Introduction to Software Development

Lect. PhD. Arthur Molnar

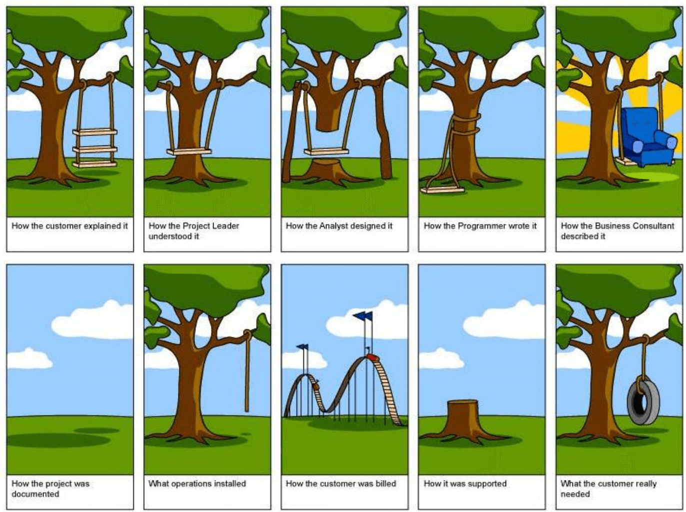
Babes-Bolyai University

arthur@cs.ubbcluj.ro

Overview

1 Introduction to software development
- Basic notions
- Simple feature-driven development process
- How to approach Assignment 2

Introduction to software development


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

Basic roles in software engineering

Programmers/Developers

- Use computers to *write/develop* programs for users

Testers/QA:

- Check the program to discover errors

Clients/stakeholders:

- Who is affected by the outcome of a project

Users

- *Run programs on their computers*

Basic roles in software engineering

Software development process

An approach to building, deploying, and maintaining software. It indicates:

- What tasks/steps must be taken during development.
- In which order?

Basic roles in software engineering

A **software development process** is an approach to building, deploying, and maintaining software.

What we will use

Simple feature-driven development process

Example problem statement

A *problem statement* is a short description of the problem being solved.

Calculator

A *teacher* (client) needs a program for *students* (users) who learn or use rational numbers. The program shall help students make basic arithmetic operations

Requirements

Requirements - define in detail what is needed from the client perspective. Requirements define:

- What the client needs.
- What the system must include to satisfy the client's needs.

Requirements

Requirements guidelines

- Good requirements ensure your system works like your customers expect. (don't create problems to solve problems!)
- Capture the **list of features** your software is supposed to do.
- The list of features must clarify the problem statement ambiguities.

Features

A **feature** is a small, client-valued function:

- expressed in the form <**action**> <*result*> <object>,
    - action - a function that the application must provide
    - result - the result obtained after executing the function
    - object - an entity within the application that implements the function
- and typically **can be implemented within a few hours** (in order to be easy to make estimates).

F1.	**Add** number to calcularor
F2.	**Clear** calculator
F3.	**Undo** last operation

Simple feature-driven development

- An **iteration** is a set period of time within a project in which you produce a stable, executable version of the product, together with supporting documentation.
- An **iteration** will result in a working and useful program for the client (will interact with the user, perform some computation, show results)

Simple feature-driven development

- Build a feature list from the problem statement
- Plan iterations (at this stage, an iteration may include a single feature)
- For each iteration
  - Model planned features
  - Implement and test features

Simple feature-driven development

- **Example iteration plan**

Iteration	**Planned feature**
I1	F1. **Add** number to calcularor
I2	F2. **Clear** calculator
I3	F3. **Undo** last operation

Iteration modelling

At the beginning of each iteration you must understand the work required to implement it. You must **investigate/analyze** each feature in order to determine work items/tasks. Then, work items are scheduled. Each work item will be independently implemented and tested.

Iteration modelling

Iteration 1 - Add a number to calculator

- For simple programs (e.g. Calculator), running scenarios help developers understand what must be implemented.
- A **running scenario** shows possible interactions between users and the program under development.

Iteration modelling

Iteration 1 - Add a number to calculator

	User	Program	Description
a		0	Shows total
b	1/2		Adds number to calculator
c		1/2	Shows total
d	2/3		Adds number to calculator
e		5/6	Shows total
f	1/6		Adds number to calculator
g		1	Shows total
h	6/6		Adds number to calculator
i		2	Shows total

Work items/tasks

- Define a task for each operation not already provided by the platform, e.g. T1, T2.
- Define a task for implementing the interaction between User and Program, e.g. T4.
- Define a task for implementing all operations required by UI, e.g. T3.
- Determine dependencies between tasks (e.g. T4–> T3–> T2–>T1, where –> means depends on).
- Schedule items based on the dependencies between them.

Work items/tasks

Possible tasks for calculator application

Task	Description
T1	Compute the GCD of two integers
T2	Add two rational numbers
T3	Implement init, add and total operations
T4	Implement user interface

Test Cases

Test case - A set of *test inputs*, *execution conditions*, and *expected results* that you identify to evaluate a particular part of a program.

Inputs: a,b	**gcd(a,b)**
2,3	1
2,4	2
6,4	2
0,2	2
2,0	2
24,9	3
-2,0	ValueError
0,-2	ValueError

How to approach Assignment 2

- You have to provide a menu that is displayed when starting the application
- Code must be divided into functions
- Each function must only do one thing
- Every non-UI function must have specification
- You must provide the required documentation
- Must be turned in no later than week 5