Project Requirements Specification

Course: CS 474 Object-Oriented Design

Project Name: RADIS

Team Members
Andrew McClellan
Austin Traverse
Kelsey Hilton

Professor: Rob Byrd, iSchool, Abilene Christian University

Last updated: 18 April 2012

Document Revision Number: 4

RADIS

1. Introduction

In this document we describe the requirements for a Radiological Application for diagnostic Imaging and Storage, which we will here by refer to by RADIS. First we describe the system concept and define its scope. Then we provide detailed requirements derived from user stories, constraints, and client specifications. Finally we provide a data model and other information management specifications. Refer to Appendix A for a glossary of terms and acronyms used in this document.

SYSTEM CONCEPT

2. Business Case & Project Vision

For the purposes of our project we are assuming that Hendrick Medical System, the client, has a need for a Radiological Application for Diagnostic Imaging and Storage. The application is intended for use by radiology specialists as a diagnostic assistant. RADIS is a type of electronic health record (EHR) system. It will maintain a database of patient history but also have image processing capabilities to aid radiologists and physicians in making a diagnosis and treating patients. RADIS will analyze images and use doctor confirmed patient history to provide a possible diagnosis. The system will also list alternate diagnoses which will be ranked on probability of correctness.

RADIS will use past history, and standard data compiled over time from a variety of patients to compare to patient data and confirm a diagnosis or come up with an alternate diagnosis that may have otherwise been missed. An accurate diagnosis means that the patient won't need to return when symptoms persist as a result of an inaccurate diagnosis. There will be fewer frustrated patients who are more likely to return and/or recommend Hendrick Medical System.

3. Stakeholders

Comment [K1]: When you are done with this section, you should have a very clear idea of who you need to talk to in order to gather requirements

Stakeholders include:
Rob Byrd, our professor, who defined the goal, guides the design, and assigns a grade.
Radiologists, the primary users of the application.
Hendrick Medical Center, the client for whom the application is being developed.

SYSTEM REQUIREMENTS

4. System Actors

Comment [K2]: . For each actor you identify and list in this section, please give a one-liner description of the actor in this section of the document, and then fully describe the actor in Appendix B using an Actor card (all blank forms can be found at the end of this document)

Radiologists: the primary user of RADIS.
Patients: the secondary user of the system.

Actors represent anything that interacts with the system; they can be human or external systems. This section lists of actors and brief descriptions. Refer to Appendix B for actor cards with full details.

Comment [K3]: Make sure to have an appendix B!

5. Scope of Work

The goal of this project is to develop a diagnostic assistant for radiologists at Hendrick Medical Center. Specific objectives have been identified with respect to the project.

1. Develop an API that incorporates image processing into an automated diagnostic help tool. The API should be open-source and reusable.
2. Identify specific type of images and their normalcy using edge-detection and histogram features.
3. Identify different diseases from a combination of patient-reported symptoms and processed image results.
4. Create a user interface to demonstrate functionality of the underlying API should follow the following business process model(BPM):
    - Starts with a patient needing to be diagnosed.
    - The patient explains symptoms.
    - Radiologist enters symptoms into patient history.
    - Radiologist determines if a scan is necessary.
    - Assistant takes images of patient with medical imaging equipment.
    - Image is processed and returns a result.
    - A report generated from image result and patient history suggests several possible diagnoses.
    - The radiologist confirms a diagnosis.
    - Diagnosis is saved in the patient's history.

Comment [K4]: At this point we are interchanging radiologists and doctors.

For further details on the BPM, please refer to visual aid in Appendix C. The visual model represents the main business process that the system will automate.

6. Functional Requirements

Use cases helped identify functional requirements for the development of the dA API and the corresponding user interface. Below is a list of use cases with a brief description of the requirements that were inspired by each.

UC1 describes the basic flow of an exam.

- The user should be able to input patient history.
- The user should be able to input an image.
- The system should return a result from the image processing.
- The system should interpret the image result and patient history to come up with a diagnosis.

UC2 outlines the process of adding a new patient to the Patient table in the database.

- The user should be able to select what type of image is to be processed.
- The user should be able to create a new patient object in a database.

UC3 describes the data entry process.
UC4 handles data management for the actual image processing.

- The user should be able to input new data.
- The system should be able to use the input data.
- The system should provide the user with a way to update existing data.
- Extraneous data should be destroyed.

UC 5 provides an elaborated exam flow.

- The user should be able to input patient history.
- The user should be able to input an image.
- The system should return a result from the image processing.
    - The result should be in the form: “normal or abnormal”
- The system should interpret the combination of the image result and patient history to come up with a diagnosis.
- The application should produce a report.
    - The report should include all previous history.
    - The report should include the original image
    - The report should include the processed image.
    - The result should include the result of the processed image.
- The report should include a diagnosis.
    - All diagnoses should be presented with justification.
        - The justification should list matching symptoms and matching image indications.
    - The report should include a list of alternate diagnoses.
        - The list of alternate diagnoses should be ordered by probability of correctness.

The use case diagram for these functional requirements can be found in Appendix D. Appendix E shows which use cases support which business process steps. Please also refer to Appendix F for the corresponding use cases with full details.

7. Non-Functional Requirements

Functional requirements are important because they affect what the system does. Non-functional requirements, on the other hand, describe the capabilities of the system. Oftentimes, non-functional requirements impact the system’s cost and must therefore, be clearly communicated with the customer. As such, non-functional requirements are specific and measurable.

The non-functional requirements listed may not be implemented due to time-constraints. The purpose of recording them here is to demonstrate that we did, in fact, think about these requirements and we now know how important it is to be precise and realistic with the client.

7.1. *Look and Feel Requirements*

7.2. *Usability Requirements*

7.3. *Performance Requirements*

7.4. *Operational Requirements*

7.5. *Maintainability and Portability Requirements*

7.6. *Security Requirements*

7.7. *Cultural and Political Requirements*

7.8. *Legal Requirements*

8. Mandated Constraints

The application will be from the perspective of a radiologist.

No more constraints have been identified at this time.

9. Relevant Facts and Assumptions

9.1. *Facts*

9.1.1. The CVIP API will only operate in Visual Studio.
9.1.2. CVIP is written in C and must therefore be wrapped in C++ to achieve hierarchical class structuring.

9.2. *Assumptions*

9.2.1. The computers must run a Windows platform and have Visual Studio installed.

Comment [K5]: There are more assumptions let's not kid ourselves

INFORMATION MODEL

10. Data Entities and Relationships

Several entities have been identified in the requirements. The attributes of each entity are listed below.

Patient: contains a record of each patient, which includes the patientID, patientName, symptoms, and diagnosis, among other attributes.
Diagnoses: contains a list of diagnoses represented by a list of common symptoms.
Image: contains histogram features, and edge-detection results of a base image.

Comment [K6]: This section is not being clearly defined. Sorry. ☹

Comment [K7]: This might be in the diagnosis table. But maybe not.

Patient-Diagnoses: One diagnosis may have many patients but a patient may only have one diagnosis.

The data entities listed in this section and their relationships are illustrated in the data model in Appendix G.

The interaction between the use cases described in the Functional Requirements above and the database tables listed in this section is represented in the matrix in Appendix H.

APPENDIX A
Glossary of Terms and Acronyms

CVIP: Computer Vision Image Processing, the foundational API for this project.

CRUD: C, R, U or D, depending on whether a use case creates, reads updates and/or deletes data.

Histogram:

Comment [K8]: Include here or not?

BPM: Business Process Model, a modeling method to describe the scope of the work to be done by the development team.

Edge-detection:

APPENDIX B
Actor Cards

Actor Specification		
Actor Name: Image Processor		
Type: Secondary	**Personality:** Boring	**Abstract:** Yes
Role Description: The image processor is a system rather than a person. It is responsible for running various tests on a given image and returning a result.		
Actor Goals: • Perform edge-detection • Compare with base data • Return a result		
Use Cases Involved with: • • •		

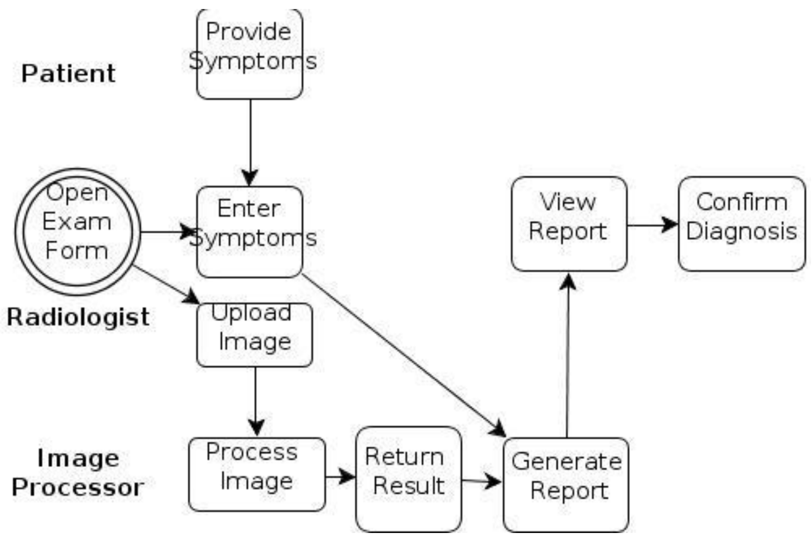
Comment [K9]: Reword, this is confusing.

Actor Specification		
Actor Name: Dr. Oz		
Type: Primary	**Personality:** Extroverted	**Abstract:** Yes
Role Description: Dr. Oz is a radiologist who works at Hendrick Medical Center. He is the user with the most interaction with the system.		
Actor Goals: • User input • Confirms final diagnosis		
Use Cases Involved with: • Every use case involves Dr. Oz. He is the primary actor because he is the main user.		

Actor Specification		
Actor Name: Jane Doe		
Type: Secondary	**Personality:** Introvert	**Abstract:** Yes
Role Description: Jane Doe is a patient who is visiting the radiologist because she is ill. She interacts with the system through the radiologist. She provides all of the necessary input data.		
Actor Goals: • • •		
Use Cases Involved with: • • •		

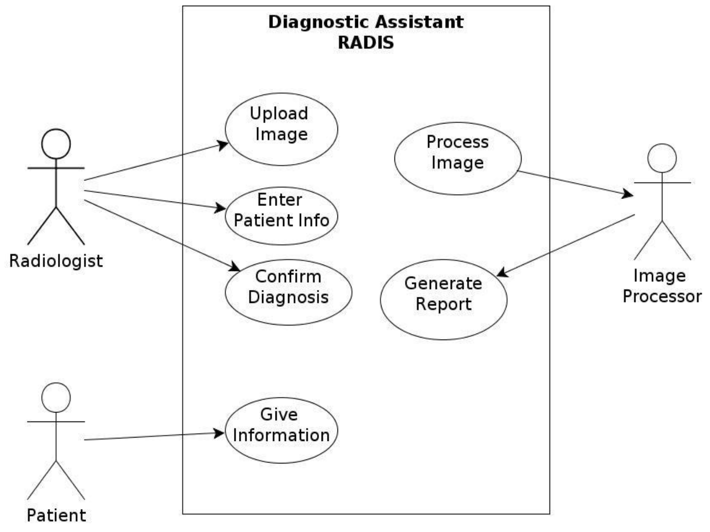
APPENDIX C

Scope of Work: Business Process Model



APPENDIX D
Use Case Diagram

Arrows from an actor to a use case indicate that the actor triggers the use case, whereas arrows from the system to an actor indicate that the system triggers the use case.



APPENDIX E
Transitional Matrix: BPM and Use Cases

BPM Steps	Use Case					
	UC 1	UC 2	UC 3	UC 4	UC 5	
Enter Patient Information	X	X	X	X	X	
Upload Image	X		X	X	X	
Save and Copy Image					X	
Process Image					X	
Display Report					X	
Confirm Diagnosis	X				X	

APPENDIX F
Transitional Matrix: CRUD (Create, Read, Update and Delete)

This matrix lists one column for every use case and a row for every database table. The cells contain the letters C, R, U or D, depending on whether the use case creates, reads, updates and/or deletes records in the corresponding table.

Entity	Use Case					
	UC 1	UC 2	UC 3	UC 4	UC 5	
Patient	C,U	C	C	C,R,U	C,U	
Image	C		C		C,R,U	

APPENDIX G
Use Cases: Functional Requirements

Use Case ID	UC1
Use Case	Simplified Exam
Elaboration	Basic flow of an exam
Actors	Patient, Physician
Description	A patient comes into the office for an exam. The physician enters patient data into the diagnostic assistant application and it returns a probable diagnosis. The physician confirms the diagnosis. The diagnosis is saved in patient's history. The physician treats the patient.
Priority	High
Non-Functional Requirements	(only those associated with this use case, if any)
Assumptions	
Source	

Use Case ID	UC2
Use Case	Add Patient
Elaboration	Base use case
Actors	Physician, Patient
Description	Add patient to database
Pre-conditions	The patient is a new patient
Flow of Events	1. Click add patient button 2. Enter first name, last name, date of birth, physician 3. Automatically assign patient ID number 4. Enter family history 5. Enter past history 6. Enter present medical problems 7. Enter allergies 8. Enter current medications 9. Save patient to database and return to main
Post-conditions	Patient history is stored in database
Alternative Flows	
Priority	High
Non-Functional Requirements	(only those associated with this use case, if any)
Assumptions	No one has the same name.
Outstanding Issues	Given time constraints there will not be an actual database

Comment [K10]: briefly describe alternative flows here for base Use Cases; extend this into complete conditional flows of events in the corresponding Elaborated Use Cases)

Use Case ID	UC3
Use Case	Data entry
Elaboration	Base use case
Actors	Physician, Patient
Description	Describe reason for visit and enter data necessary for diagnosis
Pre-conditions	The patient exists in the database.
Flow of Events	1. Find Patient in database 2. Enter date of exam 3. Enter name of physician 4. Enter symptoms 5. Upload image
Post-conditions	The reason for visit is recorded
Priority	High
Non-Functional Requirements	(only those associated with this use case, if any)
Assumptions	Symptoms will match exactly with those stored in database. Every entry is required except the image.

Use Case ID	UC4
Use Case	Exam Form
Elaboration	Base use case
Actors	Physician, Patient
Description	Describe reason for visit prepare for diagnosis
Pre-conditions	The patient exists in the database.
Flow of Events	6. Find Patient in database 7. 8. Enter first name, last name, date of birth, physician 9. Automatically assign patient ID number 10. Enter family history 11. Enter past history 12. Enter present medical problems 13. Enter allergies 14. Enter current medications 15. Save patient to database and return to main
Post-conditions	Patient history is stored in database
Alternative Flows	
Priority	High
Non-Functional Requirements	(only those associated with this use case, if any)
Assumptions	No one has the same name. Capitalization matters.

Comment [K11]: briefly describe alternative flows here for base Use Cases; extend this into complete conditional flows of events in the corresponding Elaborated Use Cases)

Use Case ID	UC4
Use Case	Image Processing
Elaboration	Base use case
Actors	Physician, Computer
Description	This use case handles standard data management events for image processing
Pre-conditions	A patient history and image exist
Flow of Events	**<u>Create Data</u>** C1. Physician uploads image C2. The original image is saved C3. The image is copied C3. The copied image is normalized(grayscale, number of pixels, contrast) C4. Preform histogram features, and edge detection **<u>Read Data</u>** R1. Compare numerical values to values that are considered standard values for the image type. These standard values exist in a database. R2. If the values are similar, return "normal" as a symptom R2.1 Else return "abnormal" as a symptom R3. Compare the patient's symptoms, which now include the result of the image processing, and compare to the diagnosis database. Return the diagnosis with the highest number of matching symptoms. R4. Return a list of alternate diagnoses ordered by the number of matching symptoms. R5. Physician confirms or enters his own diagnosis **<u>Update Data</u>** U1. Save the original and processed image to patient history. U2. Save diagnosis to patient history. **<u>Delete Data</u>** D1. Delete any intermediate copies of the processed image.
Post-conditions	An image is processed and saved in patient history
Priority	High
Assumptions	An existing API contains image processing functions

Comment [K12]: (Some Use Cases do very little processing and a lot of data management; these use cases don't have a lot of detailed flows to describe, but instead they contain a lot of data management steps – e.g., create, read, update and/or delete data; since database application developers know exactly what to do in such cases, it is often simpler to specify your flows broken down by CRUD activities and using simpler flows; since you will later need to use these use cases to identify data objects, the following use case format is appropriate for these use cases)

Use Case ID	UC5
Use Case	In-depth exam
Elaboration	Elaborated use case
Actors	Physician, Patient
Description	The physician goes through the exam recording symptoms and adding images to be processed. The information is then analyzed and results are displayed in a report.
Pre-conditions	A patient comes into the office for an exam
Flow of Events	1. Enter Patient name 2. If patient does not exist in database 2.1 Add new patient 3. Enter date of Exam 4. Enter reason for visit 5. Enter symptoms 6. If an image is needed for diagnosis 6.1 Upload image 6.2 Save and copy original image 6.3 Process image copy 6.4 Save processed image 6.5 Return image result as a symptom 7. Click generate report button to find a diagnosis 8. Display report with original and processed images as well as a diagnosis and a list of alternatives 9. Have physician confirm diagnosis 10. Save diagnosis and images to patient history
Post-conditions	Patient is diagnosed and the diagnosis and images are saved in patient history.
Alternative Flows	1. An image may not be necessary depending on the reason for visit in which case the upload should be ignored. 2. A physician may not agree with any of the provided diagnoses in which case he should be allowed to enter his own.
Priority	High
Non-Functional Requirements	(only those associated with this use case, if any)
Assumptions	Necessary images already exist.

APPENDIX H
Data Model

Use Case ID	(same ID as the base Use Case ID, but with suffix A1, A2, etc., for each alternative flow Use Case)
Use Case	
Elaboration	(indicate whether it's initial, base or elaborated use case)
Actors	
Description	(brief)
Insertion Point	(step in Base Use Case where this alternative flow should be inserted)
Pre-conditions	(clearly indicate under which conditions the alternative flow is triggered)
Alternative Flow of Events	1. 2. 3. 4. 5.
Post-conditions	
Priority	(High, Medium or Low)
Non-Functional Requirements	(only those associated with this use case, if any)
Assumptions	
Outstanding Issues	
Source	

Form for Extending Use Cases

(An Extending Use Case MUST be associated with an existing Elaborated Use Case)

Use Case ID	(same ID as the base Use Case ID, but with suffix E1, E2, etc., for each extending Use Case)
Use Case	
Elaboration	(indicate whether it's initial, base or elaborated use case)
Additional Actors	(only list actors not listed in the extended use case)
Description	(brief)
Extended Use Case	(ID and name)
Extension Point	(step in Base Use Case where it is extended)
Guard Condition	(pre-condition in the extended Use Case that causes it to execute)
Flow of Events	1. 1.1 1.2 2. 3.
Conditional Flows	4. 5.
Post-conditions	
Alternative Flows	(briefly describe alternative flows here for the Extending Use Case; extend this into complete flow of events in an Alternative Use Case form if needed)
Priority	(High, Medium or Low)
Non-Functional Requirements	(only those associated with this use case, if any)
Assumptions	
Outstanding Issues	
Source	