Systems software design

Methods of system analysis and design

Who are we?

Krzysztof Kąkol
Software Developer
PGS Software S.A.

Jarosław Świniarski
Software Developer
PGS Software S.A.

Presentation based on materials prepared by
Andrzej Ciarkowski, M.Sc., Eng.

Course objectives

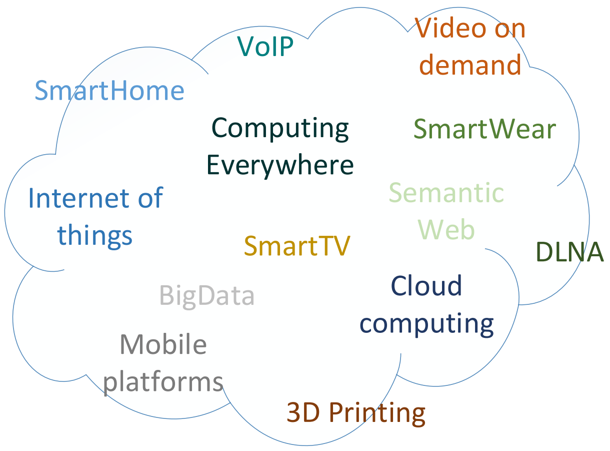
- Gathering **theoretical** knowledge
- Learning **basic techniques** of software development
- Learning about **processes in real business environment**

Course plan

1. Methods of systems design and analysis
2. IT systems project management. Scrum. Team forming
3. Version control systems. Documentation
4. Runtime configuration. Debugging and profiling
5. Basic design patterns
6. Multithreading. Operating system services
7. Network communication. Shared libraries
8. Exam ☺

Why we learn that?

IT systems are ubiquitous…



What are current top trends?

- **Project management** – Scrum
- **Systems architecture** – N-Tier, Domain-Driven Design, Hexagonal
- **Systems development processes** – Test Driven Development, Behavior Driven Development
- **Code quality and maintainability** – SOLID, Clean Architecture

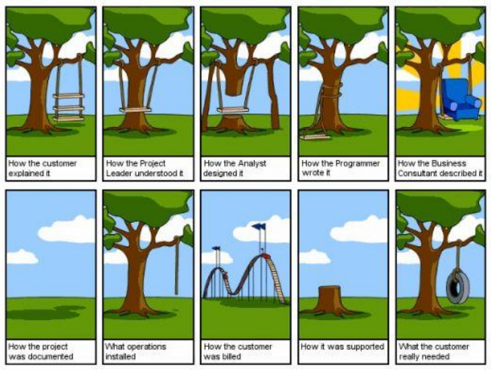
A little bit about the past…

- Waterfall project management
- Unified Modelling Language
- Formal ways of gathering requirements

- **…well… that still works, but…**

A little bit about the past…

… it can result in that:


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Unified Modelling Language

UML

- UML is a graphical language allowing to visualize system's architectural blueprints as diagrams
- UML diagrams represent two views of system model
  - Static (structural) view with objects, attributes, operations and relationships
  - Dynamic (behavioral) view focusing on collaboration between objects and state changes
- The UML diagrams are just a view (projection) of the model; the model exists without the diagrams as well, yet the diagrams allow to visualize it

UML – structural diagrams

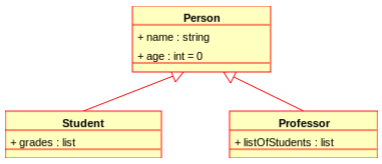
- Emphasize the things that must be present in the system being modeled
- Used for documenting the software architecture of systems
- Types
  - Class diagram
  - Component diagram
  - Object diagram
  - Composite structure diagram
  - Deployment diagram
  - Package diagram

UML – behavioral diagrams

- Emphasize what must happen in the system being modeled
- Used to describe the functionality of the system
- Types
  - Activity diagram
  - Interaction diagrams
  - State diagram
  - Use Case diagram

UML – class diagram

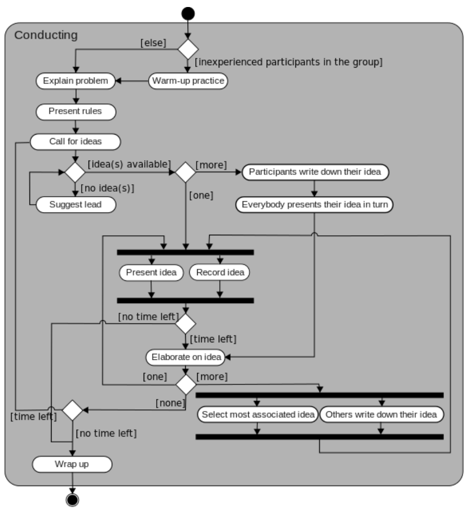
- Describes the structure of a system by showing the system's classes, their attributes, operations and relationships among objects
- The most common of UML diagrams
- Also, one of the most complex ones



13

UML – activity diagram

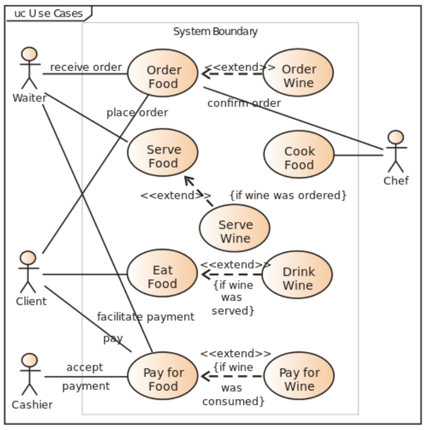
- Graphical representation of workflows of activities and actions, show the overall flow of control



14

UML – use case diagram

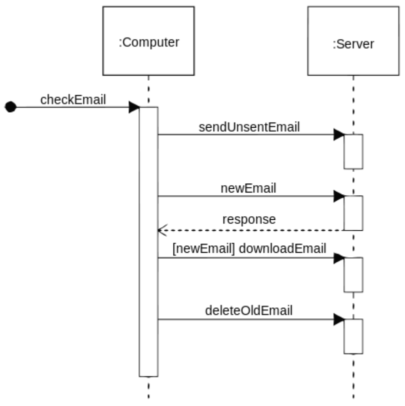
- The representation of user's interaction with the system



15

UML – sequence diagram

- Shows object interactions arranged in time sequence
- Associated with realization of use cases



16

Methods of software development

Methods of software development

- Test-Driven Development
- Red-Green-Refactor:
  - Write test first
  - Check that it doesn’t pass (Red phase)
  - Create naive implementation to make it pass (Green phase)
  - Refactor to make your code better
- Many tools supporting TDD, for different languages

Methods of software development

- Behavior-Driven Development
- Specify behaviors of the system
- Used to make specifications in constant cooperation with the customer
- Provides means to create software with highest business value

Methods of software development

Behavior-Driven Development – sample scenario:

Story: Returns go to stock

In order to keep track of stock
As a store owner
I want to add items back to stock when they're returned

Scenario 1: Refunded items should be returned to stock
Given a customer previously bought a black sweater from me
And I currently have three black sweaters left in stock
When he returns the sweater for a refund
Then I should have four black sweaters in stock
…

Domain-Driven Design

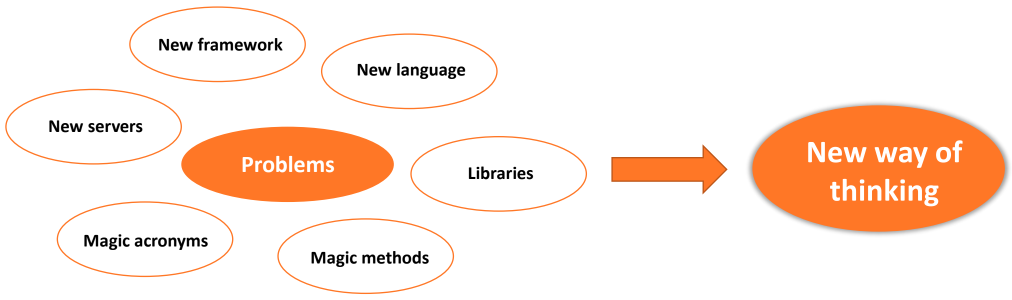
Domain-Driven Design

- **Why** Domain-Driven Design?
- **Core ingredients of** DDD
- **Building blocks**

Why DDD?

Eric Evans:

Domain-Driven Design: Tackling Complexity in the Heart of Software



Why DDD?

- Rules and patters
- Code testability
- Problems and domain language
- Experience

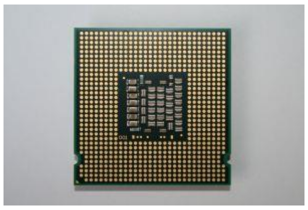
DDD core ingredients

Domain experts

- Expert in a selected domain
- Domain expert must be available for the team
- Expert provides behavior descriptions

Core domain and generic subdomains

- Core domain
- Generic/supporting subdomains

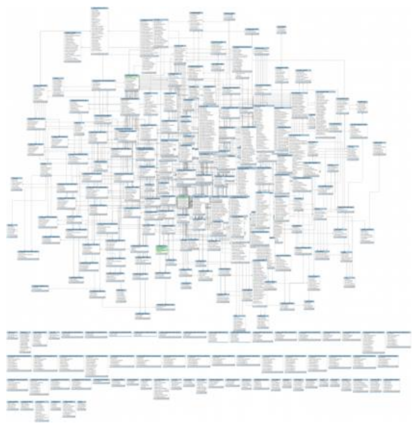


Ubiquitous language

- Created with domain expert
- Simplifies communication

Bounded context

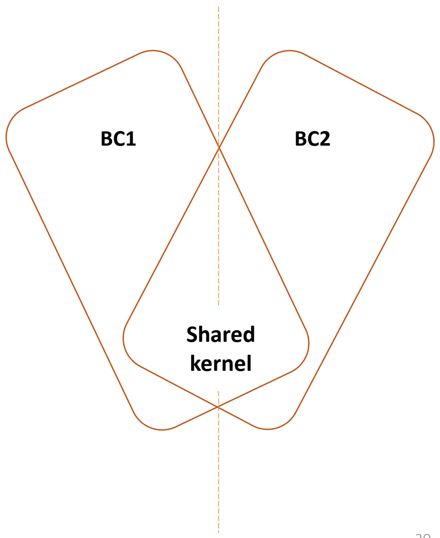
Big Ball Of Mud



29

Bounded context

- Modelling of a single domain
- Subdomain -> limited application context
- Real life:
  - Ideally – separated BCs
  - Practically – shared objects



30

Shared kernel

- Common application parts – eg. authentication
- Hard to maintain – teams must synchronize changes in shared kernel
- Shared kernel does not change very often

31

Summary

32

Problems?

- We usually need a lot of time to learn new methodologies. We must very often change **mindset**
- DDD requires learning – it provides concept, rules, patterns and processes
- DDD is best for complex subjects – it makes no sense to use DDD for CRUD-like apps
- DDD requires domain experts to be available

What are the pros?

- Flexible model and flexible code – simplified maintenance and development
- We can realize the client's vision
- DDD provides ways to solve very difficult problems
- Well organized and testable code
- Business logics strictly separated

What else is important?

- Project team organization – we'll discuss Scrum
- Design patterns – we'll speak about that later

35

Questions?

36

Krzysztof Kąkol
kkakol@pgs-soft.com

Jarosław Świniarski
jswiniarski@pgs-soft.com

www.pgs-soft.com