

Final Presentation

Software Engineering 2 Project

M. Redaelli F. Zanolì

Politecnico di Milano

February 25, 2017

- ① Introduction
- ② Requirement Analysis and Specification
 - Overview
 - UML Diagrams
 - Alloy
- ③ Design
 - Architectural Design
 - User Interface Design
- ④ Integration Test Plan
 - Overview
 - Integration Sequence Diagrams
- ⑤ Project Plan
 - Plan Contents
 - Cost Models
 - Tasks Scheduling

Introduction

Requirement
Analysis and
Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration
Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

Outline - Introduction

1 Introduction

2 Requirement Analysis and Specification

Overview

UML Diagrams

Alloy

3 Design

Architectural Design

User Interface Design

4 Integration Test Plan

Overview

Integration Sequence Diagrams

5 Project Plan

Plan Contents

Cost Models

Tasks Scheduling

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

Introduction

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

The project we have been assigned is called *PowerEnjoy* and it is a complex software system that should implement a car sharing service. In order to rationalize, clarify, and put in structured and standardized documents all the relevant concepts and informations, we designed and delivered several documents such as the **RASD**, the **DD**, the **ITPD**, and the **PPD**. These slides will only present an overview of the concepts thoroughly described in the above mentioned documents.

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

We composed the documents we had to using some tools such as:

- **TexStudio:** to compile \LaTeX document.
- **StarUML:** to draw UML diagrams.
- **Alloy Analyzer 4.2:** to checking model consistency.
- **Draw.io:** to build mockups.
- **GitHub:** for storing the project.
- **Skype:** for team collaboration

Outline - Requirement Analysis and Specification

1 Introduction

② Requirement Analysis and Specification

Overview

UML Diagrams

Alloy

3 Design

Architectural Design

User Interface Design

4 Integration Test Plan

Overview

Integration Sequence Diagrams

5 Project Plan

Plan Contents

Cost Models

Tasks Scheduling

1 Introduction

② Requirement Analysis and Specification

Overview

UML Diagrams

Alloy

3 Design

Architectural Design

User Interface Design

4 Integration Test Plan

Overview

Integration Sequence Diagrams

5 Project Plan

Plan Contents

Cost Models

Tasks Scheduling

Overview

Introduction

Requirement Analysis and Specification

Overview

UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration

Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

The aim of the software is to provide a new digital management system for car-sharing service that exclusively uses electrical cars. It can be applied to different small and big city and even in a large urban area.

This new service pretends to achieve various goals, such as:

- **G5:** Allow user logged in to see the reservation's confirmation and the time of expiration
- **G6:** A user who request a rent can abort the process whenever he/she wants
- **G9:** A non registered users can only register once to the service.
- **G10:** A user can get discount or overtaxes from his/her last rent.

Actors - TUsers, Cars and Customer Service

Below are listed the three main actors that will interact with the application once deployed:

- **Registered User:** A person who desires to subscribe to the system to benefit the full functionalities of the application.
- **Visitors:** A person that needs to move from a position to another one among the city and wants to use *PowerEnjoy* in order to do so, but has not registered yet to the service.
- **Car:** The car reserved / rented from a User that communicates with the system.
- **External Customer Service:** A team that provide technical support in case of problems related to the car.

Product Perspective

Introduction

Requirement Analysis and Specification

Overview UML Diagrams Alloy

Design

Architectural Design User Interface Design

Integration Test Plan

Overview Integration Sequence Diagrams

Project Plan

Plan Contents Cost Models Tasks Scheduling

Our *PowerEnJoy* is a **completely new product**, not based on previous ones.

It relies on **location data** received via **Internet** from each user application and car system: all the involved smartphones already have a **GPS antenna** installed inside, that communicates their position to the service.

Being a partially **distributed application**, *PowerEnJoy* requires a fully operative **Internet** connection in order to work properly, both on server and client side: **no service is intended to be provided offline**.

All the data generated by this software are stored in a database, accordingly to current normative and laws about privacy and personal data management.

In addition, several **APIs** are provided in order to allow further improvements and expansions of the software: for instance, the payment will be managed from an external entity (we supposed **PayPal**) and the map will be provided by **Google**.

- 1 Introduction
- 2 Requirement Analysis and Specification
 - Overview
 - UML Diagrams
 - Alloy
- 3 Design
 - Architectural Design
 - User Interface Design
- 4 Integration Test Plan
 - Overview
 - Integration Sequence Diagrams
- 5 Project Plan
 - Plan Contents
 - Cost Models
 - Tasks Scheduling

Introduction

Requirement
Analysis and
Specification

Overview

UML Diagrams

Alloy

Design

Architectural
Design

User Interface
Design

Integration
Test Plan

Overview

Integration
Sequence
Diagrams

Project Plan

Plan Contents

Cost Models

Tasks
Scheduling

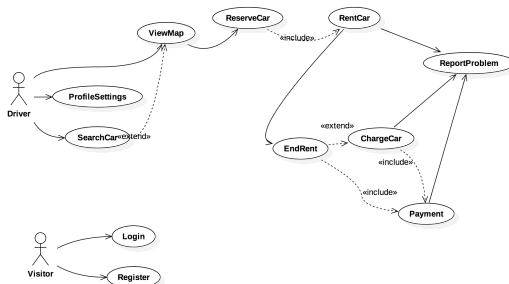
UML Diagrams

We provided a variety of UML diagrams, each type having a different purpose.

- **UML Use Case:** Shows the identified **use cases** in relation with the **involved actors**.
- **UML Sequence Diagram:** Indicates, for a given **use case**, the **interaction** between the **actors involved** and the **system**.
- **UML Class diagram:** Points out the different **software entities** involved in the application and the **relationships** between them.

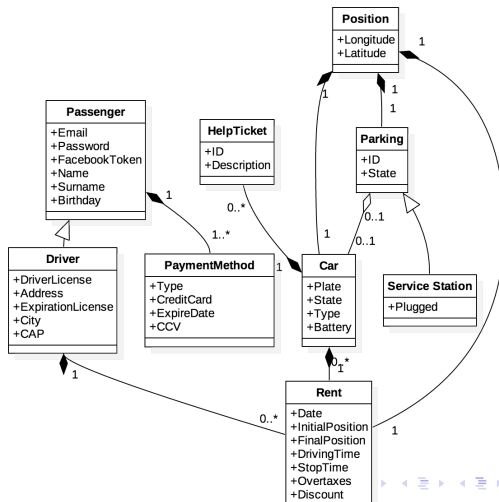
UML Use Case Diagram

This is perhaps the most useful diagram that can be designed in the early phase of the development of a software project.



UML Class Diagram

Furthermore we designed a class diagram for an early evaluation of the basic software components that consists in a sort of **Model** for *PowerEnjoy*.



② Requirement Analysis and Specification

UML Diagrams

Alloy

Architectural Design

User Interface Design

Overview

Integration Sequence Diagrams

Plan Contents

Cost Models

Tasks Scheduling

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

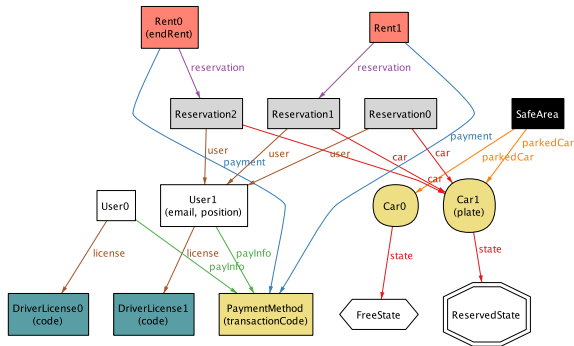
Plan Contents
Cost Models
Tasks
Scheduling

Alongside the **UML Class Diagram** we built **Alloy Models** using the **Alloy** modeling language with the help of **Alloy Analyzer 4.2**.

The tool didn't find a proof of the inconsistency of our **Alloy Models**, and that along with the **Automatic Generation** (and **Manual Verification**) of interesting worlds, made us aware of the **Consistency** of those **Models** within a reasonable level of confidence.

Alloy Simple World

Here is an example of one among the **simplest world** we generated and double checked using both **Alloy Analyzer 4.2** and **manual checking**.



- ① Introduction
- ② Requirement Analysis and Specification
 - Overview
 - UML Diagrams
 - Alloy
- ③ Design
 - Architectural Design
 - User Interface Design
- ④ Integration Test Plan
 - Overview
 - Integration Sequence Diagrams
- ⑤ Project Plan
 - Plan Contents
 - Cost Models
 - Tasks Scheduling

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

- The **Client-Server** style is used to give the many Clients connected to the Server the opportunity of sending different requests (e.g. a **Car Reservation** or **Open Car Request**).
- The **Point to Point** bidirectional communication channel is made necessary to enable the Server the delivery of various messages and requests to the Clients and the Car:
 - Generic notifications
 - Service messages
 - Internal message to the car as Open or Check the status

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

The selected software architecture follows the principles of the **Model View Controller** architectural pattern, therefore three main software components have been identified and those are:

- The **View**, the user application View that is designed to perform comfortable user experience

- The **Controller**, split in two families of components with specialized functionalities:
 - **Networking Components Family:** that is in charge of grouping the **Communication Components** and messages to the various Views, following the logic implements in the Business Components Family.
 - **Business Components Family:** In this family are included all Processing requests and either **synchronous responses** or **asynchronous events**.
- The **Model** that guarantees a high level interface to store and manage all the *PowerEnJoy* relevant data and an abstraction of **Relational Database** in a software component that is in direct connection with the **Controller**

In particular the tiers are:

- The **Client**, the mobile application designed and projected
- The **Application Server**, the main core of the system that include **Controller** and **Model** into it
- The **DB Server**, the data storage of the *PowerEnJoy*' data.
- The **Car System**, that is the interface between the physical system and the software, furnished by sensor and display in order to communicate with the user.

Component View

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

Several components has been designed to provide all the functionalities needed for *PowerEnJoy* to work. Many subsystems have been identified:

- Ride Manager,Bill Manager,Zone Manager...
- View
- Controller

Component View - UML Component Diagram

Introduction

Requirement Analysis and Specification

Overview UML Diagrams Alloy

Design

Architectural Design

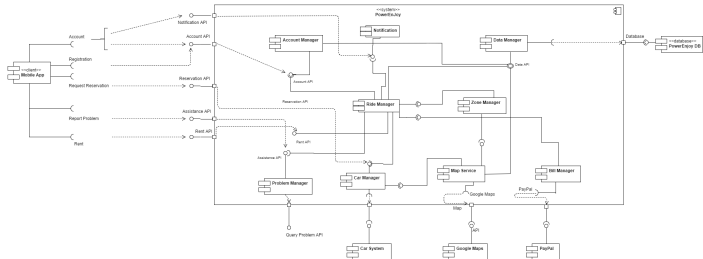
User Interface Design

Integration Test Plan

Overview Integration Sequence Diagrams

Project Plan

Plan Contents Cost Models Tasks Scheduling



Deployment View

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

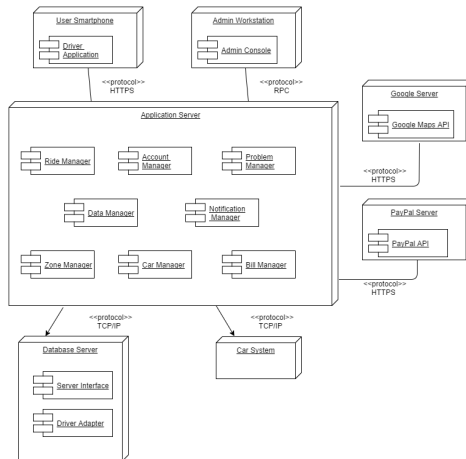
Plan Contents
Cost Models
Tasks
Scheduling

The best way found to deploy the software components identified, is to consider 5 different nodes (7 if considering the Google Server and the PayPal server) that correspond with the **Tiers** with in addition the **External Extentions**

Deployment View - UML

Deployment Diagram

The following diagram shows how **software components** are mapped into the **physical system**.



Here are proposed some of the most meaningful **UML Sequence Diagrams** with respect to show how software components interacts. The chosen functionalities are:

- Find a FreeCar
- Make a Reservation

There are other functionalities whose **UML Sequence Diagram** is not reported here for space and time constraints:

Find a FreeCar

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural Design

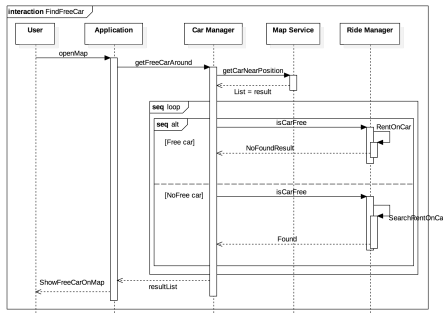
User Interface
Design

Integration Test Plan

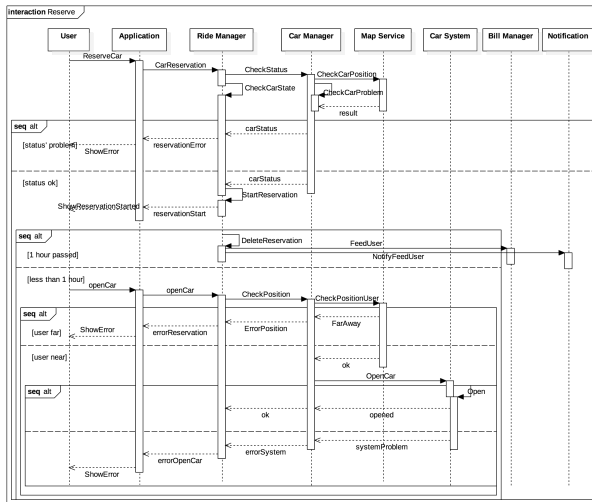
Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling



Make a Reservation



Other design decisions - Internet and Firewalls

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural Design

User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

Tasks Scheduling

GUI Design

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

In this section we provide the **most important and meaningful mockups** for every class of screens we have designed.

Car Info Mockups



Reservation Mockups



End Rent Mockups



Outline - Integration Test Plan

1 Introduction

2 Requirement Analysis and Specification

Overview

UML Diagrams

Alloy

3 Design

Architectural Design

User Interface Design

4 Integration Test Plan

Overview

Integration Sequence Diagrams

5 Project Plan

Plan Contents

Cost Models

Tasks Scheduling

Introduction

Requirement
Analysis and
Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration
Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

- ① Introduction
- ② Requirement Analysis and Specification
 - Overview
 - UML Diagrams
 - Alloy
- ③ Design
 - Architectural Design
 - User Interface Design
- ④ Integration Test Plan
 - Overview
 - Integration Sequence Diagrams
- ⑤ Project Plan
 - Plan Contents
 - Cost Models
 - Tasks Scheduling

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

Scope and Approach

This project phase is highly based on the **Design** one.

We will clearly state the order in which the software components identified in the **Component View** of the **Design** part have to be integrated one with each other in order to guarantee a well tested final software.

The **bottom-up integration testing approach** has been chosen, because for a medium sized project like *PowerEnjoy*, it is best to proceed step by step in a careful yet coherent integration strategy.

Entry Criteria

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

Before starting the integration testing of any software component that has been designed for *PowerEnJoy* system, few points have to be underlined:

- The **internal functions** of the considered component must be **unit tested** using an appropriate framework.
- We suppose that **Google Maps API and PayPal API** are well tested by **Google and PayPal** and thus we can use them without testing any further.

We have considered all the **Subsystems**. and we had divided into 3 different categories, in particular we divided the system based on the pattern used to code it.

- **Model:** All the components presented into the application server.
- **Controller:** The only difference from a higher point of view, is that in this case the **Controller** is tested on the basis of the already tested Model using **Views** actions.
- **Views:** The application is tested above the already tested **Controller** and **Model** using **UI** testing automators or manual testing.

Tasks Scheduling

Convention adopted - Blocks

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

- **Green:** This block is not dependent on any lower level component in *PowerEnJoy* and therefore it is integrated as a starting point in the current diagram.
- **Red:** This block is going to use some Drivers in order to perform a complete test on all its functionalities
- **Yellow:** This block is going to be tested without Drivers because it's at the end of the process.
- **Blue:** This block will be not tested.

System Integration Sequence

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

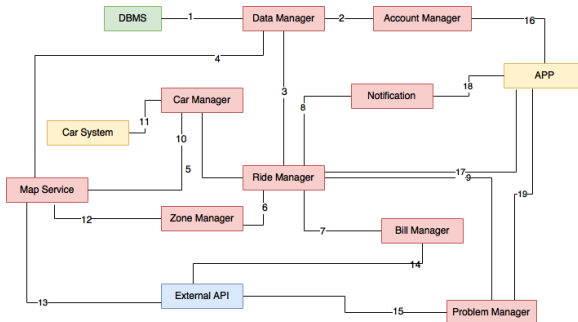
Architectural
Design
User Interface
Design

Integration Test Plan

Overview
**Integration
Sequence
Diagrams**

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling



Subsystem Integration Sequence

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration

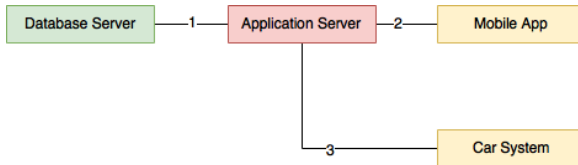
Test Plan

Overview

**Integration
Sequence
Diagrams**

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling



Program Stubs And Data Required

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
**Integration
Sequence
Diagrams**

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

In conclusion there is the need for some sample data to be in the **Database** and some sample **GPS data** are needed.

Tasks Scheduling

- ① Introduction
- ② Requirement Analysis and Specification
 - Overview
 - UML Diagrams
 - Alloy
- ③ Design
 - Architectural Design
 - User Interface Design
- ④ Integration Test Plan
 - Overview
 - Integration Sequence Diagrams
- ⑤ Project Plan
 - Plan Contents**
 - Cost Models
 - Tasks Scheduling

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

The **Project Plan** consists in tables, **Gantt diagrams**, charts and natural language descriptions of the planning, scheduling and management of *PowerEnJoy* development.

In order to estimate the project effort, we followed the assumption that the **dimension of the software** can be characterized by correlating **the kind of functionalities offered** with **the source lines of code (SLOC) of the software itself**

- ① Introduction
- ② Requirement Analysis and Specification
 - Overview
 - UML Diagrams
 - Alloy
- ③ Design
 - Architectural Design
 - User Interface Design
- ④ Integration Test Plan
 - Overview
 - Integration Sequence Diagrams
- ⑤ Project Plan
 - Plan Contents
 - Cost Models
 - Tasks Scheduling

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

The **Function Points approach**, defined in 1975 by Allan Albrecht:

- Consists in a technique to assess the effort needed to design and develop custom software applications.
- Correlates the kind of functionalities offered with the source lines of code of the software itself.

This technique consists in combining the following program characteristics to obtain a final result:

- **Internal Logic Files:** Data used and managed by the application
- **External Logic Files:** Data used by the application but generated and maintained by other applications.
- **External Input:** Elementary operations to elaborate data coming from the external environment.
- **External Output:** Elementary operations that generate data for the external environment, and they usually include the elaboration of data
- **External Inquiry:** Elementary operations that involve input and output, without significant elaboration of data.

Function Points Summary

Introduction

Requirement Analysis and Specification

Overview
UML Diagrams
Alloy

Design

Architectural
Design
User Interface
Design

Integration Test Plan

Overview
Integration
Sequence
Diagrams

Project Plan

Plan Contents
Cost Models
Tasks
Scheduling

All the calculated FP_i sums up to FP , which is the total Function Points value:

$$\begin{aligned} FP &= FP_{ILF} + FP_{ELF} + FP_{EI} + FP_{EO} + FP_{EIQ} \\ &= 80 + 10 + 35 + 16 + 8 \\ &= 149 \end{aligned}$$

COCOMO II - Parameters

Results

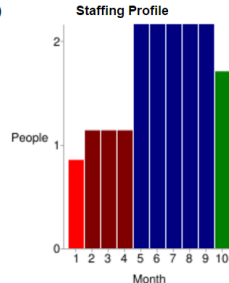
Software Development (Elaboration and Construction)

Effort = 16.4 Person-months
Schedule = 9.2 Months
Cost = \$24660

Total Equivalent Size = 7897 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.0	1.2	0.9	\$1480
Elaboration	3.9	3.5	1.1	\$5918
Construction	12.5	5.8	2.2	\$18742
Transition	2.0	1.2	1.7	\$2959



Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.5	1.2	0.3
Environment/CM	0.1	0.3	0.6	0.1
Requirements	0.4	0.7	1.0	0.1
Design	0.2	1.4	2.0	0.1
Implementation	0.1	0.5	4.2	0.4
Assessment	0.1	0.4	3.0	0.5
Deployment	0.0	0.1	0.4	0.6

Your output file is http://csse.usc.edu/tools/data/COCOMO_January_20_2017_16_05_48_617569.txt

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu

COCOMO II - Results



COCOMO II - Constructive Cost Model

Software Size

Sizing Method Function Points

Unadjusted

Function 149

Language Java

Points

Software Scale Drivers

Precedentedness Nominal

Architecture / Risk Resolution Nominal

Process Maturity High

Development Flexibility High

Team Cohesion Very High

Software Cost Drivers

Product

Required Software Reliability Nominal

Data Base Size Low

Product Complexity High

Developed for Reusability High

Documentation Match to Lifecycle Needs Nominal

Personnel

Analyst Capability Nominal

Programmer Capability High

Personnel Continuity High

Application Experience Nominal

Platform Experience Nominal

Language and Toolset Experience High

Platform

Time Constraint Nominal

Storage Constraint Nominal

Platform Volatility Low

Project

Use of Software Tools High

Multisite Development Nominal

Required Development Schedule Nominal

Maintenance Off

Software Labor Rates

Cost per Person-Month (Dollars)

Calculate

Tasks Scheduling

Task	Description	Completed?
T1a	RASD - Writing	Yes
T1b	RASD - Presentation	Yes
T2a	DD - Writing	Yes
T2b	DD - Presentation	Yes
T3a	ITPD - Writing	Yes
T3b	ITPD - Presentation	Yes
T4a	PPD - Writing	Yes
T4b	Final Presentation	Yes
T5	Implementation	No
T6	Unit Testing	No
T7	Integration Testing	No
T8	System Testing	No
T9	User Acceptance - Alpha Testing	No
T10	User Acceptance - Beta Testing	No

Gantt Diagram

