|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |

**FPT UNIVERSITY**

**CAPSTONE PROJECT DOCUMENT**

**<TNS - TaxiNet System>**

**Report #2 – Software Project Management Plan**

|  |  |
| --- | --- |
| **Group Code** | |
| **Group Member** | SE02461 – Đinh Quang Dương |
| SE02939 – Đỗ Ngọc Tuấn |
| 01657 – Trần Cảnh Quân |
| SE02580 – Trương Hoàng Hà |
| SE02895 – Đào Trung Hiếu |
| **Supervisor** | Nguyễn Văn Sang |
| **Ext. Supervisor** |  |
| **Project Code** | TNS – TaxiNet System |

- Hanoi, 01/2015 -

Table of Contents

[1 INTRODUCTION 3](#_Toc364948076)

[1.1 Purpose 3](#_Toc364948077)

[1.2 Definition and Acronyms 3](#_Toc364948078)

[1.3 References 3](#_Toc364948079)

[2 PROJECT OVERVIEW 3](#_Toc364948080)

[2.1 Project Description 3](#_Toc364948081)

[2.2 Scope 3](#_Toc364948082)

[2.3 Standard Objectives 3](#_Toc364948083)

[2.4 Milestone and Deliverables 3](#_Toc364948084)

[3 PROJECT ORGANIZATION 3](#_Toc364948085)

[3.1 Software Process Model 3](#_Toc364948086)

[3.2 Project lifecycle 3](#_Toc364948087)

[3.3 Roles and Responsibilities 3](#_Toc364948088)

[4 TOOLS AND INFRASTRUCTURES 4](#_Toc364948089)

[4.1 Hardware 4](#_Toc364948090)

[4.2 Software 4](#_Toc364948091)

[5 SCHEDULE 4](#_Toc364948092)

[5.1 Detailed Schedule 4](#_Toc364948093)

[5.2 Meeting Schedule 4](#_Toc364948094)

[5.3 Effort Estimation 4](#_Toc364948095)

[6 RISK MANAGEMENT 4](#_Toc364948096)

[7 CODING CONVENTION 4](#_Toc364948098)

# INTRODUCTION

## Purpose

This document describe shortcuts of our project functional and non-functional requirements. With this document , developer team and other companies can understand throughly about requirements and business case of the project and make it success .

## Definition and Acronyms

|  |  |  |
| --- | --- | --- |
| Acronym & Abbreviation | Definition | Note |
| Rider |  |  |
| Driver |  |  |
| Private Driver |  |  |
| Taxi Company |  |  |
| Admin |  |  |
| Fee Agent |  |  |
| Mobile Tel Comp |  |  |
| Frontend App |  |  |
| Backend App |  |  |

Table 1-1: Definitions and Acronyms

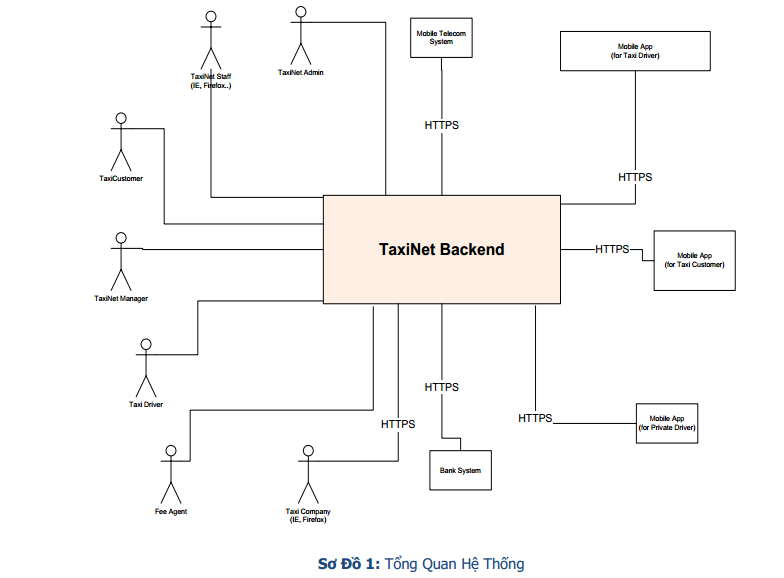
## References

1. Easy Taxi : [http://www.easytaxi.com/vn/](http://www.easytaxi.com/vn/%20%20)
2. Grasp Taxi : [http://grabtaxi.com/ho-chi-minh-city-vietnam/](http://grabtaxi.com/ho-chi-minh-city-vietnam/%20)
3. Uber : [https://partners.uber.com/drive/?utm\_source=uber.com&utm\_campaign=top\_banner](https://partners.uber.com/drive/?utm_source=uber.com&utm_campaign=top_banner%20)
4. Mondo Taxi : [http://www.mondotaxi.com/](http://www.mondotaxi.com/%20)
5. Mangocabs : [http://www.mangocabs.com/book-mango-cabs/](http://www.mangocabs.com/book-mango-cabs/%20)
6. Pasgo Taxi : [http://pasgo.vn/](http://pasgo.vn/%20)

# PROJECT OVERVIEW

## Project Description

This project provides the easiest way to call a taxi for customer with their position. Besides, it helps taxi driver to find customer, accept customer’s request and creates the shortest way to pick up customer and creats a trip with price and locations. Web applications provides management methods for taxi groups and administrator for control users and taxi drivers.



TaxiNet System includes :

* Backend application
* Mobile Application
* Banking Iteration Systems
* Mobile Network Systems
* Customers
* Taxi Drivers/ Private Drivers
* Taxi Groups/Company
* Fee agent
* TaxiNet admin

## Scope

Our project scope is included:

* Developing user requirement and software requirement specification
* Developing architecture and detailed design documents
* Coding and unit test
* Developing test case and execute system test.
* The developing software will have these following functions

TaxiNet System includes :

* Android App :
* **Customers :**

1. Register/login and sign out
2. Call the nearest/suggested/cheapest taxi driver
3. Report a taxi driver if he free but doesn’t accept request
4. Show the list of taxi-drivers around
5. Show the list of taxi-drivers who has the same trip with locations
6. Book a taxi schedule with taxi groups
7. Manage list of favourite taxi drivers
8. Record a history of phone calls and requests
9. Change options about languages and sounds
10. Send feedbacks to taxi groups or TaxiNet admin

* **Taxi Group’s drivers :**

1. Update current status and positions
2. Accept request from customers or taxi group/company
3. Register and fee payment for using services
4. Creats a long trips with price and locations
5. Show customer who send request locations and ways to go
6. Auto notifications about booking events
7. Report customer to administrator
8. Recovery password
9. Change languages and sound options
10. Record phone call and request from customers

* **Private drivers :**

1. Register and payment method
2. Creates long trip with price and location
3. Show customer’s locations and ways to go
4. Change languages and sound options

* Web app :

Web app provides management methods for taxi groups and administrator.

* **Customers :**

1. Haves all functions of mobile app
2. Add Ads register and payment for ads
3. Show histories of trips and payment with time scale
4. Creates a trips and change trip’s status until customer meets perfect driver
5. Send feedbacks to admin or taxigroups
6. Change profile informations

* **Fee Agent:**

1. Search and show the list of all users
2. Charges fee for users
3. Show history of payment with users (customer,drivers,taxi group/company) with time scale

* **Drivers :**

1. Show histories of trips and payments with time scale
2. Accept request from taxi groups
3. Creats a long trip with price and locations and change trip’s status until drivers meet perfect customers
4. Change profile informations
5. Report customer to TaxiNet admin.

* **Taxi Groups/Companies**

1. Manage taxi drivers ( add, change, remove driver’s informations)
2. Manage cars and assign car for taxi drivers’s turn
3. Get request from customers about booking schedule
4. Get feedbacks from customers for services and taxi drivers
5. Manage car park locations ( and, change, remove car park locations’s informations )
6. Creates and register ads and sales
7. Manage taxi prices and fees
8. Payment with Paypal or Banking Systems
9. Show history of trips and payments with customer , time scale and taxi drivers

* **TaxiNet’s admins :**

1. Manage users ( add,change,remove users’s informations)
2. Check documents which users’s upload to system
3. Check and validations user informations ( taxi groups , customers, private drivers )
4. Check and accept ads and sales
5. Get feedback from customers and drivers
6. Show histories of phonecalls, trips, payment of all users with time scale

## Standard Objectives

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metrics | Unit | Committed | Actual | Deviation |
| Start Date | dd-mm-yyyy |  |  |  |
| End Date |  |  |  |  |
| Duration | Elapsed days | 100 days |  |  |
| Max team size | Person | 5 | 5 | 0 |

## Milestone and Deliverables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NO | Stage | Deliver/Milestone | Deliver Date | Inspace | Final | Deliver Location |
| 1 | Inception | Deliver Report 1 |  |  |  | Supervisor |
| 2 | Inception | Project plan |  |  |  | Supervisor |
| 3 | Inception | Deliver Report 2 |  |  |  | Supervisor |
| 4 | Elaboration | User Requirment Specification |  |  |  | Supervisor |
| 5 | Elaboration | Enity Relationship Diagram |  |  |  | Supervisor |
| 6 | Elaboration | Prototype and Mock Up |  |  |  | Supervisor |
| 7 | Elaboration | Software Architecture Design |  |  |  |  |
| 8 | Elaboration | Software Requirement Specification |  |  |  | Supervisor |
| 9 | Elaboration | Deliver Report 3 |  |  |  | Supervisor |
| 10 | Elaboration | Deliver Report 4 |  |  |  | Supervisor |
| 11 | Construction | Complete Coding |  |  |  | Supervisor |
| 12 | Construction | Complete Test and Fix bugs |  |  |  | Supervisor |
| 13 | Construction | Deliver Report 5 |  |  |  | Supervisor |
| 14 | Construction | Deliver Report 6 |  |  |  | Supervisor |
| 15 | Transition | The last Document and CD Source Code |  |  |  | FPT University |
| 16 | Transition | Project Complete |  |  |  | FPT University |

# PROJECT ORGANIZATION

## Software Process Model

Due to the short time of project and the volume of work that we have to do, after discussing, our project team chose RUP as our software project model.

“The Rational Unified Process is a Software Engineering Process. It is a set of principles for software development. It is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2003. RUP is not a single concrete prescriptive process, but rather an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs. RUP is a specific implementation of the Unified Process.” (Wikipedia, 2012, IBM Rational Unified Process, viewed 26 September 2014, [http://en.wikipedia.org/wiki/ Rational\_Unified\_Process](http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process) )

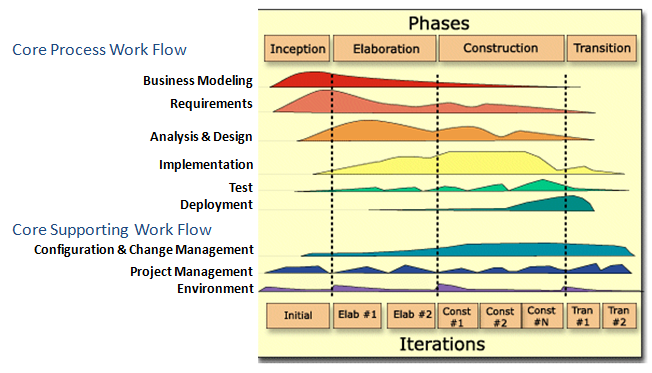


Figure 2-1: Rational Unified Process Model overview

As figure 2-1 above, the RUP model has four phases: Inception, Elaboration, Construction and Transition. Besides, it has six engineering disciplines: Business Modelling, Requirements, Analysis and Design, Implementation, Test, Deployment and three supporting disciplines: Configuration and Change Management, Project Management and Environment. “RUP is based on a set of building blocks and content elements, describing what is to be produced, the necessary skills required and the step-by-step explanation describing how specific development goals are to be achieved. The main building blocks, or content elements, are the following:

* **Roles (who):** defines a set of related skills, competencies and responsibilities of an individual or a group of individuals as a team.
* **Work Products (what):**
* **Task (how):** describes a unit of work assigned to a Role that provides a meaningful result.” (Wikipedia, 2012, IBM Rational Unified Process, viewed 26 September 2014, <http://en.wikipedia.org/wiki/Rational_Unified_Process> )

## Project lifecycle

Project life cycle of RUP model has four phases:

* **Inception Phase:** Inception is the first phase of process. In this startup phase, we should provide business case of the system and determine the scope of project. Besides, we have to create the project management plan that has project schedule, effort estimation and risk management etc. At the end of this phase, we should check the objectives of project and decide whether to continue development or not. Hence, Inception phase must be properly planned and done. Based activities of this phase
* Study business case and feasibility study of project
* Complete draft ERD of system
* Complete draft screen prototypes
* Complete draft requirements
* Determine project scopes
* Complete project management plan
* **Elaboration Phase:** The objectives of this phase are to determine appropriate architectural and construction plan for the project. The architectural decision needs to be made for the entire system, and to describe most of the requirements of system. At the end of this phase, we must examine the objectives and scopes, the choice of architecture and decide whether to proceed to the next phase. Based activities of this phases:
* Complete user requirement specification
* Complete ERD, final prototypes
* Complete Software Requirement Specification
* Complete database model
* Complete System Architecture Design
* **Construction Phase:** Construction is the third phase of RUP lifecycle. In this phase, we must have done all the coding and testing work. After coding, developers will do unit test themselves, then test team will do functional test and regression test when finishing all. Based activities of this phase:
* Complete coding and unit test
* Complete functional and regression test
* Complete user manual
* **Transition Phase:** Transition is the final phase of the RUP lifecycle. In this phase, project team has to deploy the application and give it to users. The next step is receiving feedback from users to identify the problems and then complete the system. Based activities of this phase:
* Deploy the system
* Deliver source code
* Complete all reports and documents

## Roles and Responsibilities

Supervisor

Mr.Sang NV

Team Leader

Duong DQ

Test & QA Team

Develop Team

Design Team

SRS Analysis Team

Hieu DT

Tuan DN

Tuan DN

Quan TC

Ha TH

DuongDQ

Tuan DN

Hieu DT

Quan TC

Ha Th

DuongDQ

|  |  |  |
| --- | --- | --- |
| Member | Roles | Responsibilities |
| Dương ĐQ | Leader  SRS Analysis Team  Dev Web-Leader | 1. Lead team 2. Create WBS 3. Create SRS 4. Review Report from team members 5. Creates Project Source Code package and folder 6. Manage technologies of Web UI 7. Prepare Web enviroments 8. Develop Web app UI and functions 9. Manage Source Code, hard copy version 10. Manage Progress, milestones and deliverables 11. Support other members about technical problem 12. Write report #2 and report #2 |
| Ha TH | SRS Analysis Team  Develop Web-app Team  Web-services Leader | 1. Analysis Business cases and complete SRS 2. Define Business situations and project scope 3. Complete ERD 4. Research project technologies (Hibernate, Restfull WS) 5. Prepare and creates Database enviroments 6. Develop webservice for web app and support web services for android app 7. Support other team members 8. Manage technologies of web services 9. Deliver tested code 10. Deliver software package sources 11. Write report #3 and report #6 |
| Tuan DN | Design Team Leader  Test Team  Developer Team | 1. Design mock up and prototypes 2. Review prototypes with supervisor 3. Manage prototypes 4. Prepare Android App enviroment 5. Manage UI layouts and transform prototypes to Source Code 6. Creates test cases 7. Execute test cases 8. Write report #1 and report #6 9. Research Google map API technologies 10. Develop Android UI 11. Help other team members |
| Quan TC | Design Team  Developer Team | 1. Prepare SR documents 2. Design and review prototypes and mock ups with supervisor 3. Transfrom prototypes to UI code 4. Prepare web enviroment 5. Research JSF/Primefaces technlogies 6. Develop Web UI 7. Write report #4 and #6 8. Review test case 9. Unit tests 10. Help other team members |
| Hieu DT | Developer Team  Technical leader Android App  Test & QA team | 1. Prepare risk management plan 2. Prepare effort estimate 3. Prepare android enviroment 4. Manage test cases and test planss 5. Execute test 6. Develop Android ‘s web services 7. Research Google map API technologies 8. Write report #5 and #6 9. Deliver source code and tested product 10. Review code and technical with supervisor 11. Help other team members |

# TOOLS AND INFRASTRUCTURES

## Hardware

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Hardware** | **Configuration** | **Purposes** |
| 1 | Laptop/Desktop | Core Dual 1.8Mhz or above, 2GB RAM, 100GB Hard disk | Used for development environment |
| 2 | PC Server | Core Dual 1.8Mhz or above, 2GB RAM, 50GB Hard disk | Install database server and JBoss Test server |
| 3 | Android Smartphone | ?, 12GB RAM | Install Android application for testing |
| 4 | iPhone 4 12GB | 12GB RAM | Install iOS application for testing |
|  |  |  |  |
|  |  |  |  |

## Software

1. Android Application

Android application is the mobile app for customer and taxi drivers.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Purposes** |
| 1 | ADT-Android Development Tools |  | Android Tools for development  Source: <http://developer.android.com> |
| 2 | Subclipse | 1.10.4 | Plug-in of eclipse is used to integrate with SVN server. Subclipse is an Eclipse Team Provider for the Subversion version control system.  Source: <http://subclipse.tigris.org/> | |
| 3 | Subversion Client Adapter | 1.10.1 | Plug-in of eclipse is used to integrate with SVN server. Subversion Client Adapter provides a common API for Subversion client functionality.  Source: <http://eclipse.org/subversive/> | |
| 4 | Subversion Revision Graph | 1.1.1 | Plug-in of eclipse is used to integrate with SVN server. Subversion Revision Graph for Subclipse  Source: <http://subclipse.tigris.org/> | |
| 5 | Subversion JavaHL Native Library Adapter | 1.8.8 | Plug-in of eclipse is used to integrate with SVN server  Source: <http://subclipse.tigris.org/> | |
| 6 | Subversion Client Adapter | 1.8.3 | Plug-in of eclipse is used to integrate with SVN server  Source: <http://eclipse.org/subversive/> | |
| 7 | Subclipse Integration for Mylyn 3.x | 3.0.0 | Provides integration features between Subclipse and Mylyn, such as automatic change set management and linking projects to issue tracking systemsbased on svn properties.  Source: <http://subclipse.tigris.org/> | |
|  | CollabNet Merge client | 4.0.2 | The CollabNet Merge client provides powerful Subversion merge capabilities within the Eclipse environment.  Source: <https://desktop-eclipse.open.collab.net/source/browse/desktop-eclipse/> | |
|  | SVNKit Library | 1.8.5.r2xx | SVNKit is a pure java Subversion client library that may be used through its own API or serve as a transparent replacement of native javahl bindings.  Source: <http://org.tmatesoft> | |
| 8 | SQLite Developer Lite |  |  | |

1. Backend Application

Back-end application is a Java web with Pricefaces/JSF, Hibernate , Spring framework and MySQL database. Back-end focus on admin and taxi groups for manage other users’ information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Purposes** | |
| 1 | Eclipse Juno | Juno Service Release 2 | Eclipse Java EE IDE for Web Developers.  Source: <http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/lunasr1> | |
| 2 | Subclipse | 1.10.4 | Plug-in of eclipse is used to integrate with SVN server. Subclipse is an Eclipse Team Provider for the Subversion version control system.  Source: <http://subclipse.tigris.org/> | |
| 3 | Subversion Client Adapter | 1.10.1 | Plug-in of eclipse is used to integrate with SVN server. Subversion Client Adapter provides a common API for Subversion client functionality.  Source: <http://eclipse.org/subversive/> | |
| 4 | Subversion Revision Graph | 1.1.1 | Plug-in of eclipse is used to integrate with SVN server. Subversion Revision Graph for Subclipse  Source: <http://subclipse.tigris.org/> | |
| 5 | Subversion JavaHL Native Library Adapter | 1.8.8 | Plug-in of eclipse is used to integrate with SVN server  Source: <http://subclipse.tigris.org/> | |
| 6 | Subversion Client Adapter | 1.8.3 | Plug-in of eclipse is used to integrate with SVN server  Source: <http://subclipse.tigris.org/> | |
| 7 | Subclipse Integration for Mylyn 3.x | 3.0.0 | Provides integration features between Subclipse and Mylyn, such as automatic change set management and linking projects to issue tracking systemsbased on svn properties.  Source: <http://subclipse.tigris.org/> | |
| 8 | Hibernate Tools | 3.6.0 Final | Hibernate Tools Eclipse Plugin provides development wizards, a query console and a mapping editor for Hibernate 3. Also includes hibernate-tools.jar for use with standalone Ant tools.  Source: <http://tools.jboss.org/downloads/jbosstools/juno/> | |
| 9 | CollabNet Merge Client | 4.0.2 | The CollabNet Merge client provides powerful Subversion merge capabilities within the Eclipse environment.  Source: <https://desktop-eclipse.open.collab.net/source/browse/desktop-eclipse/> | |
| 10 | JBoss Central Community | 1.1.0 | JBoss Central provides a central hub for new and experienced users to get access to quickstarts, projects, documentation resources, news blogs and access to easy installation of recommended additonal plugins. Includes all dependencies to get started with JBoss Central.  Source: <http://tools.jboss.org/downloads/jbosstools/juno/> | |
| 11 | JBoss tools and library |  | JBoss tools from <http://tools.jboss.org/downloads/jbosstools/juno/> | |
| 12 | Spring Dashboard | 3.5.0xxx | This feature provides the STS dashboard for displaying RSS feeds and the extensions page  Source: <http://projects.spring.io/spring-framework/> | |
| 13 | Spring IDE | 3.5.0.xxx | This feature provides integration for Spring IDE with Eclipse AJDT.  Source: <http://projects.spring.io/spring-framework/> | |
| 14 | Eclipse Quick search | 3.5.0.xxx | This feature provides a QuickSearch dialog that can be accessed quickly by a keyboard shortcut.  Source: <http://projects.spring.io/spring-framework/> | |
| 14 | Tomcat Plugin | 3.3.1 | Plugin for Tomcat in Eclipse  Source: <http://marketplace.eclipse.org/content/eclipse-tomcat-plugin> | |
| 15 | My SQL Database server | 5.7 | Database server  Source: <http://www.mysql.com/downloads/> | |
|  |  |  |  |  |

Table 2‑3: Software & Libraries for Web Application runtime

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Purposes** |
| 1 | Spring core | 4.1.2 | Framework for business and data access layer  Source: <http://www.springframework.org> |
| 2 | Hibernate | 4.3.7 | Libraries for Data Object mapping and access data in DB  Source: <http://hibernate.org/> |
| 3 | Log4j | 2.1 | Use for logging of the application  Source: <http://logging.apache.org/> |
| 4 | JSF (Mojarra) | 2.1 | Use for User Interface Development and runtime  Source: <https://javaserverfaces.java.net/download.html> |
| 5 | Mysql-connector-java | 5.1.17 | Use for connecting My SQL  Source: <http://dev.mysql.com/downloads/connector/j/5.0.html> |
| 6 | Jquery | 2.1.1 | Use for user interface development and runtime  Source: <http://jquery.com/> |
| 7 | My SQL Database | 5.7 | Use for Database Server  Source: <http://www.mysql.com/downloads/> |

# SCHEDULE

## Detailed Schedule

<Lịch thực thi của dự án làm bằng Miscrosoft Project>

## Meeting Schedule

Our project team 2 types of meetings :

* Meeting with supervisor : every Tuesday and Thursday from 9.00 am to 11h30am. In this meeting, we report percentages of work we did and discuss for new problem so supervisor will give us advise and references link to solve it.
* Meeting with team members : because of we stay at FU-Hoa Lac so, we meet each other every day. In this meeting, each member will report what he has done last week to others and the problems he meets (if have). All members will discuss about these problems and PM or technical leader will give the final solution for each. Finally, PM will give the tasks for all members to do and report in the next meeting

## Effort Estimation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task name** | **Worst case**  **(days)** | **Best case**  **(days)** | **Most likely**  **(days)** | **Expected case**  **(days)** |
| **Inception** | **84** | **70** | **76** | **76** |
| **Initiation** | 22 | 20 | 21 | 21 |
| **Technologies** | 26 | 20 | 22 | 22 |
| **Requirements** | 27 | 25 | 26 | 26 |
| **Develop Project Plan** | 9 | 5 | 7 | 7 |
| **Elaboration** | **152** | **130** | **135** | **137** |
| **Technology** | 9 | 7 | 8 | 8 |
| **User Requirement**  **Specification** | 7 | 5 | 6 | 6 |
| **System Requirement**  **Specification** | 26 | 20 | 22 | 21 |
| **Requirements&**  **Design** | 40 | 35 | 35 | 38 |
| **Coding** | 49 | 45 | 45 | 45 |
| **Test Document** | 21 | 18 | 19 | 19 |
| **Construction** | **99** | **87** | **91** | **92** |
| **Coding** | 50 | 45 | 45 | 46 |
| **Testing** | 49 | 42 | 46 | 46 |
| **Transition** | **17** | **13** | **15** | **15** |
| **Pre-closing** | 11 | 9 | 10 | 10 |
| **Closing** | 6 | 4 | 5 | 5 |
| **Total** | **352** | **300** | **317** | **320** |

# RISK MANAGEMENT

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Risk** | **Description** | **Root Cause** | **Potential Responses** | **Risk Owner** | **Probability** | **Impact** |
| 1 | Changing in the way of executing | Team members find out the other ways to executing the project, reduce time and cost, such as better algorithm, technology | One or some members find the better way, and they will approve that the new way is better. | Meeting for evaluating, and changing the way to execute the project if it's really better | Project team | 4 | 7 |
| **2** | Over deadline | Some members of team do not finish their tasks before deadline. | Low responsibility in working of members | Finding the reason, evaluating and giving punishment for members who over deadlines, review the estimate duration of their tasks. | Project team | 7 | 3 |
| **3** | Community Risk | Some member conflict, misunderstand each others leads to wrong actions and decessions | Lack of communications skills  Careless when reply email or messages of each others | All members  must be careful in talking with  others and writing  reports or emails.PM assigns  member to write  meeting minute  after meeting or  take note after  discussing about  issues, then send to all team  members | Project team | 4 | 7 |
| **4** | Unrealistics Schedule | Team leaders plan an unrealistics schedule so other team members can not follow to complete their tasks intime | Team leader doesn’t have experiment in creates WBS | Meeting with all team and supervisor to review and creates a plan with discussion and advice from supervisor | Project team | 2 | 3 |
| **5** | Technical risks | Short durations of learning new frameworks | Team members can not complete their task in time make the schedule slow down | Developers have to study hard and make some examples  Team needs some technical supervisor for resolve problems when studying and developng | Project team | 3 | 5 |
| **6** | Change requirements | In development time, team members discover an unrealistic function and want to change it | Team members have to change documents and report to supervisor for advice and recommend | Team members have to review SRS carefully with supervisor | Project team | 3 | 4 |

# CODING CONVENTION

We use java and android for develop so every source code have to obey to the java coding conventions

## Java Language Rules

### Don’t Ignore Exceptions

Sometimes it is tempting to write code that completely ignores an exception like this:

void setServerPort(String value) {  
    try {  
        serverPort = Integer.parseInt(value);  
    } catch (NumberFormatException e) { }  
}

You must never do this. While you may think that your code will never encounter this error condition or that it is not important to handle it, ignoring exceptions like above creates mines in your code for someone else to trip over some day. You must handle every Exception in your code in some principled way. The specific handling varies depending on the case.

### Don’t Catch Generic Exception

Sometimes it is tempting to be lazy when catching exceptions and do something like this:

try {  
    someComplicatedIOFunction();        // may throw IOException   
    someComplicatedParsingFunction();   // may throw ParsingException   
    someComplicatedSecurityFunction();  // may throw SecurityException   
    // phew, made it all the way   
} catch (Exception e) {                 // I'll just catch all exceptions   
    handleError();                      // with one generic handler!  
}

You should not do this. In almost all cases it is inappropriate to catch generic Exception or Throwable, preferably not Throwable, because it includes Error exceptions as well. It is very dangerous. It means that Exceptions you never expected (including RuntimeExceptions like ClassCastException) end up getting caught in application-level error handling. It obscures the failure handling properties of your code. It means if someone adds a new type of Exception in the code you're calling, the compiler won't help you realize you need to handle that error differently. And in most cases you shouldn't be handling different types of exception the same way, anyway.

There are rare exceptions to this rule: certain test code and top-level code where you want to catch all kinds of errors (to prevent them from showing up in a UI, or to keep a batch job running). In that case you may catch generic Exception (or Throwable) and handle the error appropriately. You should think very carefully before doing this, though, and put in comments explaining why it is safe in this place.

Alternatives to catching generic Exception:

* Catch each exception separately as separate catch blocks after a single try. This can be awkward but is still preferable to catching all Exceptions. Beware repeating too much code in the catch blocks.
* Refactor your code to have more fine-grained error handling, with multiple try blocks. Split up the IO from the parsing, handle errors separately in each case.
* Rethrow the exception. Many times you don't need to catch the exception at this level anyway, just let the method throw it.

Remember: exceptions are your friend! When the compiler complains you're not catching an exception, don't scowl. Smile: the compiler just made it easier for you to catch runtime problems in your code.

### Don’t Use Finalizers

Finalizers are a way to have a chunk of code executed when an object is garbage collected.

Pros: can be handy for doing cleanup, particularly of external resources.

Cons: there are no guarantees as to when a finalizer will be called, or even that it will be called at all.

Decision: we don't use finalizers. In most cases, you can do what you need from a finalizer with good exception handling. If you absolutely need it, define a close() method (or the like) and document exactly when that method needs to be called. See InputStream for an example. In this case it is appropriate but not required to print a short log message from the finalizer, as long as it is not expected to flood the logs.

### Fully Qualify Imports

When you want to use class Bar from package foo,there are two possible ways to import it:

1. import foo.\*;

Pros: Potentially reduces the number of import statements.

1. import foo.Bar;

Pros: Makes it obvious what classes are actually used. Makes code more readable for maintainers.

Decision: Use the latter for importing all Android code. An explicit exception is made for java standard libraries (java.util.\*, java.io.\*, etc.) and unit test code (junit.framework.\*)

### Write Short Methods

To the extent that it is feasible, methods should be kept small and focused. It is, however, recognized that long methods are sometimes appropriate, so no hard limit is placed on method length. If a method exceeds 40 lines or so, think about whether it can be broken up without harming the structure of the program.

### Define Fields in Standard Places

Fields should be defined either at the top of the file, or immediately before the methods that use them.

### Limit Variable Scope

The scope of local variables should be kept to a minimum. By doing so, you increase the readability and maintainability of your code and reduce the likelihood of error. Each variable should be declared in the innermost block that encloses all uses of the variable.

Local variables should be declared at the point they are first used. Nearly every local variable declaration should contain an initializer. If you don't yet have enough information to initialize a variable sensibly, you should postpone the declaration until you do.

One exception to this rule concerns try-catch statements. If a variable is initialized with the return value of a method that throws a checked exception, it must be initialized inside a try block. If the value must be used outside of the try block, then it must be declared before the try block, where it cannot yet be sensibly initialized:

// Instantiate class cl, which represents some sort of Set   
Set s = null;  
try {  
    s = (Set) cl.newInstance();  
} catch(IllegalAccessException e) {  
    throw new IllegalArgumentException(cl + " not accessible");  
} catch(InstantiationException e) {  
    throw new IllegalArgumentException(cl + " not instantiable");  
}  
  
// Exercise the set   
s.addAll(Arrays.asList(args));

But even this case can be avoided by encapsulating the try-catch block in a method:

Set createSet(Class cl) {  
    // Instantiate class cl, which represents some sort of Set   
    try {  
        return (Set) cl.newInstance();  
    } catch(IllegalAccessException e) {  
        throw new IllegalArgumentException(cl + " not accessible");  
    } catch(InstantiationException e) {  
        throw new IllegalArgumentException(cl + " not instantiable");  
    }  
}  
  
...  
  
// Exercise the set   
Set s = createSet(cl);  
s.addAll(Arrays.asList(args));

Loop variables should be declared in the for statement itself unless there is a compelling reason to do otherwise:

for (int i = 0; i < n; i++) {  
    doSomething(i);  
}

and

for (Iterator i = c.iterator(); i.hasNext(); ) {  
    doSomethingElse(i.next());  
}

### Order Import Statements

The ordering of import statements is:

1. Android imports
2. Imports from third parties (com, junit, net, org)
3. java and javax

To exactly match the IDE settings, the imports should be:

* Alphabetical within each grouping, with capital letters before lower case letters (e.g. Z before a).
* There should be a blank line between each major grouping (android, com, junit, net, org, java, javax).

Originally there was no style requirement on the ordering. This meant that the IDE's were either always changing the ordering, or IDE developers had to disable the automatic import management features and maintain the imports by hand. This was deemed bad. When java-style was asked, the preferred styles were all over the map. It pretty much came down to our needing to "pick an ordering and be consistent." So we chose a style, updated the style guide, and made the IDEs obey it. We expect that as IDE users work on the code, the imports in all of the packages will end up matching this pattern without any extra engineering effort.

This style was chosen such that:

* The imports people want to look at first tend to be at the top (android)
* The imports people want to look at least tend to be at the bottom (java)
* Humans can easily follow the style
* IDEs can follow the style

The use and location of static imports have been mildly controversial issues. Some people would prefer static imports to be interspersed with the remaining imports, some would prefer them reside above or below all other imports. Additionally, we have not yet come up with a way to make all IDEs use the same ordering.

Since most people consider this a low priority issue, just use your judgement and please be consistent.

### Use Spaces for Indentation

We use 4 space indents for blocks. We never use tabs. When in doubt, be consistent with code around you.

We use 8 space indents for line wraps, including function calls and assignments. For example, this is correct:

Instrument i =  
        someLongExpression(that, wouldNotFit, on, one, line);

and this is not correct:

Instrument i =  
    someLongExpression(that, wouldNotFit, on, one, line);

### Follow Field Naming Conventions

* Non-public, non-static field names start with m.
* Static field names start with s.
* Other fields start with a lower case letter.
* Public static final fields (constants) are ALL\_CAPS\_WITH\_UNDERSCORES.

For example:

public class MyClass {  
    public static final int SOME\_CONSTANT = 42;  
    public int publicField;  
    private static MyClass sSingleton;  
    int mPackagePrivate;  
    private int mPrivate;  
    protected int mProtected;  
}

### Use Standard Brace Style

Braces do not go on their own line; they go on the same line as the code before them. So:

class MyClass {  
    int func() {  
        if (something) {  
            // ...  
        } else if (somethingElse) {  
            // ...  
        } else {  
            // ...  
        }  
    }  
}

We require braces around the statements for a conditional. Except, if the entire conditional (the condition and the body) fit on one line, you may (but are not obligated to) put it all on one line. That is, this is legal:

if (condition) {   
    body();   
}

and this is legal:

if (condition) body();

but this is still illegal:

if (condition)  
    body();  // bad!

### Limit Line Length

Each line of text in your code should be at most 100 characters long.

There has been lots of discussion about this rule and the decision remains that 100 characters is the maximum.

Exception: if a comment line contains an example command or a literal URL longer than 100 characters, that line may be longer than 100 characters for ease of cut and paste.

Exception: import lines can go over the limit because humans rarely see them. This also simplifies tool writing.

### Use Standard Java Annotations

Annotations should precede other modifiers for the same language element. Simple marker annotations (e.g. @Override) can be listed on the same line with the language element. If there are multiple annotations, or parameterized annotations, they should each be listed one-per-line in alphabetical order.<

Android standard practices for the three predefined annotations in Java are:

* @Deprecated: The @Deprecated annotation must be used whenever the use of the annotated element is discouraged. If you use the @Deprecated annotation, you must also have a @deprecated Javadoc tag and it should name an alternate implementation. In addition, remember that a @Deprecated method is still supposed to work.

If you see old code that has a @deprecated Javadoc tag, please add the @Deprecated annotation.

* @Override: The @Override annotation must be used whenever a method overrides the declaration or implementation from a super-class.

For example, if you use the @inheritdocs Javadoc tag, and derive from a class (not an interface), you must also annotate that the method @Overrides the parent class's method.

* @SuppressWarnings: The @SuppressWarnings annotation should only be used under circumstances where it is impossible to eliminate a warning. If a warning passes this "impossible to eliminate" test, the @SuppressWarnings annotation must be used, so as to ensure that all warnings reflect actual problems in the code.

When a @SuppressWarnings annotation is necessary, it must be prefixed with a TODO comment that explains the "impossible to eliminate" condition. This will normally identify an offending class that has an awkward interface. For example:

// TODO: The third-party class com.third.useful.Utility.rotate() needs generics   
@SuppressWarnings("generic-cast")  
List<String> blix = Utility.rotate(blax);

When a @SuppressWarnings annotation is required, the code should be refactored to isolate the software elements where the annotation applies.

### Treat Acronyms as Words

Treat acronyms and abbreviations as words in naming variables, methods, and classes. The names are much more readable:

| Good | Bad |
| --- | --- |
| XmlHttpRequest | XMLHTTPRequest |
| getCustomerId | getCustomerID |
| class Html | class HTML |
| String url | String URL |
| long id | long ID |

Both the JDK and the Android code bases are very inconsistent with regards to acronyms, therefore, it is virtually impossible to be consistent with the code around you. Bite the bullet, and treat acronyms as words.