# Hipokratis

# Software Design Description

# version 1.0

# 01.12.2013

# Global Chain

# Ahmet KORKMAZ - 1648070 Oğuz ÖRCÜN - 1679455 Barış KENİŞ - 1746155 Erkan ONAT - 1679133

**Table of Contents**

[Preface 5](#_Toc373689611)

[1 Overview 6](#_Toc373689612)

[1.1 Scope 6](#_Toc373689613)

[1.2 Purpose 6](#_Toc373689614)

[1.3 Intended Audience 6](#_Toc373689615)

[2 Definitions 6](#_Toc373689616)

[3 Conceptual Model for Software Design Descriptions 7](#_Toc373689617)

[3.1 Software Design in Context 7](#_Toc373689618)

[3.2 Software Design Descriptions Within the Life Cycle 7](#_Toc373689619)

[3.2.1 Influences on SDD Preparation 7](#_Toc373689620)

[3.2.2 Influences on Software Life Cycle Products 8](#_Toc373689621)

[3.2.3 Design Verification and Design Role in Validation 8](#_Toc373689622)

[4 Design Description Information Content 8](#_Toc373689623)

[4.1 Information 8](#_Toc373689624)

[4.2 SDD Identification 8](#_Toc373689625)

[4.3 Design Stakeholders and Their Concerns 9](#_Toc373689626)

[4.4 Design Views 9](#_Toc373689627)

[4.4.1 Context View 9](#_Toc373689628)

[4.4.2 Composition View 9](#_Toc373689629)

[4.4.3 Logical View 9](#_Toc373689630)

[4.4.4 Depencency View 10](#_Toc373689631)

[4.4.5 Patterns Use View 10](#_Toc373689632)

[4.4.6 Interface View 10](#_Toc373689633)

[4.4.7 Interaction View 10](#_Toc373689634)

[4.4.8 State Dynamics View 10](#_Toc373689635)

[4.5 Design Viewpoints 10](#_Toc373689636)

[4.5.1 Context Viewpoint 10](#_Toc373689637)

[4.5.2 Composition Viewpoint 10](#_Toc373689638)

[4.5.3 Logical Viewpoint 11](#_Toc373689639)

[4.5.4 Depencency Viewpoint 11](#_Toc373689640)

[4.5.5 Patterns Use Viewpoint 11](#_Toc373689641)

[4.5.6 Interface Viewpoint 11](#_Toc373689642)

[4.5.7 Interaction Viewpoint 11](#_Toc373689643)

[4.5.8 State Dynamics Viewpoint 11](#_Toc373689644)

[4.6 Design Elements 12](#_Toc373689645)

[4.6.1 Design Entities and Design Attributes 12](#_Toc373689646)

[4.6.2 Design Relationships 12](#_Toc373689647)

[4.6.3 Design Constraints 12](#_Toc373689648)

[4.7 Design Overlay 12](#_Toc373689649)

[4.8 Design Rationale 13](#_Toc373689650)

[4.9 Design Language 13](#_Toc373689651)

[5 Design Viewpoints 13](#_Toc373689652)

[5.1 Introduction 13](#_Toc373689653)

[5.2 Context Viewpoint 13](#_Toc373689654)

[5.2.1 Design Concerns 13](#_Toc373689655)

[5.2.2 Design elements 13](#_Toc373689656)

[5.2.3 Example languages 14](#_Toc373689657)

[5.3 Composition Viewpoint 15](#_Toc373689658)

[5.3.1 Design Concerns 15](#_Toc373689659)

[5.3.2 Design Elements 15](#_Toc373689660)

[5.3.2.1 Function Attributes 16](#_Toc373689661)

[5.3.2.2 Subordinates Attribute 16](#_Toc373689662)

[5.3.3 Example Languages 16](#_Toc373689663)

[5.4 Logical Viewpoint 17](#_Toc373689664)

[5.4.1 Design Concerns 17](#_Toc373689665)

[5.4.2 Design elements 17](#_Toc373689666)

[5.4.3 Example Languages 17](#_Toc373689667)

[5.5 Dependency Viewpoint 19](#_Toc373689668)

[5.5.1 Design Concerns 19](#_Toc373689669)

[5.5.2 Design Elements 19](#_Toc373689670)

[5.5.2.1 Dependencies Attribute 19](#_Toc373689671)

[5.5.3 Example Languages 19](#_Toc373689672)

[5.6 Patterns Use Viewpoint 19](#_Toc373689673)

[5.6.1 Design Concerns 20](#_Toc373689674)

[5.6.2 Design Elements 20](#_Toc373689675)

[5.6.3 Example Languages 20](#_Toc373689676)

[5.7 Interface Viewpoint 21](#_Toc373689677)

[5.7.1 Design Concerns 21](#_Toc373689678)

[5.7.2 Design Elements 21](#_Toc373689679)

[5.7.3 Example Languages 21](#_Toc373689680)

[5.8 Interaction Viewpoint 21](#_Toc373689681)

[5.8.1 Design Concerns 21](#_Toc373689682)

[5.8.2 Design Elements 21](#_Toc373689683)

[5.8.3 Example Languages 22](#_Toc373689684)

[5.9 State Dynamic Viewpoint 24](#_Toc373689685)

[5.9.1 Design Concerns 24](#_Toc373689686)

[5.9.2 Design Elements 24](#_Toc373689687)

[5.9.3 Example Languages 25](#_Toc373689688)

**Index of Figures**

[Figure 1 – Use Case Diagram 15](#_Toc373689695)

[Figure 2 – Context Diagram 16](#_Toc373689696)

[Figure 3 – Component Diagram 17](#_Toc373689697)

[Figure 4 – Class Diagram I 19](#_Toc373689698)

[Figure 5 – Class Diagram II 19](#_Toc373689699)

[Figure 6 – Composite Structure Diagram 21](#_Toc373689700)

[Figure 7 – Sequence Diagram I 23](#_Toc373689701)

[Figure 8 – Sequence Diagram II 23](#_Toc373689702)

[Figure 9 – Sequence Diagram III 24](#_Toc373689703)

[Figure 10 – Sequence Diagram IV 24](#_Toc373689704)

[Figure 11 – Sequence Diagram V 25](#_Toc373689705)

[Figure 12 – Statechart Diagram 26](#_Toc373689706)

Preface

This document contains the system design information for Hipokratis, computer aided bone fracture detection system.

The document is prepared according to the “IEEE Standard for Information Technology - Systems Design - Software Design Descriptions (Scaled down version for Ceng491 course)”

This Software Design Documentation provides a complete description of all the system design and views of the Hipokratis, computer aided bone fracture detection system.

1 Overview

1.1 Scope

This is the last document before the implementation stage so it covers conceptual model for software design description, design description information content and design viewpoints.

1.2 Purpose

The purpose of this SDD is to show a detailed design explanation of the project Hipokratis. This document has the design concepts to help developers who are building this detection system and understanding the design can help other stakeholders to help building this application.

1.3 Intended Audience

The intended audience for this SDD is developers since it has a technical perspective over the design of the application.

2 Definitions

|  |  |
| --- | --- |
| **WORD** | **MEANING** |
| OpenCV | Library of programming functions mainly aimed at real-time computer vision |
| X-ray | X-ray is a radiographic image produced using the electromagnetic radiation method. |
| IEEE | The Institute of Electrical and Electronics Engineers |
| SRS | Software Requirements Specification which completely describes the behavior of a system to be developed. |
| Use Case | Use Case is a list of steps, typically defining interactions between a role and a system, to achieve a goal. |
| SDD | Software Design Description which is the complete description of the design of system. |
| CEng491 | METU Computer Engineering Department Design Course |
| MVC | Model View Controller which is an architectural pattern applicable for our software design. |
| UML | Unified Modeling Language |
| User | The actor who uses the application. |
| System | Hipokratis, computer aided bone fracture detection system |

3 Conceptual Model for Software Design Descriptions

3.1 Software Design in Context

Hipokratis is designed to help users with the detection of the bone fractures of human hand. In this system model, the user will provide an x-ray image to the system. The system will basically analyze and process that image using different filtering and detection techniques. When the system finishes all these operations, the resulting data will be shown to the user.

3.2 Software Design Descriptions Within the Life Cycle

3.2.1 Influences on SDD Preparation

The key software life cycle product that drives this software design is the SRS we have prepared. All the details and requirements are taken from there in order to prepare this document.

3.2.2 Influences on Software Life Cycle Products

This SDD influences the content of SRS of this project. It also has influences on the whole implementation phase of this computer aided bone fracture detection system. More than that, the test documentation and test plans of the system are also influenced by the SDD. In addition, the contents of SDD is taken into consideration by the developers in order to develop test cases and test procedures.

3.2.3 Design Verification and Design Role in Validation

This design should be based on the functional requirements in the SRS. The validation and verification is intended to be fulfilled by this document. Verification determines if the functional requirements in SRS are fulfilled. Validation is the determination of that the requirements for the intended use of this product are fulfilled. Validation and verification is checked by this SDD with the specifications in the SRS.

4 Design Description Information Content

4.1 Information

This section provides informations required from an SDD as follows:

* Identification of the SDD.
* Identified design stakeholders and concerns.
* Selected design viewpoints, each with type definitions of its allowed design elements and design languages.
* Design views
* Design overlays
* Design rationale

4.2 SDD Identification

An SDD shall include the information that follows:

* Scope
* References
* Context
* One or more design languages for each design viewpoint used
* Body

4.3 Design Stakeholders and Their Concerns

The stakeholders are developers who implement the system, the candidate users of the system and Ceng491 staff of Metu Computer Engineering Department. Their major concerns are:

* The implementation should be smooth since they expect a further development and usage area for this application.
* The implementation should be safe, secure, maintainable and open to future changes.
* The interface shall be easy to read and use.
* The desired results should be obtained from the developed system.

4.4 Design Views

4.4.1 Context View

This view “depicts services provided by a design subject with reference to an explicit context” (IEEE Standard). The context is defined by the elements that interact with the software like users.

4.4.2 Composition View

This view describes the way the design subject is (recursively) structured into constituent parts and establishes the roles of those parts (IEEE Standard).

4.4.3 Logical View

This view is to elaborate existing and designed types and their implementations as classes and interfaces with their structural static relationships. This view also uses examples of instances of types in outlining design ideas. (IEEE Standard).

4.4.4 Depencency View

This view specifies the relationships of interconnection and access among entities. These relationships include shared information, order of execution, or parameterization of interfaces. (IEEE Standard).

4.4.5 Patterns Use View

This view addresses design ideas as collaboration patterns involving abstracted roles and connectors. (IEEE Standard)

4.4.6 Interface View

This view provides information designers, programmers, and testers the means to know how to correctly use the services provided by a design subject (IEEE Standard).

4.4.7 Interaction View

This view defines strategies for interaction among entities, regarding why, where, how, and at what level actions occur (IEEE Standard).

4.4.8 State Dynamics View

Reactive systems and systems whose internal behavior is of interest use this viewpoint.

4.5 Design Viewpoints

4.5.1 Context Viewpoint

The context is defined by reference to actors that include users and other stakeholders.

4.5.2 Composition Viewpoint

The composition identifies the major design constituents of the design subject, to localize and allocate functionality, responsibilities, or other design roles to these constituents.

4.5.3 Logical Viewpoint

Detailed class diagram will be depicted in this viewpoint. Individual classes and the aggregation-composition relationship between those classes will be analyzed.

4.5.4 Depencency Viewpoint

A dependency viewpoint provides an overall picture of the design subject in order to assess the impact of requirements or design changes. It can help maintainers to isolate entities causing system failures or resource bottlenecks.

4.5.5 Patterns Use Viewpoint

Architechtural design patterns that construct the main structure of the system are provided in this section. Architechtural design patterns convey an image of a system. Architechtural patterns are concepts that solve and delineate some essential cohesive elements of a software architechture.

4.5.6 Interface Viewpoint

The Interface viewpoint provides information designers, programmers, and testers the means to know how to correctly use the services provided by a design subject. This description includes the details of external and internal interfaces not provided in the SRS.

4.5.7 Interaction Viewpoint

Interaction viewpoint is an essential part of our design specification since it describes the relations between entities.

4.5.8 State Dynamics Viewpoint

Reactive systems and systems whose internal behavior is of interest use this viewpoint.

4.6 Design Elements

Design entities and design attributes are stated in this subsection.

4.6.1 Design Entities and Design Attributes

Design entities and design attributes are listed as following.

* OpenCV
* Image Processor
* Database
* Repository
* Web Interface
* Image File

4.6.2 Design Relationships

OpenCV library, database, web interface are connected to each other. Without all these, the implementation of the classes of the software would be impossible.

4.6.3 Design Constraints

The constraints can arise from image files which should be handled properly and have the binary structure without any problem for the application to properly process the image without having a failure. Network connection must be supplied for the proper execution of the application. Apart from that, OpenCV and database design are the main constraints for the program.

4.7 Design Overlay

Since there is not an already-defined design view, we do not have a design overlay.

4.8 Design Rationale

In the section "4.3 Design Stakeholders and Their Concerns:", concerns of the stakeholders explained. Thus, considering those concerns the implementation must be designed to make it safe, secure and maintainable. Apart from that, the interface should be kept simple for the user to interact with the application. All this design document is prepared to satisfy the demands and concerns of the stakeholders.

4.9 Design Language

We have used The Unified Modelling Language (UML) 2.0 to make our diagrams.

5 Design Viewpoints

5.1 Introduction

In the following subsections, certain design viewpoints will be discussed in terms of design concerns. UML will be used as a design language.

5.2 Context Viewpoint

Context viewpoint of the Hipokratis Software includes the functions provided by the design of the system. These functions are used by different actors of the system in order to control the software.

5.2.1 Design Concerns

Context viewpoint of this system aims to identify the offered services and actors of the design subject. These services and actors are explained in detail in the following section.

5.2.2 Design elements

* Design entities: User, Hipokratis, Database, File Storage System.
* Design Relationships: User submits image data to Hipokratis. Hipokratis sends image to File Storage System and receives storaged image URL. Hipokratis sends and receives information about image file and image URL to the database.
* Design Constraints: This data flow must be secure.

5.2.3 Example languages

UML context diagram and use cases are as following.

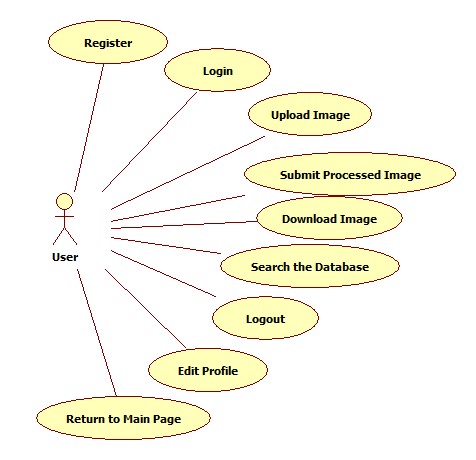


Figure 1 – Use Case Diagram

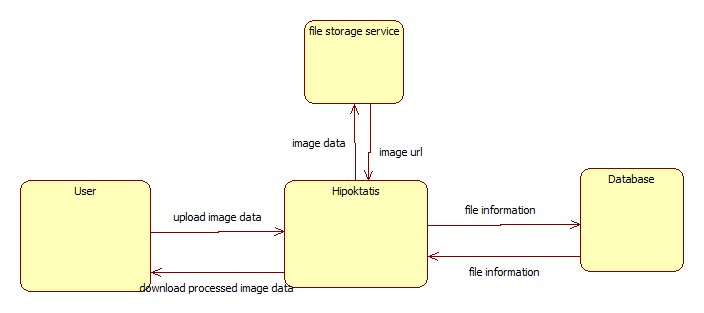


Figure 2 – Context Diagram

5.3 Composition Viewpoint

5.3.1 Design Concerns

Composition viewpoint of the Hipokratis Project includes subsystems and components of the application and describes the roles and relationships of that system components.

5.3.2 Design Elements

* Design Entities: There are three main design components in our project which are namely HipokratisWeb, HipokratisProcessor and Database. Also openCV library is required.
* Design Relationships: HipokratisProcessor is responsible for processing the x-ray image. HipokratisWeb establishes a web interface between the user, the database and HipokratisProcessor.
* Design Attributes: Design attributes are discussed in the following two chapters.

5.3.2.1 Function Attributes

Database , HipokratisProcessor and HipokratisWeb are the main components of the Hipokratis Project. HipokratisProcessor is responsable for processing the image and detecting the fractures. HipokratisWeb is responsable for providing an interaction between user , hipokratisProcessor and the Database through a browser. The database will store information about submitted images. OpenCV provides functions which are required for processing the image.

5.3.2.2 Subordinates Attribute

All of the components mentioned in the previous subsection are composed together in order to construct the main Hipokratis Project.

5.3.3 Example Languages

UML component diagram is as follows.

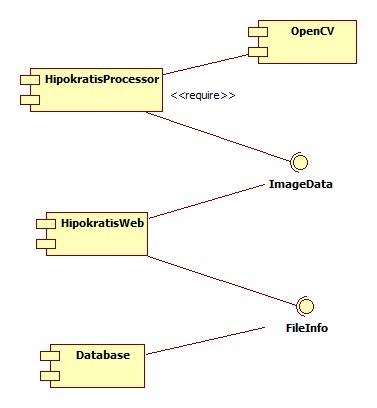


Figure 3 – Component Diagram

5.4 Logical Viewpoint

The logical viewpoint of the system aims to describe the implementation as classes with their structural relationships.

5.4.1 Design Concerns

Logical viewpoint explains the system by analyzing the classes which are constructed at the implementation step of the application.

5.4.2 Design elements

* Design Entities: Runner, Image, IOperation, GrayscaleOperation, MorphologicalOperation, EdgeDetectionOperation, GaussianFilterOperation, ShapeBasedOperation , Controller, User, Repository, FileInfo, Manager
* Design Relationships: The first part is HipokratisProcessor. All operations implement Ioperaiton. Runner is responsible for setting image operation. Image applies operation to itself. Image doesn’t know which operation it applies. Run method in Runner calls a series of setOperation and image.doOperation .

The second part is HipokratisWeb. Controller is instantiated according to request. Controller creates Manager and sets it’s attributes. Manager interacts with HipokratisProcessor. Manager sends raw image to HipokratisProcessor and gets the processed image from it. Manager creates repository. Repository object establishes connection between database and hipokratis. FileInfo is a persistant object stored in the database.

* Design Attributes: The attributes are shown in class diagram below.
* Design Constraints: Image file format must be in png , jpeg, bmp.

5.4.3 Example Languages

Class diagrams of the system are depicted as below.

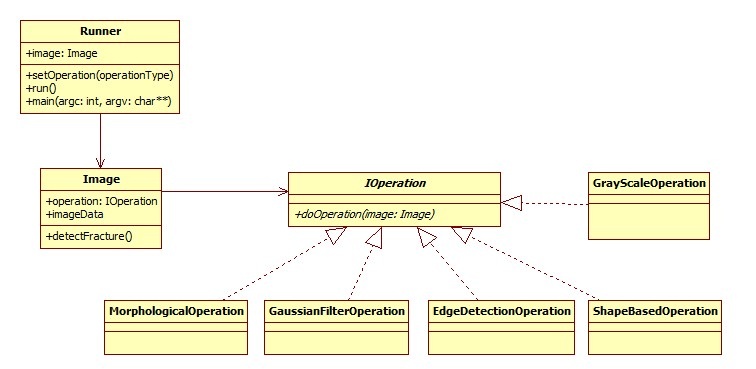


Figure 4 – Class Diagram I

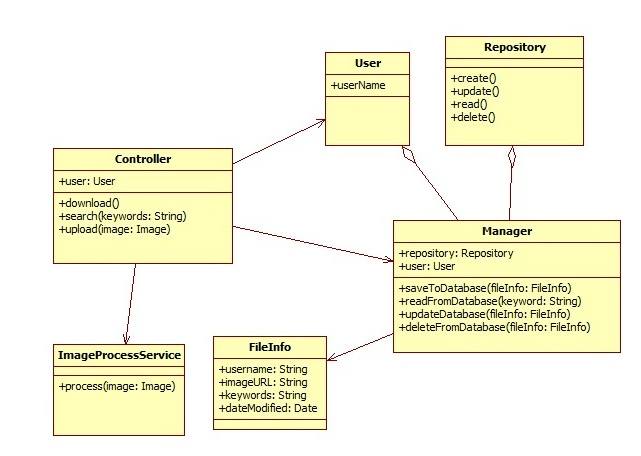


Figure 5 – Class Diagram II

5.5 Dependency Viewpoint

The dependency viewpoint specifies the relationships and dependencies between the design components of the system.

5.5.1 Design Concerns

Identifying the dependencies of the system and determining which subsystems are depends on other subsystems helps deciding the priorities in developing design entities.

5.5.2 Design Elements

* Design Entities: HipokratisWeb, HipokratisProcessor, Database, OpenCV
* Design Relationships: HipokratisWeb is dependent on HipokratisProcessor and Database. HipokratisProcessor requires OpenCV.

5.5.2.1 Dependencies Attribute

HipokratisProcessor requires OpenCV to be able to process the image and detect fractures. HipokratisWeb is dependent on Database to be able to store the submitted images. HipokratisWeb is also dependent on HipokratisProcessor to return the processed version of the uploaded image. The processed version of the image is the uploaded image with detected fractures marked.

5.5.3 Example Languages

Component diagram of the system is supplied in the subsection 5.3.3. You can refer to that subsection for the diagram.

5.6 Patterns Use Viewpoint

This viewpoint addresses design ideas as collaboration patterns involving abstracted roles and connectors.

5.6.1 Design Concerns

Hipokratis will be implemented by using Model View Controller (MVC) pattern. In this pattern, user will interact with the system by a user interface. When user makes an action this input will be handled by Controller. The controller notifies the model of the user action. Then a view manages the display of information.

As a design concern, MVC concepts and conventions must be carefully followed and implementation of classes should obey the rules of that pattern.

5.6.2 Design Elements

As a design element of the pattern use viewpoint, Model View Controller (MVC) pattern can be mentioned. MVC is a framework for building web applications using an MVC design:

* The Model represents the application core
* The View displays the data
* The Controller handles the input

5.6.3 Example Languages

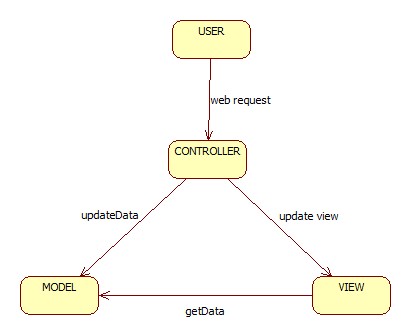


Figure 6 – Composite Structure Diagram

5.7 Interface Viewpoint

This description includes the details of external and internal interfaces

5.7.1 Design Concerns

This Interface view description of the software serves as a binding contract among designers, programmers, customers, and testers. Hipokratis Project will be introduced to the user by this interface viewpoint.

5.7.2 Design Elements

HipokratisWeb, HipokratisProcessor, OpenCV and Database will communicate with each other through FileInfo and ImageData interfaces as stated in component diagram shown in chapter 5.3.3.

5.7.3 Example Languages

Component diagram of the system is supplied in the subsection 5.3.3. You can refer to that subsection for the diagram.

5.8 Interaction Viewpoint

5.8.1 Design Concerns

In this interaction viewpoint, the way that the system components interact with each other is explained.

5.8.2 Design Elements

Runner, Image, IOperation, GrayscaleOperation, MorphologicalOperation, EdgeDetectionOperation, GaussianFilterOperation, ShapeBasedOperation , Controller, User, Repository, FileInfo, Manager, upload(), process(), download(), submit(), run(), doOperation(), search(), create(), setOperation() .

5.8.3 Example Languages

UML Sequence Diagrams for Upload:

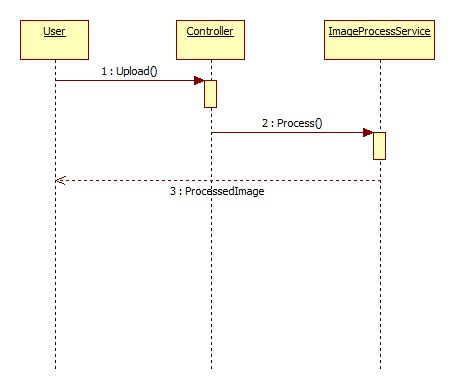


Figure 7 – Sequence Diagram I

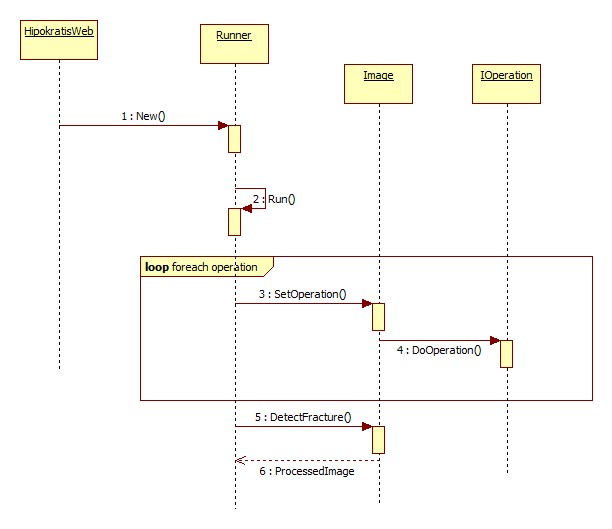


Figure 8 – Sequence Diagram II

UML Sequence Diagram for Download:

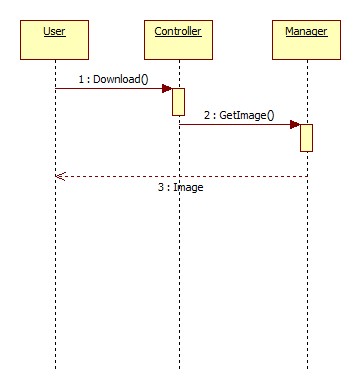


Figure 9 – Sequence Diagram III

UML Sequence Diagram for Search:

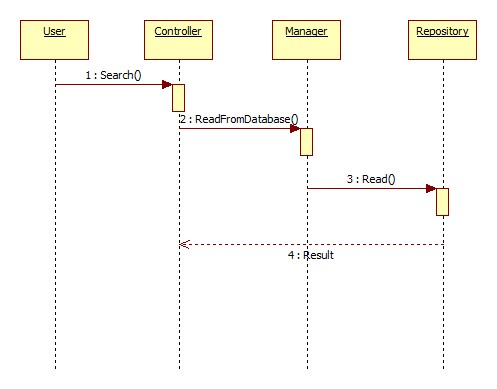


Figure 10 – Sequence Diagram IV

UML Sequence Diagram for Submit:

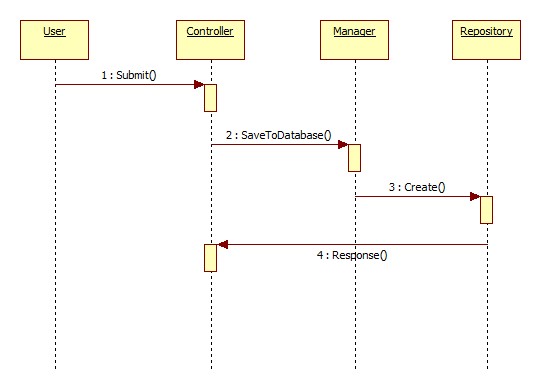


Figure 11 – Sequence Diagram V

5.9 State Dynamic Viewpoint

5.9.1 Design Concerns

In this subsection, system dynamics including modes, states, transitions, and reactions to events are described using UML statechart diagram, in terms of state dynamics viewpoint.

5.9.2 Design Elements

Design Entities: Design elements are start state, starting page state, main page state, resulting image state, search results state, save image state and end state.

Design entities and design relationships could be observed using the state transition diagram in the following subsection.

5.9.3 Example Languages

UML statechart diagram is as follows,

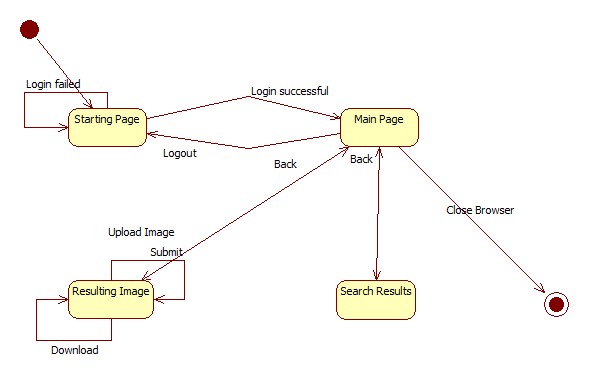


Figure 12 – Statechart Diagram