

Spring 2016

A Software Application for Cardiac-Gated Computerized Tomography Scanning

Stephen Caldwell


University of Akron, src59@zips.uakron.edu

Trevor Engelsman

University of Akron, te12@zips.uakron.edu

Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

Follow this and additional works at: http://ideaexchange.uakron.edu/honors_research_projects

 Part of the [Bioimaging and Biomedical Optics Commons](#), [Biomedical Commons](#), [Biomedical Devices and Instrumentation Commons](#), and the [Systems and Integrative Engineering Commons](#)

Recommended Citation

Caldwell, Stephen and Engelsman, Trevor, "A Software Application for Cardiac-Gated Computerized Tomography Scanning" (2016). *Honors Research Projects*. 277.

http://ideaexchange.uakron.edu/honors_research_projects/277

This Honors Research Project is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

A Software Application for Cardiac Gated Computerized Tomography Scanning

Trevor Engelsman, Stephen Caldwell

Department of Biomedical Engineering

Honors Research Project

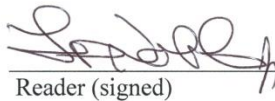
Submitted to

The Honors College

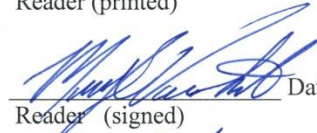
Approved:

 Date 4/28/16
Honors Project Sponsor (signed)

JAMES KESZENHEIMER
Honors Project Sponsor (printed)


 Date 4/29/16
Reader (signed)

L.D. NOBLG JR.
Reader (printed)

 Date 4/28/16
Reader (signed)

MARY C. VERSTRAETE
Reader (printed)

Accepted:

 Date 4/28/16
Department Head (signed)

BRIAN DAVIS
Department Head (printed)

 Date 4/28/16
Honors Faculty Advisor (signed)

MARY C. VERSTRAETE
Honors Faculty Advisor (printed)

Date _____
Dean, Honors College



Senior Design Project Report

April 24th, 2016

Authors:

Trevor Engelsman, Rayan Alabsi, Logan Huba, Stephen Caldwell, Ian Harker

I. Introduction

Computerized tomography (CT) scanners use X-rays and the aid of a computer to combine cross sectional images of sections or organs in the body to generate 3-dimensional images. FMI Medical Systems has requested that our design team put together a software application that can take data from an IVY Biomedical 7800 Electrocardiography (ECG) recorder, and use this information to initiate the CT scanner to take pictures of the heart during the quiescent periods (Figure 1, Figure 2). The purpose of this application is to improve the overall image quality for the CT scan by removing motion artifacts during scanning. Taking images of the heart during a period where there is little to no movement allows for the clearest pictures and this process is called cardiac gated CT scanning. Our design group saw this as an opportunity to use our knowledge of Matlab and basic programming techniques learned in our classes and apply them to a real world application.

II. Background Information

FMI is a Chinese owned company and their CT scanners are sold in the Chinese market exclusively as a reliable, inexpensive alternative to the larger brand name companies. Currently the major competitors to FMI such as Siemens and GE Healthcare have the ability to perform cardiac gated scans in their current systems and FMI is looking to match these expectations in their market. FMI's research and development department is looking to use this application with their new 64 slice CT scanner.

The purpose of cardiac-gated scans is to have an application analyze the ECG data received by an attached ECG recorder and determine the quietest and calmest portion of the heart rhythm, and to trigger an exposure during that time period. The quiescent period is estimated to be at a certain percentage, 70% as defined by FMI, of the R-R peak time of a heartbeat. The R-R peak time is the time between the QRS complexes of two different heartbeats (Figure 3). The idea here is that at 70% of the R-R peak time after the first R peak there exists a window that the heart is simply still and provides an optimal moment for taking an image.

III. Project Objectives/Goals

The scope of the project was to develop a CT scheduler software application that would determine the quiescent period of a heart beat from ECG data, determine if a cardiac gated scan is possible, and issue a command to the CT scanner to take an image during the specified time. In addition to creating this application that scheduled when the CT scanner would take an image, we needed to create an application (CT simulator) that emulated the different functions of a CT scanner. The addition of this second application is because the current CT system is still in development by FMI, and would be unavailable for testing. The CT simulator needed to be able to communicate with the CT scheduler application and provide the same functionality that a real CT scanner would supply when considering the interactions between the two applications. FMI provided our team with an ECG simulator that replicated the functionality of an IVY Biomedical ECG Recorder that produced that ECG data used in the CT scheduler. At the completion of the semester we are to provide FMI with a functional application that

meets their requirements as well as documentation of how we followed the biomedical design process.

IV. Methods and Procedures

Our team decided to use an incremental approach to create the software application. This approach allows software features to be added incrementally in small development (iterative) cycles that are repeated until the entire functionality of the application is achieved. Using this process, our group was able to divide the different parts of the process with each member working on a different part, completing each task and creating or updating the necessary documentation.

For our development process, we broke each incremental period into nine parts.

1. Select customer requirements to implement
2. Derive Engineering requirements from customer requirements
3. Define resulting use-cases
4. Perform software analysis (Class Diagrams/Sequence Diagrams)
5. Develop Verification plan
6. Design Code
7. Implement Code
8. Verify Code (engineering requirements)
9. Validate customer requirements

At the start of each cycle we selected a customer requirement(s) that we looked to complete. From here we would look to derive engineering requirements for this customer requirement. The engineering requirements describe what the application must be able to do in order to successfully meet the customer requirement.

From these engineering requirements we used Object Modeling Technique (OMT) and Unified Modeling Language (UML) to find how each part of the application is used by a user, and to analyze the processes. OMT and UML provide a standardized approach to software design and help to keep the process organized as we completed the application. The first component of this software design was use-case diagrams. These diagrams describe what the user is capable of doing during each iteration as well as the different events that the user experiences. Our team followed UML procedures as well as work done by Alistair Cockburn in order to develop the use-case diagrams which detailed: a general description, triggers, actors, preconditions, goals/outcomes, failure modes, and steps of execution.

Next, we used different forms of software analysis to ensure that our engineering requirements included all the necessary details to successfully meet the customer's requirements. Class diagrams were used to describe what must be included in the various objects within a system from a static view point. They include the various actions that each object is able to perform (functions) and what is being passed in the functions (variables). Sequence diagrams provided a dynamic approach, describing what events take place as well as the order in which each event occurs. These diagrams also list the different functions being performed at each step, giving a logical walkthrough of what happens. With the use of these diagrams, we can find any missing requirements or scenarios, reducing the time spent on reworking or implementing features down the road.

With these documents and diagrams completed we moved onto developing a verification plan for each of the engineering requirements and use-case diagrams. Each

verification plan detailed the different ways we would ensure that our engineering requirements, as well as giving a procedure on how to complete it. Next, we moved to designing and implementing the code for the application. We took the information from the previous steps and came up with how to accomplish the requirements in code form. After developing an outline of how the code would work, we worked to translate it into working code.

The last two steps to our software development process dealt with verification and validation of the code. First we used the verification plan to work through the code and make sure that each of the engineering requirements were met and worked as intended. After these engineering requirements were verified, we validated that the outcomes of the code met the original customer requirement that we set out to complete.

Having different objects that we needed to develop, we broke the project into a series of these incremental development cycles. Each cycle was planned to last for one week with an additional five days for review of the cycle, and updating of documentation. After the first week our team began to work on the next iteration cycle. We broke our project down into 7 of these cycles: three for the CT simulator, three for the CT scheduler, and one cycle that dealt with only messaging between the different applications.

V. Outcomes

We have successfully completed making the CT simulator that emulates the required functionally and processes required to communicate and work with the CT

scheduler application. We have also been able to create an application that can analyze given ECG data and determine the possibility of using a cardiac gated scan. From this the CT scheduler application is able to send a command that tells the CT scanner to take an image of the heart, successfully meeting the requirements set forth by the customer. We will be demonstrating this ability and functionality at the end of the semester. Throughout the process of design, analysis, and verification of our application, we have kept up with documentation of the various documents and diagrams showing how we followed the design process in order to accomplish the project goals. With the remaining time in the semester we will be working to improve various features of the applications, as well as completing the last of the documentation for the development process.

VII. References

1. Banas, Derek. "Object Oriented Design." *YouTube*. YouTube. Web. 09 Feb. 2016. <<https://www.youtube.com/watch?v=fJW65Wo7IHl>>.
2. "UML Diagram Types with Examples for Each Type of UML Diagrams." *Createely Blog Diagramming Articles and Tips on How to Draw Diagrams*. 2012. Web. 09 Feb. 2016. <<http://createely.com/blog/diagrams/uml-diagram-types-examples/>>.
3. Cockburn, Alistair. *Writing Effective Use Cases*. Boston: Addison-Wesley, 2001. Print.
4. "Introductory Programming Courses." *Introductory Programming Courses*. Web. 09 Feb. 2016. <<http://ocw.mit.edu/courses/intro-programming/>>.
5. "Programming Foundations Tutorials | Lynda.com." *Lynda.com*. Web. 09 Feb. 2016. <<http://www.lynda.com/Programming-Foundations-training-tutorials/1351-0.html>>
6. "Object Orientated Programming." *Wikipedia*. Wikimedia Foundation. Web. 09 Feb. 2016. <https://en.wikipedia.org/wiki/Object-oriented_programming>.

Appendices

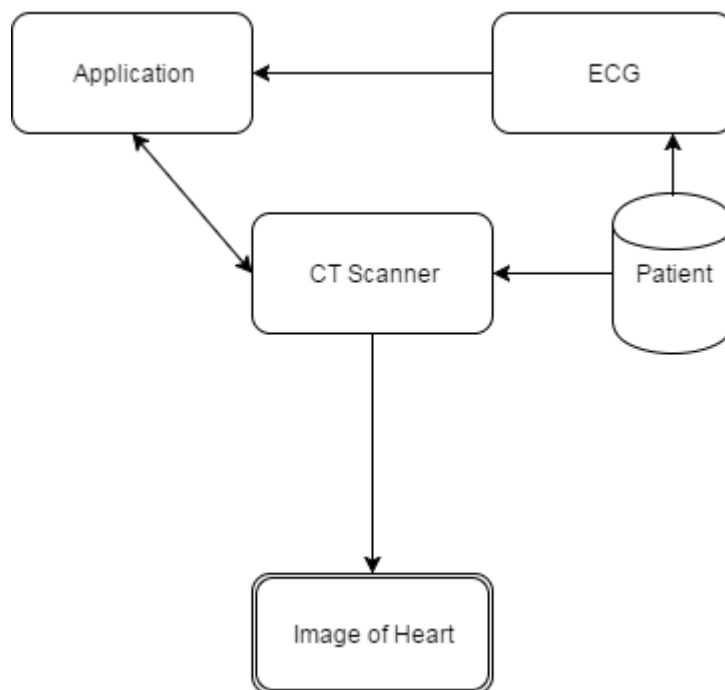


Figure 1. Top Level Diagram of Real System

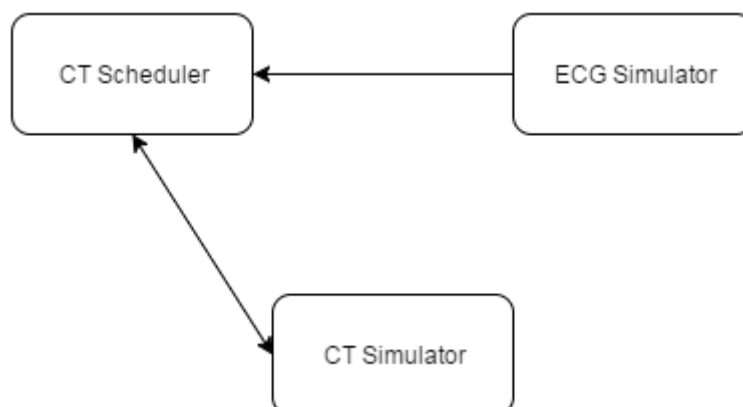


Figure 2. Top Level Diagram of Project Simulated System

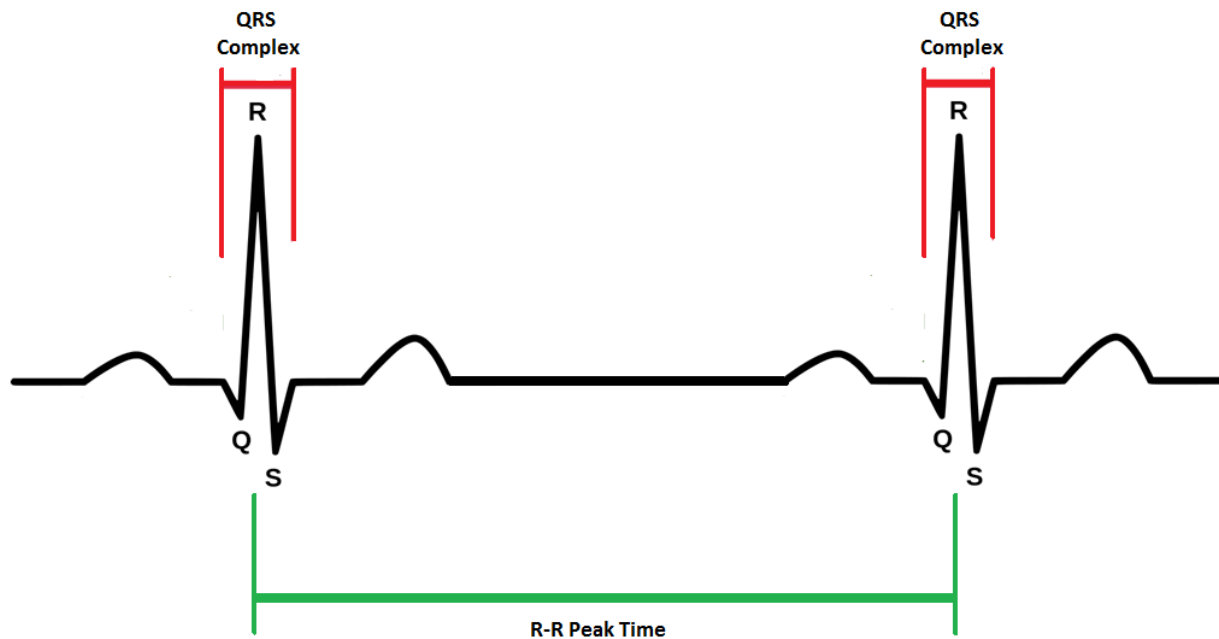


Figure 3. QRS Complex and R-R Peak Time

Functional Requirements

Customer Requirements

Need Criteria - Create a functional software application that can take data from an IVY Biomedical 7800 ECG recorder, and use this information to take exposures during the quiescent periods of the heart rhythm to improve image quality.

Required

1. Use of a high level programming language that supports OMT
2. Develop a trigger system to send an exposure initiate message
3. Exposure initiate is determined to be during the quiescent period of the heart beat (defined as 70% of R-R peak value)
4. Develop software to handle messaging between various software program
5. Develop software to determine of proper possibility of cardiac gated scanning
6. Develop a CT simulator that emulates scan control, x-ray manager and library functionality of a true CT system
7. Design an Operator UI to allow for changes to common CT parameters.
8. Use an IVY biomedical 7800 cardiac Trigger monitor and HE instruments TechPatient Cardio V4 patient simulator to create and monitor patient ECG data.

9. Display the outcome of the cardiac gated scan to a user interface.

Desired

1. Developed using C++
2. Cardiac gated possibility determined within 1 gantry rotation
3. Easy for operator to use

**Use Case
Cycle 1****Customer Requirements**

-Develop software to handle messaging between the various software programs

Use Case Description

1. An operator begins the program by starting the messaging system CoreWinAppYellowPages.

Triggers

1. The CoreWinAppYellowPages.exe has been pressed

Actors

1. CoreWinAppYellowPages
2. CT Simulator
3. CT scheduler
4. ECG simulator

Preconditions

1. An operator is available
2. CT simulator, CT scheduler, CoreWinAppYellowPages, and ECG simulator are installed within Fawkesbase Directory
3. Windows- based OS set up by FMI

Goals

1. Start the messaging system

Steps of Execution

1. Install Fawkesbase directory
2. Have FMI technician set-up for IP registry etc.
3. Install CT simulator.exe, CTscheduler.exe, ECGsimulator.exe, and CoreWinAppYellowPages.exe within file path : Local Disk\Fawkesbase\Release\RunTime
4. Run Core WinAppYellowPages, CTsimulator, CTscheduler, ECG Simulator

Engineering Requirements for Cycle 1**Messaging (General)**

- 1) Sending Messages take the form of ("Verb", "Adj", "Destination")
- 2) Receive messages take the form of ("Verb","Adj")

3) Messages are handled using CoreWinSubFramework.dll

CT Scheduler Engineering Requirements:

Use alongside Code Outline and Table 4 for further details

1. A Diagnostic Log Window is displayed on the startup of CT Scheduler. Any adjustments to variables, messages, or states are logged with timestamps, and printed to Diagnostic Log Window. These changes are looked for every 25 ms (1 tick) for verification purposes.
2. The CT scheduler must be able to receive a "Prepare Initiate" message (Table 3#1) from the CT simulator in order to enter Startup Phase (see 10-15)
3. CT scheduler must be able to send a request for scan parameters (Table 3 #2) to the CT simulator. This is to determine Scan Count in Engaging Phase (see 29-33)
4. CT scheduler must be able to receive response from CT simulator with updated parameters (table3#3) Response will not be immediate due to user input into GUI.
5. CT scheduler must be able to send a "Prepare Complete" Message to the CT simulator (Table 3#4) once Startup Phase is complete (see 15)
6. CT Scheduler must be able to receive "Scan Initiate" message from CT simulator (Table 3#5) in order to enter Recording Phase (see 16-18)
7. CT scheduler must be able to send "Exposure Initiate" message to CT simulator (Table 3#8) this is the culmination of the Engaging Phase (see 29-33)

Use Case Cycle 2

Customer Requirements

- Develop a CT simulator that emulates scan control, x-ray manager and library functionality of a true CT system.
- Design an Operator UI to allow for changes to common CT parameters.

Use Case Description

1. An operator enters required variables to prepare the CT simulator provided by the CT simulator GUI using a computer keyboard. The CT simulator checks, and

confirms these variables, and displays them. The operator then submits these values. The CT simulator is then in the preparations stage.

Triggers

1. An operator selects preferred scan protocols, and submits the requested CT simulator parameters.

Actors

1. An operator who submits required CT simulator variables.
2. A CT Simulator GUI that allows these variables to be entered, and displayed.
3. A CT simulator program that receives these values, and controls messaging between it and CT scheduler.

Preconditions

1. An operator is available.
2. Variables (parameters) are known.
3. Necessary CT simulator program is installed on computer.
4. Computer keyboard and screen is available.

Goals

1. Set the CT simulator in its complete prepared state, and ready for execution.

Steps of Execution

1. An operator is requested to make a scan protocol selection from options displayed on CT simulator GUI.
2. The Operator is requested to enter parameter values using the CT simulator GUI.
3. The operator submits parameter values to the CT simulator.
4. The CT simulator confirms input values.
5. CT simulator receives request for parameters from CT Scheduler.
6. CT simulator sends requested parameters to CT scheduler.
7. CT simulator requests confirmation of received parameters from CT scheduler.
8. The CT scheduler updates "Prepared" status.
9. CT simulator GUI displays "Start Scan" option for selection by CT operator.
10. CT simulator GUI awaits CT operator to select "Start Scan" button.

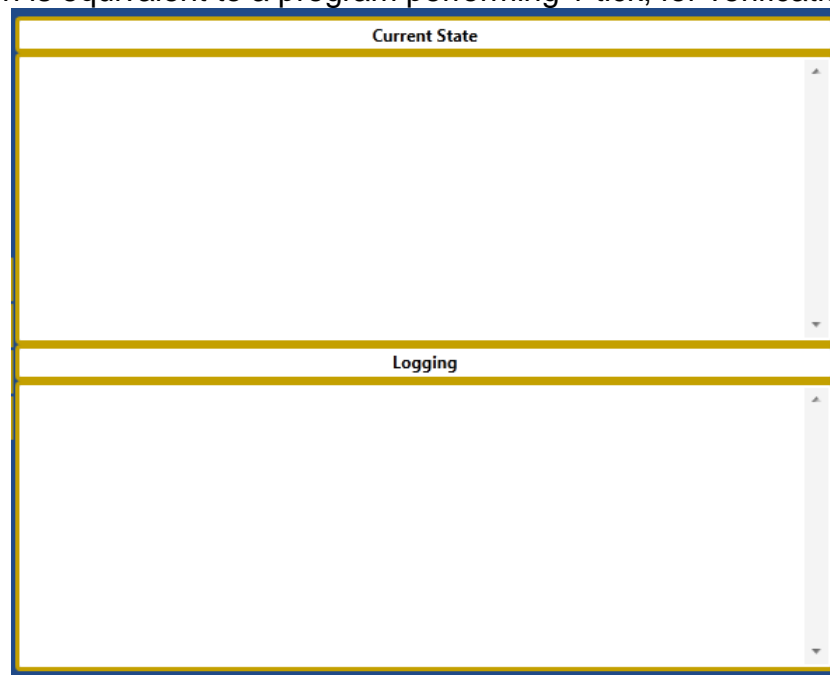
Engineering Requirements Cycle 2

Logging

- 1) Time stamp take the form `##:##:##:###` using hour:minute:second:millisecond
- 2) Verb take the form "`|`" followed by "Verb" followed by "`=`" or ""
- 3) Adjective is a string of any length

CT Simulator


- 1) Load CoreWinSubMessaging.dll and CoreWinSubFramework.dll into CT simulator and computer registry using CoreWinAppYellowPages.
 - a) To perform registry, execute the following functions using the public class cFramework in the CoreWinSubMessaging.dll.
 - i) Functions of gProgramName and gMessagingName should both be assigned with a string variable "CT Simulator", which establishes the messaging destination/origin variable "CT Simulator" for use with CoreWinSubMessaging.dll
 - ii) Function InitializeMessaging(), If registry with CoreWinSubMessaging.dll with a return value of 0, and if it fails it will return a value of 1.
- 2) The Diagnostic Log Window is displayed on the startup of CT Simulator.
 - a) Diagnostic window labeled 'Current State' is where any state transitions, scan variable modifications, and any messages received and sent will be logged for verification purpose.
 - b) Diagnostic window labeled 'Logging' is where current state, scan variable modifications and any messages received and sent will be logged every 25 ms, which is equivalent to a program performing 1 tick, for verification



purpose.

- 3) CT simulator must be able to display data grid boxes for operator to enter variables 1, 2, 7,8,9,10,11 from Table 1. The grid boxes are preloaded with default parameters found in table 1. The user has the option to change these variables within the grid boxes. The user must then press the push button labeled 'submit' to confirm the entry of the variables to the .dll. A 'prepare initiate' message (Table 3 #1) is sent to CT scheduler.

Cardiac Gaited Scans



Parameter	Value
kVp	80
mA	250
Rotation_Speed	120
RR_Percentage	0.6
Scan_Gating_Number	5
Retrospective_Limit	2

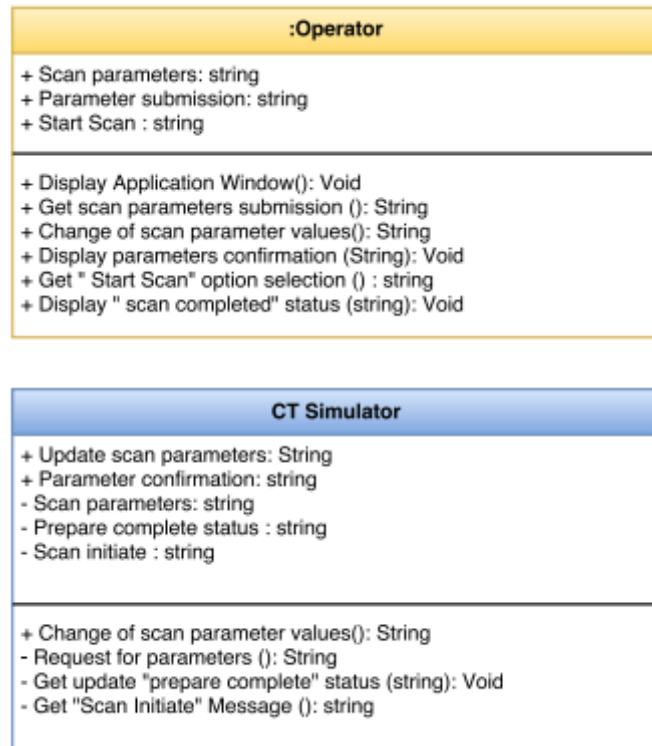
Submit

StartScan

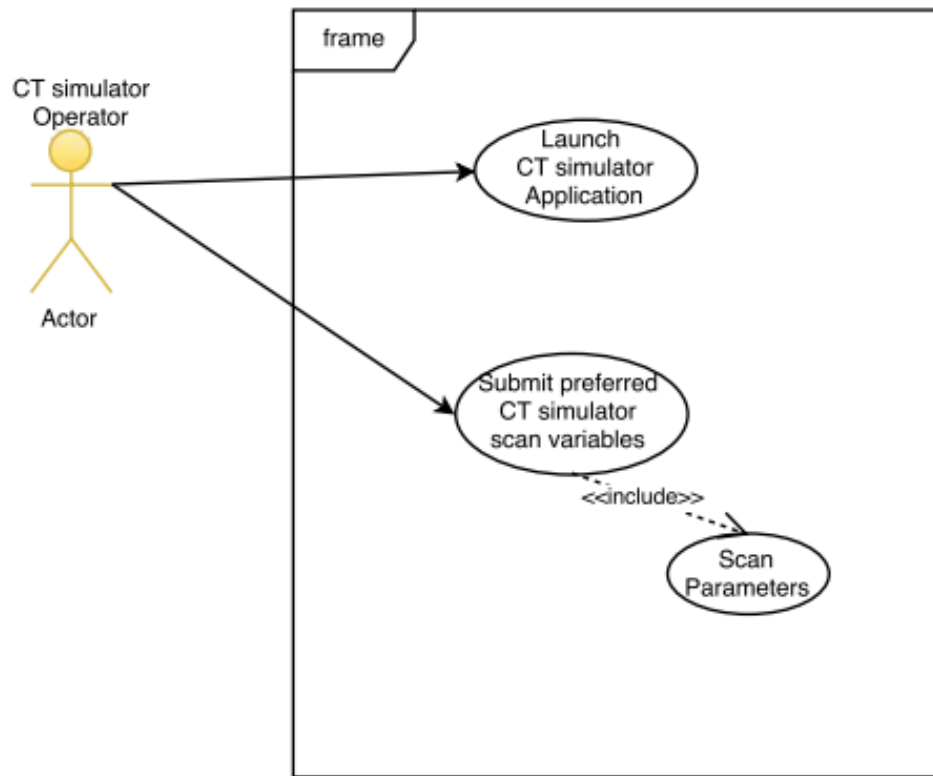
Start Logging

Stop Logging

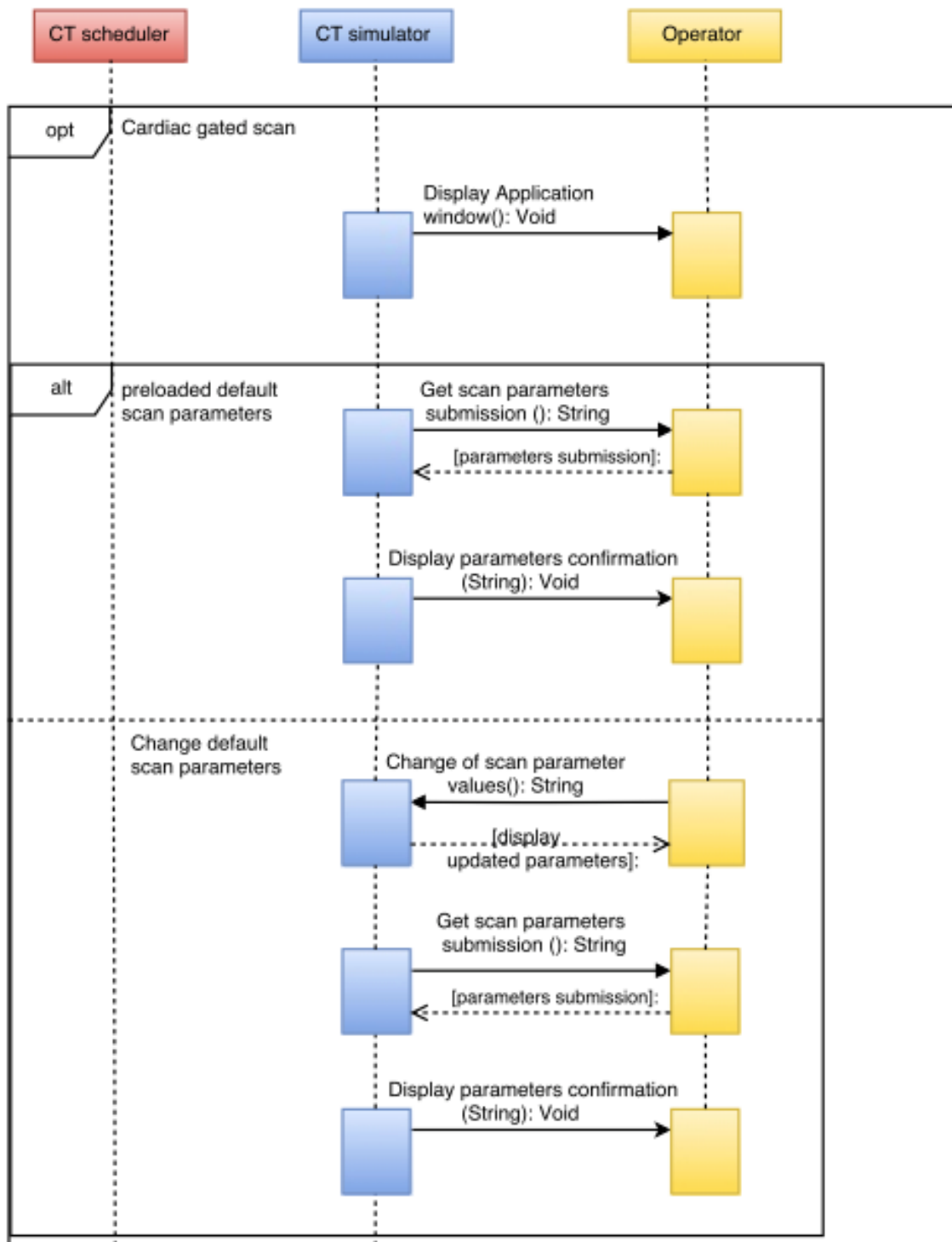
- a) Figure shows data grid boxes for corresponding input protocols
- b) 'Submit' pushbutton is located at the bottom right



Class Diagram



Use Case Diagram



Sequence Diagram

USE CASE

Cycle 3

Customer Requirements

- Develop a CT simulator that emulates scan control, x-ray manager and library functionality of a true CT system.
- Design an Operator UI to allow for changes to common CT parameters.

Use-Case Description

1. CT simulator has confirmed that system has been prepared and received a confirmation message from CT scheduler. Operator then has option to push “start scan” button to put CT simulator into scan state.

Triggers

1. CT simulator has received prepare complete from CT scheduler.

Actors

1. Operator who pushes “start scan” button.
2. A CT simulator GUI that displays a “start scan” button.
3. A CT Scheduler that sends messages to CT simulator.
4. CT simulator that handles messaging between CT scheduler and CT simulator.

Preconditions

1. Operator has submitted valid parameters.
2. “Submit” button has been pressed.

Goals

1. Confirm that system has been prepare and set state of CT simulator into scan state.

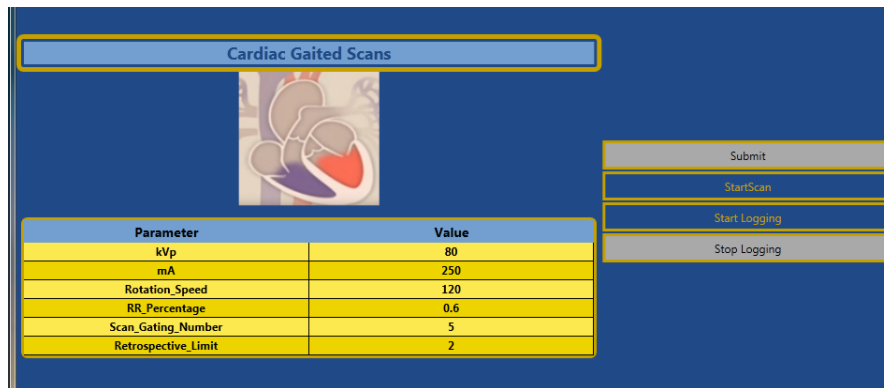
Steps of Execution

1. CT simulator waits for prepare complete message from CT scheduler.
2. CT simulator receives prepare complete message from CT scheduler.
3. CT simulator GUI displays option for start scan.
4. Operator presses start scan.
5. CT simulator is placed into scan state.

Engineering Requirements Cycle 3

Prepare

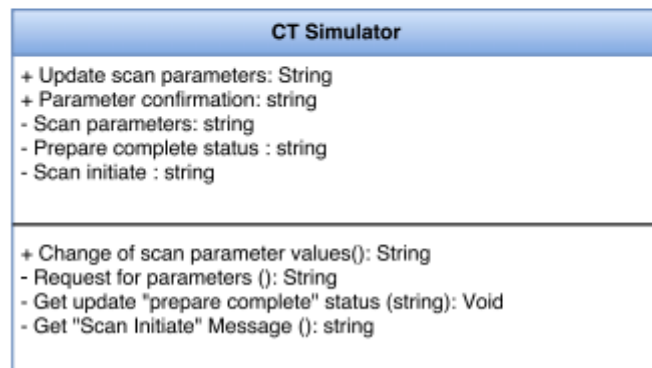
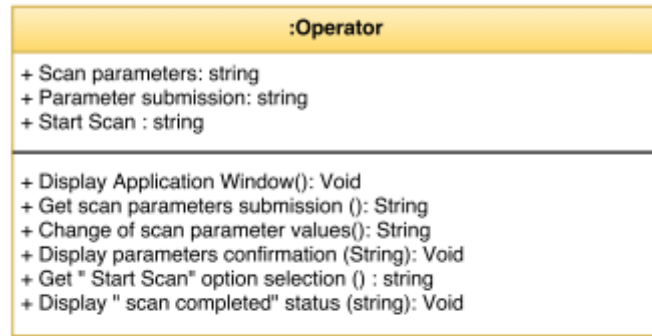
1. CT simulator must be able to receive requests from CT scheduler for parameters (Table 3 #2).
2. CT simulator must be able to send response for CT scheduler parameter requests (Table3 #3)
3. CT simulator must be able to receive request from CT scheduler to update 'Prepare Complete' status, checking for message every 25 ms (1tick).(Table 3 #4)
4. Static text to left of grid boxes showing 'Cardiac Gated Scan'. The CT simulator must be able to display a push button labeled 'Start Scan' once "Prepare Complete" variable has been updated to 'True' in.dll. CT simulator checks if 'Prepare complete' variable has been updated every 25 ms (1tick) (Table 3 #4). Button is located below Submit Button



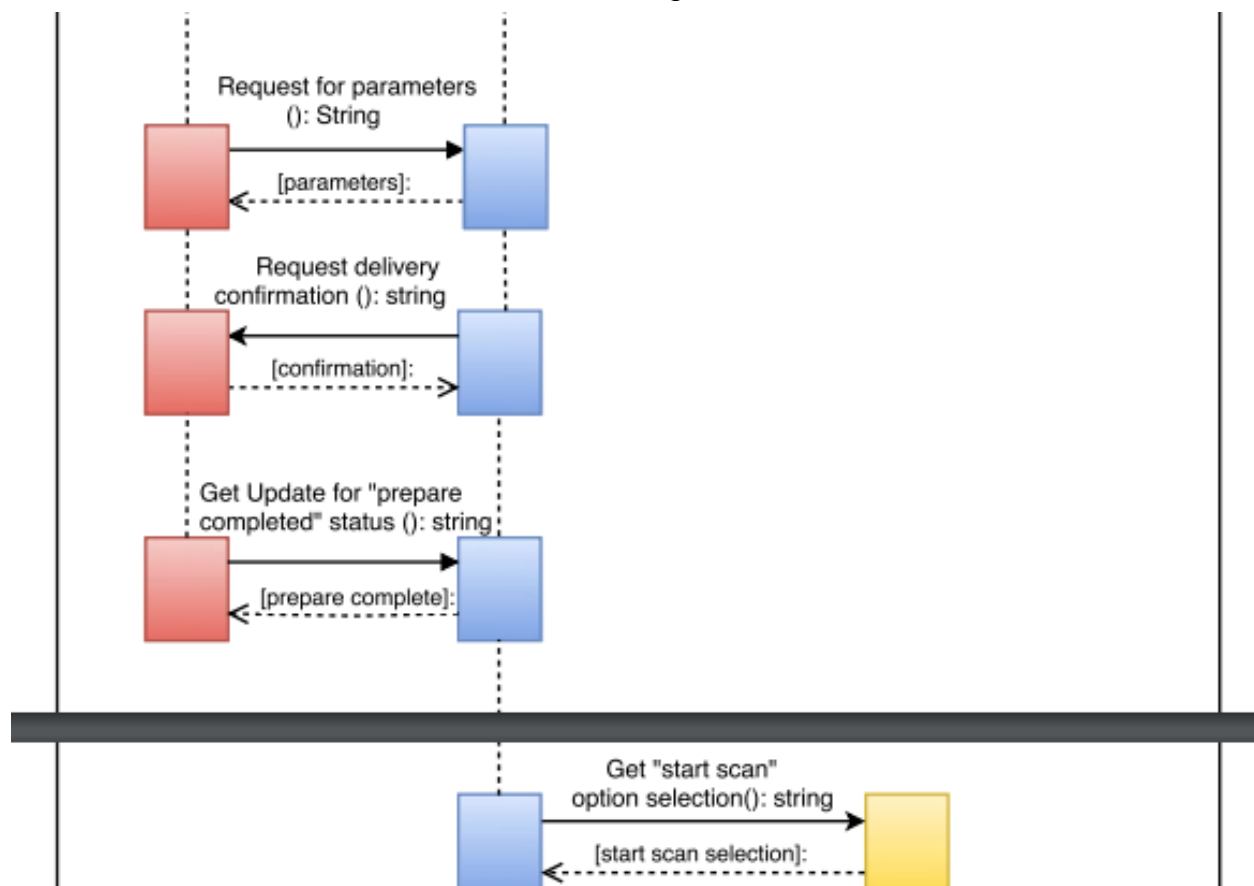
Cardiac Gated Scans

Parameter	Value
kVp	80
mA	250
Rotation_Speed	120
RR_Percentage	0.6
Scan_Gating_Number	5
Retrospective_Limit	2

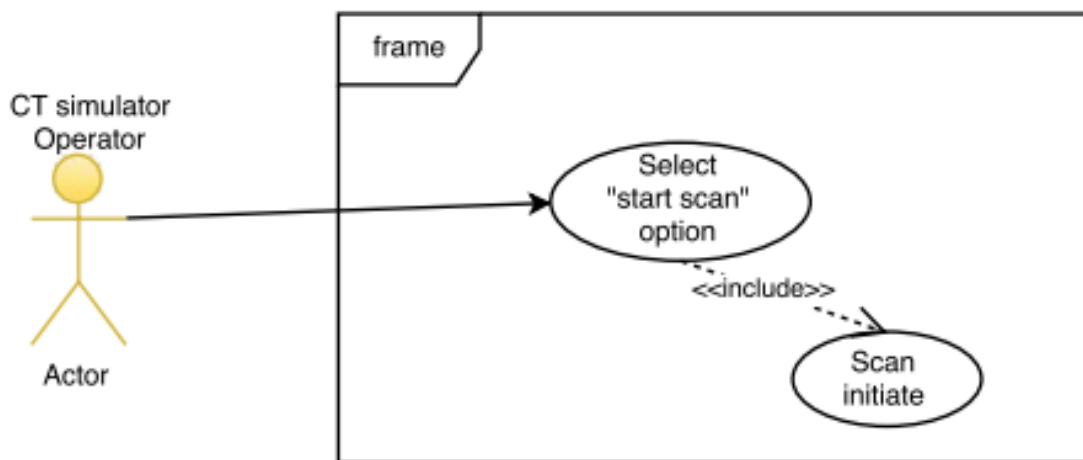
Submit
StartScan
Start Logging
Stop Logging



Class Diagram



Use Case Diagram



Sequence Diagram

Use Case Cycle 4

Customer Requirements

-Display the outcome of the cardiac gated scan to a user interface.

Use-case Description

1. CT simulator has taken the required scan count defined in parameters. CT simulator GUI displays that the scan is complete.

Triggers

1. CT scheduler sends message to CT simulator stating results of scan.

Actors

1. Operator monitoring GUI.
2. CT simulator managing messages with CT scheduler.
3. CT scheduler sending messages to CT simulator.

Preconditions

1. Retrospective limit is not met.
2. Number of required images is met.

Goals

1. Display on GUI that cardiac gated CT scan was successful.
2. CT simulator state is set to idle.

Steps of Execution

1. CT scheduler is in scanning state taking cardiac gated scan images.
2. Scan count is met & retrospective limit was not met.
3. CT scheduler updates that scan is complete and sends message to CT simulator.
4. CT simulator receives message.
5. CT simulator displays that the cardiac gated scan was successful on GUI.
6. CT simulator state is set to idle.

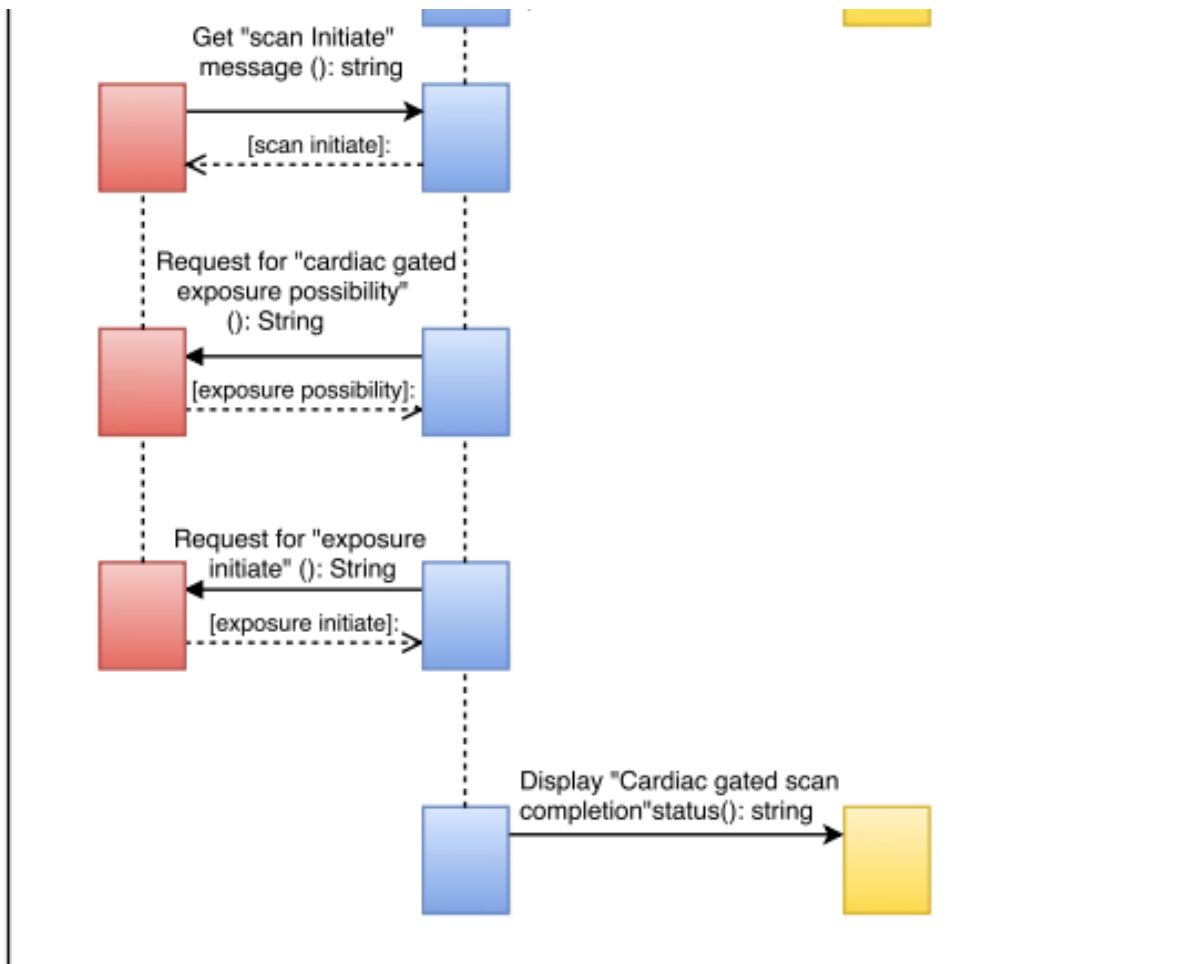
Engineering Requirements Cycle 4

1. CT simulator must be able to accept update messages for cardiac gated exposure possibility' and 'exposure initiate' messages until Retrospective Limit, Scan gating number is reached or abort message received.
2. CT simulator must be able to display status of 'cardiac gated scan completion status' (table 3) on GUI when Retrospective Limit, Scan gating number is reached or abort message received. CT simulator must check to see if any of these options are true every 25 ms (1tick) (Table 3 #9)

:Operator
<ul style="list-style-type: none"> + Scan parameters: string + Parameter submission: string + Start Scan : string
<ul style="list-style-type: none"> + Display Application Window(): Void + Get scan parameters submission (): String + Change of scan parameter values(): String + Display parameters confirmation (String): Void + Get " Start Scan" option selection () : string + Display " scan completed" status (string): Void

CT Simulator
<ul style="list-style-type: none"> + Update scan parameters: String + Parameter confirmation: string - Scan parameters: string - Prepare complete status : string - Scan initiate : string
<ul style="list-style-type: none"> + Change of scan parameter values(): String - Request for parameters (): String - Get update "prepare complete" status (string): Void - Get "Scan Initiate" Message (): string

Class Diagram



Sequence Diagram

Use Case Cycle 5

Customer Requirements

- Develop software to determine of proper possibility of cardiac gated scanning.
- Use an IVY biomedical 7800 cardiac Trigger monitor and HE instruments TechPatient Cardio V4 patient simulator to create and monitor patient ECG data.

Use case description

1. CT scheduler start up and initial data requests.

Triggers

1. CT scheduler program is started.

Actors

1. CT scheduler application shortcut
2. ECG Simulator providing ECG data
3. Local Network Port
4. CT simulator DLL

Preconditions

1. ECG simulator is running.
2. ECG port is registered to DLL.
3. ECG is sending data to local network port.
4. CT simulator is running variables have been submitted.

Goals

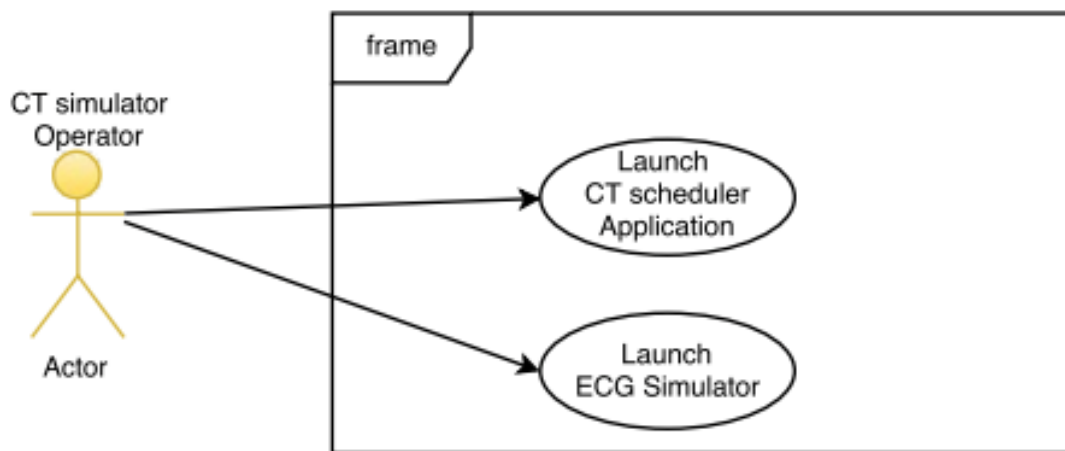
1. CT scheduler starts and looks for ECG data from specified network port.
2. CT scheduler finds ECG data and stores it as a variable.
3. Continuously look for ECG data at port while application is running.
4. CT scheduler receives parameters.

Steps of Execution

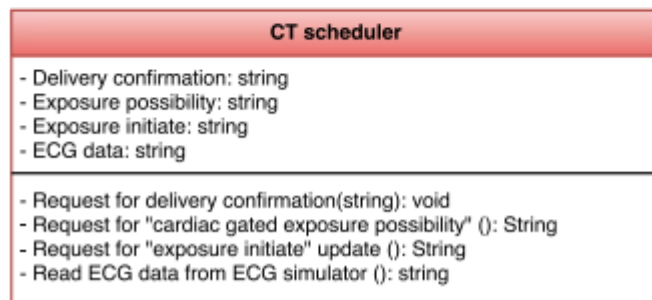
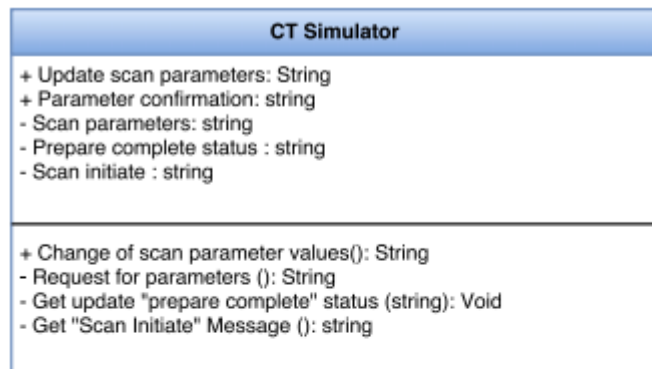
1. Run CT scheduler application.
2. CT scheduler searches for specified local network port.
3. CT scheduler retrieves ECG data from port and stores as variable.
4. CT scheduler continues to retrieve data from port every 25ms.
5. CT scheduler receives request to prepare initiate.
6. CT scheduler requests parameters from CT simulator DLL.

Engineering Requirements Cycle 5

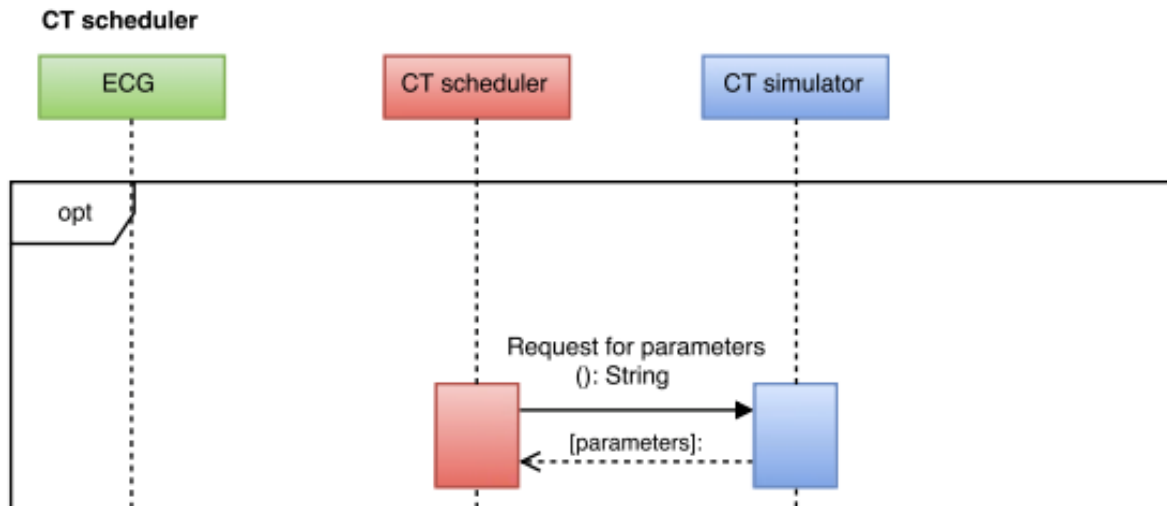
1. CT Scheduler must be able to register to the Ivy Biomedical 7800 ECG determined port (Proposed Ethernet Protocol.docx).
2. CT scheduler must be able to read from the ECG TCPIP port (Proposed Ethernet Protocol.docx) which is updated every 250ms, and re-write each chunk to ECGtemp1 (Table 4#1) Start-up phase (prior to scanning, prepare initiate received): ECG data is read and R-peak value threshold is determined



Use Case Diagram



Class Diagram



Sequence Diagram

Use Case Cycle 6

Customer Requirements

- Develop software to determine of proper possibility of cardiac gated scanning.
- Use an IVY biomedical 7800 cardiac Trigger monitor and HE instruments TechPatient -
- Cardio V4 patient simulator to create and monitor patient ECG data.

Use case description

1. Determination of Cardiac gated scan possibility and updating prepared status.

Triggers

1. Prepare initiate message received from CT simulator.

Actors

1. CT simulator handling messaging
2. ECG Simulator
3. CT scheduler
4. ECG local network port

Preconditions

1. ECG simulator is sending ECG data.
2. 'Submit' button on CT simulator GUI has been pressed.

3. CT scheduler is on.

Goals

1. Determine the possibility of a cardiac gated scan.
2. Set CT scanner to prepared state.

Steps of Execution

1. CT scheduler reads data from local port.
2. Data is analyzed by CT scheduler to determine R-R peak values.
3. A threshold value for R-R peak values is determined.
4. Determination is based on total time between R-R peaks values.
5. Message stating possibility of cardiac gated scan is sent to CT simulator.
6. CT scanner is set to prepared state.

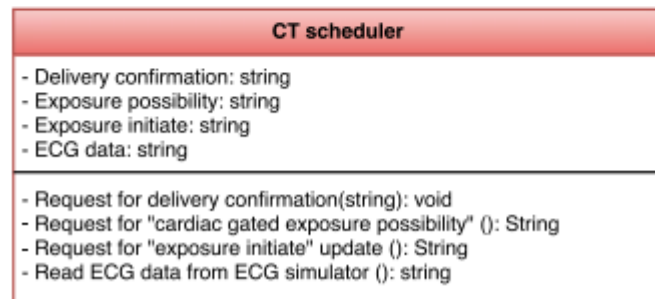
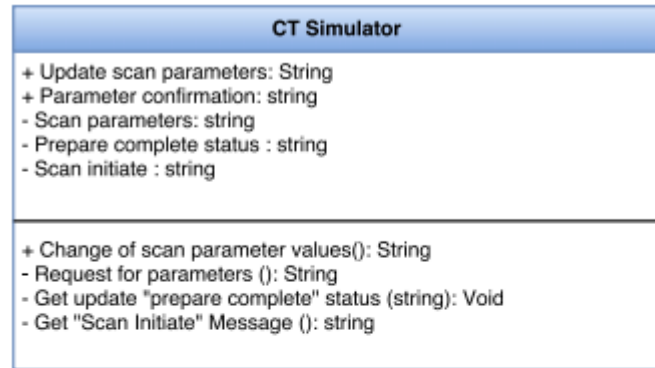
Engineering Requirements Cycle 6

CT Simulator

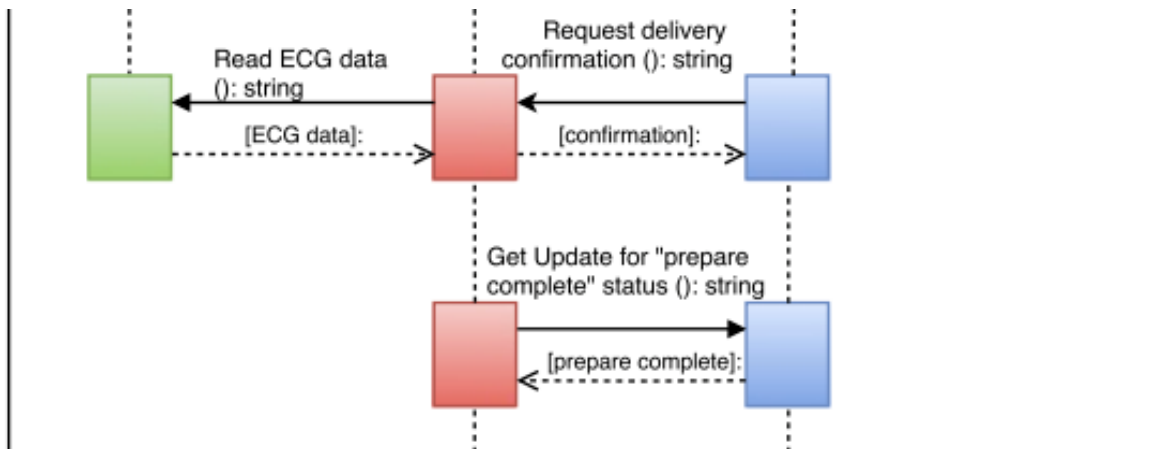
1. CT simulator must be able to send 'Scan Initiate' message to CT scheduler (Table 3 #5)
2. CT simulator must be able to receive request to update 'Cardiac gated exposure possibility' message (table 3). Message is received every 25ms (1tick) (Table 3 #8)
3. CT simulator must be able to receive request to update 'Exposure initiate' message by CT scheduler. Message is received every 25ms (1tick) (Table 3 #7)

CT Scheduler

1. ECGtemp1 (Table 4#1) data is copied every 250ms to ECGtemp2 (Table 4#2) to be used for data manipulation.
2. ECGtemp2 is manipulated into zeros, and 1 individual maximum value per chunk. Maximum of ECGtemp2 designated as tempthresh (Table 4#3)
3. After manipulation, the data from ECGtemp2 is added to the end of Log (Table 4#5)
4. Tempthresh is added to the end of threshdet (Table 4#6)
5. After 120 chunks of data have been collected (120 repeats of steps 10-13 see tick in Table 4#4), Threshold is calculated based on the maximum and range of threshdet (the compilation of each chunk's max).
6. Once Threshold (Table 4#7) has been determined Startup Phase is complete, Log is cleared, Prepare Complete sent.



Class Diagram



Sequence Diagram

Use Case Cycle 7

Customer Requirements

- Develop a trigger system to send an exposure initiate message.
- Exposure initiate is determined to be during the quiescent period of the heart beat (defined as 70% of R-R peak value).
- Develop software to determine of proper possibility of cardiac gated scanning.
- Use an IVY biomedical 7800 cardiac Trigger monitor and HE instruments TechPatient Cardio V4 patient simulator to create and monitor patient ECG data.

Use case description

1. ECG data analysis and scan initiate timing determination.

Triggers

1. Start Scan button has been pressed.

Actors

1. CT simulator handling messaging
2. ECG Simulator
3. CT scheduler
4. ECG local network port

Preconditions

1. CT scheduler is prepared.
2. ECG simulator is providing data to port.

Goals

1. Determine moment for exposure initiate message to be sent to CT simulator.
2. Update scan parameters such as retrospective limit and scan count.
3. Send trigger message "exposure initiate".

Steps of Execution

1. Data from ECG is compared to threshold value previously determined.
2. R Peak values are found and location in time determined.
3. CT scheduler determines when next possible Image can take place.
4. If image is not possible, Retrospective limit count is updated.
5. CT scheduler send exposure initiate message at determined time.
6. Scan count is updated.
7. System repeats until scan count is reached.

Engineering Requirements Cycle 7

Recording phase (during scanning, scan initiate received): ECG data is read and compared to Threshold

1. Phase can only happen once "Scan Initiate" Message from CT simulator has been received, Startup Phase has been completed, and ECGtemp is updated.
2. ECGtemp1 data is copied to ECGtemp2 to be used for data manipulation.
3. If the max of ECGtemp2 is above Threshold, set all other data values in chunk to be zero and add ECGtemp2 to the end of Log. Set variable Max to true (Table 4#8). Otherwise set all values to zero and add to the end of Log.

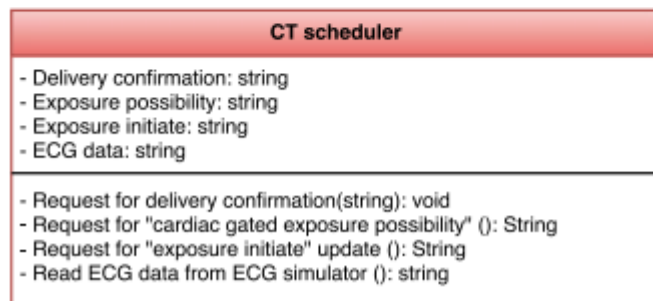
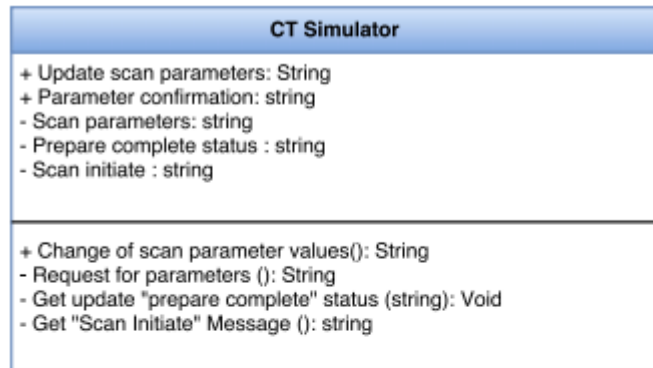
Calculating phase (during scanning): Log is read and time until scan engage is determined

1. Phase can only happen when Max is true and variable Engage (Table 4#9) is false.
2. Copy Log to Log2 (for data manipulation see Table 4#10) scan Log2 right to left for variables greater than zero.
3. Set P2 (Table 4#11) as position of first max reached within Log2
4. Set P1 (Table 4#12) as position of second max reached within Log2
5. dt (Table 4#13) is calculated by the difference between P2 and P1 (time between R peaks)
6. t1 (Table 4#14) is calculated as 70% of dt (time from R to engage)
7. L1 (Table 4#15) is calculated as position difference between end of Log2 and P2 (time-position of R peak within last chunk received)
8. LT (Table 4#16) is calculated as the summation of L1, 0.25ms, and LS (Table 4#17) - the tunable system lag (LT is total lag)
9. te (Table 4#18) is calculated as the difference between t1 and LT (te is time until scan engage)
10. Set Max false and end Calculating phase

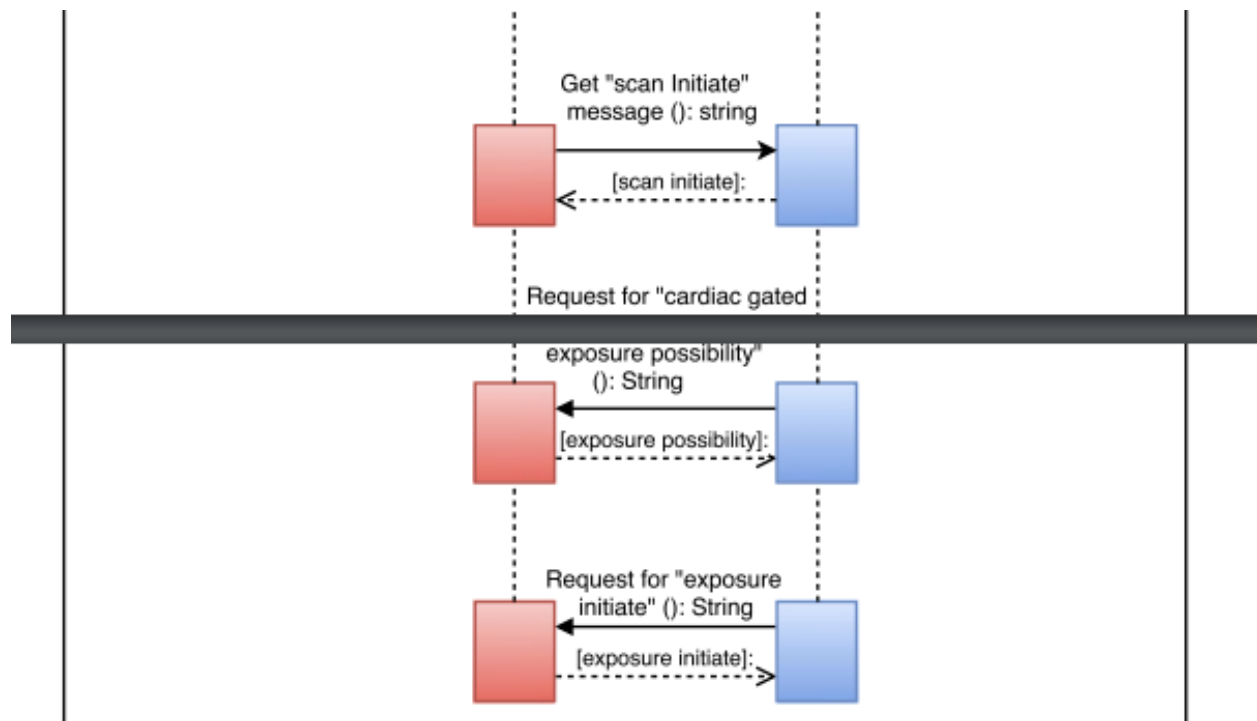
Engaging phase (during scanning): Time until scan engages (te) and CT system are synchronized for scan to occur

1. Phase can only happen when 'te' is updated and ScanCount is less than determined value (determined by CT based on operator inputs for scan type)
2. Set Engage to true and Stopwatch (Table 4#19) to countdown from 'te'
3. If CT is ready and stopwatch is not zero, wait.

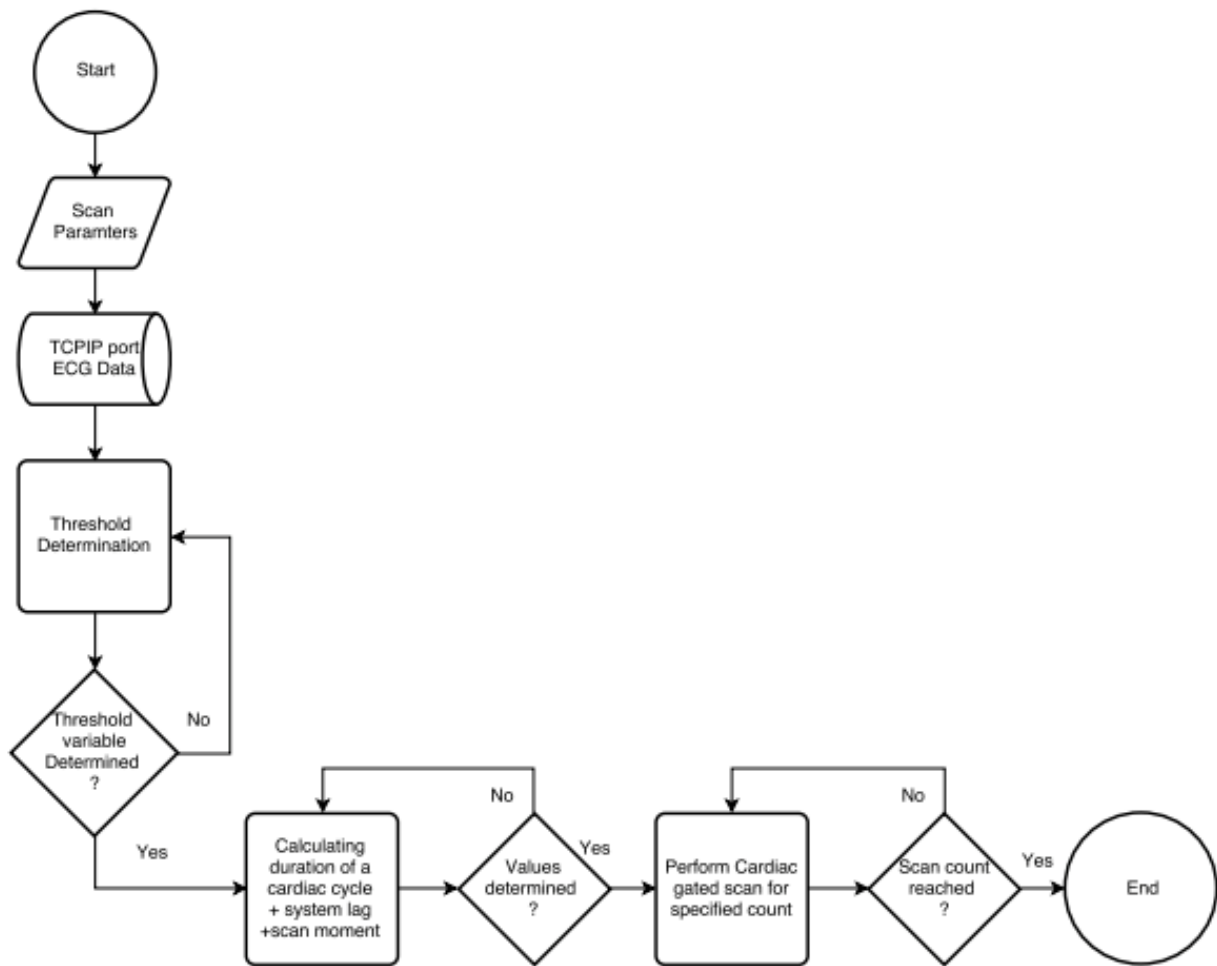
4. If CT is ready and stopwatch is zero, send Scan Trigger. Increase Scan Count (Table 4#20) by one.
5. When Scan Count is equal to determined value, set stopwatch to false, end phase and end scan.



Class Diagram

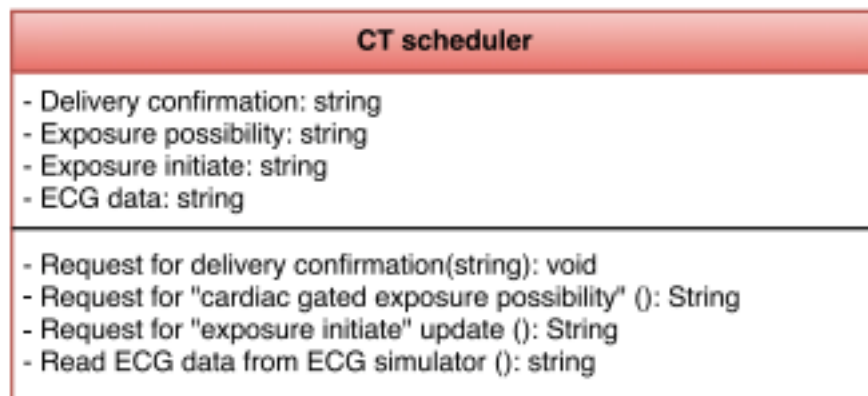
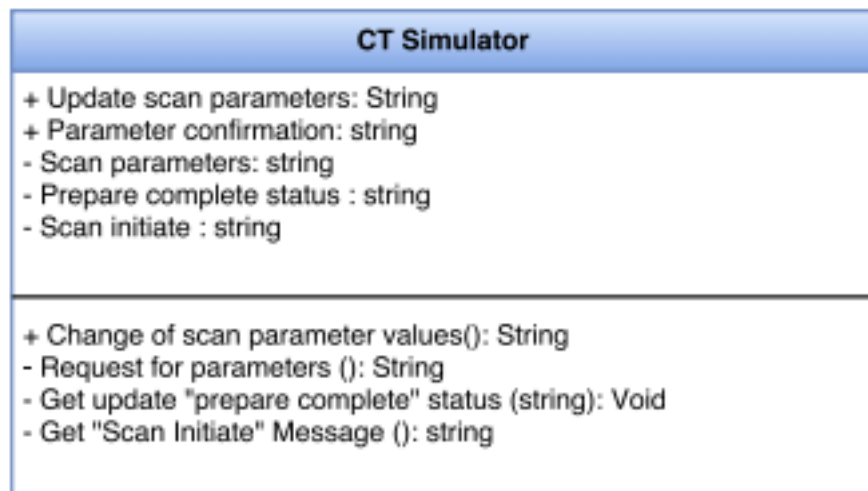
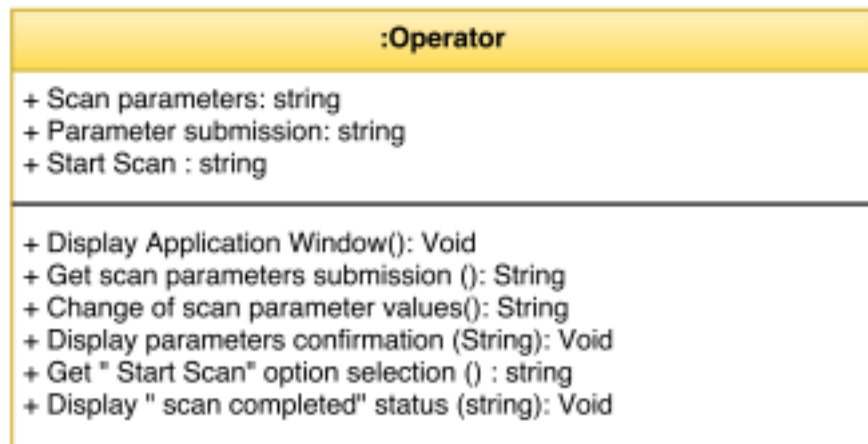


Sequence Diagram

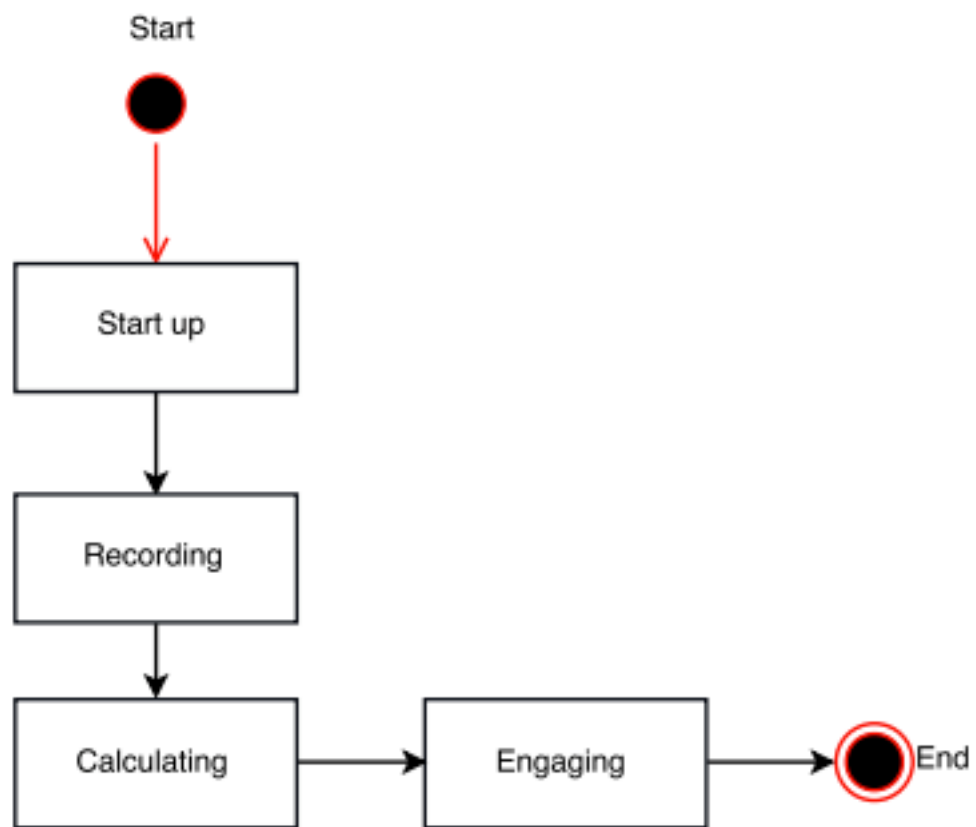


Algorithm Flowchart

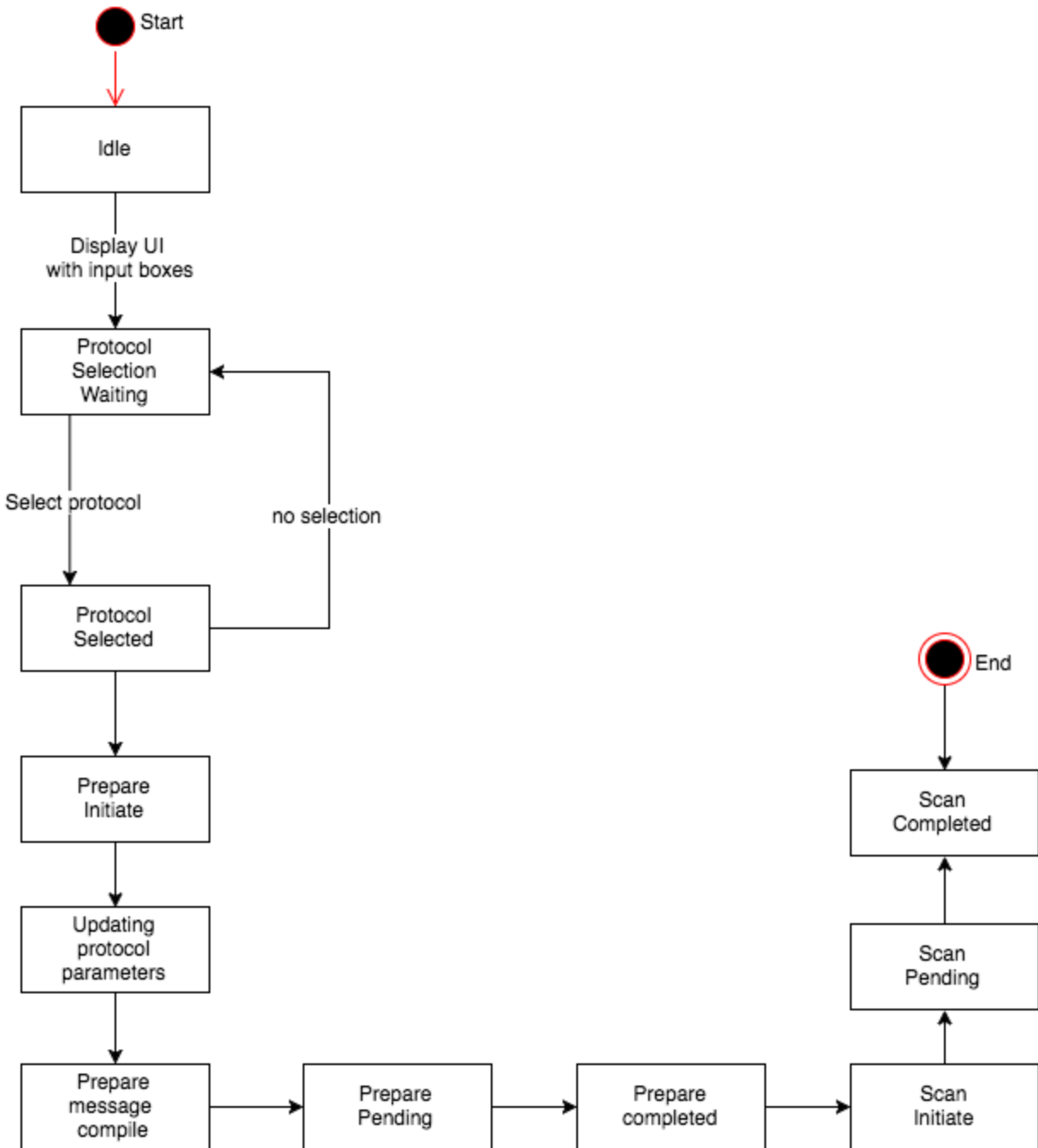
Additional Diagrams



Overall Class Diagram

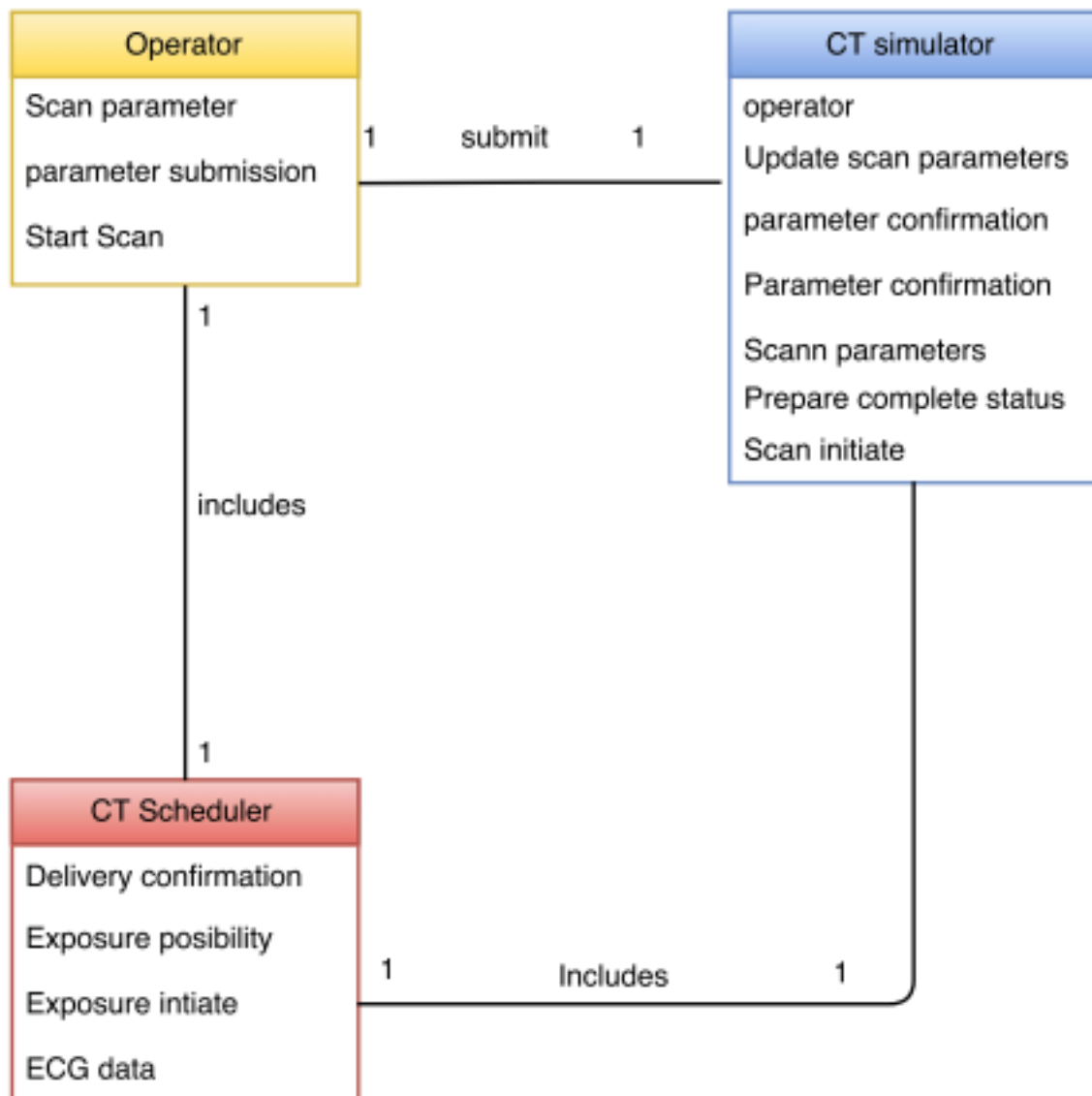


CT Scheduler State Machine Diagram



CT Simulator State Machine Diagram

Object model



Object Model Diagram

Reference Tables

Table 1. Scan Parameter Descriptions

	Name	Range	Description
1	Scanner Voltage	80-140	<p>The anode voltage, expressed in kilovolts to be used in the exposure</p> <p>Received: Preparation State</p> <p>Applied: Scanning State</p>
2	Scanner Current	Typical is between 50 and 400, but limit may vary depending of power calculation which used Scanner Current and Scanner Voltage	<p>The anode current, in milliamps associated with the exposure.</p> <p>Received: Preparation State</p> <p>Applied: Scanning State</p>
3	X-Ray Exposure Time	Maximum value is limited based on the present calculated tube heat capacity which use the Heating Curves provided by the X-Ray tube manufacturer to project final tube heat capacity, which must remain under a value of 0.90	<p>The amount of time that x-rays should be exposing</p> <p>Received: Preparation State</p> <p>Applied: Scanning State</p>
4	X-Ray Exposure Mode	"Theta" or "Timed" or "Gated"	<p>Tells X-Ray Manager what mode to become so it can perform the proper protocols</p> <p>Received: Preparation State</p> <p>Applied: Preparation and Scanning States</p>
5	Encoder Trigger	0 to 2240	<p>Sets the encoder tick, which is related to angle, that x-rays should start to be exposing</p> <p>Received: Preparation State</p> <p>Applied: Scanning State</p>

6	Integration Limit	depends on what acquisition time the detector is placed in	Number of 912x16 data sets which are acquired during a single exposure. 912x16 is the dimension of a single data capture
7	Rotation speed	10 to 150	The speed, values of Revolutions/Rotations Per Minute, at which the gantry will be rotating during the data acquisition Received: Preparation State Applied: Scanning State
8	RR Quiescent Percentage	0 to 100 (%)	The placement of the quiescent period, which is percentage along the patients R-R peak data, where exposures should start. The R-R peak data is heart rhythm data where each new R peak is the 100% of the previous rhythmic data on the 0% of the current/next rhythmic data Received: Preparation State Applied: Preparation and Scanning State
		0 to 1 (decimal form of %)	
9	Scan Gating Number	1 to the length of the patients heart	The number of sequential scan need for complete acquisition of the patients heart
10	Retrospective limit	2 to 5	The amount of attempts that prospective gating is attempted during the prospective protocol before the protocol is shifted to retrospective Received: Preparation State Applied: Scanning State

11	Columation Position	16x1.16, 16x0.58, 8x0.58, 4x0.58,2x0.58	<p>The position of the collimator prior to scan start.</p> <p>Received: Preparation State</p> <p>Applied: Scanning State</p>
----	---------------------	---	--

Table 2. Program Start-up Order

	Program Name
1	CoreWinAppYellowPages
2	ECG Simulator
3	CT Scheduler
4	CT Simulator

Table 3. Messaging Descriptions

#	Name	Description
1	Prepare Initiate	<p>Takes inputs in UI and saves them to .dll</p> <p>Received: Preparation State</p> <p>Applied: Preparation State</p>
2	Scan Parameter request	<p>CT scheduler asks for parameters listed</p> <p>Received: Preparation State</p> <p>Applied: Preparation State</p>
3	Scan Parameter Response	<p>Ct Simulator responds to request and sends requested variables</p> <p>Received: Preparation State</p> <p>Applied: Preparation State</p>

		<p>The status of the CT Scheduler to allow for exposure times to be determined after the input of required input variables</p> <p>Received: Preparation State</p>
4	Prepare Complete	<p>Applied: Scanning State</p>
		<p>Sets the system into the Scanning State</p> <p>Received: Prepared State</p>
5	Scan initiate	<p>Applied: Scanning State</p>
		<p>Sets the System into Exposure State</p> <p>Received: Scanning State</p>
6	Exposure initiate	<p>Applied: Scanning State, updated through Scanning state</p>
		<p>Sets the System into Exposure State</p> <p>Received: Scanning State</p>
7	Exposure initiate Response	<p>Applied: Scanning State, updated through Scanning state</p>
		<p>Updates the possibility of a cardiac gated exposure to be completed by the CT scanner</p> <p>Received: Scanning State</p>
8	Cardiac gated exposure possibility	<p>Applied: Scanning State, updated through Scanning state</p>
		<p>Derives the success of the cardiac gated scan</p> <p>Received: Scanning State</p>
9	Cardiac Gated Scan Completion Status	<p>Applied: Scanning State, updated through Scanning state</p>

Table 4. CT Scheduler Variable Descriptions

Table	Variable	Description
1	ECGtemp1	Raw data from ECG. Time and voltage. Updated every 250ms in 250ms segments.
2	ECGtemp2	Copied from ECGtemp1. Filtered to find max peak, set rest to zeros.
3	tempthresh	The max of each chunk (each ECGtemp2)
4	tick	Number of chunks received. Runs to 120.
5	Log	Compilation of each manipulated ECGtemp2.
6	threshdet	Compilation of maximum from each chunk.
7	Threshold	Calculated from threshdet. R-peak threshold value. Determined in Startup.
8	max	Status true when max of ECGtemp2 is above Threshold.
9	Engage	Status true when Calculating Phase is finished and Engaging Phase is occurring.
10	Log2	Copy of Log made when max is true.
11	P2	Position of most recent max within Log2.
12	P1	Position of second-most recent max within Log2.
13	dt	Calculated from P2 and P1. Time between R peaks (also is cardiac cycle duration).
14	t1	Time between R peak and quiescent period.
15	L1	Time between last R peak and end of last received chunk. Represents lag.
16	LT	Calculated from L1 and LS. Represents total lag.
17	LS	Adjustable system lag. Accounts for non-direct ECG connection.
18	te	Calculated from t1 and LT. Time until scan engage moment.
19	Stopwatch	Counts down from given value to

		zero (in seconds).
20	Scan Count	Determined in CT Simulator based on user input. Number of exposures needed.

State Descriptions

CT Simulator

- **Idle**
 - At this state, the application is already to be launched and currently not being used by any program.
- **Protocol selecting waiting & Protocol selected**
 - The application will already pass protocol selecting waiting, and is at protocol selected state, since it automatically selects the option of performing a cardiac gated scan, and begins logging upon startup of application.
- **Prepare initiate**
 - At this state, the user already submitted the required scan parameters to the CT simulator by selecting the “Submit” option. The application at this point started the procedure of preparing the CT simulator for a cardiac gated scan.
- **Updating Protocol parameters**
 - The CT simulator takes the submitted scan parameters, updates and stores them.
- **Prepare message compile**
 - The CT simulator prepares Message containing scan parameter values to be sent to the CT scheduler. Once message is prepared, the CT simulator send the prepare message to the CT scheduler.
- **Prepare pending**

- The CT simulator waits for a delivery confirmation from the CT scheduler of the sent prepared message.
- **Prepare completed**
 - Once CT scheduler updates the CT simulator with the delivery confirmation, the CT simulator is now at this state, and finished the prepare procedure. At this point the CT simulator enables the “Start Scan” option for the user to select once ready to start the cardiac gated scans.
- **Scan initiate**
 - When the user selects the “Start scan” option, the CT simulator is at this state and starting the post prepares procedure.
- **Scan pending**
 - In this state the CT scheduler is using the algorithm in order to determine cardiac gated exposure possibility, and determine exposure initiate once completed. The CT simulator at this point is waiting for the CT scheduler to update the Scan completion status.
- **Scan completed**
 - Once CT scheduler completes its tasks, it updates the scan completion status in the CT simulator. The GUI then displays this message to user for confirmation.

CT Scheduler

- **Startup & initialization**
 -
- **Reading ECG Data**
 -
- **Calculating**
 -

- Engaging

-

Verification & Validation Plans

Software Verification & Validation plan And Testing Protocols

Overview

The Verification and validation plan will be used to test certain specifications of the software in order to verify that the system satisfies, and delivers the designed requirements, as well as to validate that it accomplishes its purpose and meets all customer requirements.

The plan is going to consist of the following main sections:

- CoreWinApp
- ECG Simulator
 - TCPIP Port
- CT Scheduler
 - Window
 - Messaging
 - Algorithm
- CT Simulator GUI
 - Window
 - Current Status

- Diagnostic Log
- CT Simulator (whole)

The goal of this plan is to insure that the listed main sections pass their assigned testing protocols which will be specified in a separate document.

CoreWinApp Yellowpages

CoreWinApp is considered the manager of all messaging between the different applications used in this demonstration (ECG Simulator, CT scheduler, CT simulator GUI). In this plan we will test its ability to unlock the dll. , register to messaging, and enable operator to use messaging. The user must have this application running before moving on further with the demonstration.

ECG Simulator

ECG Simulator is an application that is provided to our group by FMI medical systems, and it represents an application that simulates a patient's heart rhythm. This heart rhythm will be used with the other developed applications (CT simulator, CT scheduler) in order to perform a cardiac gated scan. In this plan we will tests its ability to send ECG data to a designated port (TCPIP), and allow the CT scheduler to read the ECG data from that port.

CT Scheduler

The CT scheduler is the application developed by the team using an algorithm to determine the possibility to perform a cardiac gated scan at the quiescent period of the heartbeat. This is done by collecting ECG data from our ECG Simulator, as well as input from CT operator through the developed CT simulator GUI. The CT simulator will be sending inputs to CT scheduler, and receiving output to display to the user containing information once exposure possibility, and the completion of scan is done. In this plan we will tests its ability to

launch the application, send and receive messages to and from the CT simulator GUI, read ECG data from the TCPIP port, and test the algorithm's ability to determine exposure possibility and perform a cardiac gated scan.

CT simulator Operator Interface GUI

The CT simulator Operator Interface GUI is an application developed for the purpose of simulating a real life CT scanner. The application allows the user to submit desired parameter values used for the cardiac gated scan, and observe the processes that the simulator goes through on the diagnostic log window. The diagnostic log window updates the application status every 25ms to verify that it follows exactly the designed tasks, and validate that it is successfully simulating the real life CT scanner. The Operator Interface GUI window will include the following features: Parameter names, initial parameter values, Submit option for parameter submission, Start Scan option, Start Logging and Stop Logging which gives control to the user over updating the Diagnostic Logging window, Current Status window which updates the status of the application with a timestamp, and a Diagnostic Logging window which displays an update of all application processes with a timestamp and is going to be used throughout the verification processes for this application. The statuses that the CT simulator will go through and be shown on the Current status window are the following: Idle, Protocol Selection Waiting, Protocol Selected, Prepare Initiate, Updating Protocol Parameters, Prepare message Compile, Prepare Pending, Prepare Completed, Scan Initiate, Scan Pending, Scan Completed. In this plan we will test the CT simulator ability to launch the application, display status updates every 25ms, accept user inputs, register to the dll, and manage messaging between it and the CT scheduler.

Testing Protocols

CoreWinApp

1- Tested Object: CoreWinApp (dll.)

- **Features to be tested**

- Test the Launch of the application to unlock the dll, and insure the registration of the other applications associated.
- Test its ability to display information on the application window related to applications registered with it along with timestamps.
- Test its ability to handle messaging between the different applications associated with it.

- **Testing approach:**

- Locate the CoreWinApp yellow pages shortcut on the computer desktop.
- Using the computer mouse pad, double click on the CoreWinApp shortcut. This will result in opening the CoreWinApp window to confirm that the program is running successfully.
- Run any application associated with the CoreWinApp, and observes the applications window to confirm the registration with that application along with a timestamp.
- Confirm by using the other applications that the messaging is handled properly, by observing that input values from one application, matches the output values or results in the other.

- **Expected Output**

- The applications should launch successfully, unlock the dll, and register any application associated with it.
- Be able to display the name of registered applications with time stamps for user on the application window.

- **Pass/Fail criteria**

- The object passes the test protocol if it successfully displays the application window, unlocks the dll, registers to associated applications, and display names of registered applications with time stamps. Moreover, the
- When trying to run the program when it was already launched, it will state that “it is unable to setup a network component during installation”, meaning the applications failed to launch again.

ECG Simulator

1- Tested Object: TCPIP Port

- **Features to be tested**

- Test its ability to store ECG data to the TCPIP port for the CT scheduler to read from.
- Test its ability to update ECG data every 250ms.

- **Testing approach:**

- The CoreWinApp must be already launched, as well as, the CTSchedulerApp, and must be registered to the dll. This can be confirmed by observing the CoreWinApp window to locate the name of applications registered along with a timestamp.
- Locate the ECGSimulator shortcut on the computer desktop.
- Using the mouse pad, double click on the ECGSimulator shortcut, which will display a computer command window that contains information which includes: port number that the server is running at, local end point, and displays the current status of waiting for connection to the port.
- Wait for the connection to be accepted. Once the connection is accepted, the command window will confirm that and display the path number.

- **Expected Output**

- The ECGSimulator successfully connects to the TCPIP port, and stores ECG data to the port for the CTSchedulerApp to read from.

- **Pass/Fail criteria**

- The object would pass the test protocol if the ECGSimulator was successfully able to connect to the TCPIP port, and send ECG data to the port for the CTSchedulerApp to read from.
- The object would fail the test protocol if the ECGSimulator was not able to connect to the TCPIP port, or send ECG data to the port for the CTSchedulerApp to read from.

CT Scheduler App

1- Tested Object: window

- **Features to be tested**

- Test its ability to display the CTSchedulerApp window.
- Check the ability of the window to clearly show titles until ejected by user.

- **Testing approach:**

- The CoreWinApp must be already launched, as well as, the CTSchedulerApp, and must be registered to the dll. This can be confirmed by observing the CoreWinApp window to locate the name of applications registered along with a timestamp.

- **Expected Output**

- The Application launches, and registers to the dll. successfully until ejected by user.

- **Pass/Fail criteria**

- The application would pass the test protocol if it was able to successfully launch the application window until it is ejected by user.
- The application would fail the test protocol if it was not able to successfully launch the application window, or was not able to keep the application launched until it is eject by user.

2- Tested Object: Messaging

- **Features to be tested**

- Test the CTSchedulerApp ability to send and receive messages to and from dll.
- Test its ability to update auto scribe features.

- **Testing approach:**

- The CoreWinApp must be already launched, as well as, the ECGSimulator, and Operator Interface, and must be registered to the dll. This can be confirmed by observing the CoreWinApp window to locate the name of applications registered along with a timestamp.
- Launching the utilWinTestMessagingLibraryCSharp.exe application, this is a program that allows for dynamic registration to the messaging dll, and register as CTSchedulerApp. This will allow the user to receive and send message as if they are the CTSchedulerApp.
- Run the CT Simulator using the Operator Interface application, and confirm messaging between the CT simulator and CTSchedulerApp by observing the utilWinTestMessagingLibraryCSharp.exe application. This will display information regarding messages sent and received, from and to what application. The user can confirm that the sent input message correctly corresponds with the output or received message.

- **Expected Output**

- The user ability to confirm that sent an input message correctly corresponds with the output received message using the utilWinTestMessagingLibraryCSharp.exe application.

- **Pass/Fail criteria**

- The object would pass the test protocol if the user was able to confirm that sent input messages correctly corresponds with the output received message.
- The object would fail the test protocol if the user was not able to confirm that sent input messages correctly corresponds with the output received message.

3- Tested Object: Algorithm

- **Features to be tested**

- Test its ability to read and analyse ECG data to determine R-Peak values, and R-R duration time.
- Tests its ability to determine exposure possibility, and trigger the CT simulator to perform a cardiac gated scan.

- **Testing approach:**

- The CoreWinApp must be already launched, as well as, the CTSchedulerApp, CT simulator, and ECG simulator, and must be registered to the dll. This can be confirmed by observing the CoreWinApp window to locate the name of applications registered along with a timestamp.
- The operator has the option to use the preinstalled initial parameter values (KVp, mA, Rotation Speed, RR percentage, Scan Gating number, Retrospective Limit), or to change any of these values to their desired values by using the mouse pad and clicking on the row they want to change, and clicking again on the value box to enable editing using the computer keyboard.

- Once operator confirms their desired parameter values, using the mouse pad, select the Submit option to update the CT simulator with parameter values. The operator can locate the latest update of the parameter values, and system state on the Current Status window for verification. At this point the system will go through Prepare Initiate, Updating Protocol Parameters, and Prepare Message Compile, and is currently in the Prepare Pending states.
- Wait for it to be in the prepared state. Once the CT simulator is in the prepared state, it will enable the Start Scan option. The CT simulator then controls messaging between the different applications involved, and then updates the prepared complete messages once it is in the prepared state. Then using the computer mouse pad, select the Start Scan option to start the execution of the cardiac gated scan.
- The user then observes the status update of the completion of cardiac gated scan, to confirm that the cardiac gated scan was successfully completed.
- **Expected Output**
 - Analyze ECG data to determine R-Peak values, and R-R duration time needed for triggering the CT simulator.
- **Pass/Fail criteria**
 - The object would pass the test protocol if it was able to determine the needed values (R-Peak, R-R time), and trigger the CT simulator to perform a cardiac gated scan.
 - The object would fail the test protocol if it was not able to determine R-Peak value, R-R duration time, or scan position result in not accurate.

CT Simulator GUI

1- Tested Object: Window

- **Features to be tested**

- Test its ability to display the CT simulator application window.
- Check the ability of the window to clearly show titles, parameters, parameter values, current status log, and the diagnostic log.
- Test the window's ability to display application's internal information to user.

- **Testing approach:**

- Locate the Operator Interface shortcut on the computer desktop.
- Using the computer mouse pad, double click on the Operator Interface shortcut.
This will result in opening the Operator Interface window for Cardiac Gated Scan.

- **Expected Output**

- The application window should clearly display the features listed above to the user.

- **Pass/Fail criteria**

- The application window passes the test protocol if it successfully displays the application window with all its features, as well as, keeps the application window launched until ejected by user.
- The application window fails the test protocol if it did not successfully open the window, failed to display all assigned features, or failed to keep window open until the end of the scan.

2- **Tested Object:** Scan Parameters

- **Features to be tested**

- Test its ability to display name, and value of scan parameters.
- Test its ability to include, and display initialized scan parameter values.
- Tests its ability to allow user to edit any undesired values using the computer keyboard.

- Test its ability to submit scan parameter values, and allow Current Status window to display the submission of these values.
- Test its ability to only accept scan parameter values that are within their specified range (table#1. V3).
- **Testing approach:**
 - The CoreWinApp must be already launched, as well as, the Operator Interface and must be registered to the dll.
 - The Operator Interface application window must be already launched and displayed to user.
 - By observing the scan parameter section on the application window, the operator can confirm that the system successfully and clearly displays the following initial scan parameter names and values: (KVp=80, mA=250, Rotation Speed=120, RR percentage=0.6, Scan Gating number=5, Retrospective Limit=2).
 - Using the mouse pad, the user can confirm the ability to change scan parameter values by clicking on the row they want to change, and clicking again on the value box to enable editing using the computer keyboard.
 - Using the mouse pad, the user can confirm the submission, and update of scan parameter values by selecting the Submit option to update the CT simulator with parameter values. The operator can locate the latest update of the parameter values with timestamps on the Current Status window for verification. This will also display the rejection of scan parameter values if they were submitted out of range based on (table# 1. V3).
- **Expected Output**

- The Scan parameter section should be clearly displayed to user, include initial scan parameter values, allow editing by user, and submission to CT simulator for update.

- **Pass/Fail criteria**

- The object would pass the test protocol if it was able to successfully display initial parameter names and values, allow editing by user, and allow the submission of these values to CT simulator for update.
- The object would fail the test protocol if it failed to display initial parameter names and values, allow editing by user, and allow the submission of these values to CT simulator for update.

3- **Tested Object:** Current Status

- **Features to be tested**

- Test its ability to display any updates within the system.
- Test its ability to show timestamps along with every update of status. The statuses that the CT simulator will go through and be shown on the Current status window are the following: Idle, Protocol Selection Waiting, Protocol Selected, Prepare Initiate, Updating Protocol Parameters, Prepare message Compile, Prepare Pending, Prepare Completed, Scan Initiate, Scan Pending, Scan Completed.
- Test its ability to automatically being updating, and logging upon start of the application.
- Test that every input entered to the system, match the output shown on the current status log.

- **Testing approach:**

- The CoreWinApp must be already launched, as well as, the Operator Interface and must be registered to the dll.
- The Operator Interface application window must be already launched and displayed to user.
- By observing the Current Status window, the operator can confirm that the window is automatically being updating, and logging upon start of application. At this point the system has already passed Idle, Protocol Selection Waiting, and Protocol Selected states, and is currently in waiting for parameter submission to go to Prepare initiate state.
- For the user to confirm the remaining statuses, the operator has to submit parameter values.
- Using the mouse pad, select the Submit option to update the CT simulator with parameter values. The operator can locate the latest update of the parameter values, and system state on the Current Status window for verification. At this point the system will go through Prepare Initiate, Updating Protocol Parameters, and Prepare Message Compile, and is currently in the Prepare Pending state.
- Once the CT simulator is in the prepared state, it will enable the Start Scan option. The CT simulator then controls messaging between the different applications involved, and then updates the prepared complete messages once it is in the prepared state. Then using the computer mouse pad, select the Start Scan option to start the execution of the cardiac gated scan.

- **Expected Output**

- By following the listed steps, the current status window should be able to display all updates that are manually submitted by user, or automatically updated within the system.

- The Current status window should be able to display status update with timestamps throughout the procedure, until the end.
- **Pass/Fail criteria**
 - The Current Status window would pass the test protocol if it was able to successfully go throughout the procedure, and perform updates with no lagging until the end of the scan.
 - The Current Status window fails the test protocol if it does not update status, updates the incorrect status, or missing timestamps.

4- **Tested Object:** Diagnostic log

- **Features to be tested**
 - Test its ability to update the system status accurately every 25 ms.
 - Test its ability to show timestamps along with every update of status. The statuses that the CT simulator will go through and be shown on the diagnostic log window are the following: Idle, Protocol Selection Waiting, Protocol Selected, Prepare Initiate, Updating Protocol Parameters, Prepare message Compile, Prepare Pending, Prepare Completed, Scan Initiate, Scan Pending, Scan Completed.
 - Test its ability to automatically begin updating, and logging upon startup of the application.
 - Test that every input entered to the system, match the output shown on the diagnostic log window.
- **Testing approach:**
 - The CoreWinApp must be already launched, as well as, the Operator Interface and must be registered to the dll.

- The Operator Interface application window must be already launched and displayed to user.
- By observing the Diagnostic log window, the operator can confirm that the window is automatically being updating, and logging every 25ms upon start of application. At this point the system has already passed Idle, Protocol Selection Waiting, and Protocol Selected states, and is currently in waiting for parameter submission to go to Prepare initiate state.
- For the user to confirm the remaining statuses, the operator has to submit parameter values.
- Using the mouse pad, select the Submit option to update the CT simulator with parameter values. The operator can locate the latest update of the parameter values, and system state on the Diagnostic log window for verification. At this point the system will go through Prepare Initiate, Updating Protocol Parameters, and Prepare Message Compile, and is currently in the Prepare Pending state.
- Once the CT simulator is in the prepared state, it will enable the Start Scan option. The CT simulator then controls messaging between the different applications involved, and then updates the prepared complete messages once it is in the prepared state. Then using the computer mouse pad, select the Start Scan option to start the execution of the cardiac gated scan.
- **Expected Output**
 - By following the listed steps, the diagnostic log window should be able to display all updates that are manually submitted by user, or automatically updated within the system every 25ms.
 - The diagnostic log window should be able to display status update with timestamps throughout the procedure, until the end every 25ms.
- **Pass/Fail criteria**

- The diagnostic log window would pass the test protocol if it was able to successfully go throughout the procedure, and perform updates every 25ms with no lagging until the end of the scan.
- The diagnostic log window fails the test protocol if it does not update status every 25ms, updates the incorrect status, or missing timestamps.

5- Tested Object: CT Simulator (whole)

- **Features to be tested**

- Test its ability to register to the CoreWinApp dll.
- Test the Operator Interface's ability to run all the features associated with it altogether successfully (window, current status window, Diagnostic log window, and Scan Parameters).

- **Testing approach:**

- The CoreWinApp must be already launched, as well as, the CT scheduler, and ECG simulator, and must be registered to the dll. This can be confirmed by observing the CoreWinApp window to locate the name of applications registered along with a timestamp.
- Locate the Operator Interface shortcut on the computer desktop.
- Using the computer mouse pad, double click on the Operator Interface shortcut. This will result in opening the Operator Interface window for Cardiac Gated Scan, which will also update the registration of this application with the dll. On the CoreWinApp with its name and timestamp.
- The Operator Interface will automatically be logging and updating current status upon start up, and waits for parameter submission by operator. At this point the system has already passed Idle, Protocol Selection Waiting, and Protocol

Selected states, and is currently in waiting for parameter submission to go to Prepare initiate state.

- The operator has the option to use the preinstalled initial parameter values (KVp, mA, Rotation Speed, RR percentage, Scan Gating number, Retrospective Limit), or to change any of these values to their desired values by using the mouse pad and clicking on the row they want to change, and clicking again on the value box to enable editing using the computer keyboard.
- Once operator confirms their desired parameter values, using the mouse pad, select the Submit option to update the CT simulator with parameter values. The operator can locate the latest update of the parameter values, and system state on the Current Status window for verification. At this point the system will go through Prepare Initiate, Updating Protocol Parameters, and Prepare Message Compile, and is currently in the Prepare Pending state.
- Wait for it to be in the prepared state. Once the CT simulator is in the prepared state, it will enable the Start Scan option. The CT simulator then controls messaging between the different applications involved, and then updates the prepared complete messages once it is in the prepared state. Then using the computer mouse pad, select the Start Scan option to start the execution of the cardiac gated scan.

- **Expected Output**

- The CoreWinApp should launch successfully, unlock the dll, and register any application associated with it.
- CoreWinApp should be able display the name of registered applications with time stamps for user on the application window.
- The Operator Interface application window should clearly display the features listed above to the user.

- The Scan parameter section should be clearly displayed to user, include initial scan parameter values, allow editing by user, and submission to CT simulator for update.
- The current status window should be able to display all updates that are manually submitted by user, or automatically updated within the system.
- The Current status window should be able to display status update with timestamps throughout the procedure, until the end of the cardiac scan.
- The diagnostic log window should be able to display all updates that are manually submitted by user, or automatically updated within the system every 25ms.
- The diagnostic log window should be able to display status update with timestamps throughout the procedure, until the end every 25ms.
- **Pass/Fail criteria**
 - The test protocol would pass if CoreWinApp successfully registers to applications associated with it, along with displays of confirmation with timestamp.
 - The test protocol would fail if the CoreWinApp is not able to register applications to its dll. and fails to display registry confirmation.

Code Description & Requirements Verification

CT Simulator Operator Interface GUI (A)

1. **Protocols A1**
 - a. **Display (XML, C#) A1.1**
 - i. **Main Window A1.1.1**

```

<Window x:Class="OperatorInterface.MainWindow"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="clr-namespace:OperatorInterface" <----- How I can use
objects and attach them to the window dynamically

    Title="Operator Interface" Height="800" Width="800"

    Background="{StaticResource FMISolidDarkSkyBlue}"

    WindowStartupLocation="CenterScreen">

    <Grid Name="gridMain">

```

ii. Menu A1.1.2

```

<DockPanel>

    <Menu DockPanel.Dock="Top" Name="mMain">

        <MenuItem Header="_File" Name="miFile">

            <MenuItem Header="E_xit" Name="miFileExit"
Click="MenuItem_Click" />

        </MenuItem>

        <MenuItem Header="_Help" Name="miHelp">

            <MenuItem Header="_About" Name="miHelpAbout"
Click="miHelpAbout_Click" />

            <MenuItem Header="_Messages" Name="miHelpMessages"
Click="miHelpMessages_Click" />

            <MenuItem Header="_Configuration" Name="miHelpConfiguration"
Click="miHelpConfiguration_Click" />

            <MenuItem Header="_Timers" Name="miHelpTimers"
Click="miHelpTimers_Click" />

        </MenuItem>

    </Menu>

    <StackPanel></StackPanel>

</DockPanel>

```

THIS WILL CREATE AND SET UP THE PLACEMENT REGION FOR DISPLAY

```
<Grid Name="GUIdisplayRegion" Margin="0,21,0,22">
```

THIS WILL CREATE SECTION OF DISPLAY REGIONS BETWEEN THE CONTROLS WE USE AND THE LOGGING (USING "*" MAKES IT DYNAMIC)

```
<Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="100*" />
```

```
<ColumnDefinition Width="100*" />
```

```
</Grid.ColumnDefinitions>
```

THIS WILL CREATE THE CONTROL REGION AND CREATES 2 REGIONS

```
<Grid Name="OperatorControlRegions" Grid.Column="0">
```

```
<Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="125*" />
```

```
<ColumnDefinition Width="50*" />
```

```
</Grid.ColumnDefinitions>
```

- Grid.Column="0" references the 1st column of the GUIdisplayRegion Grid

DYNAMIC ATTACHMENT OF AN OBJECT

```
<local:ScanParameterPanel x:Name="scanparameterpanel" Grid.Column="0" Margin="0"/>
```

- THIS IS WHERE THE USER WILL BE ABLE TO EDIT PARAMETERS
- local:ScanParameterPanel IS THE DYNAMIC BINDING OF THIS DISPLAY
- x:Name="scanparameterpanel" INITIALIZE A VARIABLE THAT CAN BE REFERENCED IN C# CODE FOR OBJECT BASED BINDING OF THIS DISPLAY
- Grid.Column="0" references the 1st column of the OperatorControlRegions Grid

THIS CREATES THE GRID THAT WILL DISPLAY PUSH BUTTONS FOR THE OPERATOR

```
<Grid Name="ButtonControlGrid" Grid.Column="1" Margin="0,6,0,0">
```

```
<StackPanel VerticalAlignment="Center">
```

```

        <Button Name="StartPreperation" Content="Submit"
            Height="auto" Width="auto" Click="StartPreperation_Click" />
        <Button Name="StartScan" Content="StartScan"
            Height="auto" Width="auto" Click="StartScan_Click"
IsEnabled="False" />
        <Button Name="StartLogging" Content="Start Logging"
            Height="auto" Width="auto" Click="StartLogging_Click" />
        <Button Name="StopLogging" Content="Stop Logging"
            Height="auto" Width="auto" Click="StopLogging_Click" />
    </StackPanel>
</Grid>

```

- Grid.Column="1" references the 2nd column of the OperatorControlRegions Grid
- Click="....." ARE ALL FUNCTIONS IN THE CODE WHICH ARE RESPONSIBLE FOR HANDLING LOGIC WHEN CLICKED ON

iii. Dynamic attachment of an object A1.1.3

```

<local:StateControl x:Name="statecontrol" Margin="0,21,6,22"
Grid.Column="1" />

```

- THIS IS WHERE THE USER WILL BE ABLE TO SEE ANY LOGS
- local:StateControl IS THE DYNAMIC BINDING OF THIS DISPLAY
- x:Name="statecontrol" INITIALIZE A VARAIBLE THAT CAN BE REFERENCED IN C# CODE FOR OBJECT BASED BINDING OF THIS DISPLAY
- Grid.Column="1" references the 2st column of the OperatorControlRegions Grid

MAIN WINDOW INITIALIZATION (GUI IS ABLED TO BE USED)

```

</Grid>
</Window>

```

iv. Diagnostic Log Window A1.1.4

The LOGGING DISPLAY DECLARATION - UserControl is
Windows recommended method for WPF applications using
MVVM

```
<UserControl x:Class="OperatorInterface.StateControl"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="300">
```

THIS WILL CREATE AND SET UP THE PLACEMENT REGION FOR DISPLAY

```
<Grid>

    <Grid.RowDefinitions>

        <RowDefinition Height="auto"/>

        <RowDefinition Height="*/>

        <RowDefinition Height="auto"/>

        <RowDefinition Height="*/>

    </Grid.RowDefinitions>
```

THIS PLACES A BOX WITH TEXT LETTING THE USER KNOW WHAT THE
TEXT BELOW REFERENCES

```
<TextBox Text="Current State"

    HorizontalContentAlignment="Center"

    HorizontalAlignment="Stretch" Grid.Row="0"/>
```

THIS WILL CREATE A RICHTEXTBOX, WHICH ALLOWS FOR COLOR AND
SIZE MANIPULATION TO HAPPEN

```
<RichTextBox Name="StateDisplay" HorizontalAlignment="Stretch"
    VerticalAlignment="Stretch" VerticalScrollBarVisibility="Visible" Grid.Row="1"/>
```

- THIS IS WHERE THE USER WILL BE ABLE TO WHAT STATE
CHABGES IN THE APPLICATION
- x:Name="StateDisplay" INITIALIZE A VARAIBLE THAT CAN BE
REFERENCED IN C# CODE FOR OBJECT BASED BINDING
OF THIS DISPLAY
- Grid.Row="1" references the 2nd Row of the Grid

THIS PLACES A BOX WITH TEXT LETTING THE USER KNOW WHAT THE
TEXT BELOW REFERENCES

```
<TextBox Text="Logging" HorizontalContentAlignment="Center"
HorizontalAlignment="Stretch" Grid.Row="2"/>
```

THIS WILL CREATE A RICHTEXTBOX, WHICH ALLOWS FOR COLOR AND SIZE MANIPULATION TO HAPPEN

```
<RichTextBox Name="LogDisplay" HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" VerticalScrollBarVisibility="Visible" Grid.Row="3"/>
```

- THIS IS WHERE THE USER WILL BE ABLE TO WHAT THE APPLICATION IS IN (LOGS NEW LINE ROUGHLY EVERY 25 ms)
- x:Name="LogDisplay" INITIALIZE A VARIABLE THAT CAN BE REFERENCED IN C# CODE FOR OBJECT BASED BINDING OF THIS DISPLAY
- Grid.Row="4" references the 4th Row of the Grid

LOGGING DISPLAY INITIALIZATION (GUI IS ABLED TO BE USED)

```
</Grid>
```

```
</UserControl>
```

b. Parameters A1.2

The SCAN PARAMETER DISPLAY DECLARATION - UserControl is Windows recommended method for WPF applications using MVVM and object oriented programming

```
<UserControl x:Class="OperatorInterface.ScanParameterPanel"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:Heart="clr-namespace:OperatorInterface"
mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="300">
```

THIS WILL CREATE AND SET UP THE PLACEMENT REGION FOR DISPLAY THAT Will AUTOMATICALLY BIND EACH OBJECT RIGHT BELOW THE PRIOR OBJECT

```
<Grid>
```

```
<StackPanel VerticalAlignment="Center">
```

THIS PLACES A BORDER AROUND A BOX WITH TEXT LETTING THE USER KNOW WHAT PROTOCOL HAS BEEN SELECTED

```
<Border DockPanel.Dock="Top" Margin="0"
VerticalAlignment="Center" HorizontalAlignment="Stretch">
```

```

BorderThickness="5" BorderBrush="{StaticResource
FMISolidDarkButter}" CornerRadius="5">

    <TextBlock Text="Cardiac Gaited Scans"
    FontWeight="ExtraBold" FontSize="18"
    Foreground="{StaticResource
    FMISolidDarkSkyBlueForegroundBrush}"
    Background="{StaticResource
    FMISolidLightSkyBlueBackgroundBrush}" Margin="3"
    HorizontalAlignment="Stretch"
    TextAlignment="Center"/>

</Border>

```

THIS WILL CREATE A DYNAMIC REGION TO DISPLAY A HEART BEATING

```

<Heart:HeartDisplay x:Name="Heartdisplay" Width="150"
Height="150" />

```

THIS WILL CREATE A DYNAMIC DATAGRID

```

<DataGrid Name="ScanProtocolGrid" Style="{DynamicResource
FMI_Data_Grid}" ItemsSource="{Binding}" Width="auto" Height="auto"
AutoGenerateColumns="False" HorizontalAlignment="Stretch"
HorizontalContentAlignment="Stretch" FontWeight="Bold"
TextBlock.TextAlignment="Center" Padding="0" Visibility="Visible"
VerticalAlignment="Center">

    <DataGridTextColumn Binding="{Binding parameter}"
    IsReadOnly="True" Width="*" Foreground="Black"
    CanUserReorder="False" CanUserSort="False">
        <DataGridTextColumn.Header>
            <DataGridColumnHeader Content="Parameter"
            Height="auto" Padding="0" Margin="0"
            HorizontalAlignment="Stretch"
            HorizontalContentAlignment="Center"
            TextBlock.TextAlignment="Center"/>
        </DataGridTextColumn.Header>
    </DataGridTextColumn>

    <DataGridTextColumn Binding="{Binding value}"
    IsReadOnly="False" Width="*" Foreground="Black"
    CanUserReorder="False" CanUserSort="False">
        <DataGridTextColumn.Header>
            <DataGridColumnHeader
            Content="Value" Height="auto" Padding="0"
            Margin="0" HorizontalAlignment="Stretch"
            HorizontalContentAlignment="Center"
            TextBlock.TextAlignment="Center"/>
        </DataGridTextColumn.Header>
    </DataGridTextColumn>

```

</DataGridTextColumn.Header>
</DataGridTextColumn>

- THIS IS WHERE THE USER WILL BE ABLE TO EDIT AND UPDATE THE SCAN PARAMETERS PRIOR THE INITIAL SCAN PROCESS

SCAN PARAMETER PANEL INITIALIZATION (SCAN PARAMETER PANEL IS ABLED TO BE USED)

</DataGrid.Columns>

</DataGrid>

</StackPanel>

</Grid>

</UserControl>

2. Messaging A2

- a. Outbound (utilities region) A2.1**
 - i. Prepare A2.1.1**
 - ii. Scan A2.1.2**
- b. Inbound (messaging region) A2.2**
 - i. Prepare com A2.2.1**
 - ii. Scan Initiate A2.2.2**
 - iii. Exposure Initiate A2.2.3**
- c. Auto Scribe Status (configuration region) A2.3**

CT Scheduler (B)

1. Messaging B1

- a. Prepare B1.1**
 - i. GUI B1.1.1**
 - ii. X-ray Manager B1.1.2**
- b. Scan B1.2**
 - i. GUI Initiate B1.2.1**
 - ii. GUI Complete B1.2.2**

B. Constraints and Limitations

Our design process was also difficult for us to understand in the beginning because much of the application was either left open for interpretation or clearly defined without the possibility of change (cardiac gating vs other options). While the initial plan

seemed very straight forward, it soon became very difficult to accomplish because we had a large end goal, but none of the specifics needed to reach this goal.

Software development is complex even for experienced software engineers, and has been a difficult chore for our team to overcome. While we have experience with writing code, we do not have a formal education in software design. This has required additional effort to work to understand the requirements of software design.

C. Timeline

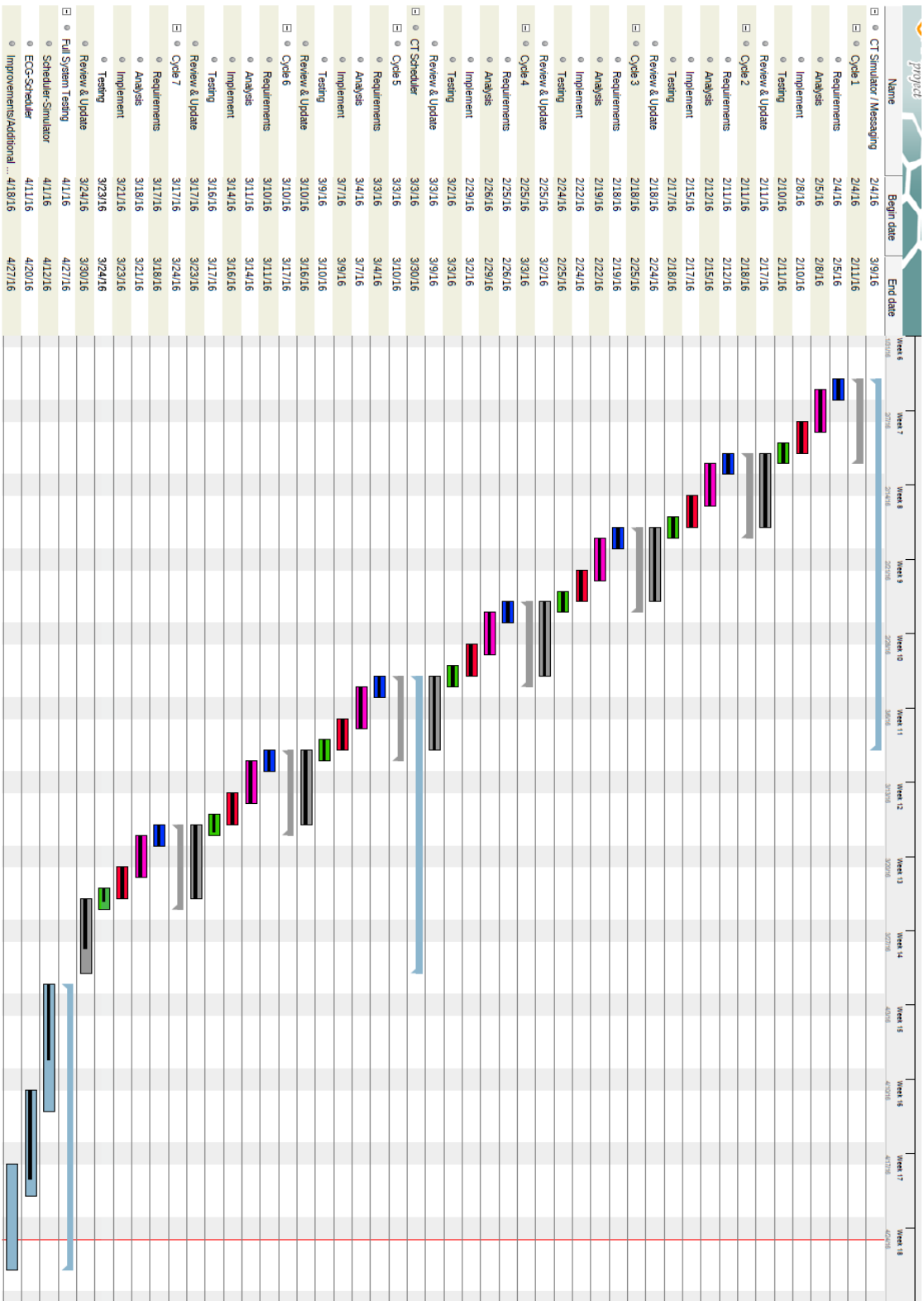
ECGCT Spring Semester

Apr 25, 2016

Tasks

2

Name	Begin date	End date
CT Simulator / Messaging	2/4/16	3/9/16
Cycle 1	2/4/16	2/11/16
Requirements	2/4/16	2/5/16
Analysis	2/5/16	2/8/16
Implement	2/8/16	2/10/16
Testing	2/10/16	2/11/16
Review & Update	2/11/16	2/17/16
Cycle 2	2/11/16	2/18/16
Requirements	2/11/16	2/12/16
Analysis	2/12/16	2/15/16
Implement	2/15/16	2/17/16
Testing	2/17/16	2/18/16
Review & Update	2/18/16	2/24/16
Cycle 3	2/18/16	2/25/16
Requirements	2/18/16	2/19/16
Analysis	2/19/16	2/22/16
Implement	2/22/16	2/24/16
Testing	2/24/16	2/25/16
Review & Update	2/25/16	3/2/16
Cycle 4	2/25/16	3/3/16
Requirements	2/25/16	2/26/16
Analysis	2/26/16	2/29/16
Implement	2/29/16	3/2/16
Testing	3/2/16	3/3/16
Review & Update	3/3/16	3/9/16
CT Scheduler	3/3/16	3/30/16
Cycle 5	3/3/16	3/10/16
Requirements	3/3/16	3/4/16
Analysis	3/4/16	3/7/16
Implement	3/7/16	3/9/16
Testing	3/9/16	3/10/16
Review & Update	3/10/16	3/16/16
Cycle 6	3/10/16	3/17/16
Requirements	3/10/16	3/11/16
Analysis	3/11/16	3/14/16
Implement	3/14/16	3/16/16
Testing	3/16/16	3/17/16
Review & Update	3/17/16	3/23/16
Cycle 7	3/17/16	3/24/16
Requirements	3/17/16	3/18/16
Analysis	3/18/16	3/21/16
Implement	3/21/16	3/23/16
Testing	3/23/16	3/24/16
Review & Update	3/24/16	3/30/16
Full System Testing	4/1/16	4/27/16
Scheduler-Simulator	4/1/16	4/12/16
ECG-Scheduler	4/11/16	4/20/16
Improvements/Additional Features	4/18/16	4/27/16



D. Budget

At this time there has been no use of the provided budget. There are no current plans to use the provided budget.