# Distributed software development
# Documentation instructions

**Version 1.0**

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 2013-10-21 | 1.0 | Initial version | Juraj Feljan |

# Introduction

## 1.1 Purpose of this document

The purpose of this document is to provide instructions on what to include in the documents that have to be written during the course. The instructions are given for the following documents, each in a separate section:

- Project plan,
- Requirements definition,
- Design description,
- Acceptance test plan,
- Test report,
- Final report.

For the Summary week report and the Minutes of Meeting documents, detailed templates are provided instead of instructions. The type and contents of additional documentation not listed here (for instance, project policies, installation guides etc.) should be agreed upon together with the project supervisor.

## 1.2 Intended Audience

The intended audience is:

- the course students that write the documents,
- the project staff that grades the documents.

## 1.3 References

The instructions listed here are based on the instructions used in the Software Engineering 2: Project Teamwork course at Mälardalen University, with permission from the course teacher, Jan Carlson.

## 2. Project plan

From the project plan document, the reader should quickly get a high-level understanding of what the project is about, and get some basic facts such as who the customer is and who is in the project group. The document should also specify how the work will be organized and present a time plan for the project. Before submitting, make an internal review. Does it give a clear, detailed picture of how your project is organized and how you plan to carry out the work? If you gave the document to another team, would they be able to carry out the project according to it?

At least the following items should be addressed:

- Background and objectives (who is the customer, what is the purpose of the project from the customer's perspective).
- Project group (members, contact info, roles and responsibilities).
- Organization and communication (meetings, forms of contact, routines, how should worked hours and results be reported, configuration management).
- Development process.
- Planned deliverables and milestones, both internal and external.
- Division of work into activities.
- Time plan (use a suitable diagram, for instance a Gantt chart)
- Quality assurance (any planned activities or routines for ensuring quality, both regarding the final product and other deliverables).
- Project risks.

Next we list some additional definitions and clarifications that might be useful when writing the project plan.

A milestone is a time point in the project used to check if the project is on track. Usually it corresponds to the end-point of an activity (or several activities). Milestones such as "80% of the code done" are useless because they cannot be checked. You cannot measure what is 80% of the code, because the amount of code that still has to be developed is uncertain. Milestones such as "alpha prototype" are useless if they do not describe what the alpha prototype should contain.

A deliverable is a concrete output in the project. Deliverables are usually delivered at the end of an activity (or several activities). They can be external (sent to the customer) or internal (necessary for the team, but not delivered to the customer). Deliverables are usually delivered at a milestone, but there can be deliverables that are delivered at a different time point than on a milestone. Also, there can be milestones that do not have any deliverables associated, but still correspond to an activity (or several activities) being finished. In other words, a milestone corresponds to 1-N activities that must be finished prior, and/or to 0-N deliverables that must be delivered. A revised document is a new deliverable.

An activity is a distinct logical thread in project work. For instance, an activity can be developing the graphical user interface of your system. Activities are further broken down into actions, which are concrete steps a particular person needs to carry out before a certain deadline. For instance: “John will develop the login window by Friday noon” is an action under the “Graphical user interface” activity. In the project plan document, the project should be broken down into activities. You will define the actions during the project and they will be reported in the Minutes of meeting and Summary week report documents.

## 3. Requirements definition

The purpose of this document is to define what you are supposed to develop. This will be used initially to ensure that you have a common view of the task before you (within the group, and between the group and the customer). It will later be used to guide the development, and it is also one criterion which can be used to measure the quality of the delivered system. Remember that requirements should focus on what to do, not how to do it. Avoid describing solutions.

The following items are probably relevant for most projects:

- Short background (refer to Project plan document for details).
- High level description of the domain and the problem.
- If the project is about extending or improving something that exists, give a description of the existing system, or the context: any existing or planned entities that the developed system should interact with but which will not be developed in the project (for example, physical equipment or an existing database).
- High-level description of the desired functionality (e.g., captured by a UML use-case diagram and described in more detail by UML sequence diagrams). In particular, it can be good to focus on the desired interaction between the systems and the users. A possible format for the use cases can be:

Use case name: <this is a unique, meaningful name>
Participating Actors: <the persons or external devices/legacy software interacting with your system>
Flow of Events: <describes the interaction between the participating actors and the system>
Exceptions: <cases that correspond to the occurrence of some problem in the normal flow>
Special Requirements: <any QoS requirement associated with the execution of this use case>

- Detailed list of requirements. If you use Scrum, this corresponds to the initial product backlog (and some of the things listed below are somewhat different).
- Give each requirement a unique ID.
- Assign a priority level to each requirement.
- Each requirement is often stated as a name, a description, a motivation and the source (e.g., from customer or identified by the project group).
- Make sure all requirements are possible to validate, i.e., that there is a way to easily determine if a requirement is satisfied or not, avoid vague and ambiguous requirements such as "the system should be sufficiently fast".
- Requirements should be understandable for both the customer and the group, avoid being too technical.
- There can be both functional and non-functional requirements.
- The requirements should be complete, i.e., any system satisfying all requirements should be satisfactory to the customer. In reality, this is very difficult to achieve if interpreted strictly, but you should at least aim in that direction.

## 4. Design description

The design description should capture important design decisions you have made in the project, and should provide a good basis for understanding the code. A possible reader (in addition to the course staff) would be a new member joining the group, or someone who will maintain or evolve the system further.

The following items are relevant in most projects, but you should think about what is important to capture (maybe some of this can be skipped, and possibly additional items should be added):

- Short background (refer to Project plan document for details).
- High-level description of the system to be developed and the desired functionalities (refer to Requirements definition document for details).
- System overview, especially if the developed software is part of a larger system (e.g., describing the hardware and any equipment controlled by or providing input to the system, and detailed descriptions of interfaces to other systems or existing parts of the system not developed within the project).
- Software architecture. What does the overall decomposition of the software into modules/units/components look like (from a static and dynamic perspective, i.e., what are the units and how do they interact)? What is the rationale (motivation) for this design? Depending on your project, you might also want to describe concerns such as major system states, persistent data, access control, synchronization and timing, error handling, security, etc. Use appropriate figures and diagrams (for instance suitable UML diagrams) to define and exemplify. Make sure to explicitly show the connection between the requirements and use cases on one hand and the software modules on the other.
- Detailed software design. For each (or some) of the architectural units, provide a detailed description of its internal design from a static and dynamic perspective i.e., what are the parts (e.g. classes) and how do they interact. Depending on your project, you can describe your database model, Web page organization, key algorithms, protocols between different components etc. Use appropriate figures and diagrams (for instance suitable UML diagrams) to define and exemplify.
- Graphical user interface. Describe the structure and general layout of the interface. You can show the interface by providing mockups (which can later be substituted with screenshots).

## 5. Acceptance test plan

Acceptance testing is done to determine if the requirements are met. The acceptance test plan should outline the overall plan for test activities to be performed by the customer (or together with the customer), motivated by the requirements. Each test should be described detailed enough so that someone else could carry out the test based on these instructions.

The test specifications will differ a lot between different projects, and you have to decide what tests are relevant in your particular project and how to describe them in the best way. The following items give some ideas of what you could address:

- As always, you need some background and references to other related documents.
- Give a high level overview of the test activities, and motivations why these are relevant and sufficient.
- For each test, define detailed instructions how to carry out the test and how test results are measured (pass/fail and what is the criteria, letting the test subject answer a set of questions, etc.). Remember to relate the tests to the requirements.

## 6. Test report

The test report captures and summarizes the test results. The structure of the test report should be defined together with the test specification. Then, as the tests are carried out, the report is filled with information. Finally, when testing is over, the results are summarized. This document does need not to be as self contained as the other documents. Just describe what the document is, and refer to the test specification.

## 7. Final report

The final project report should summarize the experiences of the project, both in terms of the produced results and of the project work. Identify as many parameters/metrics as possible that can be used to illustrate the project work. These metrics should be both on the level of the project, on the level of the two distributed sub-teams and on the level of individual team members. The final report should heavily relate to the project plan document. Make an analysis of the results with respect to the original project plan, show in what aspects the original plan was followed and in what aspects the project deviated from the plan. Use suitable figures, diagrams and tables.

At least the following items should be addressed:

- Background information and references to other related documents.
- Project results.
    - Give an overview of the results produced in the project.
    - List the produced deliverables.
    - Refer to the Requirements definition document and state how many of the requirements have been satisfied (fully or partially).
    - Comment on missing functionality and possibilities for improvements and extensions.
- Project work
    - Organization and routines (refer to project plan and comment if there have been any changes during the project).
    - Total project effort (e.g., in person-days) and how it was distributed over different activities.
    - Worked hours per group member.
    - Other suitable metrics.
    - Positive experiences (for example successful routines or techniques).
    - Improvement possibilities (for example, what would you do different if you were to do a similar project again?).