modeling the object-oriented code structure
Interaction model	Interaction diagram, activity diagram, Collaboration diagram	Functional	Modelling system interoperability with the environment	Modeling interoperability of system components
State model	State Transitions Diagram	Behavioral	Modeling system response to events	Modeling interoperability of system components
Timing model	Sequence Diagram Timing Diagram	Dynamic	Modeling Event Scenarios	Modeling time-dependent events and processes
System architecture model	Component Diagram, Deployment diagram	Static	Not used	Modeling the hardware and software structure of the system
Code structure model	Package diagram, Object diagram	Static	Not used	Modeling the physical code structure
Data structure model	ERD diagram, Class diagram	Static	Not used	Modeling the database schema
Navigation model	WND Diagram, Storyboard	Functional	Not used	Model transitions between interface components

Application designing – pros & cons

Pros

- Faster understanding of customer needs
- Detect potential technical problems faster
- Possibility to choose the optimal solution
- Ability to understand complex application structure
- Saving on code modifications
- Facilitating later care and development

Cons

- Difficulty in application with indefinite requirements
- Deferred delivery of the running application to the client
- Risk of unrealizability of the project (detaching from implementation possibilities)
- Costs for the development of project documentation
- Deferred return of the development costs

Modern approaches to application design (1)

- RAD – Rapid Application Development
  - Implementation without an earlier design
  - Design = development
  - Advantage – efficiency
  - Drawback – dependency from the IDE (Integrated Development Environment) – No project portability
- Modeling in the IDE
  - Using the modeling tools included with the IDE
  - Limited modeling capabilities
  - Advantages:
    - Easy to translate design into code
    - Ease of creation of project with working code
  - Drawback – dependency from the IDE (Integrated Development Environment) – No project portability

Modern approaches to application design (2)

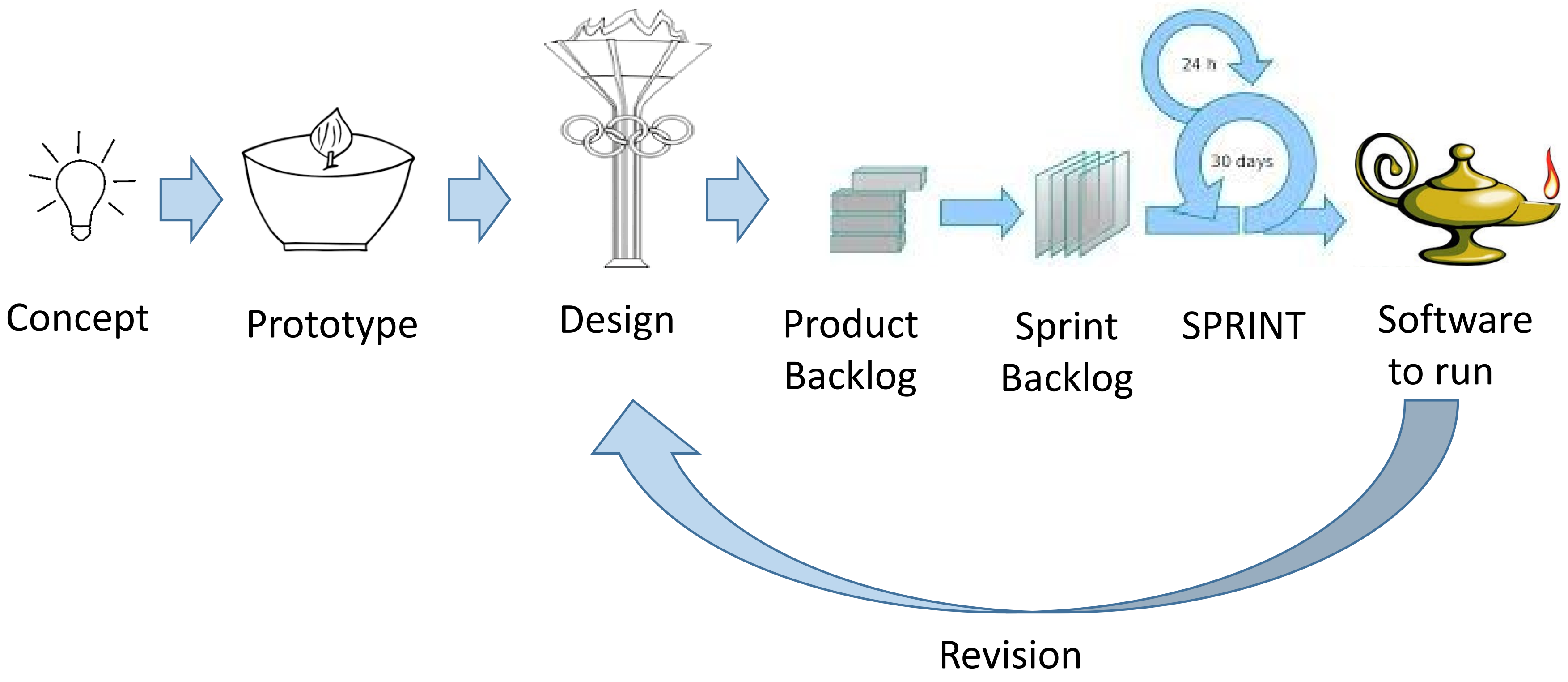
- Using Design Pattern:
  - Design pattern – proven design solution
  - Manual implementation needed
  - Advantage – versatility of application
  - Disadvantage: High cost for encoding
- Use frameworks
  - Framework – implemented design pattern + recipe for proven implementation solution
  - Advantage – Large choice, efficiency in typical applications
  - Threats: Incompatibility of different frameworks, lack of knowledge, immaturity of solutions
  - Disadvantages: Lack of flexibility of use, risk of default of the schedule
- Conclusions for application:
  - The use of frameworks is effective, but very risky.
  - You might want to use design patterns when the frameworks are not sufficient.

Designing in agile methodologies?

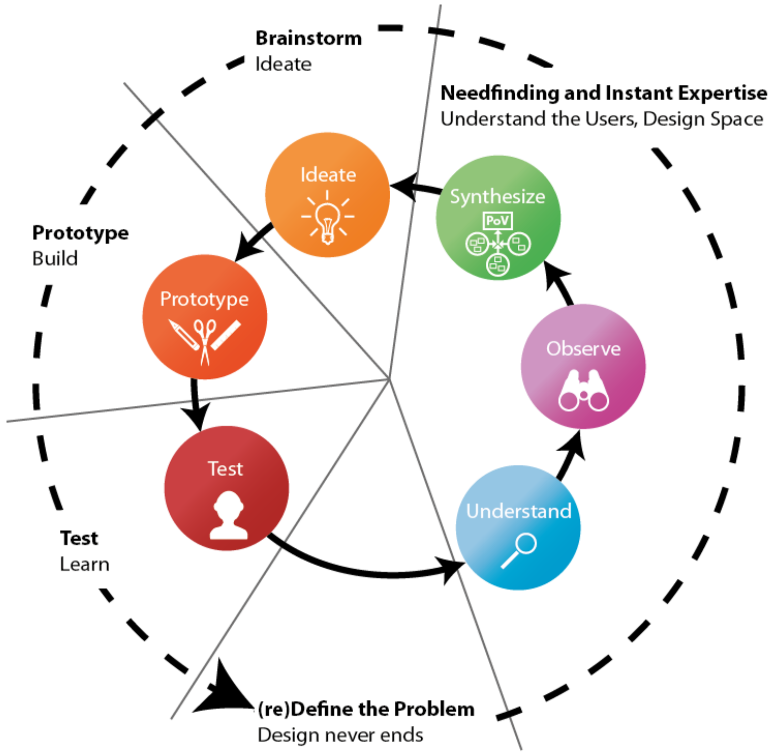
- *Extreme programming, SCRUM* – Iteration planning (not designing)
- *Test Driven Development* – Tests designing
- *Feature Driven Development* – functionality design?

Design in SCRUM?

In larger IT projects:



Design Thinking

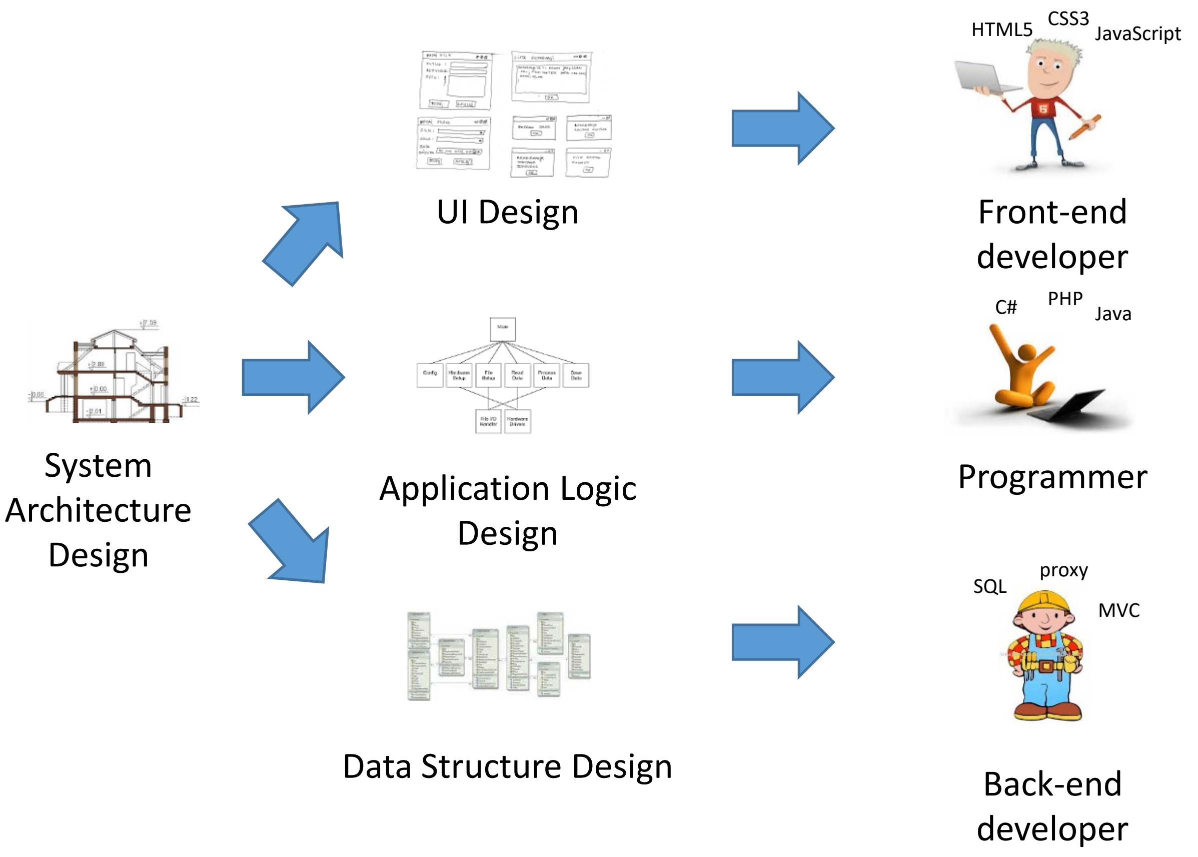


Conclusion:

Design is accompanied by manufacturing even if it is not explicitly exposed

z: Jumpstarting Scrum with Design Thinking
Christophe Vetterli et. al.
http://www.researchgate.net/publication/255710860

Multilayer Design – Division of Developer Skills



<u>Opportunities:</u>
- ✓ Work parallelization
- ✓ Skill division
- ✓ High skill use

<u>Threats:</u>
- ✓ Lack of coordination
- ✓ Misunderstandings
- ✓ Project inconsistency

<u>Conclusion:</u>
- ✓ Full-stack developers needed

Specific Web application design issues

The requirements specification must specify:

- System architecture requirements (may arise from enterprise Organization)
- Portability requirements
- Performance requirements (difficult to determine – need to estimate) and scalability
- Protection and safety requirements – to remind the customer
- Accessibility and globalization requirements (e.g. user interface languages)

Portability issues

- Users can have a variety of hardware, including the older (less efficient).
  - ➢ *What will be the minimum system requirements?*
- Clients can run on different operating systems.
  - ➢ *On what systems is the application running?*
- Some applications may be designed for mobile devices (responsiveness – match the interface to your device's capabilities).
  - ➢ *How to keep your apps responsive?*

Performance issues

- Data must be transmitted over a distance over different bandwidth.
  - ➢ *How to minimize the amount of data transferred?*
- The server must often handle multiple clients.
  - ➢ *Whether and how to divide the processing between the server and clients?*
  - ➢ *Whether and how to divide the functionality into multiple servers?*
- The server must provide a quick response to user requests.
  - ➢ *How to store the results of calculations for reuse?*
  - ➢ *Whether and how to divide the functionality into multiple servers?*

Scalability issues

- It is difficult to predict the actual load of the server.
  - *How to divide functionality into multiple servers?*
  - *How to enable server inclusion as needed?*
  - *How to protect system from overload?*
- The amount of stored data can be enormous.
  - *Which database engine to use?*
  - *How to increase data server capacity?*
  - *Apply Database LAX?*

Security issues

- The Web server is visible from the outside.
  - ➢ *How to ensure that users are authenticated?*
  - ➢ *What user roles to plan and which permissions to grant them?*
  - ➢ *Whether to restrict external access to the server?*
- Customer data is generally confidential.
  - ➢ *How to protect personal data?*
  - ➢ *What data does the administrator have access to?*
  - ➢ *How to prevent users from accessing other users data?*
  - ➢ *How to ensure safe communication?*
- A Web server can become an object of hacker attacks.
  - ➢ *How to prevent server ports from scanning?*
  - ➢ *How to protect server from break-in?*
  - ➢ *How to reduce potential attack opportunities?*

Safety issues

- Servers may crash.
  - *Whether to provide backup servers?*
- Customer data can be destroyed.
  - *How and when to make backups?*
  - *Where to store backups?*
  - *Whom to give data recovery privileges?*
- Internet links may crash or become inadequate.
  - *How to provide backup Internet connections?*

Accessibility and globalization issues

- The website may have users with different options and preferences.
  - ➢ *What group of users is this service for?*
  - ➢ *How to ensure readability and intelligibility for users?*
  - ➢ *How to ensure readability for color-sensitive users?*
  - ➢ *Is it necessary to provide a version for the visually impaired?*
- The website can be accessed from all over the world.
  - ➢ *Is it necessary to have availability 7/24?*
  - ➢ *When and how to do server maintenance?*
- The website may be available to people from different cultures.
  - ➢ *What language versions to implement?*
  - ➢ *What standards of data exchange to provide?*
  - ➢ *Whether and how to adapt to different cultural sensitivities?*

Literature

- Pressman R.S.: *Software Engineering, A Practitioner's Approach.*
- Booch G., Rumbaugh J., Jacobson I.: *UML. User Guide*.