Context Project Guideline

(for students and lecturers)

Introduction

The goal of the context project is to perform a software project specified by an external party in a given context. Students will experience theories of software engineering and teamwork by applying them to a concrete case. This includes requirements engineering, architecture and software design as well as the implementation and presentation of the software system.

Learning objectives

The context project has the following process, presentation, and product specific learning objectives. After completing this course the student will be able to:

Process

- Work in a team following an iterative and incremental software development process (the process is determined by the group)
- Work together with the users and stakeholders from a non-IT context.
- Analyze and evaluate a problem in a non-IT context
- Acquire the information and knowledge on the given non-IT context from the literature and experts in order to design and develop the solution
- Evaluate the requirements of different stakeholders
- Plan each iteration of the project, reflect and evaluate the planning regularly, and adjust the planning to changes in the project and the environment
- Reflect on his/her contribution to the project and the final product
- Form a vision of the role of IT in the given context and its role for solving the given problem

Product

- Analyze and document the system requirements in a requirements document
- Design and document the architecture for the given problem in a non-IT context
- Specify an iterative and incremental test and implementation plan
- Develop a prototype and/or final product according to the requirements, design, implementation and test plan
- Continuously test and evaluate the solution

Presentation

- Report in oral form to his/her peers, supervisors, users and customers
- Report in written form to his/her peers, supervisors, users and customers

Available context projects and coordinators for 2012

In 2011 the following context projects are available:

1. Health Informatics / Medical Informatics (Willem-Paul Brinkman)
2. Crisis Management: Improving Resilience to Incident Management, Search & Rescue Missions, and Public Safety (Catholijn Jonker, Niek Wijngaards)
3. Mijn Cultureel Erfgoed (Alan Hanjalic)
4. Programming Life: Synthetic Biolog (Dick de Ridder, Marcel Reinders)

List of deliverables

During the projects, the teams need to submit a number of deliverables that will be used for the assessment of the group. Basically, for each deliverable the team first submits a draft version to the teaching assistants. They will provide feedback on the draft that then needs to be integrated by the students within one week. After one week the group submits the final version, which will be considered for the grading.

General guidelines for the documents:

For each document, the following requirements apply. Each document should contain:

- Title page (title of the document, name and id of the students, datum, version number)
- Short abstract about the contents of the document
- Table of contents
- Document specific content
- References to literature and other sources of information

List of deliverables and schedule:

The list of the deliverables and corresponding deadlines for the submission of the draft and final versions is:

- Requirements: Analysis and Design (draft: Q3/W3, final: Q3/W4)
- Architectural Design (draft: Q3/W4, final: Q3/W5)
- Test and Implementation Plan (draft: Q3/W5, final: Q3/W6)
- Peer-Reviews (Q3/W4 and Q4/W4)
- Lightweight SCRUM Plans (before each iteration)
- Final Report (draft: Q4/W7, final: Q4/W8)
- Final Product (version: Q4/W2, final: Q4/W8)
- Final Presentation (Q4/W9, exact date will be announced)

The following provides a guideline on how to structure each deliverable (might be adapted for a particular context).

Requirements: Analysis and Design

1. Introduction
2. Current system
3. Proposed system
   - 3.1. Overview
   - 3.2. Functional requirements
   - 3.3. Nonfunctional requirements
   - 3.4. Constraints (“Pseudo requirements”)
   - 3.5. Analysis models
     - 3.5.1. Use case model
       - 3.5.1.1. Use case diagram
       - 3.5.1.2. Use case descriptions plus scenarios
     - 3.5.2. Business object model
     - 3.5.3. Dynamic models (describe the main scenarios with sequence, state, or activity diagrams)
   - 3.6. User interface
4. Glossary

Architectural Design

1. Introduction
   1.1. Purpose of the System
   1.2. Design Goals
   1.3. Definitions, acronyms and abbreviations
   1.4. References
   1.5. Overview
2. Current software architecture
3. Proposed software architecture
   3.1. Overview
   3.2. Subsystem Decomposition (which sub-systems and dependencies are there between the sub-systems?)
      3.2.1. Interface (API) of each sub-system
   3.3. Hardware/Software Mapping (mapping of sub-systems to processes and computers, communication between computers),
   3.4. Persistent Data Management (file/ database, database design)
   3.5. Global Resource Handling and Access Control for the different actors
   3.6. Concurrency (which processes run in parallel, how do they communicate, how are deadlocks prevented?)
   3.7. Boundary Conditions (how is the system started and stopped, what happens in case of a system crash)

Test and Implementation Plan

1. Introduction
2. MoSCoW (prioritization of requirements: Must, Should, Could, Wont)
3. Implementation and tests
   3.1. Order of implementation of features
      3.1.1. Iterations
      3.1.2. Milestones
   3.2. Test plan
      3.2.1. Unit testing
      3.2.2. Integration testing
      3.2.3. Acceptance testing
4. Risk analysis (what are the risks for the successful implementation of the system?)

Peer-Reviews

In the fourth week of each quarter the students perform a peer-review of all team members in the project group including herself (max. 1 A4). The review contains for each team member an evaluation of:

- the know-how the team member is bringing into the group
- the way of working
- the collaboration with the other team members

Lightweight SCRUM Plans (before each iteration)

Iterations in the implementation phase of the system last for two weeks. Before each iteration of the implementation of the system, the project group submits a plan to the teaching assistant. The planning comprises:

- The selection of a set of features (important features first).
- A List of the tasks for each feature.
- The assignment of students to tasks.
- An estimation of the effort per task.

- The actual effort per task (after the iteration is done).
- Short reflection on the main problems and adjustments of the iteration planning

The planning of the previous iteration is used as input for the planning of the next iteration. At the end of the quarter the plans are added to the final report. The students are free to use the template for the planning.

Final Report

1. Introduction
2. Key problem(s) and solution(s) - highlights
3. Reflection on the teamwork
4. Individual reflection on the project (max. 1 A4 per team member)
5. Lightweight SCRUM Plans

Final Product

The source code should be well structured and include unit tests. The quality of the source code (including test code) will be checked once during the development phase in Q4/W2 and at the end of the project QW4/W9. The evaluation of the quality is performed by the Software Improvement Group (SIG). The feedback given for the version in Q4/W2 should be considered for the final version.

Final Presentation

The main purpose of the final presentation is to demonstrate the final product as well as to show the highlights (key problems and their solutions) and reflect on the development process. Each presentation lasts for 20 minutes and needs to include a live demonstration of the system. The presentation is followed by 10 minutes of discussion.

Implementation

The implementation of the system will start in W6/Q3. The implementation follows an incremental and iterative software development process.

Planning

Each iteration needs to be planned according to the lightweight SCRUM planning documents described before. At the end of each iteration the plan is checked: which tasks features have been finished, which tasks are still open. The results of the check are considered for the planning of the next iteration.

Test-driven

The implementation should be test-driven: first, implement a test for a feature and then start implementing the feature. Make use of a xUnit test framework.

Integrate the different parts of your implementation as soon and as much as possible. Use build environments such as ant or maven. Develop corresponding integration tests to check whether the integrated system works.

Always have a running version

The implementation follows the concept of "Always have a running system". After each iteration, there will be a tool demonstration session in which each project group gives a 10 minutes demo of the most recent implemented feature(s). After the demos

follows a 20 minutes discussion of the design, implementation, tests, and iteration planning.

Assessment

The following describes the assessment of the project groups. Basically, the grade is given to the group.

Calculation of the Grade

The formula for determining the grade per project group is:

Overall score = 0.5*Product score + 0.4*Process score + 0.1*Presentation

The product score is calculated by:

- Requirements: Analysis and Design Document (20%)
- Architectural Design Document (15%)
- Test and Implementation Plan (15%)
- Final Product evaluated by SIG – see below (20%)
- Regular Tool Demonstrations - features (20%)
- Context seminar report (10%)

The process score is calculated by:

- Lightweight SCRUM Plans (40%)
- Final Report (20%)
- Peer Reviews (20%)
- Regular Tool Demonstrations - process and planning (20%)

The presentation score is obtained from the final presentation.

Final Product evaluation by SIG:
The source code and test code of the final product (if it has been implemented with PHP, Java, C, C++, or C#) will be evaluated by the Software Improvement Group (SIG). The criteria used by SIG are:

- Readability of the code (naming of classes, methods, attributes and variables)
- Structure of the code (size of classes and methods)
- Complexity of the code (complexity of methods, depth of inheritance trees)
- Documentation of the code

In projects developed with other technologies for which an automated assessment by SIG is not possible, the Final Product will be assessed through the context teacher using the same criteria.

Plenary presentations

The best presentation from each context project will be selected and awarded with a presentation at the plenary session. The plenary session is open to all students who participated in the course and people involved. Exact date is to be announced.

Demo market

There will be a demo market organized in parallel with the plenary session. Aimed location of the demo market will be the ground floor hallway of the faculty. Definitive location will be announced. The software can be presented with laptops and/or posters. Tables and poster boards will come available.

Organization of the Context Projects

Each context project is done in groups of 5 students. A maximum of 4 groups per context project is allowed. Students can choose their three preferred context projects, however the final assignment will be done by the context project teachers. Students with a higher number of ECTS will be given the preference to do their selected context project.

In general, the process and product is the responsibility of the group (e.g., group determines the agenda of the meetings). Similar, request for feedback and support should come from the group.

Role of the teacher

- Provides a general description of the assignment:
  - goal
  - problem statement
  - prerequisites
  - references and contact persons
- Regular meetings with the TA (weekly),
- Make sure that the groups follow the software development process
- Supports and guides the external advisor
- Overall responsibility

Role of the teaching assistant (TA)

They represent the daily coach of the project groups:

- TA gives feedback on draft versions of the documents, ideas and solutions (but should not provide solutions).
- TA meets weekly with the students (initiative in meetings should come from the students, make sure that they prepare for each meeting).
- TA checks the progress and planning documents

Role of the external advisor

The external advisor should give feedback on the development process and the final product. Feedback on the development process can be one or two times during the implementation phase. Feedback on the product can be one time during the implementation phase (during a demo session) and at the presentation of the final product.

Context Seminar

The context seminar runs in parallel to the context project and provides (additional) information on the problem context. We recommend to give the context seminar within the first five weeks. The information should help the students in defining the requirements and architecture of the system.

The result of the seminar is a report that accounts for 10% of the product score. The requirements and structure of the report depend on the problem context and will be determined by the teacher. We recommend a submission deadline and the end of five weeks or end of the third quarter latest.

End-terms

Based on the learning objectives the following end-terms are used:

- The student should be able to perform a ‘state of the art’ analysis on the topic addressed in the project assignment.
- The student should be able to select and justify the appropriate programming techniques for the end product.
- The student should be able to analyze relevant scientific literature and discuss the implications (written and verbally) in the context of the project.
- The student understands the principals of the (lightweight) scrum approach for software engineering and should be able to manage a project accordingly.
- The student should be able to present, concisely and critically, a software product from technical, functional and end-user perspective.

Project Coordinators

dr. Martin Pinzger, m.pinzger@tudelft.nl
dr. Nick Guldemond, n.a.guldemond@tudelft.nl