SOFTWARE REQUIREMENT SPECIFICATION

OF

MUSIC RECOMMENDATION SYSTEM

CENG HISTORY X

HACER NİHAL TARKAN

AYŞE AYBÜKE TAŞDİREK

ASENA OK

BİRANT ALTINEL

TABLE OF CONTENTS

1. Introduction

This software requirement specification (SRS) report expresses complete description about Recommendation System Project sponsored by AGMLab. This document includes all the functions and specifications with their explanations to solve related problems as a project of Middle East Technical University Computer Engineering Department.

1.1 Problem Definition

We are working on some common problems the recommender systems face and trying to improve the current solutions on these problems to increase the accuracy of the recommendations. The main issues would be huge, dense & sparse data sizes. There also may be issues about similarity, calculations, and data integration. To make the picture more clear, if we are to specify data integration process as an example, to integrate data from different sources requires combining heterogeneous data sources under a single query interface which raise an important issue.

The problem about huge and dense data is that it may lead to the need of more complex calculations with inefficient speed. Therefore we need to work heavily on different paradigms, algorithms, and databases to create an efficient solution even for very large data sets.

Sparsity of data is also a major issue, because no matter how well the algorithms are, if there is no data to process, this system will be of no use. Therefore we need to create solutions to also deal with these situations.

1.2 Purpose

The purpose of this document is to present a detailed description of the Recommender System that we will design and implement. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate This document is intended for both the stakeholders and the developers of the system.

The corresponding environments that this project will be integrated to are supposed to and probably will have a large number and variety of users in which many of them need such system to find whatever it is they need. The users may or may not be aware of their need for a recommendation feature on the software or website they are using, but such features can increase efficiency and save time to users, while they are looking in a place where there is a large number and variety of content which causes them to waste a lot of time to find what they need. From another point of view, there might be users that don't exactly know what they are looking for and such situations can also make a useful solution out of this project.

1.3 Scope

The goal is to design a recommender system on music domain. The system will be composed of server side components and client side components. The server-side component will manage the database operations and algorithms that produce recommendation results. The client-side components will be graphical interfaces that are integrated into corresponding larger systems.

Generally, a music recommender system consists of three key components: users, items and user-item matching algorithms. So, the background of our product is becoming more important than foreground component. At the background, the system will gather data from both users and items. To be more specific and precise, those data can be classified as follows:

- User Modeling [1] : Suggest that user data can be categorized into 2 domains: User Profile Data and User Listening Experience Data (shown in Table 1 and Table 2).[2]:

Data Type	**Example**
Demographic	Age, marital status, gender etc.
Geographic	Location, city, country etc
Psychographic	Stable: interests, lifestyle, personality etc. Fluid: mood, attitude, opinions etc.

Table 1. User Profile Data Classification

Type	Percentage	Features
Savants	7	Everything in life seems to be tied up with music. Their musical knowledge is very extensive.
Enthusiasts	21	Music is a key part of life but is also balanced by other interests.
Casuals	32	Music plays a welcome role, but other things are far more important.
Indiff erents	40	They would not lose much sleep if music ceased to exist, they are a predominant type of listeners of the whole population.

Table 2. User Listening Experience Data Categorization

- Item Modeling [1] : The second component of recommender systems is music item. The music metadata classified into three categories as in table below:

Item Type	Example
Editorial Metadata (EM)	the cover name, composer, title, or genre etc.
Cultural Metadata (CM)	Similarity between music items
Acoustic Metadata (AM)	Beat, tempo, pitch, instrument, mood etc.

Table 3. Music Item Metadata

By using both user and item related data and relationship between them, our software will produce meaningful music recommendations to users.

As we can see from the development of music recommender systems over the past years, the results are becoming more personalized. Using only music itself and user ratings are no longer sufficient. In recent years, great amounts of work have been done in music perception, psychology and music's effects on human behavior. Undoubtedly, music always has been an important role in people’s life and people have greater access to it. All of these highlight that music recommender is an effective tool for saving our time to find music for our personal interests.

1.4 User and Literature Survey

After that e-commerce gained popularity of selling products and the development of new generation mobile phones and innovative inventions like tablets, shopping become easier than before. Therefore, new kind of advertising techniques came up. Recommendation systems are one of the most popular techniques nowadays. They are useful for both the company and the user, because they increasing product sales and reducing the time spend on shopping. However, they have some problems because of huge data. Main problems are being unable to get really relevant results and being unable to get results in reasonable time.

Up to now there are a lot of methods developed to resolve the problems stated above such as collaborative filtering, content-based filtering. However they have some weaknesses. For example, collaborative filtering has the problem called cold start and means that recommendation system cannot produce any suggestion or recommendations. This problem occurs when items are provided in the system but there are few customers and few or no rankings. And the other example for content-based filtering, if the content is in lack of enough information to distinguish the items precisely, the recommendation cannot be precisely done. On the other hand, our system should find the most accurate recommendations.

Our potential users that we desire to help their problems are companies that use internet utilities to sell their products. And by proposing our product to these companies, we will reach to users of the websites that companies use. Therefore we will reach to both customers and companies with our system.

1.5 Definitions, acronyms, and abbreviations

SRS	Software Requirement Specification
Neo4j	Neo4j is a robust transactional property graph database.
Java	Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.

Cypher	Cypher is a Graph Query Language.
Neoclipse	Neoclipse is a standalone workbench application to interact with Neo4j (database directory or server).
Windows	Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.
Netbeans	NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5.
GUI	Graphical User Interface
Eclipse	A multi-language Integrated development environment (IDE)

1.6 References

[1] Song Y., Dixon S., Pearce M. "A Survey of Music Recommendation Systems and Future Perspectives", Page 397, 2012 London

[2] Jean J., "Aucouturier and Francois Pachet. Music Similarity Measures: What is the Use. In Proceedings of the ISMIR", Pages 157–163, 2002.

[3]Mortensen M., "Design and Evaluation of a Recommender System", Pages 26-28, 2007, Norway

[4] Javega M., "Content-based Music Recommender System", Page 35, 2005

[5]Whitman B., "How Music Recommendation Works and Doesn't Work", Retrieved from http://notes.variogr.am/post/37675885491/how-music-recommendation-works-and-doesnt-work

[6]Alexander C., "Risk Based Software Development", 2005, Retrieved from http://www.reliablesoftware.com/weblog/blogger.html

1.7 Overview

Following section of this document will focus on describing the system in terms of product perspective, product functions, user characteristics, assumptions and dependencies. In the third section, we will address specific requirements of the system, which will enclose external interface requirements, functional requirements of the system, performance requirements , and other requirements.

2. Overall description

In this part, background information about specific requirements of the system will be provided briefly. General issues that affect the product and outline of the functional requirements will be mentioned, too. In short, this section will mainly give information about product perspective, product functions, constraints, assumptions and dependencies.

2.1 Product perspective

Recommendation system depends on data come from websites users and companies items. This program has different type of users, so there is functionality differences between users will occur with respect to item data. The recommendation systems used by companies have differences for efficiency. Our recommendation system should work efficiently according to music data. So, there exists a user interface that is suitable for music recommendations and this interface will be an e-commerce website. Our system will be working in background. Once the recommendation system finds an accurate result, it will be shown on the interface.

In terms of hardware, recommendation system will be embedded in a website. To use or benefit from recommendation system, user should enter from a personal computer, mobile device with internet connection, tablet etc.

In terms of software, our recommendation system will run on personal computers, smart phones etc. That is, it will run on any device with internet connection. The system will work both Windows and Unix operating systems. Moreover, it will be implemented making use of database management tools such as Neo4j.

This brief information of interfaces is explained in more detailed below.

2.1.1 System interfaces

First of all, the application needs to have music data to make recommendation to user and user data to evaluate. User data is necessary to make a really relevant recommendation. The system will use this information to make inference about recommendation by getting user's actions. Rating value of a music data is a derived value obtained by an algorithm.

2.1.2 User interfaces

There will be one type of user. Therefore there are no differences between users in terms of functionality, visualization and interface.

The user interface is only depend on the websites designers. The website that our system is working on the background of it can offer different opportunities to its users. For example, a website can offer the right to choose background image to each of its users. However the architecture will probably be the same for all users.On the other hand, the interface will be in some number of steps such as user login, some top hundred lists, search options etc. But the first step should be user login to activate the recommendation system. Then the recommendations will be specified according to the algorithm that depends on user actions. These actions are determined by some parameters such as the time that the user listened the music. Some of these actions are listened, downloaded, listened before. If a music is downloaded before the other musics' rating value in the album will increase.

Up to now, we mentioned about how the recommendation will be done, but making recommendation more recognizable is also an important point. After getting a recommendation to user, the system should display it on the interface. However, if the recommendations cannot get user's attention, all the work done up to now might come to nothing. Therefore, they should be displayed on the user interface spectacularly.

2.1.3 Hardware interfaces

The recommendation system can work on any internet connected device. These devices should have some limit requirements to make the application run effectively. We expect that the processor speed and internet speed are high.

2.1.4 Software interfaces

First of all the system will work on any platform. Internet connection is a must to reach the system. Moreover, most of the application will be coded by Java. Java APIs of database management tools such as Neoclipse, which is a standalone workbench application to interact with Neo4j. Moreover, some query languages like Cypher will be used. Some tools and software are listed below.

Name	Version number	Source
NEO4j	2.0.0-M06	http://www.neo4j.org/
Netbeans	7.4	https://netbeans.org/
Windows	7/8	Microsoft Company
Cypher	-	http://www.neo4j.org/
Neoclipse	-	http://www.neo4j.org/
Java	SE 7	Oracle Corporation

2.2 Product functions

Use case diagram of the recommendation system and the other subsystems are revealed in below diagram. Steps are gathered in distinct entities, the functions of which are stated in further subsections.

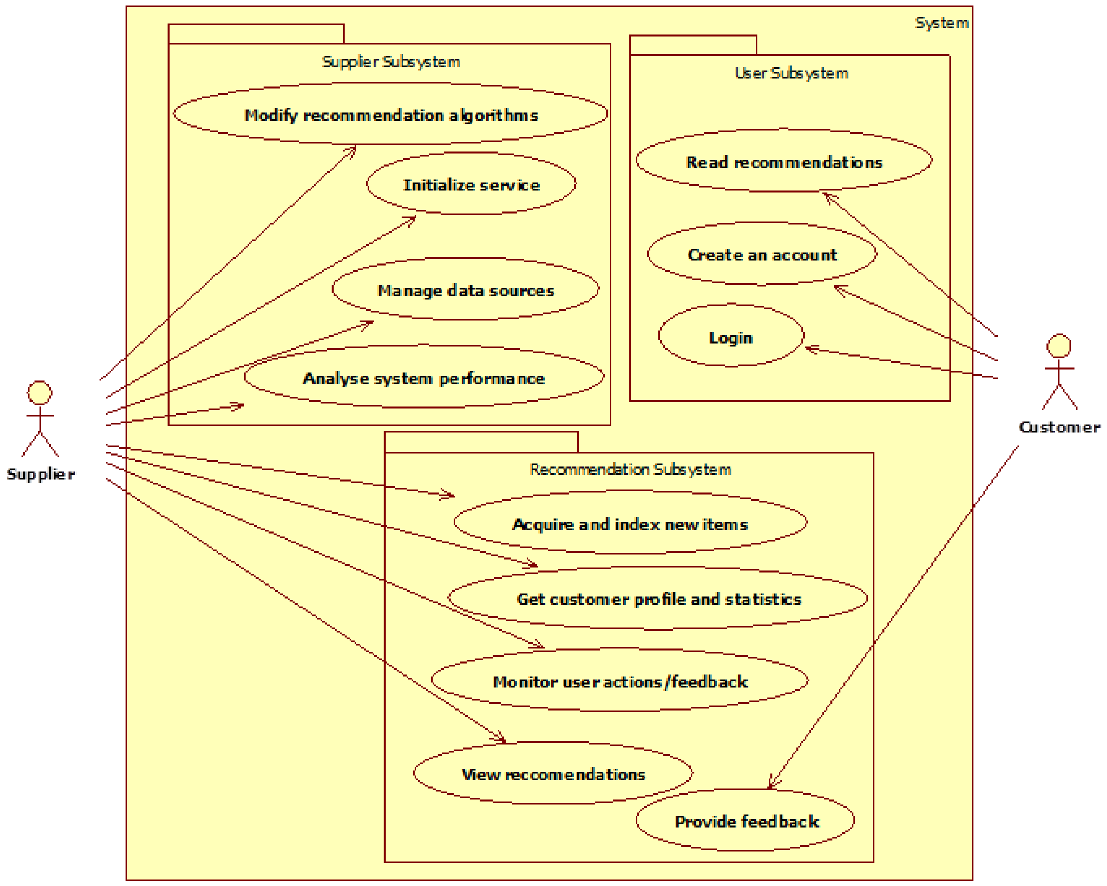


Figure 2.2.1: Use Case Diagram

2.2.1 Acquire and Index New Items

This function inserts new items by indexing them like the previously saved items. This functionality may be achieved by a query language. After adding this item to database, this item will also ranked according to its album id for users who listens that album.

2.2.2 Get Customer Profile and Statistics

This function provides customer profile information and statistics to supplier of the whole system. Actually, this function provides the whole information needed to make a recommendation.

2.2.3 Monitor User Actions/Feedback

The system should monitor user actions as the system receives implicit feedback through it.

2.2.4 View Recommendations

To make user tend to make use of an opportunity of a recommendation, monitoring is an important point. Here supplier uses this functionality to get attention of the user on a recommended item by displaying it in an attractive way.

2.2.5 Provide Feedback

The users must be able to read reviews and provide feedback to the system. This feedback is crucial to provide more accurate recommendations in time.

2.3 Constraints

- Since we need user profile data while developing the product, to find real time and sufficient data can be a problem for developer because of regulatory policies.

- Millions of data will be needed to test the software. At this stage developers will need huge amount of disk space and clusters.
- We will not design any specific interfaces for the product. It will be suitable for the application that people listen music from. So, developers have to consider common components of these applications so that the software can be integrated to any of this application easily.
- The application gathers real time user profile information from user accounts. Therefore, it must be reliable and keep those data in safe. Moreover, the system will produce new data about users depending on their behavior on the web. Security of this resulting data must be provided by the software also.
- Most important concern of the system is producing accurate recommendations. To provide expected accuracy and to handle with sparse and huge data at the same time is critical.

2.4 Assumptions and dependencies

As stated in the previous section which is constraints, there are several requirements like music data, user data, database management tool etc. To accomplish activating recommendation system, these requirements should be provided. However, the case of having all these software and hardware provide, still we might have some difficulties to test the system without strong internet connection. Losing connection is an important problem, because our system will work on online platform. If the internet connection cannot be supplied as requested, the whole sales policy will fail and our system will be useless.
We stated that we will use Java as a programming language to code our system. However, we can also study in other platforms. For example, we might study in .NET Framework.

3. Specific requirements

This section will describe the software requirements in detail as subsections which are interface requirements, functional and non-functional requirements.

3.1 Interface Requirements

Since there is just one type of user, the application will have only the user interface. Basically the user interface will direct the user through steps and will display the recommendations found by our algorithm. The steps of the user interface are the following;

Supplier Subsystem

This user interface is only for supplier, which is company. Supplier can do initialize service, analyze system performance, manage data sources, and modify recommendation algorithms operations.

User Subsystem

This user interface is for customers, that is the people who have an account or who want to be a member of the website. A user can do create an account, read recommendations (for user who has an account), log in operations.

Recommendation Subsystem

This user interface is the one which we all interested in, because our algorithm will work in the background of it. This interface will be used by both customers and suppliers. Customer cannot do many operations, but their feedbacks are very important to create a relevant recommendation. Users can only use provide feedback operation. Suppliers can do more operations, and these are acquire and index new items, get customer profile and statistics, monitor user actions and feedbacks, view recommendations. View recommendations operation can be used after our algorithm gives the result.

3.2 Functional Requirements

In this section, we will explain the major functions of the Recommendation System.

3.2.1 Client

The client-side of the system will be an application with a user interface that is integrated into a music listening website or application. This application gathers the information from users, investigates some actions of the users, and provides the connection with the server. This application is the client-side interface of the Music Recommender, so it does not include the functionalities of the host music environment such as playing music etc.

- **Requesting recommendations**

  The client-side application must allow user to request recommendations manually, and interact with the server to receive recommendations.

- **Evaluating songs**

  It must be able to evaluate songs and send appropriate information to the server.

- **Investigating user**

  The system will follow the interaction of users with music items., and send these obtain back to the server.

- **Display recommendations**

  The application must display the recommendations that are obtained from the server to the user in a proper way by providing a GUI.

3.2.2 Server

The server-side system will hold the entire data in a graph database, and must include all functionality to perform operations on this database, receive requests from the clients, evaluate, create and send recommendations etc.

- **Handle recommendation requests**

  The server application shall obtain and handle requests for recommendations.

- **Store evaluations**

  The server application shall receive and store music evaluations

- **Data storing**

  The server application shall be able to store the newly retrieved data to the database.

- **Recommend using content based filtering**

  The server application shall be capable of producing recommendations by interpreting the content and evaluations by actual user.

- **Recommend using collaborative filtering**

  The server application shall be capable of producing recommendations by interpreting the evaluations given by actual user and other similar users.

- **Recommend using contextual collaborative filtering**

  The server application shall be capable of producing recommendations by interpreting contextual information given by the users and evaluations given by the actual user and other similar users. [3]

3.3 Non-functional Requirements

The non-functional requirements of the system are explained below as performance requirements and design constraints.

3.3.1 Performance requirements

- **Accuracy**

  Since we will give the priority to the accuracy of the software, the performance of the Music Recommender will be based on its accuracy on recommendations.

- **Failure handling**

  System components may fail independently of others. Therefore, system components must be built so they can handle failure of other components they depend on. [4]

- **Openness**

  The system should be extensible to guarantee that it is useful for a reasonable period of time.

- **Security**

  Sensitive information should be kept in safe.

3.3.2 Design constraints

- Language

______The product should be_integratable with any music websites. Also we will handle with a lot of parameter and object. At this point, we need an object-oriented and commonly used language. As stated above, Neo4j will be used for server side and it has a Java API. So, Java programming language will be used for these aims.

- Hardware Constraints

______The_system will be integrated with a website. To use recommendation system, user should enter from a personal computer, mobile device with internet connection, tablet etc.

- Software System Attributes

    ➔ Usability

    The software will be embedded in a website. It should be scalable designed to be easily adopted by a system.

    ➔ Reliability

    The system should have accurate results and fast responses to user’s changing habits.

    ➔ Security

    User profile information will be used, so data security is one of the most important concern of the system.

4. Data Model and Description

This section describes information domain for the Music Recommender.

4.1 Data Description

Data objects in the Music Recommender that will be managed/manipulated by the software are described in this section with their attributes using class diagrams.

4.1.1 Data objects

Music recommendation system roughly has 5 types of data objects, namely User, Item , Recommendation, UserManager, EvaluationMeasures.

4.1.1.1 User

This object will hold the information of a specific user; id, name, age etc.

User
UserID:int UserName: String UserGender: String UserAge:String UserLocation: String

getUserID()
getUserName()
getUserGender()
getUserAge()
getUserLocation()
addReview()
updateUserProfile()

Table 4.1.1.1.1 User Class Diagram

4.1.1.2 Item

This object will hold information about each specific music item; id, name, artist, album, genre etc.

Item
itemID:int itemName:String itemReview:itemReview
getReview()

Table 4.1.1.2.1 Item Class Diagram

4.1.1.3 Recommendation

This object will hold a specific rating value that a user gave to a item, including the required information of the user and item along with a rating value.

Recommendation
predictedRating:float averageRating:float itemReview:float

setPredictedRating()
serAverageRating()
getItemReview()

Table 4.1.1.3.1 Recommendation Class Diagram

4.1.1.4 UserManager

This object handles the user addition and deletion operations.

UserManager
userManagerID: int

getUserManagerID()
addUser()
deleteUser()
deleteUser()

Table 4.1.1.4.1 UserManager Class Diagram

4.1.1.5 EvaluationMeasures

This object handles the operations about evaluating the data.

EvaluationMeasures
precision:float recall:float mse:mse

getPrecision()
getRecall()
getMse()
getMap()

Table 4.1.1.5.1 EvaluationMeasures Class Diagram

4.1.2 Data dictionary

We have given details about description of object attributes at section 4.1.1. Therefore, it is not necessary to mention about them again.

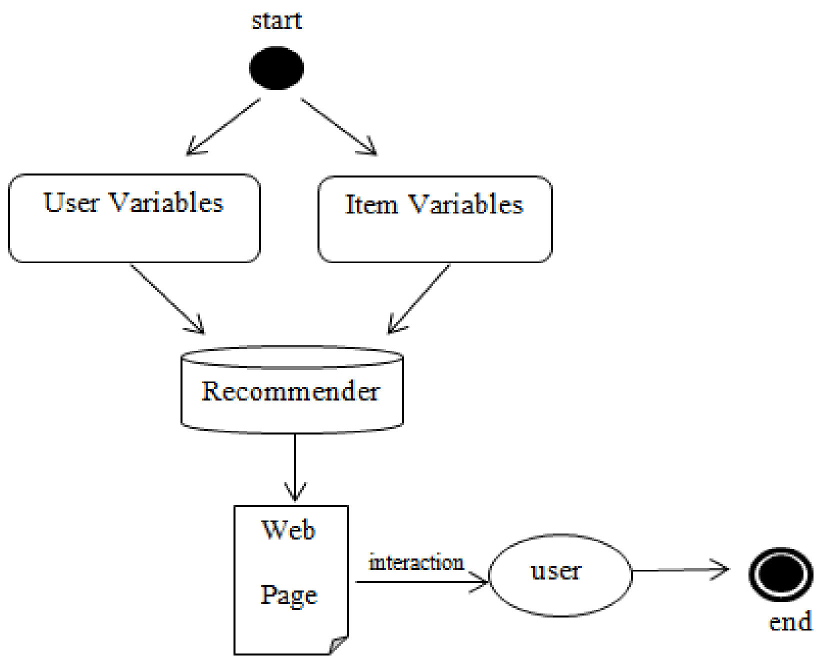
5. Behavioral Model and Description

5.1 Description for software behavior

At the first stage, recommendation system depends on data come from websites users and companies items. After that, it will start to generate accurate recommendations. These recommendations may depend on artist and song similarity. This is a suggestion of what other artists or songs are similar to the one you are looking at. This approach will be used when the system do not have sufficient information about user. Also the recommendations may be produced by using personalized data. This means that the system will have a “user model” (your activity on a service – plays, skips, ratings, purchases) for each user. This model is a list of songs or artists that the service does not think you know about yet that fits your profile. The other way of provide useful suggestions to users is playlist generation is playlist generation. The most consumers of music discovery are using some form of playlist generation. This is different from the above two in that they receive a list of items in some order (usually meant to be listened to at the time.) [5]

The system presents these results to the end user by using GUI of the system that recommender system is embedded in.

5.2 State Transition Diagrams



6. Planning

In this part of the document, the structure of the team responsible from the project, the basic schedule, and the process model will be presented.

6.1 Team Structure

We plan to divide the workload equally at the technical side. To contact academic staff and AGMLab, to search for project competitions and to write paper, we will do job sharing according to our field of interests. The basic structure of workload of team as follows:

Asena Ok: Algorithms, contact academic staff, research challenges
Aybüke Taşdirek: Algorithms, contact academic staff, research challenged
Birant Altinel: Database, contanct AGMLab, academic research
Nihal Tarkan: Database, contanct AGMLab, academic research

6.2 Estimation (Basic Schedule)

Task Name	November	December	January	February	March	April	May
Project Planning	2 week						
Learning and Improvement	2 weeks	1 week					
Learning Tools		3 weeks					
Design			3 weeks				
Algorithms			1 week	3 weeks			
Coding &Database Designing				1 week	4 weeks	4 weeks	
Testing							2 weeks

6.3 Process Model

In this project, spiral model, as can be seen in Figure 6.3.1 will be used.[6] The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts.

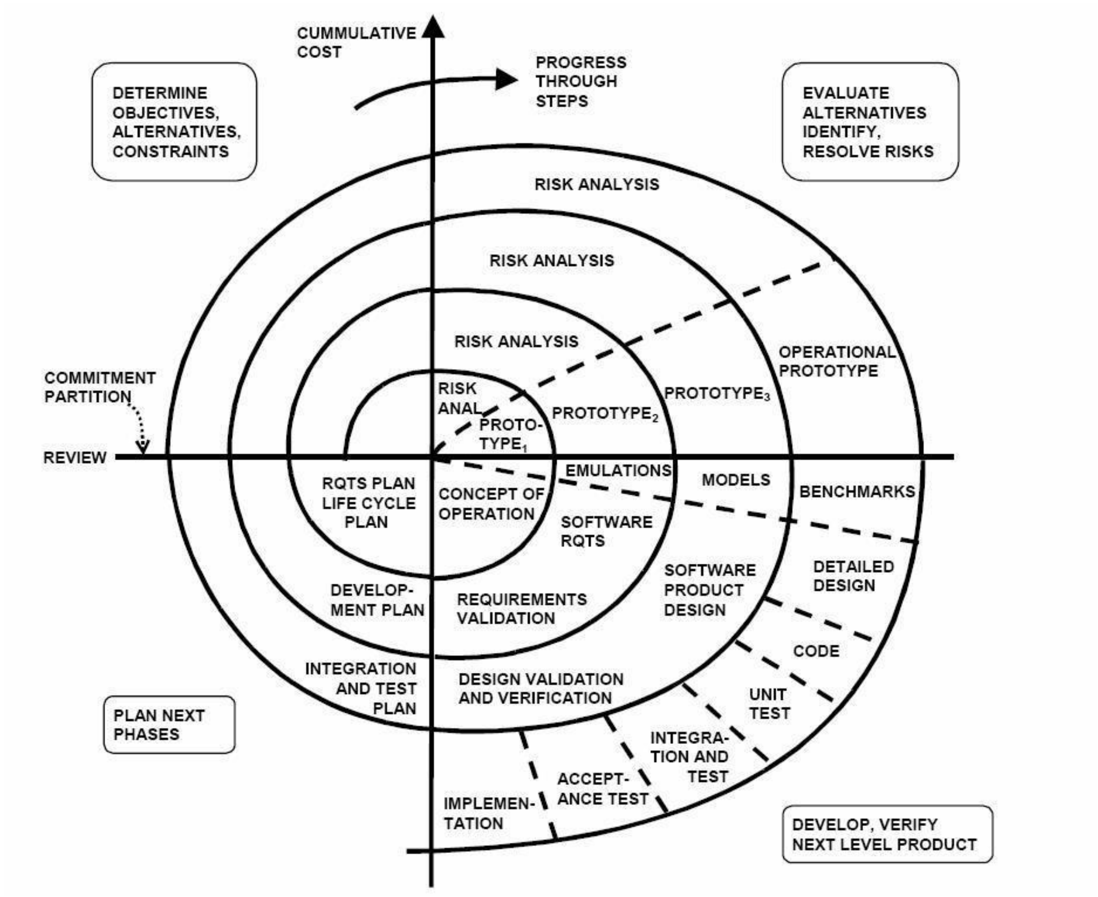


Figure 6.3.1: Spiral Diagram

7. Conclusion

Software Requirement Specification is an important part of the process of project development. Moreover, it is a prerequisite for creating the following design documentation. In this document, we have provided the information about general product description, data elements that the product deals with, specific requirements like product's interfaces and the functions will be implemented. In addition general behavior of the product has been explained in order to make it easy for the customer to understand to product's usage clearly. This document has been created through the help of various researches and depending on the demands of the customer. However, some little specifications are prone to be changed in the future.