Decision Tree Engine App

Project Plan

Group 3:

Valerio Lucantonio, Julieth Castellanos,Nathan Chape,Tamara Dancheva,Anna Enbom,Eric Johansson, Niklas Sjöqvist

1 Introduction

Decision Tree App Engine is the project assigned to our group, group 3, for Software Engineering Project Teamwork course at Mälardalens Högskola. The purpose of this project is to create an application that allows who want to make an investments, in different fields, to evaluate through “tree surveys” how much he/she is sure about the investment. End users should get from the survey statistics on how many of the questions answered that ”approve” an investment, how many that ”don’t approve” and how many that still are ”uncertain”. The application must offer also to an admin the possibility to create these surveys.
This document will go through a presentation of the group, our organization and used tools, planned effort and deliverables, quality assurance (section 2), then the technical section about the applications: background, architecture, functional and non-functional requirements, traceability validation and verification (section 3).

2 Project Organization

Project manager

Valerio Lucantonio has been assigned this position and will therefore take a leading role within the team to help coordinate tasks and to help initiate decision making. He will also function as the groups main spokesperson towards the steering group.

Customer relations

Anna Enbom volunteered for this position and has since been responsible for communication with the client and for setting up meetings with the client.

Github maintenance

Tamara Dancheva is responsible for setting up and maintaining the groups GitHub repositiory.

Development team

All project members are part of the development team. The purpose of the team is to design, implement and test the application. They are all responsible for assigning themselves tasks, ask for help from other members when they’re in need and to actively participate in the project. All previously listed members and Julieth Castellanos, Nathan Chape, Eric Johansson and Niklas Sjöqvist will be part of the development team.

2.1 Group Organization and tools

Communication within the group is done through email and the facebook message system. The group shares documents between each other on Google-drive and three different backlogs will be stored on ASANA[5]. One for tasks that are to be implemented in the application which are done on an individual level. Another for team coordination containing tasks that affects the entire team and a final one which is used to create and plan meetings and meeting agendas. These backlogs are divided into separate folders. Code is shared through Github's online services and documents that are to be shared with the client are done so by using dropbox.

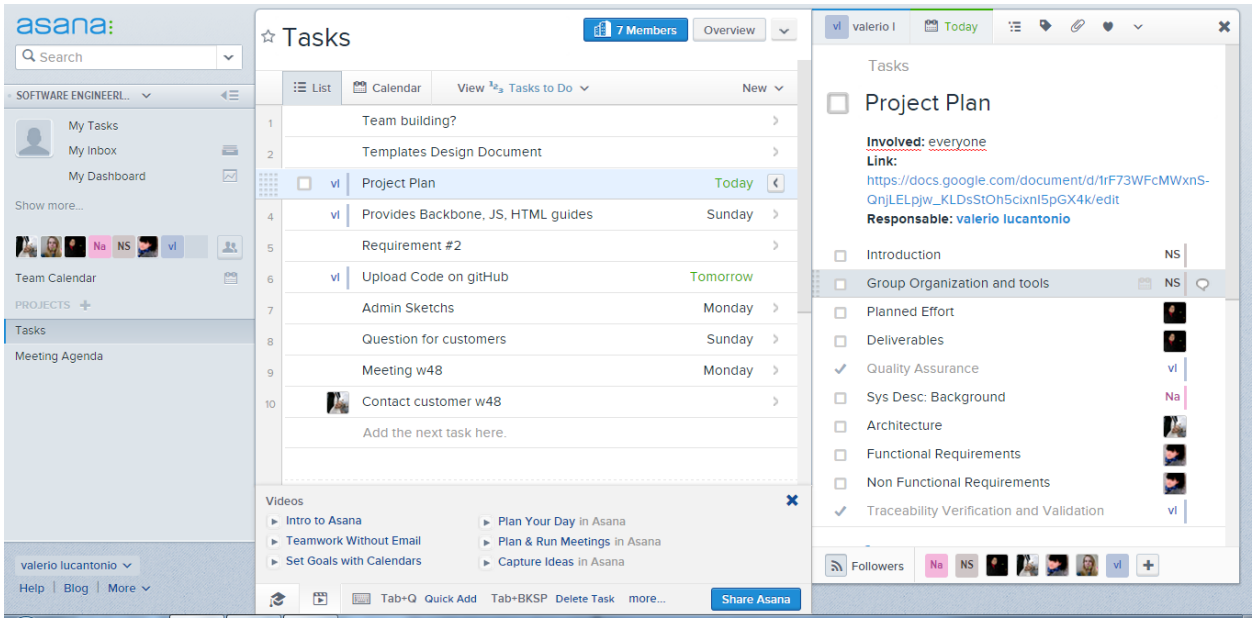


Fig.1: Asana backlog: Organization of this document

Meetings are organized and planned through the use of ASANA[5] which, in turn, is linked to our respective e-mail. When needed, meetings can also be planned through the facebook message system where the group has created a chat room for quick communication.

Work hours and results are to be presented verbally or in text on the weekly meeting when the presentations for steering group are constructed. If a member is unable to do so they should ahead of time notify a team member (preferably the project manager) about this and give them the information required for them to present their work hours and results.

Backbone[3] and Cordova[4] are both planned to be used to develop the application. The team will be using pair programming[6] as often as possible to simplify the process of divisioning the

labour. During the implementation process it's highly likely that a few members will take respectively leading roles in different fields reflecting their areas of knowledge such as user testing, prototyping and continual design.

2.2 Planned effort

The criterion used to estimate the cost or effort required to design, code and test the software is called Function Point (FP) metric. Function points are derived "using empirical relationships based on countable measures of software's information domain and qualitative assessments of software complexities". The information domain value is taken from the Data Flow Model shown in Appendix A. The computing function points are given by the following table, taken from [1]

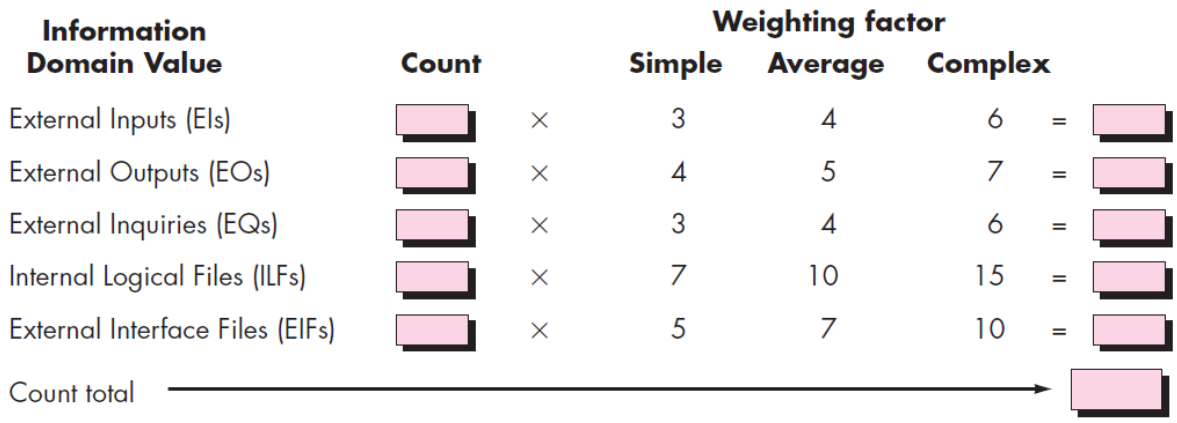


Fig.2:Computing function points

The data, along with the appropriate complexity is:

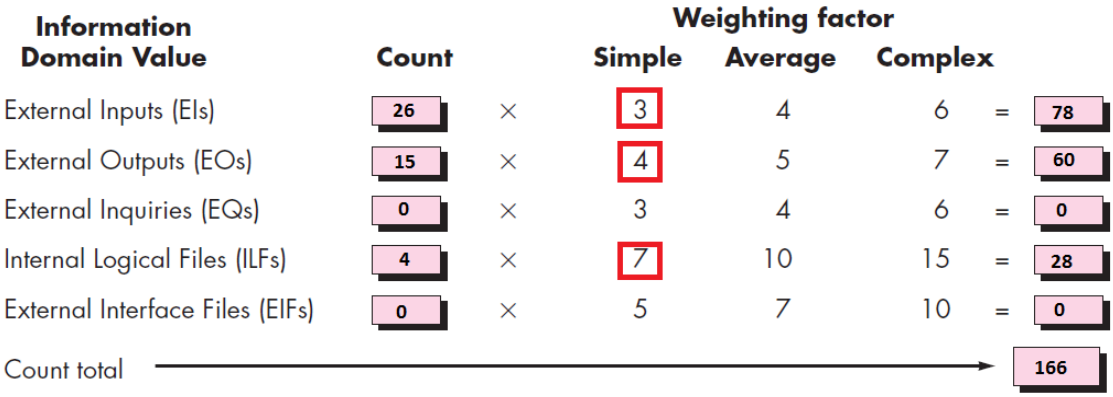


Fig 3: Function points for the software "Generic decision tree engine"

The ”Count total” value represents the number of function points that the implemented program is going to have in average. This number has to be adjusted using the equation:

$$FP = Count\ total\ x[0.65 + 0.01x\sum(F_i)]$$

where Fi is a number according to the criteria given in Appendix B

$$FP = 166\ x[0.65 + 0.01x(12)]$$
$$FP = 166\ x[0.72] = 119.52$$

So, the total amount of FPs is estimated in 119,52. We can round this number to 120.

The project group is composed by 7 people, which decided collectively to work in all the phases of the project equally. It means that we can get the number of FPs produced by every person, using the next formula:

$$Effort\ per\ person = \frac{Total\ FP}{Ammount\ of\ people}$$

$$Effort\ per\ person = \frac{120}{7} = 17,14 \simeq 17$$

Assume that past data indicates that the average of FPs produced for each person-week is 3,5. So, we can calculate the number of weeks that the team is going to take designing, coding and testing the software:

$$Number\ of\ weeks = \frac{17}{3,5} = 4.85 \simeq 5$$

So, 5 is the number of weeks (aproximadamente) required to design, code and test the “Generic decision tree engine”.

The project involves other activities that are listed in the following table:

Activity	Vol (person-week)	Number of weeks required	Comments
Project Management and Documentation (It includes Power point presentations to the steering group and group presentations, as well as deliverable and user documentation)	7	1	• Combined efforts reduce cost
Meetings (It includes meetings with the steering group, presentations with the group	7	0,5	• The people who belong to the group are going to be in all presentations, with some exceptions*
Requirements	7	1	• The client have a clear picture of the product required
Designing, coding and testing	7	5	• Important to associate this tree activities in order to get knowledge of the product and be fast in the development • Being novices means almost everyone must study new technology
Total weeks working		7,5	

*Valerio Lucantonio is not going to be in the city during weeks 1 and 2, but he is going to work in distance. So he is going to miss the final presentation.
*Eric Johansson is not going to be in the city during weeks 51,52 and 1, but he is going to work in distance. So, he is going to miss the meetings with the steering group in the corresponding weeks.

Fig.4: Main activities of the project

The 7,5 weeks required to carry out the project are represented in the following way:

Participant	Week									Total
	46	47	48	49	50	51	52	1	2	
Julieth Castellanos	10	20	20	20	20	20	0	20	20	150
Nathan Chape	10	20	20	20	20	20	0	20	20	150
Tamara Dancheva	10	20	20	20	20	20	0	20	20	150
Anna Enbom	10	20	20	20	20	20	0	20	20	150
Eric Johansson	10	20	20	20	20	20	0	20	20	150
Valerio Lucantonio	10	20	20	20	20	20	0	20	20	150
Niklas Sjöqvist	10	20	20	20	20	20	0	20	20	150
Total hours required in the project										1050

Fig.5: Working hours of the group

2.3 Deliverables

Course deliverables	**Date**
Project plan	20/11/2014
Design description (first version)	4/12/2014
First prototype	4/12/2014
Test specification	18/12/2014
Design description (final version)	8/12/2014
Final prototype	8/12/2014

Client deliverables	**Date**
Requirements document (first version)	20/11/2014
First prototype	4/12/2014
Final prototype	8/12/2014
User Manual	12/12/2014

Activities and dates are correlated in the Gantt Diagram, shown in appendix C

2.4 Quality Assurance

In this analysis quality assurance should be guaranteed from three points of view:

- Product quality: we will ensure the quality of the product formally for what concerns the engine and his algorithm, while for the front-end we will have several meeting with the customers, in which we will expose mock ups and sketches of the applications in order to get more feedbacks from the real users.
- Deliverables quality : we will ensure the quality of deliverables using our previous knowledge acquired during other courses of software engineering. We will use documentations referring to some templates from previous courses.

- Process development quality: we are trying to achieve a good quality of the process development by having at least two meetings for week, and trying to reach always agreements on the most important problems. Moreover, we are using several tools (Asana, Google Doc, Dropbox) to share all the possible information to let all the group members' aware of the current status of the project. In this course the time has a very crucial roles, and usually time and quality are divergent. In order to respect deadlines and to deliver all the required materials we are organizing documents with Asana assigning the task to one among us that will be the responsabile and dividing the sections of the document and assigning them among us. For the implementation we will make three sub-groups of two people implementing the same feature and all the features will be managed in a product backlog in asasa with the same pattern of the document.

3 System Description

3.1 Background

As stated in the introduction, our goal is twofold:

- to develop a web based application that allows an administrator to easily create *decision trees* (*see* Appendix A) for various applications, e.g. Economic analysis, Industry and Production, Automation… etc.
- and secondly, to develop a mobile application that allows a end users to access these trees, and answer some questions about investments in their respective domains. The application should be available for Android and IOS devices.

The application for the administrator will be deployed for one administrator, and it will be a web application (probably running locally). Instead the application for end users will be android and ios applications.

3.1.1 Existing System

The client has stated that they have, internally, developed an excel based application that they use to develop decision trees. Whilst conceptually, a basis for the current application we are developing, it is unlikely there will be structural - beyond the formal decision tree structure - similarities between the two.

In regards to the end user interface, the client has supplied us with a *user interface* demonstration that we could possibly use in our development of the mobile application.
While data storage, in some form (e.g. database), will be necessary for testing. It is currently unclear how the client intends to store data used and created in the final product.

3.2 Architecture

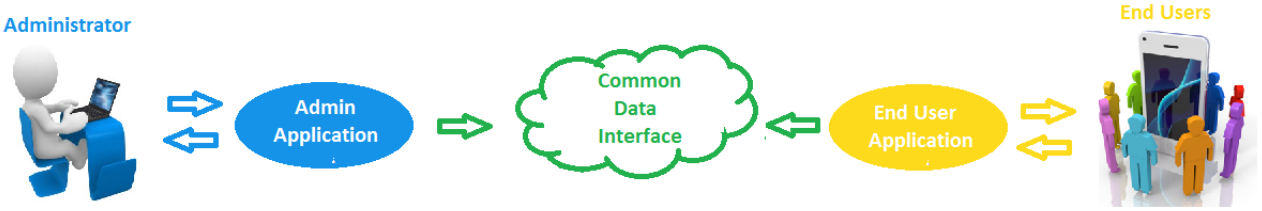


Fig.6: System Architecture

We will develop a decision tree engine application for the administrator to develop trees. According to the customer, there will only be one administrator (Andreas) that will develop all the trees and he will be using the admin application through a desktop computer or laptop.
There will also be a application for end-users. The end-user will go through the questions in the tree and receive statistics on how many of the questions answered that ”approve” an investment, how many that ”don’t approve” and how many that still is ”uncertain”. This is in percentage %. Easiest way is by putting weight on the answers.
We have decided to develop the end-user application in Cordova. Cordova is a hybrid cross-platform tool for mobile applications. It is called hybrid because it can be deployed like a native app (for Android and IOS devices) but it is made like a web page (HTML, CSS and JavaScript) and executed in a browser environment. The advantages of an approach like this are that the same code can be used for all mobile platforms and almost all the native functionalities can be accessed. Maintenance is also assured because the source code is only one and every operations of updating will produce consistent final apps. We have also decided to use the Backbone.js as a business logic layer. Backbone.js is a lightweight framework. Some call it a MVC (Model View Controller) framework, some call it a MV* framework. Backbone.js is dependent on Underscore.js and jQuery.

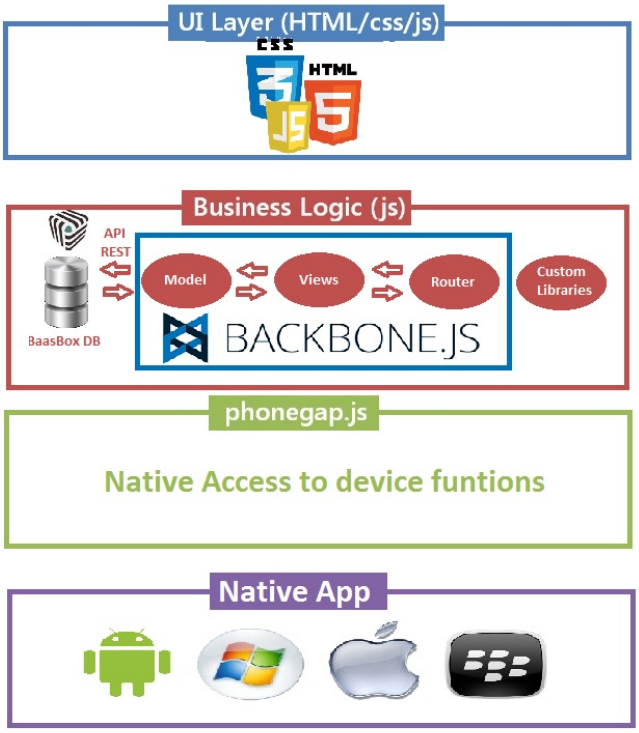


Fig.7: Cordova Architecture

The admin tool will probably be a web application as well, so that we can reuse as much code as possible.We will use Backbone outside Cordova to be consistent with the front end application and to make reuse of the tree structures. We are still not sure how the applications will communicate each other because we need yet to discuss this with the customer, but we have two different proposal: local or remote DB.

We have not yet decided on the structure of the tree and how to store trees. We made a simple sketch on how it might look (not a finished solution):

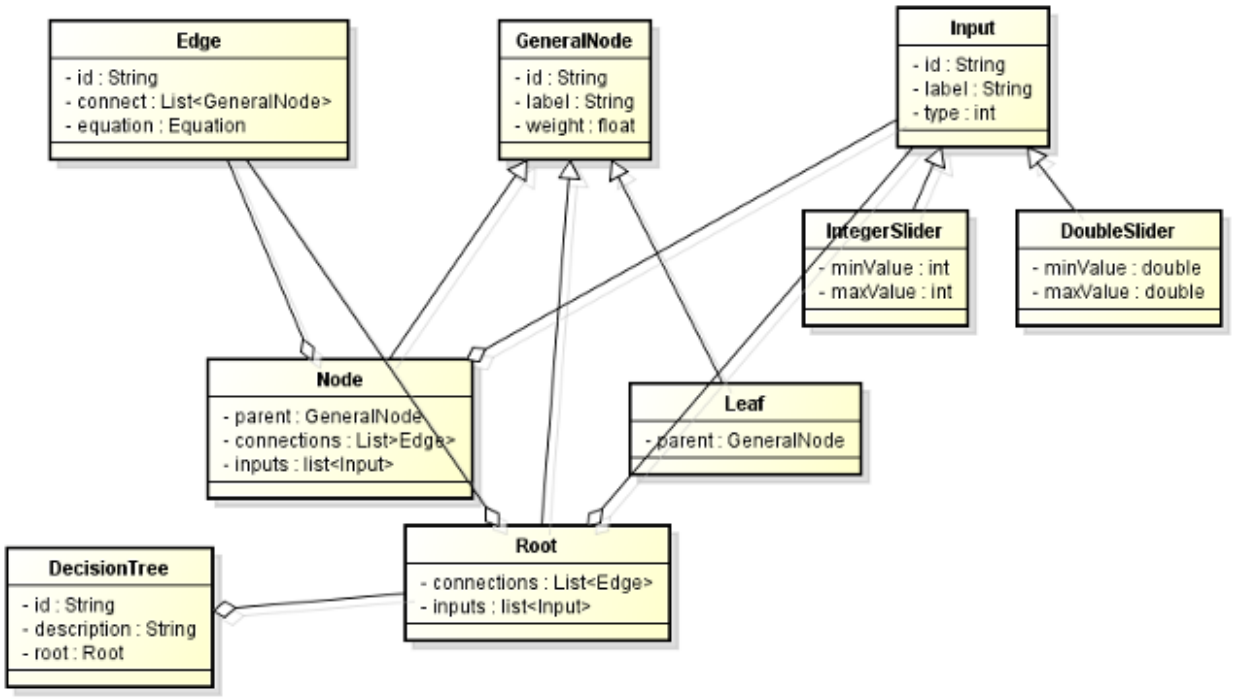


Fig.8: Tree Engine Model

3.3 Functional Requirements

A tree creator/editor, engine, and a web application (in which the engine is integrated) to test the functionality should be developed. There are two main categories of actors in this project: application end users and application creators.

The end users should be able to access the surveys offline. They should be able to continue working on a survey they have started or change the answers on a survey they have completed. Once a survey is completed, users should be able to forward it to an email address. Other features include changing the language, subscribing for updates, having access to tips and advices and a general description of the application.

The app creators should be able to integrate the engine in the application they create that is to be used by the users. They use the engine API calls to generate the view that lets the users fill the survey (by interpreting the tree that is generated by the tree creator/editor) and to evaluate the user's answers according to a predefined formula. They use the tree

creator/editor to create and edit trees, add and edit questions and answers and create a formula that is used to generate the final result.

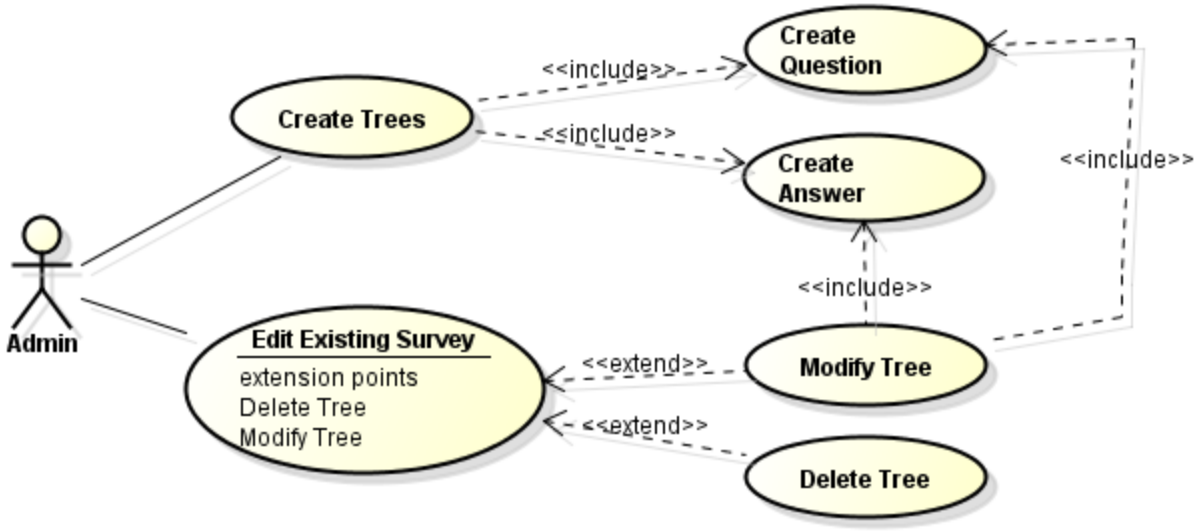


Fig.9: Admin Use Case Diagram

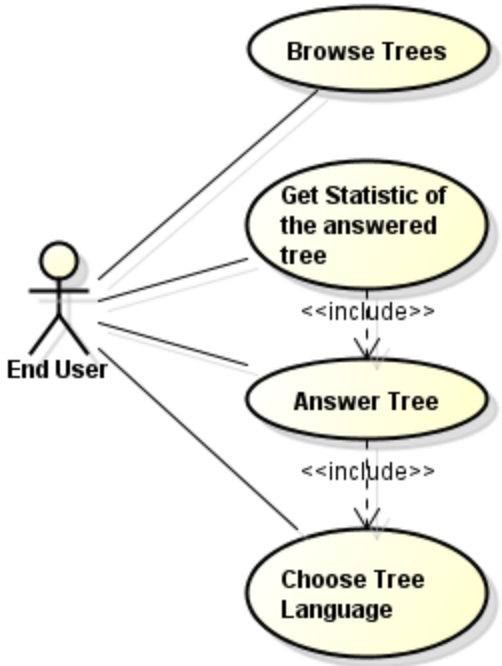


Fig.10:End User Use Case Diagram

3.4 Non-Functional Requirements

3.4.1 Usability

The applications targeted for both the users and the administrators should be simple, highly intuitive and easy to understand. The graphical user interface should be non-distracting, with easily readable font. From an aesthetic point of view, the look of the application does not have to be embellished.

The administrator application should be implemented in a way so the admin with no programming skills can easily create and edit decision trees. A clear, easy to edit view of the tree has to be provided. Drag and drop is desirable.

As for the user's application, the application should contain information about how to use the application and an option to check out a question's specific description, making it more user-friendly.

3.4.2 Reliability

The reliability of the decision tree engine is of great importance. The engine should always produce a correct tree structure and correct end results after completing the survey according to the formula provided by the administrator who created the survey.

3.4.3 Testability

Testability is necessary in order to make sure that the engine is reliable and always producing the correct output. Therefore modularity of the code and separation of concerns are essential, the modules have to be as isolated as possible to ease the process.

3.4.4 Portability

Portability is one of the main requirements in this projects since the goal is to make the user application accessible for as wider audience as possible deploying it in a multiple different platforms (Android, IOS).

3.4.5 Reusability

The engine source code should be independent of the user interface, therefore making the code easily reusable. This is one of the main requirements for this project.

3.4.7 Extensibility

Since work on the engine will probably be resumed after the hand-in of the code, it is very important that the code leaves space for an easy way of adding new features and modification of existing features.

3.4.8 Documentation

In order to assure reusability, maintainability and extensibility, extensive documentation has to be provided including: project plan, design description, test specification, project report. The source code has to be easily comprehensible, including a lot of comments , sufficient code documentation and has to be written following a set of code standard rules.

3.4.9 Price

The project's prototype has initially been developed using the Xamarin Forms framework. Because of the high price for the license, a cheaper alternative for creating a cross-platform solution is sought. Using open-source frameworks like Cordova and libraries like Backbone is therefore a suitable choice.

3.4.10 Performance

The performance is not of great importance in this project.

3.5 Traceability, Verification and Validation

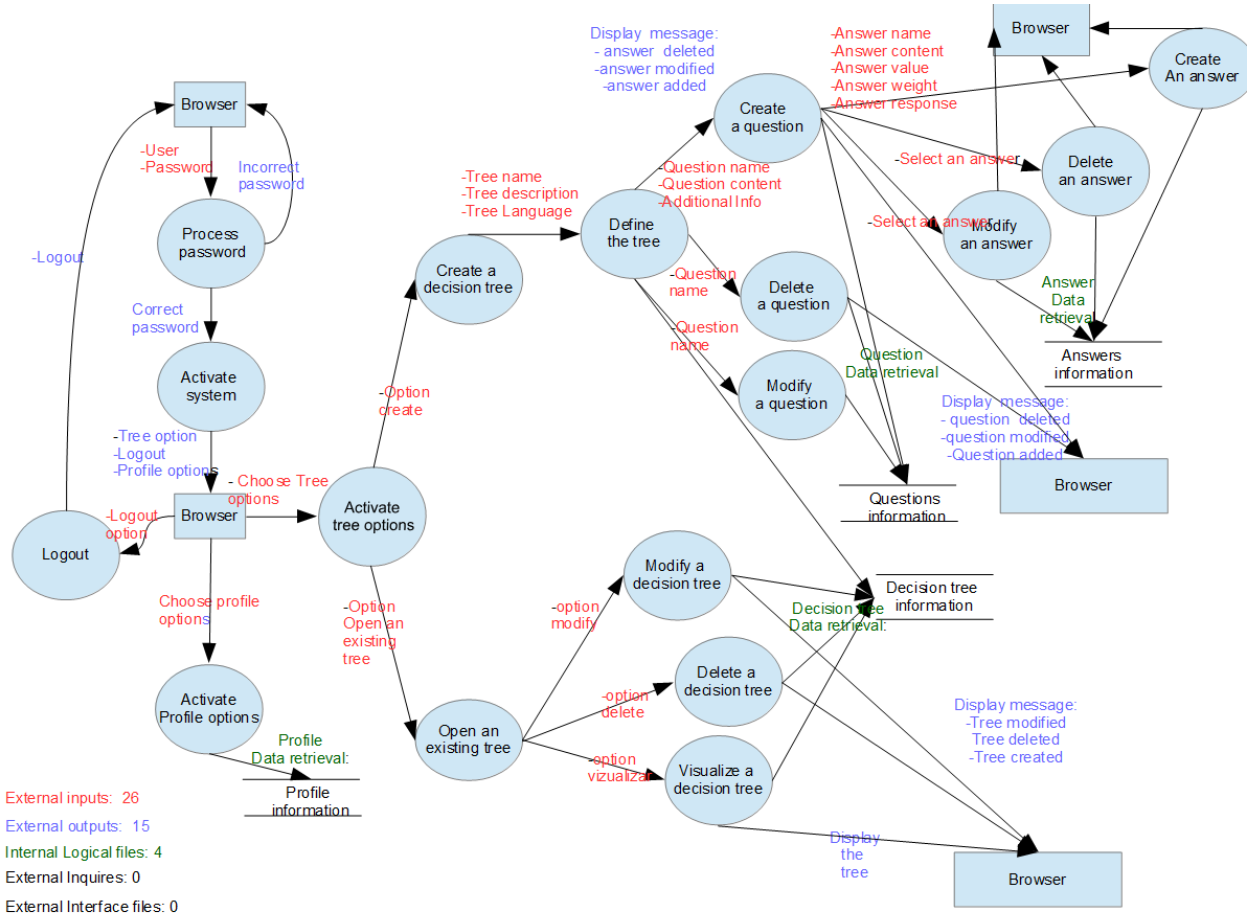
We will manage these three important aspect of this project in this way:

- Traceability: for requirements and of course implementation traceability we will use asana. We will use a product backlog to distribute the work among us, and each functionality will be assigned to a member of the group. We can talk in this case, since it is mostly a mobile application, of views instead of functionalities. We will develop views independently from each other so that it is possible to assign each view among us and to trace it through the backlog we are going to create.
- Verification: in order to verify that the application will meet its requirements and that the process of building it is the correct one, we will follow the strict pattern provided from the technology. We will divided our work and follow backbone patterns systematically to assure the product will be build right.
- Validation: we will validate our product by having weekly meetings with customers to see if the product meets his expectations. The validation process will start as soon as the requirements definition and will keep consistency during the development of the code. Of course we are right organized with an agile approach so we know that the process of validation is really important, and that suddenly the work of an entire week could be useful. To manage this problem we need to be sure, before we start coding, that the expected result and final result will be the same. We will manage this with granularity on features so that we

will start to implement a new feature only if both the logic and the presentation layers are well defined according to the customer.

Appendices:

Appendix A: Data flow model for the “Generic decision tree engine”

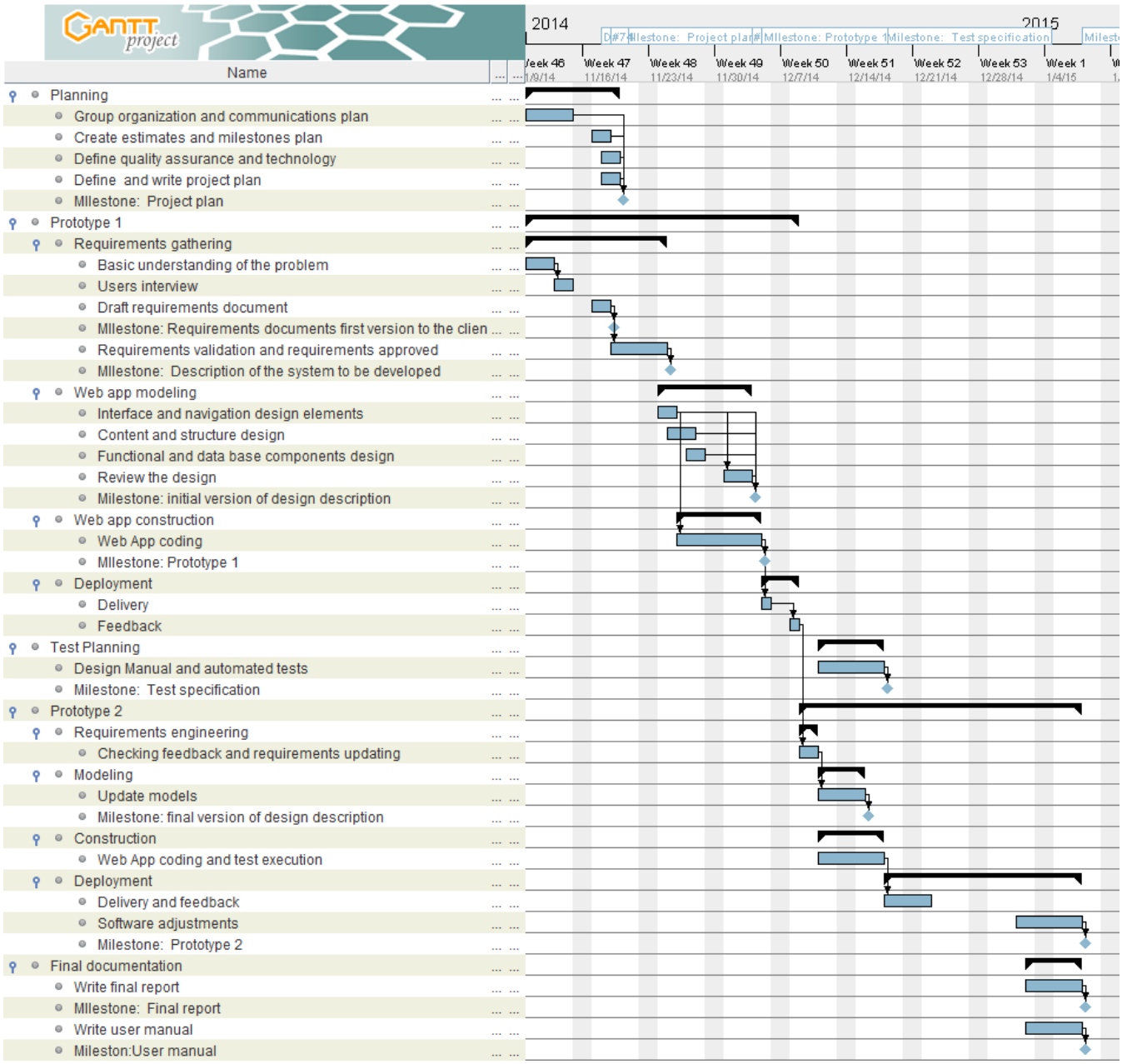


Appendix B: Value adjustment factors (VAF)

Each question is answered using a scale that ranges from 0 (not important or applicable) to 5 (absolute essential)

Question	**Weight**
Does the system require reliable backup and recovery?	1
Are specialized data communications required to transfer information to or from the application?	1
Are there distributed processing functions?	0
Is performance critical?	1
Will the system run in an existing, heavily utilized operational environment?	0
Does the system required online data entry?	0
Does the online data entry require the input transaction to be built over multiple screens or operations?	0
Are the ILFs updated online?	1
Are the inputs, outputs, files or inquires complex?	1
Is the internal processing complex?	2
Is the code designed to be reusable?	2
Are conversion and installation included in the design?	0
Is the system designed for multiple installations in different organizations?	0
Is the application designed to facilitate change and ease of use by the user?	3
Total (F_i)	12

Appendix C: Gantt Diagram



References:

[1] Pressman, R. S. (2010). Product Metrics. In *Software engineering: A practitioner's approach* (7th ed., pp. 620–623). New York: Mc Graw Hill.

[2] "What is Eclipse?". Eclipse.org. Retrieved 2014-11-20. http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fint_eclipse.htm

[3] "BACKBONE.JS" 2014-02-20. Retrieved 2014-11-20. http://backbonejs.org/

[4] "About Apache Cordova™". Retrieved 2014-11-20. http://cordova.apache.org/

[5] "LEARN The ASANA Basics". Retrieved 2014-11-20. https://asana.com/guide/learn

[6] "Pair Programming"Retrieved 2014-11-20. http://www.extremeprogramming.org/rules/pair.html