University of Pretoria

COS 301 - Software Engineering

The Savage Ru's

Software Requirements Specification and Technology Neutral Process Design

Author(s):	*Student number(s):*
Jodan Alberts	14395283
Mark Klingenberg	14020272
Una Rambani	14004489
Ruan Klinkert	14022282

May 24, 2016

Contents

1 Introduction

This is the software requirements specification for the vizARD Augmented Reality application being developed for EPI-USE Labs by The Savage Ru's.

2 Vision

EPI-USE Labs (henceforth referred to as "the client") intends for the VizARD application to be used by a large variety of mobile device users across both Android and iOS platforms. VizARD helps to simplify the analysis of numerical data through visualization, in the form of automatically generated 3D graphs.

Fundamentally, the system will allow a user to take a picture of a table of numerical data which he/she may need to interpret. The application will then use OCR (Optical Character Recognition) to read the data from the picture. It will then decide on an appropriate graph for the type of data and generate a graph for the data. After the graph is generated, it will project a 3D model of the graph onto the image (or, ideally, onto a live stream of the paper) for the user to view.

Additionally, the system will allow users to send images (or screen captures) of generated graphs to other devices via popular social media channels.

Typically usage will be as follows:

- The user (possibly a businessman) finds tabular data he/she would like to analyse more easily.
- The user opens the app.
- Once the app is open and loaded, the user takes a picture of the table he/she would like to analyse.
- The user receives a notification that the graph has been generated and the generated graph is displayed on the screen (mapped onto the paper).
- The user taps on the "Share" button and is presented with several options through which he/she can share the graph.
- An option is selected and an image of the graph is sent to the other user.

3 Background

It is much simpler for us to recognize patterns and make quick analysis of data if it is presented to us in visual form. A simple example for the use of such an application would be a principal at a school who is presented with the Mathematics results of a particular grade for several quarters, such an application would make it very simple for him to quickly visualize the numeric data and see the trend.

The problem at hand is that there is a lot of information to go around and so little time to process. In a society that demands us to make decisions quickly, it would be wise to have a tool that aids the decision making process by making the information easier to digest and that is what vizARD intends to do.

Potential users could range from students, researchers, people in business, managers at stores and anyone else who would like to visualize data on the go.

4 Architecture Requirements

In this section we discuss the software architecture requirements, including access and integration requirements, quality requirements and architectural constraints.

4.1 Access Channel Requirements

The system will function as an offline application. As such there will not be any outside system with direct access to the application. There are, however, access channels for user. Specifically users will gain access to the system via two mobile device operating system:

- Android OS
- iOS

Furthermore any data that must be sent between the Operating System and the application will be done via the native APIs for each OS. On Android this will consist of the numerous Java APIs that is built into the system, and will likely be used to interact with the camera, file system, and (for sharing) the data connection. On iOS an Objective-C API suite will be used - in keeping with the native development philosophy of iOS.

4.2 Quality Requirements

4.2.1 Performance

4.2.2 Reliability

4.2.3 Scalability

4.2.4 Usability

4.2.5 Auditability

4.2.6 Security

4.3 Integration Requirements

The VizARD app will integrate with Android OS, and iOS, and use the suite of Android APIs which accompany the OSes. Specifically, APIs will be used to integrate with the sharing functions in order to share to different social media platforms and messaging apps. Additionally the app will gain access to the file system and camera via the built-in APIs. Furthermore,

Java APIs will be used to integrate between the various systems which make up VizARD's functionality. For instance between OpenCV and Tesseract for the OCR.

4.4 Architecture Constraints

- Android
- iOS

Although no other specific constraints are specified, it is implied that the systems used must all be cross-platform to allow for the two different interfaces (Android and iOS). As such, the AR Engine, OCR Engine and 3D Library must be OS independent.

4.5 Use case prioritization

4.5.1 Critical

- Taking a picture
- OCR (Optical Character Recognition)
- Automatic Graph Suggestion Algorithm
- Graph Generation
- Mapping Graph to Page

4.5.2 Important

- Live Augmented Reality Mapping
- Editing Graphs
- iOS Application
- Social Media Sharing

4.5.3 Nice to Have

- Opening Previous Graph

4.6 Use case/Services contracts

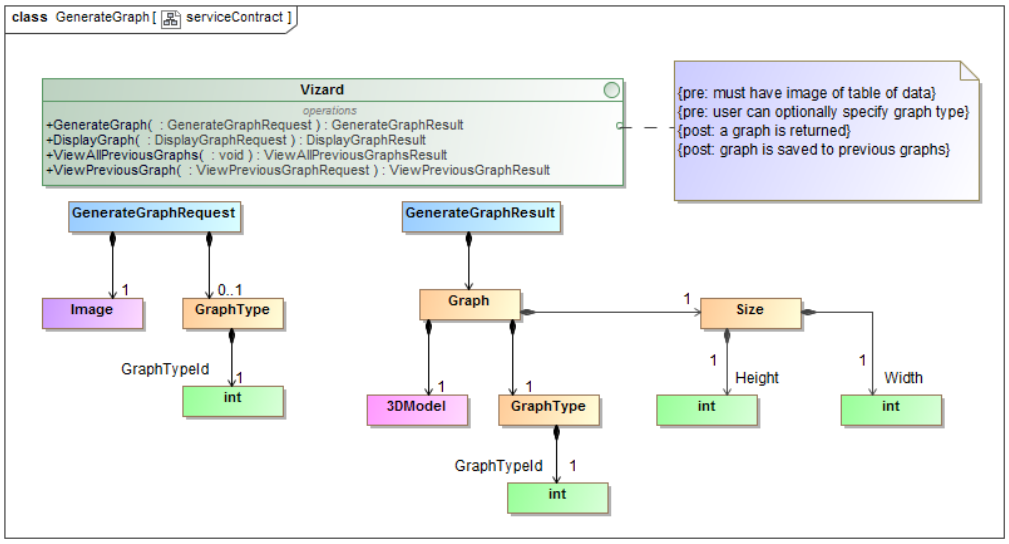

Figure 1: Services Contract : GenerateGraph

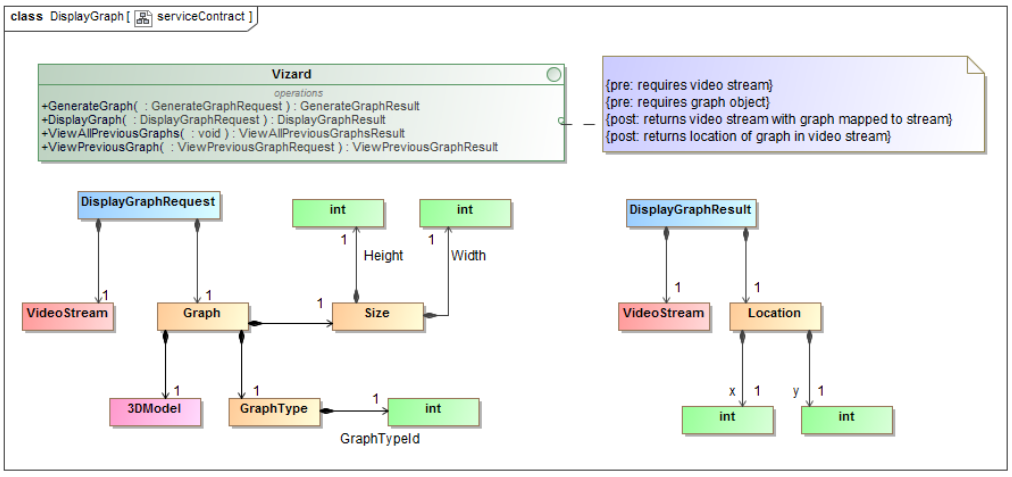

Figure 2: Services Contract : DisplayGraph

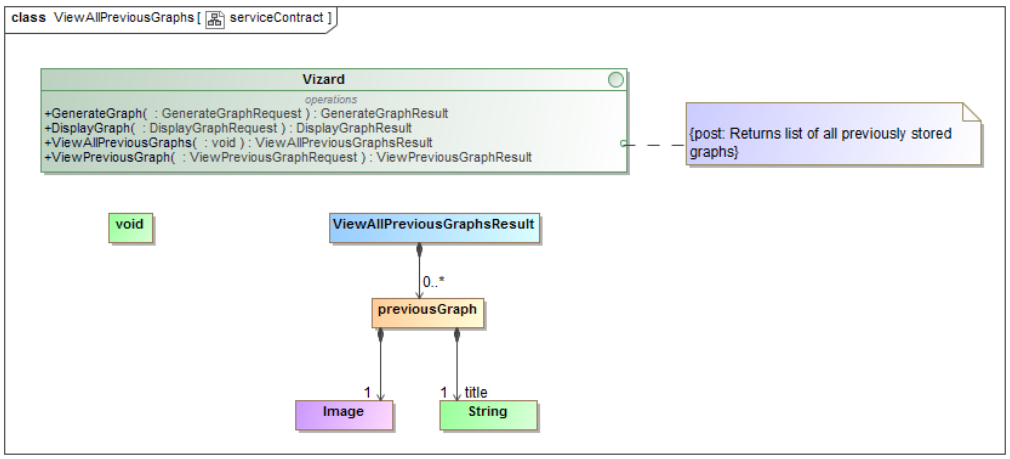

Figure 3: Services Contract : ViewAllPreviousGraphs

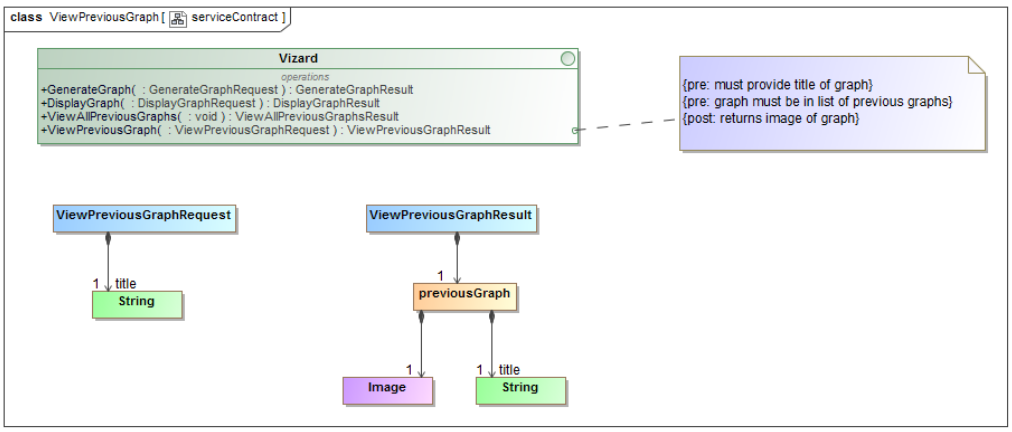

Figure 4: Services Contract : ViewPreviousGraph

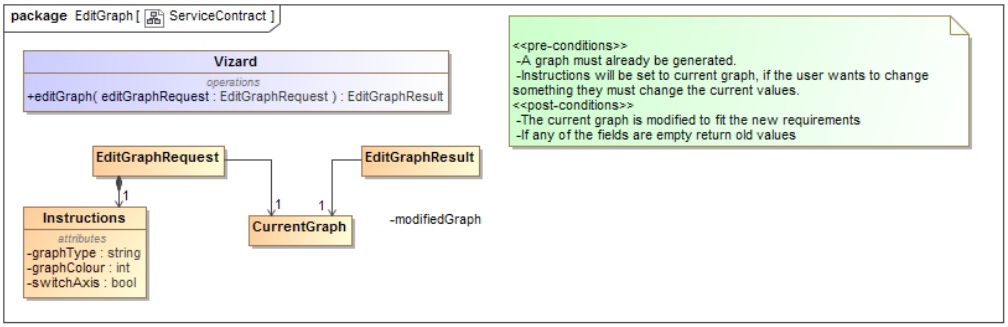

Figure 5: Services Contract : EditGraph

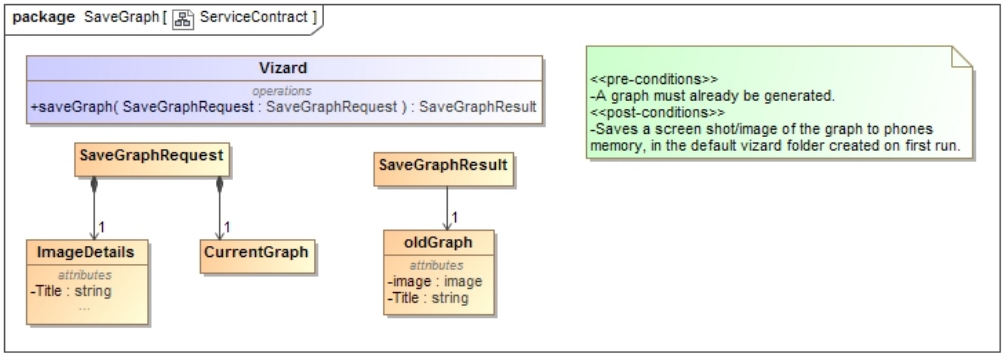

Figure 6: Services Contract : SaveGraph

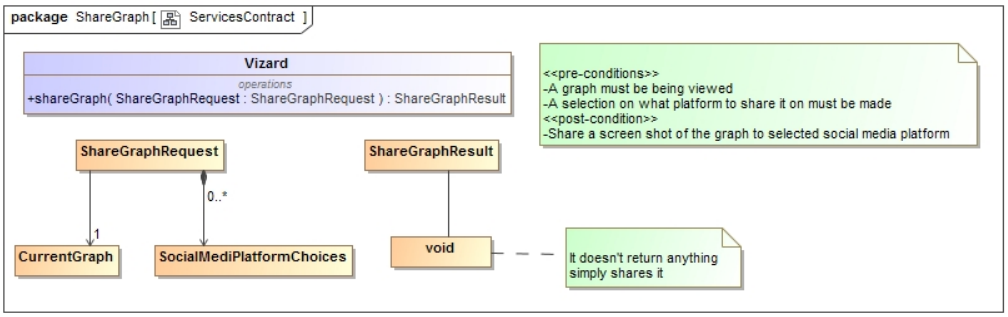

Figure 7: Services Contract : ShareGraph

4.7 Required functionality

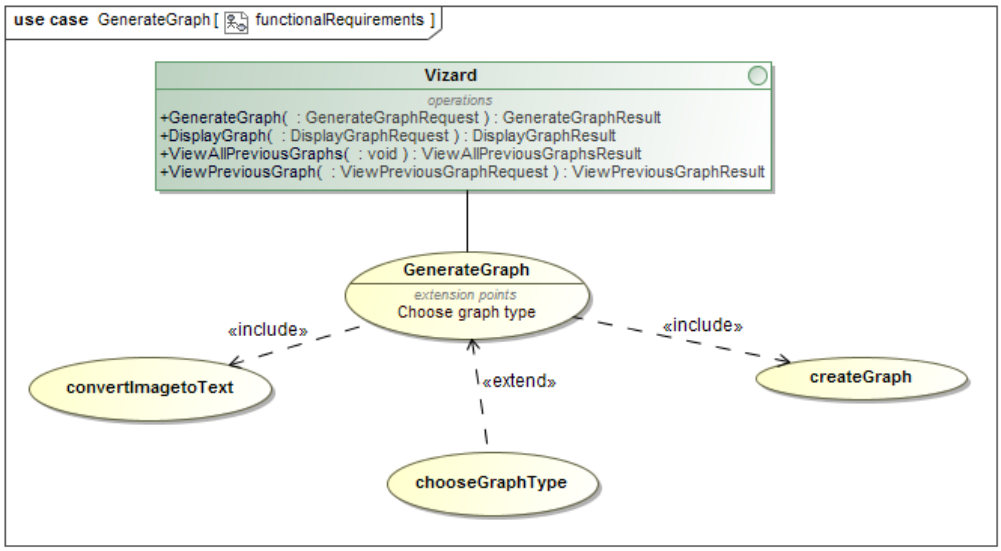

Figure 8: Required functionality : GenerateGraph

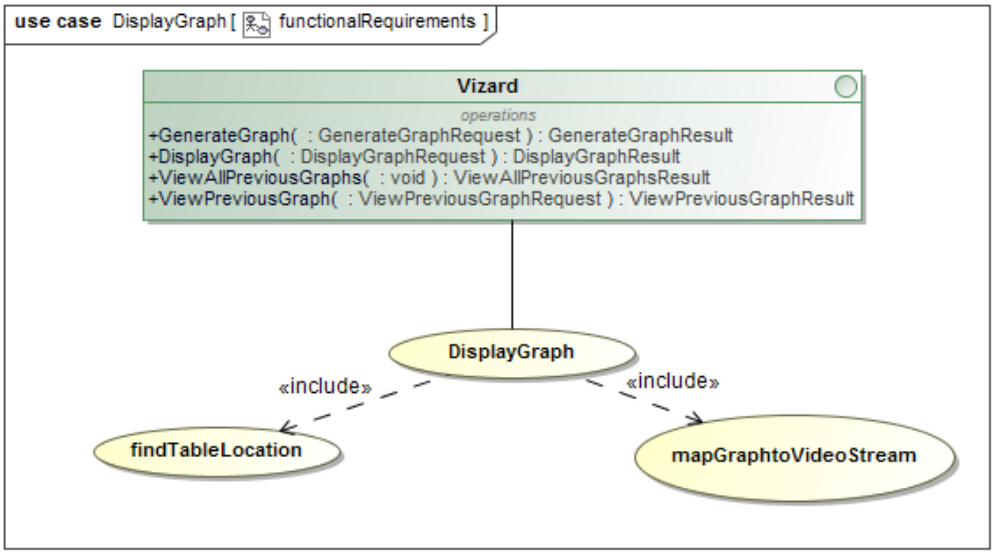

Figure 9: Required functionality : DisplayGraph

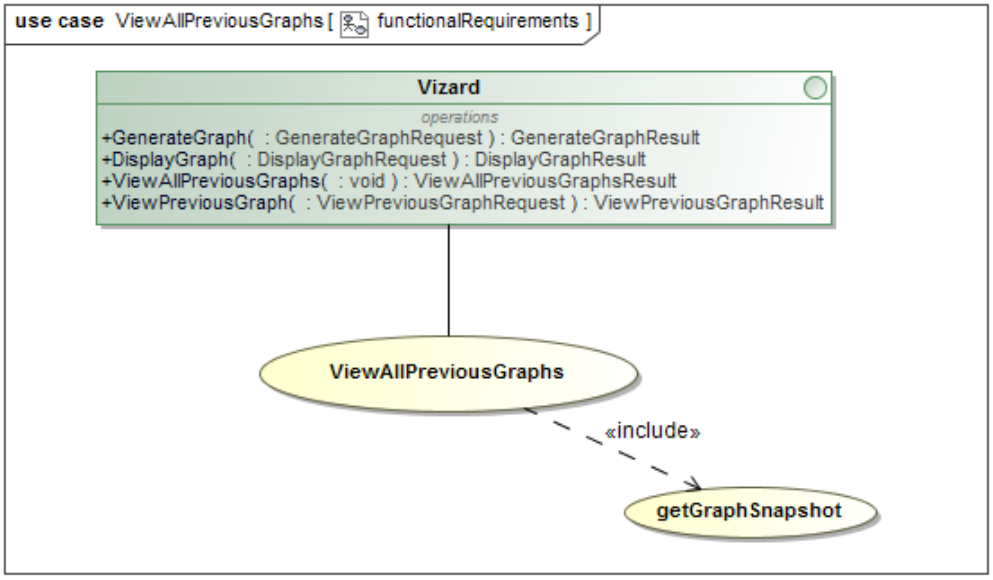

Figure 10: Required functionality : ViewAllPreviousGraphs

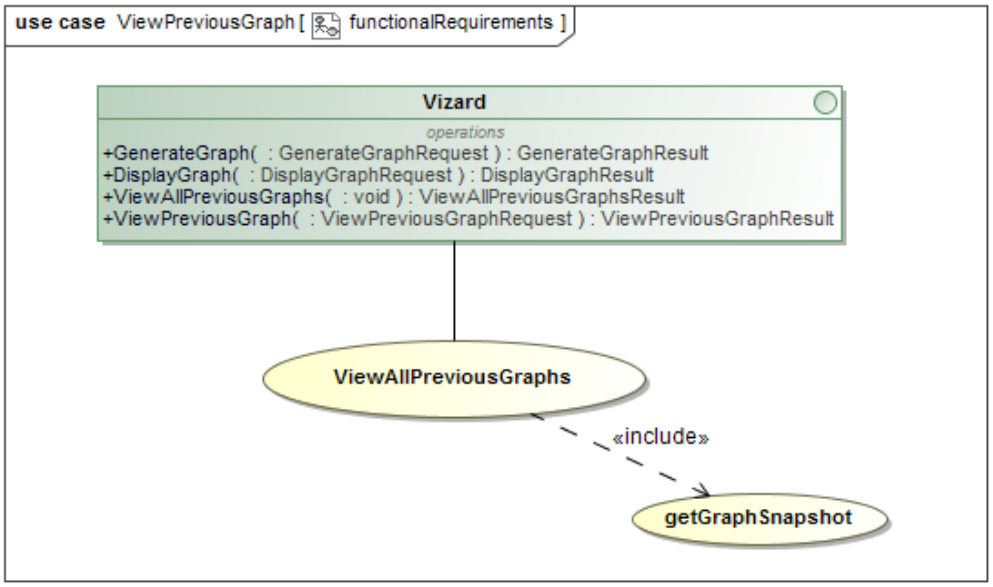

Figure 11: Required functionality : ViewPreviousGraph

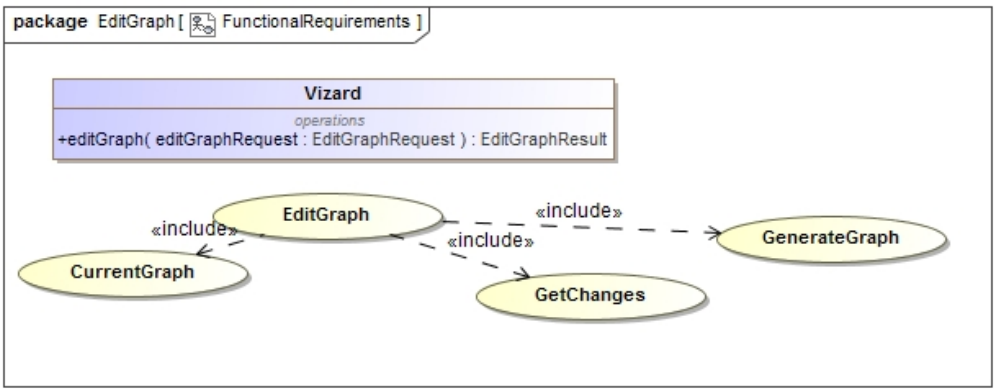

Figure 12: Required functionality : EditGraph

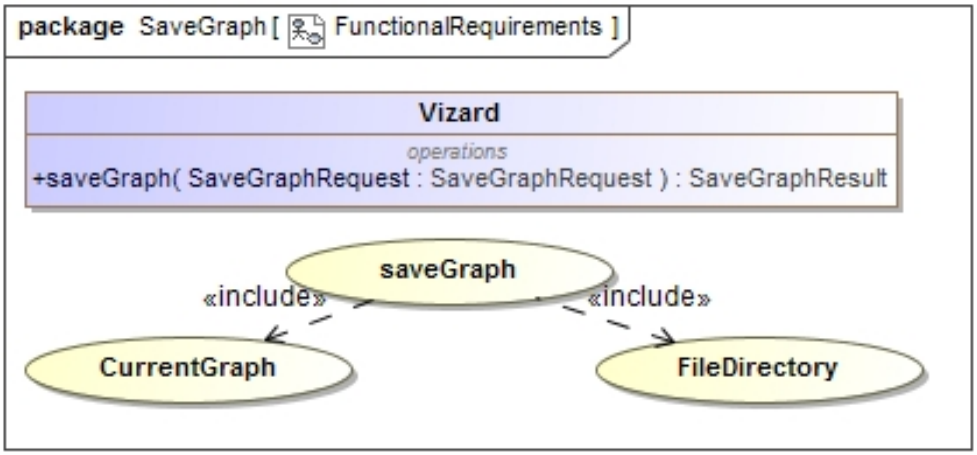

Figure 13: Required functionality : SaveGraph

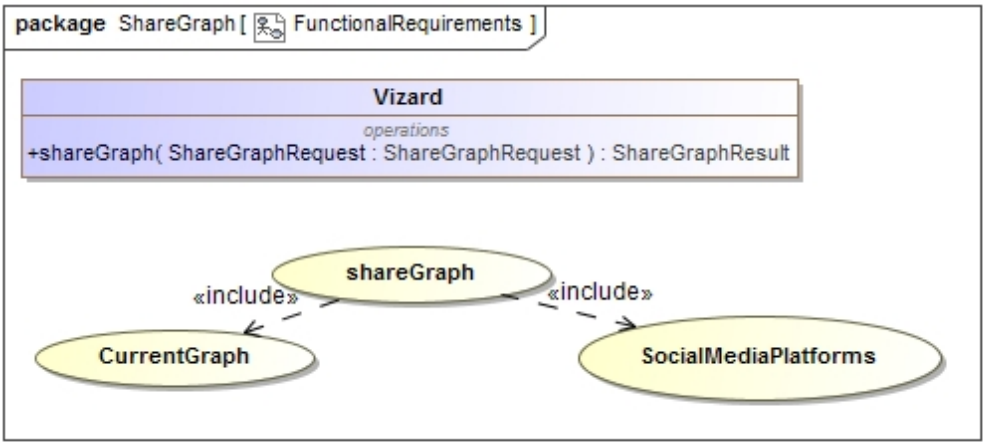

Figure 14: Required functionality : ShareGraph

4.8 Process specifications

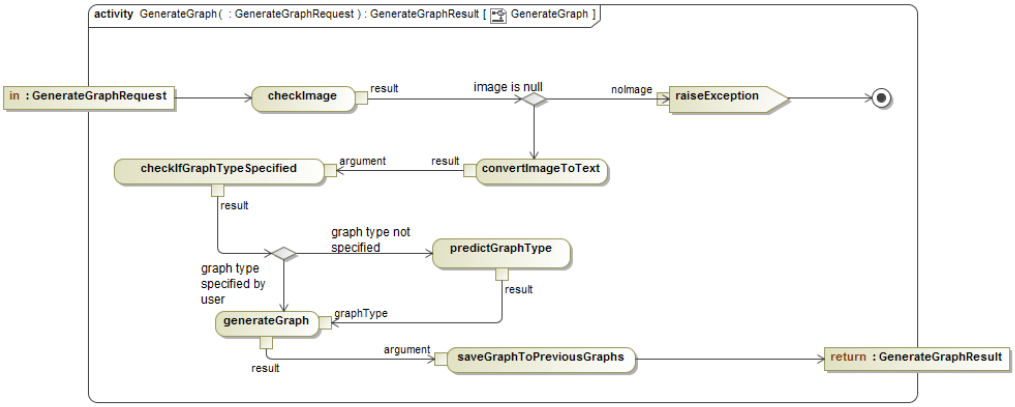

Figure 15: Process specifications : GenerateGraph

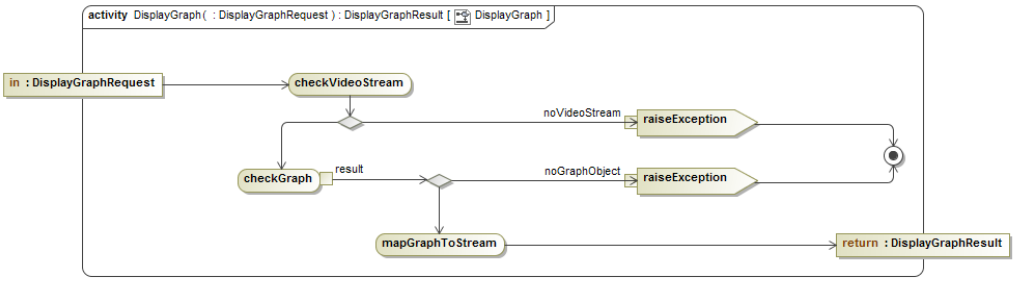

Figure 16: Process specifications : DisplayGraph

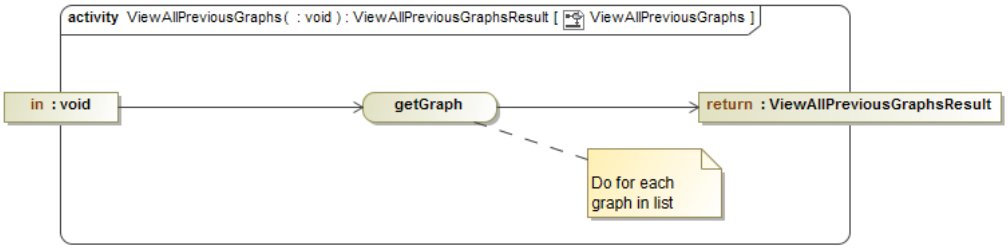

Figure 17: Process specifications : ViewAllPreviousGraphs

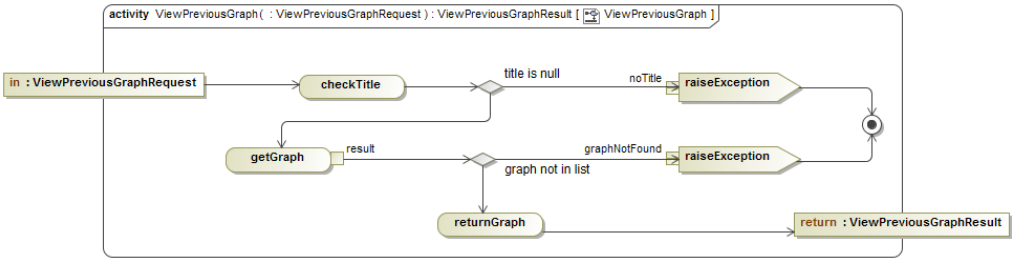

Figure 18: Process specifications : ViewPreviousGraph

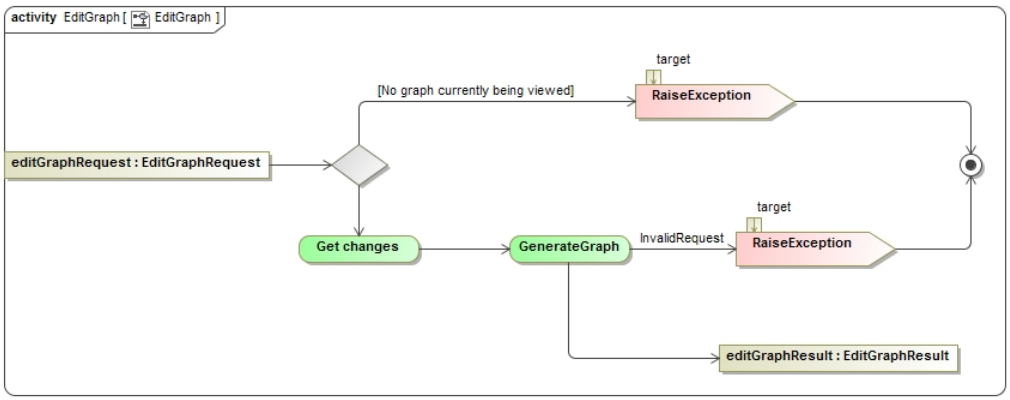

Figure 19: Process specifications : EditGraph

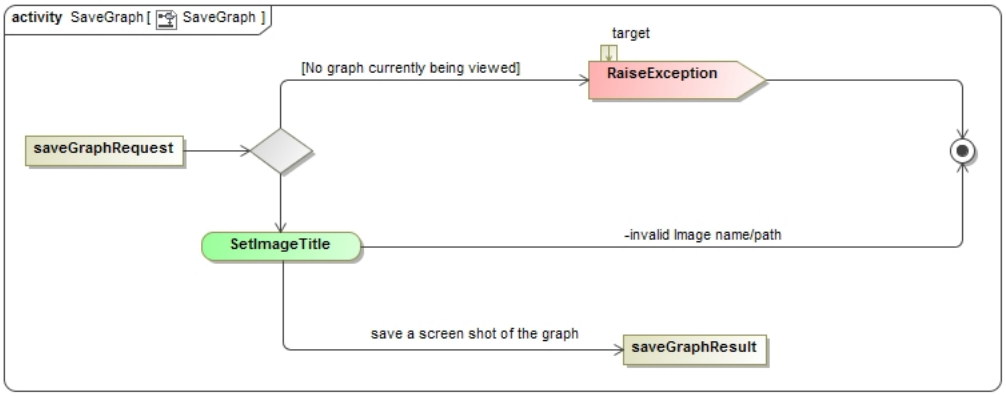

Figure 20: Process specifications : SaveGraph

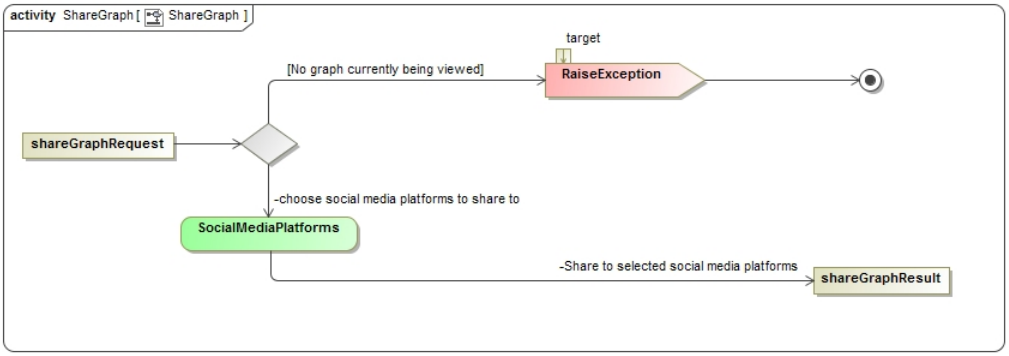

Figure 21: Process specifications : ShareGraph

4.9 Domain Model

5 Software Architecture

5.1 Architectural Patterns or Styles

5.2 Architectural Tactics or Strategies

5.3 Use of Reference Architectures and Frameworks

5.3.1 Web 2.0 Reference Architecture

5.4 Access and Integration Channels

5.5 Technologies

6 Open Issues