Navigation

- Home
- » Guides
- » Software Development
- » Software Development Life Cycle
- » Software Development Processes

Search Go

Table of Contents

Software Development Processes¶

Authors: Gökmen Göksel, Semen Cirit, Renan Çakırerk, Beyza Ermiş

Version: 0.1

../../../_images/Graph.png

Process Objectives and Policies¶

Analysing, designing, developing and testing the requirements to bring out the software product that is demanded by the users or customers.

Process Scope¶

It covers the software development activities. It ends with the closure of the project.

Process Owner¶

The Project Leader who takes the responsibility of the software's development and the software development team whom worked with.

Process Relations¶

This process is carried out in parallel with the Operational Coordinator..

Process Inputs/Outputs¶

Inputs¶

- Project Plan
- Every kinds of document that is necessary for the realization of the project and is delivered by the customer or user
- Software Test Plan

Outputs¶

- System requirements specification
- Software requirements specification
- Software design description document
- Interface design description document
- Database design description document

Process Resources¶

Human Resources¶

- Technical Coordinator is responsible for giving final approval for the technical infrastructure and giving the technology direction for the related project.
- Operational Coordinator is responsible for balancing and controlling the appropriateness of the project to software processes and progress and timing of the work with other projects and releases.
- Project Leader, is choosen by the "Technical Coordinator and Operational Coordinator based on the content of the job.
- Developers involved in the project, are choosen in conjunction with the Operational Coordinator at the request of Project Leader.
- Project Leader and the developers are responsible for determining the software requirements of the project and making competitor analysis, software design, software development, unit testing, system intergration and packaging.
- Project Leader is the follower of these works and the reporter to the upper unit (Operational Coordinator ve Technical Coordinator) at the same time.

Process Activities¶

Project Initialization¶

Project initialization starts with the statement of requirement description includes the main function of the software with one or two sentences made by the person or people who requested the software. This explanation could be illustrated as below:

"It is necessary a graphical user interface for the user management in the system."

A Project leader is determined after the person who requested the software convey this information to the Operational Coordinator.

After the Project Leader is determined, a representational code name is determined for the project and with this name a project is opened under Pardus Project Tracking System (tracker.pardus.org.tr). It could be expressed as a sub-project of another project in accordance with the project requirements. It will be appropriate the participation of the responsable of the superior project -even as a consultant- to the project in question in this condition. Furthermore, depending on the privacy status of the project, it is created a working directory at one of the code repositories "internal" or "uludag" under the subdirectory related to the topic. The information and documents related to all the studies should be stored under this directory. All progress should be carried out over the Project Tracking System and all the information and documents should be stores under the project's working directory. Even if it has become absolute that a document put in the directory is not necessary anymore, it should not be removed for the purpose of recording; it should be presented to the required units within the "Project Final Report" when the project at available version or the project is at the end. In addition, an e-mail list could be opened for the purpose of the project issues' discussion, in the circumstance of the need for the project.

The working repository should be opened under "internal" or "projects" repositories together with the specified code name depending on the privacy status of the project.

Proprietary Projects¶

All the codes and documents related to the proprietary projects appear in a closed project directory.

Open Projects¶

Following the coding process of the project, a Git repository is opened with the project code name under Pardus Git server for the project. All the documentations that are prepared and developed up to this stage are moved to this repository. All the codes and documentations will be developed after this stage are stored/updated in this repository.

If the decision of making the project an open project after the start of the project as a secret project is given; all the information and documentation that are stored in the "internal" up to this stage are moved to git repository presented in open.

Establishment of the System Requirements¶

1. The first system requirements are conveyed by the "External Projects Coordinator" if the project is including external projects, otherwise they are conveyed by the Operational Coordinator according to the demands of new features requests from users.
2. The system requirements table is established by the detailed description of the priority requirements that are targetted to be fulfilled at the user side (its priority, it is demanded for

which user or customer etc.)

3. System requirements are reviewed and refined by Project Leader, system engineer, External Project Coordinator (in the condition of external project) and Operational Coordinator

Competitor Analysis¶

1. Project Leader determines the competitors that will be analyzed with the approval of Technical Coordinator.
2. Project Leader determines a team related to the issue with the approval of “Operational Coordinator” for the competitor analysis. (This team is the probable development team of the project, then it could be additions or ommisions to this team)
3. On behalf of obtaining certain results from the Competitor Analysis, proof of concept could be performed by the person/people related to the issue and the results of these processes will shape the software requirements substantially.
4. In addition, in this process, the technical constraints are stated by the team that will perform the project and the software requirements could be changed based on these constraints.

Establishment of the Software Requirements¶

1. Based on the system requirements and competitor analysis, the information is shared between a group consisted of experienced developers related to the issues and the software requirements are begined to be constituted.
2. The first version of the software requirements is reviewed and refined by the Project Leader, system engineer and Technical Coordinator

Table of Requirements versioning¶

When the system and software requirements reaches a certain maturity (this stage is decided by the cooperation of the developers and the demanders of the project), the requirements table should not be changed except for minor corrections and/or unless there is a very unusual situation. If replacement is necessary, it is needed to get the approval of both he developers and the demanders of the project, and the changes could be only presented with the new version of the requirements table.

Prioritization of requirements¶

After filling the requirements table, a process related to prioritization of requirements begins with the participation of project leader, the demander(s) of the project and the team performed the competitor analysis. This process bases the ranking of the stages have to be done to reach the condition that performs the first sentence of the demand in the most basic sense.

Determination of the Project Team¶

Project leader constitutes a team according to the size and the urgency of the software needs together with “Operational Coordinator”

Technical Analysis¶

The steps after this stage are in the responsibility of the team that will carry out the project. The process begins with the technical analysis and decision of the probable operation(s)/algorithm(s) could be applied made by the project team. At the end of this process, it should be expressed definitely the method(s), library or the other application(s) decided to be used, covers the user requirements completely. The requirements table could be used as a check list for this process. The planning should also be done about the module(s) or additional software(s) that will be rewritten or making suitable for use in necessary conditions.

Task Sharing¶

Task-sharing is carried out for projects developed in a modular structure. This sharing process begins with the demands of the project leader and the member(s) of the technical team. After determining the task allocations and the module responsables in a certain matter, they are recorded to the project documentation and the work tracking system.

Work Tracking¶

The project schedule should be updated in each of every technical meetings with the participation of the person or people responsible for coding, if necessary the date shifting or postponing conditions should be comitted to the calendar with the approval of the person or people claiming the project. This processes are performed by the project leader. This calendar should be operated on Pardus Project Tracking System (tracker.pardus.org.tr) and it should be known that the entire development team will continue to the task sharing through this calendar. The responsibility of the unspecified conditions in project tracking system belong to the person who have been on that part of the task. The person/people who demands the project also could get the project status information from the system instantaneously.

Pardus Software Life Cycle and Work Breakdown¶

This section is to learn how we name the sections that make up a software project and how to manage them.

Every software has a lifecycle. To develop a project we will use a balanced combination of Incremental, V Shaped and Prototyping software development life cycles specialized for Pardus.

You can see details from Software Development Life Cycle image.

A Pardus software project is made up of several sections:

Phase¶

Defines the major activities that every project must have. Every Phase is a **sub-project** of the main project. Each phase has a beginning and ending date.

A minimal Pardus software project **must** have the following phases:

1. **Requirements Phase**: Each project starts with the Requirements Phase.

   The requirements phase include:

   - Gathering Requirements
   - Preparing the requirements specifications document (RSD)
   - Requirements Analysis

2. **Design Phase**: The second phase is usually the Design Phase. In some projects there can also be an equipment preparation phase before or after this phase.

   A software design should be detailed so the programmer that translates the design to a programming language does not have to make any decisions while developing the software.

   The design should be supported with UML diagrams such as:

   - Activity Diagrams
   - Interaction Diagrams
   - Class Diagrams
   - Sequence Diagrams

   Software design are divided in two:

   1. **High Level Design**

      A high-level design provides an overview of a solution, platform, system, product, service, or process. The purpose is the show the big picture.

      High level design includes:

      - Architecture
      - Components
      - Interfaces
      - Networks
      - High level data flow
      - Optional: Classes

   2. **Low Level Design**

      A low-level design contains the details of the high level design. At the end of the low level design, the code should be **all but written**.

      Low level design includes:

      - Classes
      - Associations between classes
      - Member and non-member functions
      - Member variables

    - Any other detail that are involved in testing

3. **Implementation Phase**:

   Implementation phase is translating the design to a programming language without making any design decisions.

4. **Testing Phase**:

   Testing phase includes:

   - Unit testing (test cases will be run automatically by **Buildbot**)
   - Alpha tests
   - Beta tests
   - Acceptance test

   Tests can run parallel with the Implementation Phase.

5. **Maintenance Phase**:

   Maintenance Phase starts after the first version of the software (after Beta version) is released.

   Includes:

   - Updating the software
   - Fixing bugs
   - Optimizations
   - Implementing new technologies

Milestone¶

Milestones are **tasks** which defines the major **deliverables** of each phase.

- Milestones should be numbered as “P.M”. P indicates the Phase number that the milstone belongs to and M indicates the milestone number.
- A milestone must have a starting and ending date.
- Milestones should be clear and specific in scope. Which means one must understand what will be required to complete the milestone.
- A milestone should be measurable.
- A milestone can be assigned to an individiual or a group of individuals.

Milestone Template

Milestone 2 of Phase 1 will be written as: 1.2 Module XYZ is completed - (01/01/2011 - 15/02/2011)

Step¶

Defines each **sub-task** to complete a milestone.

- A step must have a starting and ending date.
- A step should be measurable.
- A step can be assigned to an individiual or a group of individuals.

Step Template

Step 2 of Milstone 3 of Phase 5 will be written as: 5.3.2 Module XYZ is completed - (01/01/2011 - 15/02/2011)

A Minimal Pardus Project Example¶

This section tries to explain how to apply the Pardus Software Development Lifecycle.

1. Determine which phases there will be in the project.
2. Requirements phase begins.
   - Gather information about the project requirements.
   - Prepare the Requirements Specification Document.
   - Requirements analysis.
3. Estimate the begining and ending dates.
4. Write down project milestones. These are the highest level milestones like “coding is finished”, “first prototype is released”, “testing is over”, “project is finished”.
   - Create a project schedule.
5. Design phase begins.
   - High level design begins. Estimate the begining and ending date of the high level design. - *Please refer to the High Level Design section*
   - Low level desing begins. Estimate the begining and ending date of the high level design. - *Please refer to the Low Level Design section* - Breakdown the low level design into steps (sub-tasks)
   - Assign each step to an individual or a group with deadlines written.
   - Enter each step to tracker.pardus.org.tr
   - Determine how many prototypes are needed. - Which steps together will make the Prototype 1 (P1) - Which steps together will make the Prototype 2 (P2) - ...
   - According to the step deadlines that build up to make a prototype determine the prototype release date.
   - Determine which prototype will be considered as Alpha release.
   - Prepare the Design Specification Document.
6. Implementation phase begins.
   - Writing unit tests before coding is prefered.
   - According to the design document and Pardus Coding Standarts Document create the overall file and folder structure.
   - Leave unit testing to Buildbot.
   - Each step is reviewed by the **Review Board**
7. The following will be applied when each protoype is ready.

  - Overall testing is done.
  - Requirements compliance tests are done.
  - Prototype review is done.
8. Alpha release is ready.
  - Alpha tests are done. - White Box, Black Box... etc.
9. Beta release is ready.
  - Beta tests are done. - Usability test.. etc.
10. Integration tests are applied.
11. Next version after Beta is out and the project is over.

  - If any tests fail, maintenance is applied.

Equivalent Terms in Pardus Project Tracking System (tracker.pardus.org.tr)¶

As said above, every project should be registered to the Redmine based Pardus Project Tracking System (tracker.pardus.org.tr).

In Redmine the sections above are translated as:

- Phase -> Sub-Project
- Milestone -> Task
- Step -> Sub Task
- Prototype -> Version (Prototypes will be discussed in the next section)

Coding Process¶

Preparation of Standard Software Documents¶

To maintain the standard software development processes and to achieve the coding process through these standards, the reports/documents listed below should be prepared at the beginning of the coding stage and should be included in the project documentation. It should not be forgotten that all the developing process will be done based on these documents. It could be made changes on these documents in need and these documents should be in their own version numbers.

The task of preparing these documents are performed by the specified person/people in the project team and if necessary, by the participation of the demander(s) of the project. As with all other work, the work tracking system should be used for the preparation and development process of these documents. These documents could be shared between the module developers.

- UML Diagram
- Flow Diagram
- Use-Case Diagram
- Object Diagram carried out the case of developer team
- Scenarios determination of Classes and relations between Classes
- Methods, Variables and Parameters In this type of definitions, it could be followed the skeleton code and the certification method can be accessed via this code

- Determination of data fields (Database Selection, Tables, Indexes and Table Relations) The Database requirements specified in the project requirements should be based on.

Defining Classes¶

A detailed class table is prepared for the all functions that are planned to be used at the beginning of the coding process, in terms of this table, API/library design is determined in a flexible way as possible and in accordance with the requirements of the project. The API/library design should be documented besides and it should not be forgotten that it will be the most comprehensive document of the project in technical sense.

Defining Interfaces¶

A similar one of the class table, which is defined for the properties and processes will be used, is prepared for the interfaces, if necessary. This table could include the interfaces that is necessary to cover the requirements specified in the user requirements document, the tasks of these interfaces and in addition of these, a mockup will be prepared for the interface. Furthermore, in the projects that need multi-interface, the interface transitions should be indicated on a flow diagram.

The features that are determined when designinig the interface, should be prepared to fulfill the user requirements. It could be back to user requirements table if possible differences emerge, however this is not a preferred process. It should not be forgotten that the main goal of the interfaces is to fulfill the requirements.

Determination of External Dependencies¶

A report including all libraries will be used as external and their dependencies should be prepared an it should be updated for each change has been done. The content of this report should not be in a conflict with the constraints specified in the requirements table. For example;

> “The product should be able to work in Linux Systems.”

the external dependencies of a project with such a constraint in the above sentence should be able to compatible with the Linux systems.

Testing Process¶

During the coding process, for each module performed; the module’s code is tested primarily by its developer. These tests should be carried out as described in the “Software Tests” document and for each module, its developer’s approval, about the module is carried out before that version, should be gained.

Module tests include the following testing processes. The person/people who will carry out the test procedures should be specified in advance in the project document.

- Unit tests
- Requirement tests
- Installation tests
- Test team's tests
- User group tests

Testing processes described above, also for the system that all modules are combined, performed under the Project leader's control prior to the specific versions/targets.

Determining Versions¶

Accepted Versions¶

In the process of coding, specific target dates are determined for the below version conditions in a way that could be updated and renewed in constantly.

Prototype Version¶

- This version is stored in the code repository in a flexible way; during the development of this version, a code directory structure is used; the detailed information about this structure is indicated in Pardus Coding Standarts document.
- For this version, every developer could create his/her own working directory (branch), this process could be described besides either per developer or per functions will be planned to be carried out in accordance with the requirements of the project
- This version could have a version number (like 1.0, 2.0) in itself. This version numbers are handled independent of the ultimate version numbers.
- The tag mechanism is used effectively, if supported (Git supports), in the storage will be used for the project. This tagging could be carried out for every prototype version. The standarts of the versioning are also placed under the title of Version Numbers.
- In this release, the expectation from the project is primarily fulfill the project initial sentence. Prototype version could be determined as a target version for also some of the detailed requirements introduced based on this sentence.
- Prototype version could reveal clear data in the course of the project and in some circumstances, it could even lead the project to return the process of the project requirements. To determine these kinds of situations, the demander(s) of the project should be included to the possible tests of Prototype Version.
- The API/library or interfaces developed in Prototype Version could be changed completely after "Prototype Presentation" meetings will be held with the participation of the "Technical Coordinator", Test Team and the person/people claiming the project; although they are the basis for final version. However, it should not be forgotten that such conditions affects the project calendar in a negative way.
- Prototype version could not be ready for the end-user and it could include

codes/interfaces that does not meet the interface standards or coding standards. The outputs need to be in the most basic level. Code or user documentations could be deficient.
- Software Development Document is prepared in the prototype version and it is published in “developer.pardus.org.tr”.

Alpha Version¶

- This is the first version released after the adoption of the prototype version and any return cannot be made in the project after the release of this version.
- This version is stored in the code repository prepared in accordance with the Pardus Coding Standarts document. If necessary, code tree inherited from the prototype version could be re-designed as ststed in the document. After the alpha release completed, any changes can be made on the code directory tree structure.
- For this version, every developer could create his/her own working directory (branch), this process could be described besides either per developer or per functions will be planned to be carried out in accordance with the requirements of the project
- This version should get version number together with the main version. Near a version number such as “0.1”, the “a” phrase could be added; like “0.1a”. This version numbers could not be handled indepentent of the ultimate version’s numbers.
- During this release, all the modules planned to be presented in the final version should be determined, its requirements should be extracted and the prototype working should be completed. At the end of alpha version, if a new feature request occurs, a new version is aimed after the launch of final version.
- Alpha version’s requirements come to an end with the completion of all features. In other words alpha phase ends with a feature freeze, indicating that no more features will be added to the software. There may exist bugs.
- This version could be presented to a specific user group against approval of a contract including the risks will be received to get feedback.

Beta Version¶

- Beta Version shows that the last point of the project came to and any feature cannot be added on the way to final version. After beta version, it is not possible to come back to a stage at the requirements level. In such cases, after reveal of the final version, a new version (version number) is aimed.
- The API/library or interfaces developed in the beta version cannot be different from the final version. Beta version is a stabilization phase, so during this version, it is possible to make only bug fixing, translation update or graphics update. For this reason, the Beta Version has great importance for the project.
- In this version, it is expected that all the prototypes developed at alpha version are made into product.
- This version should get version number together with the main version. Near a

version number such as “0.1”, the “b” phrase could be added; like “0.1b”. This version numbers could not be handled indepentent of the ultimate version’s numbers.
- This version could be presented to a specific user group against approval of a contract including the risks will be received to get feedback.
- Beta version should be ready for the user, it cannot include codes/interfaces that does not meet the interface standards or coding standards. Code and user documents should be completed. Such conditions are deterrent conditions for release.
- Software Design Document prepared in the prototype version cannot be updated after the beta version. If an update is necessary, a new version will be developed after the final version should be waited.

Final Version¶

- In this version, the fatal errors detected at the beta release, and improvement errors such as related to translations and visual deficiencies should be resolved. For the errors lead to completely change the substructure and need addition of new features, a new version is aimed after the release of the final version.
- After final version, it is not possible to come back to a stage at the requirements level. In such cases, after reveal of the final version, a new version (version number) is aimed.
- The project’s distribution processes (packaging, repository requirements, dependencies) should be completed with this version.
- This version gets the main version’s number, it does not get any suffix.
- Code, user documentations, installation documentations, technical support documentations and project web page should be completed. Such conditions are deterrent conditions for release.

Version Numbering¶

The version numbering standard will be applied in alpha version and after alpha version is as follows;

(Main Version Number).(Sub Version Number).(Revision Number).(Build Number)

Need to change of Main Version Number is decided according to the content of changes made compared a version before. The addition of a new module is not available in the previous version to the system is a sufficient reason to increase the Sub Version Number. Revision Number could be increased after each change that should be included in the application; for this number, change number in the repository could be based on. Build Number is important for users and test team rather than developers, this number is increased in every condition that the product is packaged and resend to the user/tester. And rather than versioning, it is updated independent of the project, depending on the package management sytem that the packaging process is done and repository conditions is located at.

Success Criteria¶

1. The rate of requirement change after the suspension of the software requirements (%)
2. The number of problem/bug related to the software contained in the Project Progress Reports
3. The number of bugs contained in developer tests
4. The number of bugs contained in the customer approval tests
5. The number of software process deviation during the project (in terms of practices are not carried out)
6. The number of reproduced software work/product

Documentation¶

The code documentation carried out during the coding process belongs to the developer who is responsible for the development of that module. This documentation should be done as described in the Pardus Coding Standarts document.

The preparation of the “Help Documents” needed for presentation of the codes to the users is also included within the business plan of that module’s responsable. Several meetings or co-workings with participation of all the members of the team could be necessary for the “Help Documents” related to all system.

The responsability of the preparation of the documents needed for the installation of the system and making it ready belongs to the development team; however, it could also be applied to suggestions of the person/people demanding the project. The installation requirements will emerge as a result of these documents should be stated in accordance with the constraints defined in project requirements.

All the steps and technical requirements are necessary for the maintenance should be stated in “Maintenance Document” and it should not be forgetten that the target group of this documentation is technical staff.