

**Software Requirements Specification**

**For**

**Fitness/Weather App**

**Prepared by Group G**

**Team Fitness**

**November 11, 2018**

***Copyright © 2002 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.***

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table of Contents ..........................................................................................................................** | | | **ii** |
| **Revision History ...........................................................................................................................** | | | **iii** |
| **1.** | **Introduction ..............................................................................................................................** | | **1** |
| 1.1 | | Purpose ............................................................................................................................................. | 1 |
| 1.2 | | Document Conventions .................................................................................................................... | 1 |
| 1.3 | | Intended Audience and Reading Suggestions .................................................................................. | 1 |
| 1.4 | | Project Scope ................................................................................................................................... | 1 |
| 1.5 | | References ........................................................................................................................................ | 1 |
| **2.** | **Overall Description ..................................................................................................................** | | **2** |
| 2.1 | | Product Perspective .......................................................................................................................... | 2 |
| 2.2 | | Product Features .............................................................................................................................. | 2 |
| 2.3 | | User Classes and Characteristics ..................................................................................................... | 3 |
| 2.4 | | Operating Environment .................................................................................................................... | 3 |
| 2.5 | | Design and Implementation Constraints .......................................................................................... | 3 |
| 2.6 | | User Documentation ........................................................................................................................ | 3 |
| 2.7 | | Assumptions and Dependencies ...................................................................................................... | 3 |
| 2.8 Budget……………………………………………………………………………………………...4 | | |  |
| **3.** | **System Features .......................................................................................................................** | | **4** |
| 3.1 | | Project Features ................................................................................................................................ | 4 |
| 3.2 | | Use Cases ......................................................................................................................................... | 7 |
| 3.3 | Architectural Style………………………………………………………………………………..15 | |  |
| **4.** | **External Interface Requirements .........................................................................................** | | **16** |
| 4.1 | | User Interfaces ............................................................................................................................... | 16 |
| 4.2 | | Hardware Interfaces ....................................................................................................................... | 16 |
| 4.3 | | Software Interfaces ........................................................................................................................ | 16 |
| 4.4 | | Communications Interfaces ........................................................................................................... | 16 |
| **5.** | **Other Nonfunctional Requirements .....................................................................................** | | **16** |
| 5.1 | | Performance Requirements ............................................................................................................ | 16 |
| 5.2 | | Safety Requirements ...................................................................................................................... | 16 |
| 5.3 | | Security Requirements ................................................................................................................... | 17 |
| 5.4 | | Software Quality Attributes ........................................................................................................... | 17 |
| **6.** | **Other Requirements ..............................................................................................................** | | **17** |
| **Appendix A: Glossary..................................................................................................................** | | | **17** |
| **Appendix B: Analysis Models .....................................................................................................** | | | **17** |
| **Appendix C: Issues List ...............................................................................................................** | | | **17** |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Sameh Heinen |  | Use Case 1 | 1.1 |
| Steven Moore-Vountas | Oct 14 | Use Case 2  Domain Model | 1.1 |
| Edouard Theoux |  | Use Case 3 | 1.1 |
| Hagop Avakian |  | Use Case 4 | 1.1 |
| Anastasiia Drozdova | November 2 | Use Case 5 | 1.1 |

**1. Introduction**

**1.1 General**

This document puts together the software requirements specifications of Fitness app that is implemented for an Android device as a part of COMP 354: Introduction to Software Engineering. The goal of this fitness app is to be connected to Endomondo to be able to retrieve information and correlate it with some weather data. The main idea behind this application is to provide the user with the information about what days would be good to exercise, depending on the weather and the user’s past data.

**1.2 Deadlines**

|  |  |
| --- | --- |
| Date | Summary |
| 14/10/2018 | Data Extraction - approach |
| 4/11/2018 | Iteration # 1 - Fitness App |
| 3/12/2018 | Iteration #2 - Fitness/Weather App |

**1.3 General Constraints**

**Task 1 requirements (fitness)**

● Fitness is tracked with Endomondo (separate application)

● Fitness data is must be scraped from Endomondo’s JSON data (requires login info)

● Only activity is cycling

● Analyze average speed and/or total time

● Display data points as a chart

**Task 2 requirements (weather)**

● analyze temperature and wind speed

● Weather from April to October/November only

● Weather must be collect for Montreal or nearby region

● Display data points as a chart

**Task 3 requirements (integration)**

● Application login capabilities

● Analyze patterns between fitness and weather data

● Predict if today is a favorable day for fitness according to weather forecast

● Does performance go up or down after breaks?

**Eventual tasks**

● Documentation (follow given template/sample)

● Price estimation

**Notes**

● Preferably Android O

● Use of libraries/APIs allowed

● Data is stored on the device, not on the cloud

● No encryption required

● Use 10/20 days moving averages for data

**1.4 Cost**

We estimate the app implementation of the task 1 (fitness app) to be \_\_\_\_\_\_\_.

We used the constructive cost model to approximate the total cost to develop the first task.

The estimation details are showed below.

Based on expert judgment from Dr. Rilling, we believe the project will be of organic size, with very little innovation, and with a size of 500 lines. Since this is not a very big project, the deadlines are spread out enough so that we consider that they are not so tight : we have a period of two months from start to finish.

To estimate the Fitness app project, the combination of bottom-up estimation and an algorithmic mode, more precisely Boehm’s COCOMO will be used.

The Boehm’s COCOMO consists of evaluating the

COCOMO organic small project

Effort = a\*size^b\*c

a = 2.4

b = 1.05

size = 1

c = adjustment factor

Is there anything about this being a team project that drives the adjustment factor?

effort = 2.4 person/months, therefore, split into 5 people, it would be half a month? (edited)

so 80 hours per person?

the average hours per month is 174

(174 hours/month) / (2.4 person/month) = 72.5 person-hours

A Person-Month is one month of effort by one person. In the olden days, a Person-Month would have been called a Man-Month or a Staff-Month. In standard COCOMO, there are exactly 152 hours per Person-Month

ok found that

152hours/month\*2.4 person-months = 364.8 person-hours

364.8/5 = 72.96 hours per person! About 73 hours per person to get this project done... WOAH!

If everyone could calculate how many lines of code we would need for their own use case, we could have an even better estimation.

**1.5 References**

**2. Overall Description**

**2.1 Product Perspective**

The product is being developed for ProfitRUS, which would like to introduce a new UML drawing tool, extended from the Violet UML Editor open source project, for developers. The new UML tool will implement the following features previously non-existant within Violet UML:

Login: will ask the user for a username and password in order to view statistics.

Statistics: will provide statistics for specific diagrams

Constraints: diagrams will have to follow rules as outlined for UML diagrams

**2.2 Product Features**

**Login:** 1. Create new login

1. Login as existing user
2. Access statistics

**Statistics:** 1. Display pie chart showing size distribution of classes

1. Display pie chart showing number of outgoing messages per object
2. Calculate averages for statistics
3. Display window with all statistics from other groups
4. Display warnings

**Constraints:** 1. No empty activation bars in Sequence diagrams

1. No return messages from objects that have been called
2. Identify GRASP control pattern
3. Classes cannot have multiple different recursive relationships
4. No bi-directional composite or aggregation relationships between two classes
5. Notifications of problems
6. Login required to view statistics

**2.3 User Classes and Characteristics**

The users of the ProfitsFromUML drawing tool will vary from Software Engineering students to Software Engineering professionals. We will categorize the users of the UCM drawing tool as **Power Users, Intermediate Users** and **End Users**. The **Power User**is highly educated,experienced and has a high level of technical expertise. He/She is most likely a professional Software Engineer possibly a leader of a group of Software Engineers working on a project. He/She will have the highest security/privilege level and will use the product more than any other user. Therefore, the Software Engineer is the favored user class of the UCM drawing tool. The **Intermediate User**is educated, has some technical expertise and some experience. He/She is aworking Software Engineering professional or Graduate Student, and will have some privileged access. He/She will use the product some of the time. The **End User** has low or no technical expertise and low privileges. Possibly a student or customer, he/she will use the product very minimally and just for reference.

**2.4 Operating Environment**

The operating environment will be Microsoft Windows 10, running on the x64 hardware platform. Eclipse IDE Version 4.6.1 (Neon) will be used. Other software components include Eclipse Java Development Tools, Eclipse Plug-In development environment, etc.

**2.5 Design and Implementation Constraints**

Java will be used to implement ProfitsFromUML. For the visualization aspect of the project, the user will have the ability to read statistics, display pie charts showing size distribution of classes in the class model, and the ability to display pie charts showing the number of outgoing messages per object in the sequence diagram. The visualization will also calculate averages for statistics. A display window will also be shown with all statistics from other groups. Warnings will also be displayed: High CBO for the class model, and large number of outgoing messages from an object for the sequence diagram.

**2.6 User Documentation**

The user documentation standard we will use is **List or Reference**. For further documentation regarding the base software, the user can refer to software documentation for Violet UML Editor [2], which this software is based on.

1. <http://en.wikipedia.org/wiki/Software_documentation>

**2.7 Assumptions and Dependencies**

The Feature class is composed of 3 sub-classes. The User class is composed of 3 sub-classes.

The object oriented method of inheritance is assumed here.

Other assumptions and dependencies:

The class diagram contains all features present in Violet UML Editor.

The development requires the Microsoft Windows operating system.

There will be 3 user types: Power, Intermediate and End.

There will be limited access to the statistical results.

We will likely used swing/awt libraries to generate pie charts and tables.

**2.8 Budget**

2.8.1. Visualization Budget

We estimate our budgetfor the visualization part of the project will be **$7532.76**. We used constructive cost model to calculate the total cost during developing the extended version of the Violet UML Editor. The estimation details are shown below.

Based on visualization part specification, we believe our responsible part is organic, the size of it is small with very little innovation, and the deadlines is not very tight. So, for the equation Effort = ∗ ∗ , we pick = 2.4, = 1.05.We estimated the total lines of code will be around 1000 lines. We figured this out by estimating every single small function instead of the whole system that our domain model has, then add all the estimates up. For variable c, it based on the following 15 criteria/factors. After multiplying this criteria/factors together, we get c = 0.653885011. We also estimate that the visualization

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  | Rating | | | |  |  |  |  |
| Cost Drivers |  | Very | | Low |  | Normal | | |  |  | High | |  | Very | | Extra |
|  |  | low | |  |  |  |  |  |  |  |  |  |  | High | | High |
| Product Attributes |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Required software reliability | 0.75 | |  | 0.88 |  | 1.00 | |  |  | 1.15 | |  | 1.40 | |  | / |
| Database size | / | |  | 0.94 |  | 1 |  |  |  |  | 1.08 |  | 1.16 | |  | / |
| Product complexity |  | 0.70 |  | 0.85 | 1.00 | | |  |  |  | 1.15 |  | 1.30 | |  | 1.65 |
| Computer Attributes |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Execution time constraint | / | |  | / |  | 1.00 | |  |  | 1.11 | |  | 1.13 | |  | 1.66 |
| Main storage constraint | / | |  | / |  | 1.00 | |  |  | 1.06 | |  | 1.21 | |  | 1.56 |
| Virtual machine Volatility | / | |  | 0.87 |  | 1.00 | |  |  | 1.55 | |  | 1.30 | |  | / |
| Computer turnaround time | / | |  | 087 |  | 1.00 | |  |  | 107 | |  | 1.15 | |  | / |
| Personnel Attributes |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Analyst capabilities | 1.46 | |  | 1.99 |  | 1 |  |  |  | 0.86 | |  | 0.71 | |  | / |
| Applications experience | 1.29 | |  | 1.13 |  | 1 |  |  |  |  | 0.91 |  | 0.82 | |  | / |
| Programmer capability | 1.42 | |  | 1.17 |  | 1.00 | |  |  |  | 0.86 |  | 0.70 | |  | / |
| Virtual machine experience |  | 1.21 |  | 1.10 |  | 1.00 | |  |  | 0.90 | |  | / | |  | / |
| Programming language experience |  | 1.41 |  | 1.07 | 1.00 | | |  |  |  | 0.91 |  | 0.82 | |  | / |
| Project Attributes |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use of modern programming | 1.21 | |  | 1.10 |  | 1.00 | |  | 0.91 | | |  | 0.82 | |  | / |
| practices |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use of software tools | 1.21 | |  | 1.10 | 1.00 | | |  | 0.91 | | |  |  | 0.83 |  | / |
| required development schedule | 1.23 | |  | 1.08 | 1.00 | | |  | 1.04 | | |  |  | 1.10 |  | / |

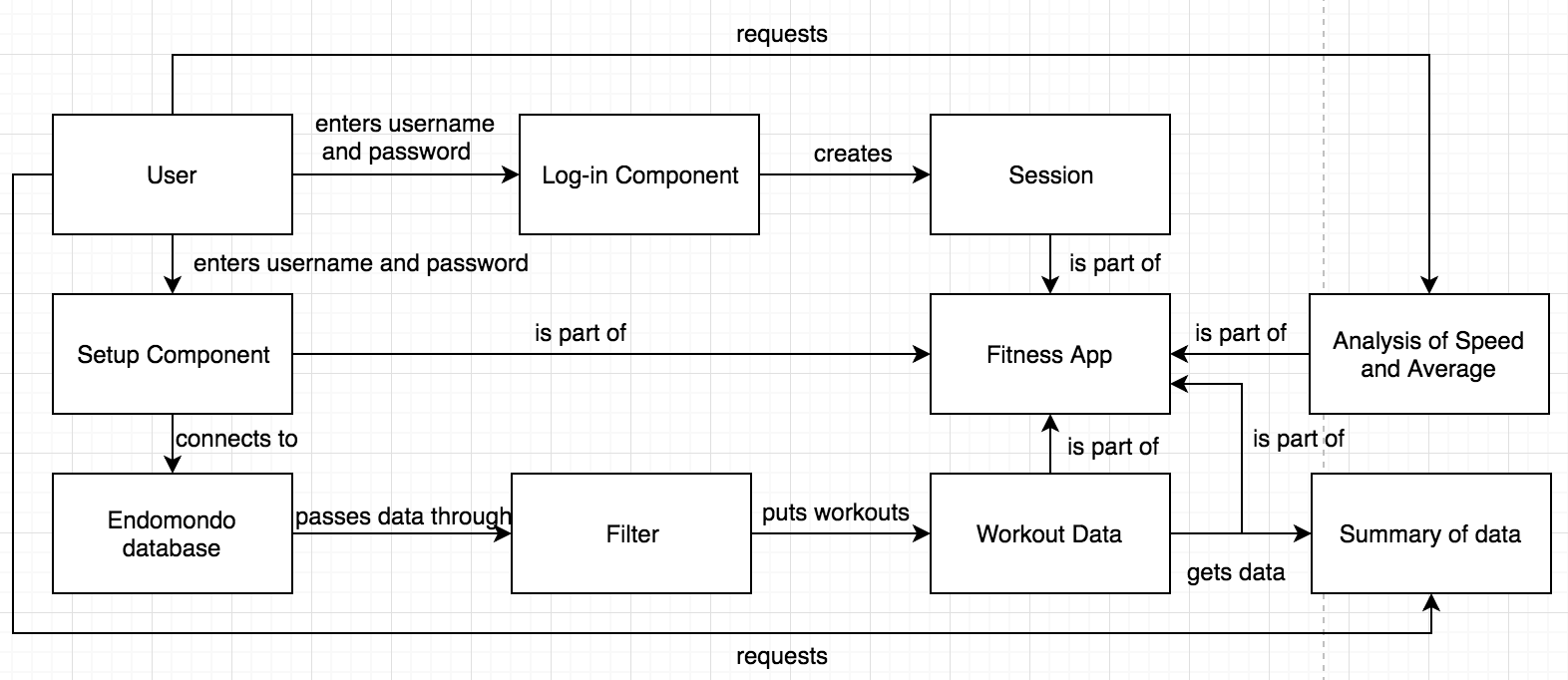
part of the project will be completed within two months. Thus, since the monthly salary is 4800, the total estimated cost for our responsible part is:2.4 ∗ 11.5 ∗ 0.653885011 ∗ 4800 = 7532.76.

**3. System Features**

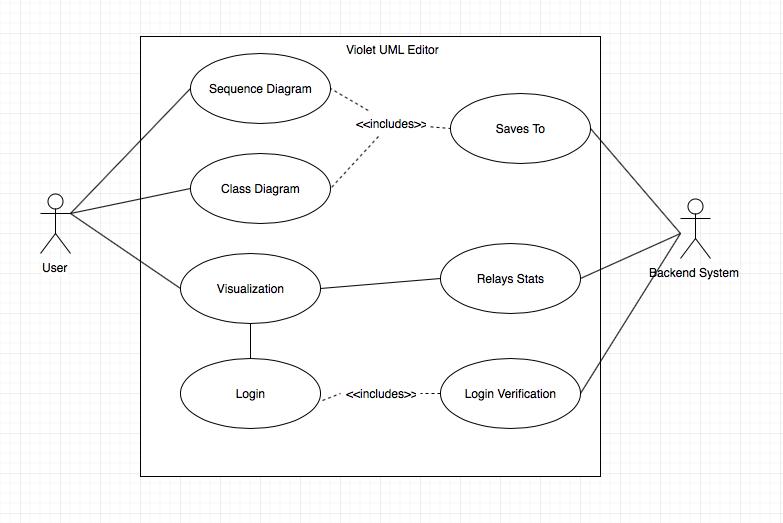
ProfitFromUML has a central goal of offering its user(s) reliable and readable statistics. These statistics will be presented graphically for the user, in an intuitive manner.

**3.1 Project Features**

*3.1.1 System Domain Model*



*3.1.3Use Case Diagram*



**3.2 Use Cases**

A few use case scenarios will be presented, each with a full dressed scenario, a system sequence diagram, and a diagram including GRASP patterns.

**3.2.1. Use Case 1 – Log-In**

Fully Dressed Scenario

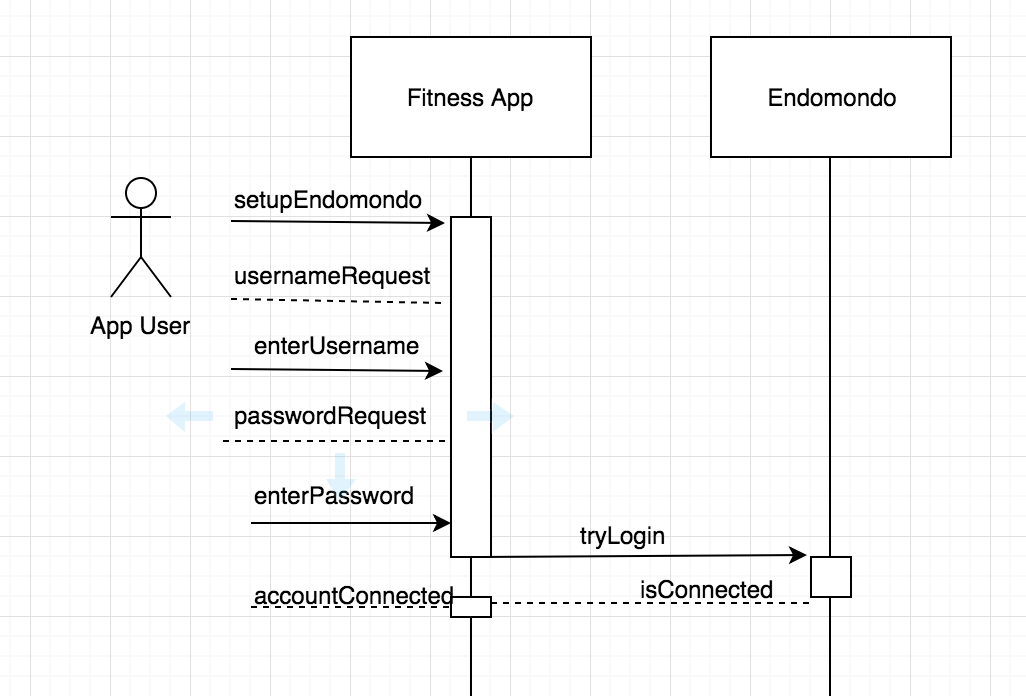
System Sequence Diagram

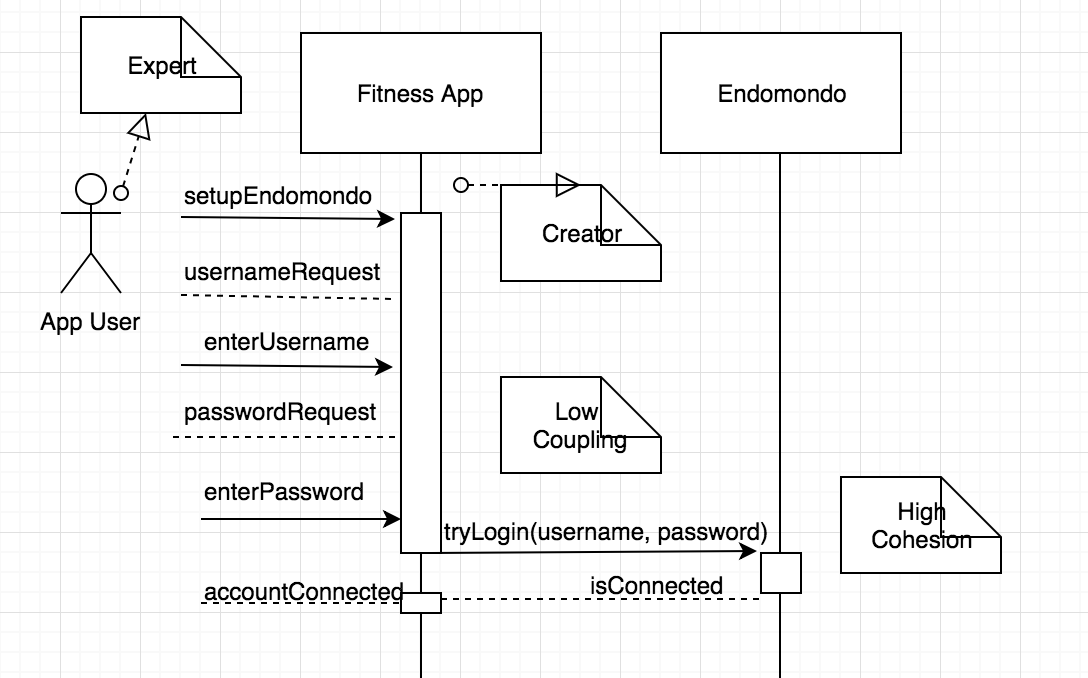
Sequence Diagram with GRASP Patterns

**3.2.2. Use Case 2 –Setup**

|  |  |  |
| --- | --- | --- |
| **Number** | 2 | |
| **Name** | Connect to Endomondo | |
| **Summary** | Enter the username and password and fetch information from the external database | |
| **Priority** | Very High | |
| **Preconditions** | User has logged into the Fitness application | |
| **Postconditions** | Application is connected to Endomondo and is ready to upload data | |
| **Primary Actor(s)** | User | |
| **Secondary Actor(s)** | Endomondo User Database | |
| **Trigger** | User wants to connect Endomondo account to Fitness Application | |
| **Main Scenario** | **Step** | **Action** |
|  | 1 | User logins to the Fitness Application |
|  | 2 | Selects the Endomondo Setup |
|  | 3 | Enters username and password |
|  | 5 | Clicks Enter to connect to Endomondo |
| **Extensions** | **Step** | **Branching Action** |
|  | 3.1 | No Endomondo account associated to username or password |
|  | 3.2 | Message shows: ‘Enter a valid username/password, or go to <https://www.endomondo.com/forgotpassword> if forgot password’ |
| **Open Issues** |  | none |
|  |  |  |

Fully Dressed Scenario

System Sequence Diagram

Sequence Diagram With GRASP Pattern

**3.2.3. Use Case 3 – Filter**

Fully Dressed Scenario

System Sequence Diagram

Sequence Diagram with GRASP Patterns

**3.2.4. Use Case 4 – Summary**

Fully Dressed Scenario

System Sequence Diagram

Sequence Diagram with GRASP Patterns

**3.2.5. Use Case53 – Statistics**

Fully Dressed Scenario

System Sequence Diagram

Sequence Diagram with GRASP Patterns

**3.3 Architectural Style**

For the visualization portion of the Violet UML editor, the batch sequential architectural style will be used. This style allows data to be processed sequentially by separate programs. Each step will be run to completion before the data is processed by the subsequent step. This seemed like the most logical way to perform the visualization. The visualization will begin by receiving data in the form of a text file, from either the the Sequence diagram, or the Class model diagram. After deciding which of the two components the data came from, it will begin to process the data, producing the necessary tables and pie charts to display the data. This will all be done sequentially; each step must wait for the previous to finish before continuing.

The pipe-and-filter architecture style was also considered for the visualization due to its sequential nature of data processing. However, in this style, the data stream is constant, which is not necessary for the visualization – data will be given in a single shot. For the visualization, it is important that only one program is handling the data at once.

To produce the final visuals, which will likely include a table and pie charts, external libraries will be used. We will implement graphical classes (swing, awt), as well as a library that will allow us to create pie charts (JFreeChart). Using these libraries will save us a considerable amount of time.

**4. External Interface Requirements**

**4.1 User Interfaces**

*<To be determined>*

**4.2 Hardware Interfaces**

ProfitFromUML will be Java based developed and will be cross platform supported under the Java Virtual Machine (JVM).

**4.3 Software Interfaces**

ProfitsFromUML wis developed under Java 8 and will require a compatible Java Runtime Environment (JRE) to be installed on the running machine.

The visualization will also feature an export to image file (.jpeg) which will create and image of the statistics visualization. An image viewer is required to open the images of the visuals.

**4.4 Communications Interfaces**

*<To be determined>*

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

ProfitFromUML is a real - time application, thus the computation needs to minimize response time. All features directly related to diagram such as the ones presented in the uses cases need to be performed instantly.

**5.2 Safety Requirements**

As a productivity software product, no special safety measure is necessary.

**5.3 Security Requirements**

Individual diagrams are tied to a specific account. The user must be logged in to access and edit files.

**5.4 Software Quality Attributes**

Software development focus will primarily be on the correctness of the product and if time allows, the ease of testing.

**6. Other Requirements**

No other requirements are to be noted for this project.

**Appendix A: Glossary**

*<To be determined>*

**Appendix B: Issues List**

No issues to report as of Now.