<<Git Guys>>

<Finding and Reporting Information Console>

Software Design Document

Version <1.0>

<December 9,2020>

Document Control

Approval

The guidance team and clients shall approve this document.

Document Change Control

Initial Release:	1.0
Current Release:	1.0
Indicator of Last Page in Document:	</3
Date of Last Review:	12/9/2020
Date of Next Review:	12/8/2020
Target Date for Next Update:	12/9/2020

Distribution List

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members: Dr. Salamah

Customer: Cyber Experimentation & Analysis Division (CEAD)

Software Development Team: Alex Vasquez, Luis Soto, Andrew Clanan, Isaias Leos, Jacob Padilla

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
0.1.0		Team	Assignments: Alex Vasquez ● Section 5,6,7 ● 2.1 ● Document Formatting Andrew Clanan ● Section 3 ○ CRC Cards 4-6

			● Revision of Entire Document **(DETAILED)** Isaias Leos ● Section 3 ○ CRC Cards Event, System, Task, Subtask. ● Revision of Entire Document **(DETAILED)** Luis Soto ● Section 4 ○ CRC Cards 10-12 ● Revision of Entire Document **(DETAILED)** Jacob Padilla ● Finish Entire Introduction (Section 1) ● Section 2.2 ● Section 4 ● Revision of Entire Document **(DETAILED)**
0.2.0	12/8/2020	Alex Vasquez	Revised CRC Cards, Contracts, and Collaboration Graph
0.3.0	12/8/2020	Jacob Padilla	Introduction section 1
0.4.0	12/8/2020	Isaias Leos	Modified Collaboration Graph to be visible.
0.4.1	12/8/2020	Isaias Leos	Added CRC card and more information regarding section 3.
0.4.2	12/9/2020	Team	Review #1, Revisions, Final Review
0.5.0	12/9/2020	Isaias Leos	Added Class Description for System + contracts
0.6.0	12/9/2020	Isaias Leos	Added Class Description for Subtask Manager + Contracts
0.7.0	12/9/2020	Isaias Leos	Fixed Collaboration Graph for small inconsistency i.e. arrows coming from wrong direction.
0.8.0	12/9/2020	Alex Vasquez	Subsystem 5,6,7
0.9.0	12/9/2020	Isaias Leos	Reviewed more sections and fixed issues with post/pre conditions and formatting of the crc cards.

0.10.0	12/9/2020	Luis Soto Jacob Padilla	4. Detailed Description of Subsystem(Field Assignment) Includes: Added Class Description for Finding + contracts Added Class Description for progress + contracts Added Class Description for analyst + contracts Added Class Description for archive + contracts Added Class Description for report Added Class Description for System + contract Added Class Description for System + contracts Added Class Description for Progress server + contracts Added Class Description for Archive server + contracts
0.11.0	12/9/2020	Alex Vasquez	Section 2.0 and 2.1
0.12.0	12/9/2020	Alex Vasquez	Formatting for document, document hygiene.
0.13.0	12/9/2020	Alex Vasquez	Section 1.4
0.14.0	12/9/2020	Isaias Leos	Improved section 1.2
1.0	12/9/2020	Team	Last team review (EVER)

Table of Contents

1. Introduction

1.1 Purpose and Intended Audience

The purpose of the Software Design Document (SSD) is to aid in the development of the design and structure of FRIC. The SDD illustrates how the system can be separated into components to help make the implementation and guidance of the design manageable.This document is only intended to be viewed by the Cyber Experimentation and Analysis Division team (CEAD), the Development Team, and the Guidance Team. Contents shall not be disclosed, discussed, or shared with any individuals/parties that are not mentioned above.

1.2 Scope of Product

The Cyber Experimentation & Analysis Division acknowledges the complexity and time it takes in order to complete a cyber engagement. This includes managing task assignments, monitoring progress, documenting vulnerabilities discovered during cyber engagements, and generating reports that present potential issues to CEAD's clientele [1]. CEAD requested a system that would aid in the management of tasks, collection of evidence, and generated reports during a cyber engagement.

The system will provide the ability to manage task assignment, keep track of progress, manage the collection of evidence during engagements, and generate custom reports. FRIC is offered through a web application that can only be accessed by a local network.

Benefits of FRIC include providing the capability to allow a user to be able to create or access a new event using FRIC. Storing data entered into a database, while so having the ability of moving data to other databases. Allow the documentation of Events, Systems, while also logging all actions done within the system. Allow the management of tasks and subtasks given within the event and having a field to document the vulnerabilities into a finding. The system shall also provide a notification based on due dates and task progress while also allowing synching between analysts to share findings between each other. Lastly archiving and saving information from previous cyber events to see how previous vulnerabilities were documented in the past. The goal of FRIC is to assist the CEAD team with documenting and reporting the aforementioned vulnerabilities.

The development of FRIC was broken down into 5 iterations following the requirements provided on the SRS and Client memo. The following link provides detail on how those iterations were implemented:

https://minersutep-my.sharepoint.com/:v:/g/personal/ajclanan_miners_utep_edu/Eerb5zcjOgVCtTYUD548S40BRRxib0ciJqUqvGzOWMmqVg?e=SCMgxp

1.3 References

1. Tai Ramirez, Elsa, Findings and Reporting Information Console (FRIC) [SRS] El Paso, TX: University of Texas at El Paso, 2020

1.4 Definitions, Acronyms, and Abbreviations

Definitions

Word	Definition
Responsibility	Something an object (server) does for other objects (clients)
Contract	Set of cohesive responsibilities that a client depends on
Subsystem	Group of classes that collaborate among themselves to support a set of contracts
Class	An extensible program-code for creating objects, providing initial values for state variables, and implementations of behavior (member functions or methods)
Collaboration Graph	A collaboration diagram is a type of visual presentation that shows how various software classes interact with each other
Protocol	Set of signatures (method name, input parameters, return type) for methods to be implemented.

Acronyms and Abbreviations

Acronyms

FRIC	Finding and Reporting Information Console
SDD	Software Design Document
CEAD	Cyber Experimentation and Analysis Division

SRS	Software Requirements Specification
CRC	Class Responsibility Collaborators

Abbreviations

e.g.	example

1.5 Overview

The Document is separated into 7 different sections: Introduction, Decomposition Description, and Detailed Description of the subsystem. Introduction states the purpose of the document, scope of FRIC, any references, syntax, and overall view of the document. Decomposition Description states the collaboration Diagram and subsystem description. Detailed Description of the subsystem contains Subsystem cards, and CRC cards including contracts with protocols.

2. Decomposition Description

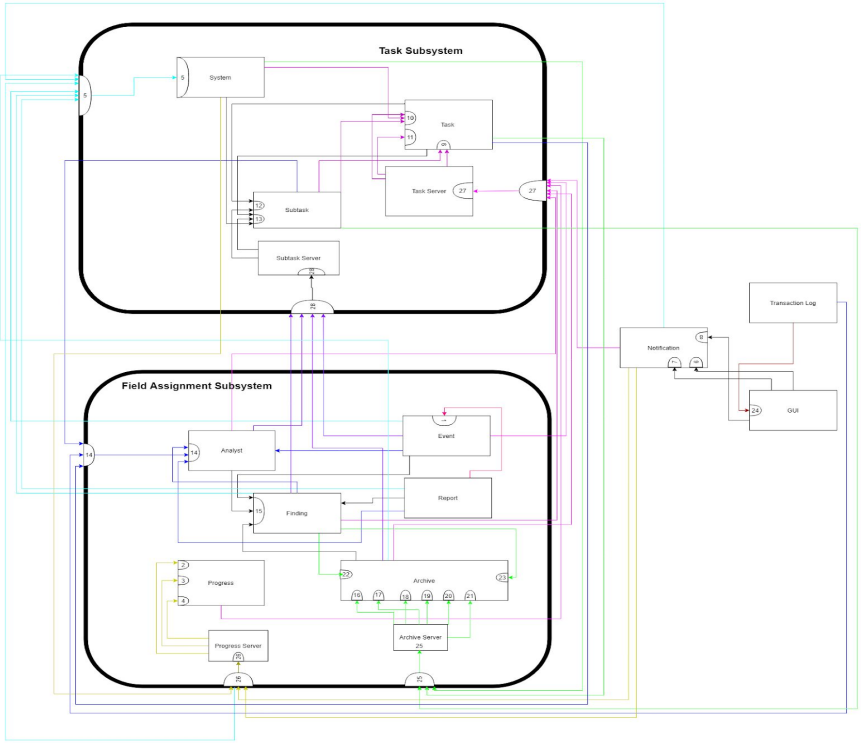
The following section contains information concerning the Subsystems of FRIC. Designers and Maintainers may use this information to deploy updates and patches in future versions. FRIC is two Major Subsystems and three freestanding classes. The collaboration between the class and subsystems are represented by lines that are pointing to the various contracts, which are represented by bubbles inside of the class.

2.1 System Collaboration Diagram

FRIC has two major Subsystems which are: Task Subsystem and Field Assignment Subsystem. Outside of these two Subsystems we have 3 classes which are Notification, Transaction Log, and GUI.

Task Task Subsystem has 3 contracts in which outside entities can use. Contract 27 is associated with the Task Server which is the superclass of Task. Any request made to Task Server can be a request to Task. The same logic applies for the other two contracts of this Subsystem which are contracts 5 and 28.

Field Assignment subsystem also has 3 contracts that outside entities can use. These contracts are 14,25, and 26. These contracts are all for superclasses associated with a class inside. They include Archive Server, Progress Server, and Analyst.



Link to graph: https://drive.google.com/file/d/1nPD4Smo4JVVmiQwgEOao-zxmWDHYXHua/view

The system has been structured so that the Analyst and everything the analyst can produce and or do is put into a subsystem, while the hierarchy with the exception of the Event itself is put into another. Two entities exchange common requests to the various classes inside of each subsystem, this is also another reason the development team decided to group the classes in this manner. There are 3 freestanding classes that have functionality that is not closely related with the other two subsystems which are Notification, GUI, and Transaction Log. These 3 classes do not relate with each other enough in what they request so the development team decided they were best suited to be alone.

2.2 Subsystem Descriptions

Field Assignment Subsystem

The event subsystem is a way to abstract all the interactions taking place between Event, Task, Subtask, System, Progress, and Archive.

Classes:

- Event
- Report
- Archive
- Archive Server
- Analyst
- Finding
- Progress
- Progress Server
- Finding
- Analyst

Contracts of subsystem:

- 14 Analyst
- 26 Progress Server
- 25 Archive Server

Field Assignment Subsystem listed in Section 4 of Design Document

Task Subsystem

The finding subsystem is a way to abstract all the interactions taking place between Finding and Report.

Classes:

- System
- Task
- Task Server

- Subtask
- Subtask Manager

Contracts:

- 5 System
- 27 Task Server
- 28 Subtask Manager

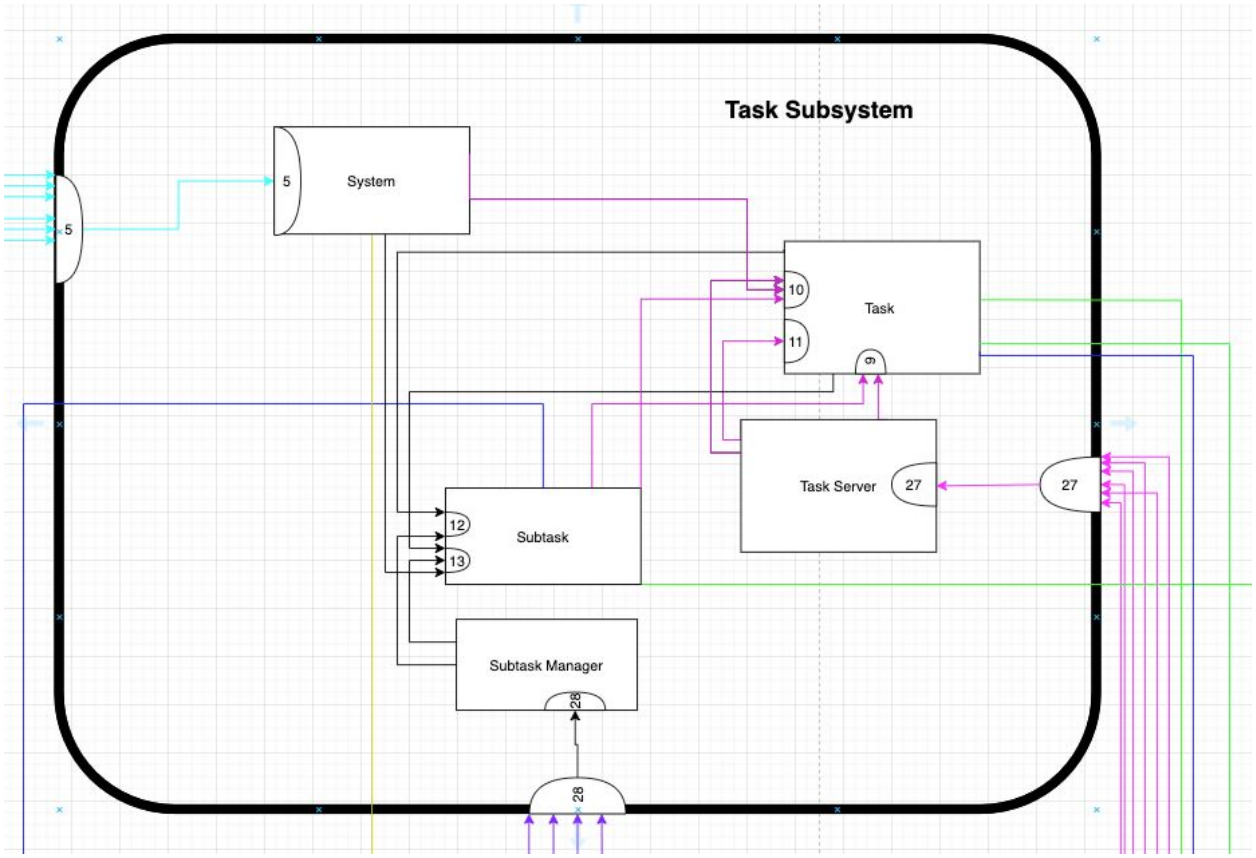
Task Subsystem listed in Section 4 of Design Document

3. Detailed Description of Subsystem: Task Subsystem

This will be the main component of the F.R.I.C system. This subsystem will be what the analyst will be using to save and add information regarding the cyber engagement specified by CEAD.

3.1 Task Subsystem Description (subsystem cards)

The name of the subsystem will be called Task Subsystem and the purpose of this system is to allow analysts to manage systems and work on the necessary/assigned task and subtask for a given event. This system will be working together with another subsystem to document all necessary information regarding said cyber event. Analysts will be using the majority of these systems to identify and populate cyber attacks that said analyst will be working on.



3.2 Class Description <System>

Class Name: System	
Description: This class represents the systems being tested by CEAD analysts.	
Private Responsibilities: ● Archive System ● Restore System ● Add System ● Edit System ● Knows Tasks for System ● Knows Subtasks for System ● Knows current System Progress	**Collaborations**: Archive(22) Archive(18) Task(10) Subtask(27) Progress(4)
Superclass: N/A	
Subclass: N/A	

3.2.1 Contract <Provide Systems>

Class Name: System	
Superclass: N/A	
Collaboration Diagram Reference: 5	
Class Description: This class represents the systems being tested by CEAD analysts	
---Contracts---	
Contract Name: Provide Systems	
Contract Description: This will provide all information regarding the system object.	
---Protocols---	
Responsibilities: 1. Provide System Information a. Provide System information	**Collaborations**:
Method Signature: public static Event provideSystem(){}	

Algorithmic Description: This will obtain a system from the given systemList, which requires there to be at least one system inside that list. It will also return that system and ensure that there is a system that will be returned.
Pre-conditions: //@ requires systemList.length != 0
Post-conditions: //@ ensures Systems != null && > \old(systemList.length) == systemList.length
Comments: **System Information includes:** Knows System description, Knows System Name, Knows System Location, Knows System Router, Knows System Switch, Knows System Room, Knows Test Plan, Knows Confidentiality, Knows Integrity, Knows Availability, Knows Access, and Knows Mitigation

3.3 Class Description <Task>

Class Name: Task	
Description: This class will contain all information related to a Task.	
Private Responsibilities: ● Knows Task Information ● Create Task ● Edit Task ● Archive Task ● Restore Task ● Demote Tasks to Subtask ● Knows Collaborator	**Collaborations**: Archive(21) Archive(16) Analyst(14)
Superclass: Task Server	
Subclass: N/A	

3.3.1 Contract <Associate Tasks to Subtasks>

Class Name: Task
Superclass: Task Server

Collaboration Diagram Reference: 9	
Class Description: This class will provide all the information related to the task class.	
--Contracts--	
Contract Name: Associate Tasks to Subtasks	
Contract Description:This contract associates a task to a subtask.	
--Protocols--	
Responsibilities: 9. Associate Tasks to Subtasks a. Obtained Specified Subtask b. Set a link between the task to subtask.	**Collaborations**: Subtask(27) Subtask(27)
Method Signature:public static boolean assocTaskToSub(int subtaskID, int taskID){}	
Algorithmic Description: Using the IDs of a task and subtask we can create a database record that uses those IDs as the primary thus forming an association between them.	
Pre-conditions: //@ requires subtaskID > 0 && taskID > 0	
Post-conditions: //@ensures Arrays.asList(task.listOfAssocSubtks).contains(subtaskID)	

3.3.2 Contract <Provide Task>

3.3.3 Contract <Provide Task Due Date>

Class Name: Task
Superclass: Task Server
Collaboration Diagram Reference: 11
Class Description: This class will provide all the information related to the task class.
---Contracts---

Contract Name: Provide Task Due Date	
Description:Given a task, return the due date.	
---Protocols---	
Responsibilities: 11. Provides Task Due Date a. Obtain due date of Task. b. Provide due date of Task	**Collaborations**:
Method Signature:public static Date taskDueDate(int taskID){}	
Algorithmic Description: Using the ID of a task we can obtain all the information for a task and therefore the task due date. Which will need to be a valid date.	
Pre-conditions://@ requires taskID > 0	
Post-conditions: //@ensures task.dueDate != null	

3.4 Class Description <Subtask>

Class Name: Subtask	
Description: This class will contain all information related to a Subtask.	
Private Responsibilities: ● Create Subtask ● Edit Subtask ● Archive Subtask ● Restore Subtask ● Promote Subtasks to Task ● Knows Collaborator ● Assign a Subtask to Analyst	**Collaborations**: Archive(20) Archive(17) Analyst(14) Analyst(14)
Superclass: Subtask Manager	
Subclass: N/A	

3.4.1 Contract <Associate Subtask>

<table>
<tr><td colspan="2">Class Name: Subtask</td></tr>
<tr><td colspan="2">Superclass: Subtask Manager</td></tr>
<tr><td colspan="2">Collaboration Diagram Reference: 12</td></tr>
<tr><td colspan="2">Class Description: This class will contain all information related to a Subtask.</td></tr>
<tr><td colspan="2">---Contracts---</td></tr>
<tr><td colspan="2">Constract Name: Associate Subtask to Task</td></tr>
<tr><td colspan="2">Contract Description: This contract associates a subtask to a specified task, so when looking at task information we will be able to view subtasks related to it.</td></tr>
<tr><td colspan="2">---Protocols---</td></tr>
<tr><td>Responsibilities:
1. Associate Subtask to Tasks
a. Obtained Specified Task
b. Obtained Specified Subtask
c. Set a link between the Subtask to Task.</td><td>Collaborations:

Task (10)

Task(9)</td></tr>
<tr><td colspan="2">Method Signature:public static boolean assoSubToTask(int subtaskID, int taskID){}</td></tr>
<tr><td colspan="2">Algorithmic Description: This contract will accept the subtaskID and taskID and will look up both IDs and add an that ID into an array that is contained inside the subtask.</td></tr>
<tr><td colspan="2">Pre-conditions: //@ requires subtaskID > 0 && taskID > 0</td></tr>
<tr><td colspan="2">Post-conditions: //@ ensures Arrays.asList(subtask.listOfAssocTasks).contains(taskID)</td></tr>
</table>

3.4.2 Contract <Provide Subtask>

Class Name: Subtask	
Superclass: Subtask Manager	
Collaboration Diagram Reference: 13	
Class Description: This class will contain all information related to a Subtask.	
---Contracts---	
Constract Name: Provide subtask	
Contract Description: Provide all information that makes up a subtask.	
---Protocols---	
Responsibilities: 13. Provide Subtask a. Provide Subtask Information	**Collaborations**:
Method Signature: public static Subtask provideSubtask(){}	
Algorithm Description: This will obtain a subtask from the given subtaskList, which requires there to be at least one system inside that list. It will also return that system and ensure that there is a subtask that will be returned.	
Pre-conditions: //@ requires subtaskList.length() > 0	
Post-conditions: //@ensures subtask != null	

3.5 Class Description <Subtask Manager>

Class Name: Subtask Manager
Description: This class will handle requests from outside subsystems.

Private Responsibilities:	**Collaborations**:
Superclass: N/A	
Subclass: Subtask	

3.5.1 Contract <Handle Request>

Class Name: SubtaskManager	
Superclass: N/A	
Collaboration Diagram Reference: 28	
Class Description: This class will manage and redirect requests for subtasks.	
---Contracts---	
Constract Name: Associate Subtask to Task	
Contract Description: This contract associates a subtask to a specified task, so when looking at task information we will be able to view subtasks related to it.	
---Protocols---	
Responsibilities: 2. Handle Subtask Request a. Identify the request b. Redirect request. c. Depending on the request i. Return Subtask d. Or i. Associate Subtask to Task	**Collaborations**: Subtask(12) Subtask(13)

Method Signature: public static void handleRequest(){}
Algorithmic Description:
Pre-conditions: 1. Requests must not be empty 2. There must be at least one subtask in FRIC 3. Subtasks must be in the active event. 4. There must be at least one task in FRIC
Post-conditions://@ensure \result(conditionMet) == true

3.6 Class Description <Task Server>

Class Name: Task Manager	
Description: This class will manage the responsibilities of the task class.	
Responsibilities: ● Handle all requests made to the task class	**Collaborations**: Task 9,10,11
Superclass: N/A	
Subclass: Task	

3.6.1 Contract <Task Server>

Class Name: Task Server

Superclass: N/A	
Collaboration Diagram Reference: 27	
Class Description: This class will manage the responsibilities of the task class	
---Contracts---	
Constract Name: Task Server	
Contract Description: This contract manages the requests made to task.	
---Protocol---	
Responsibilities: Handle Task Request ● Identify request ● Send request to appropriate contract: ○ Associate tasks to subtask ○ Provide task ○ Provide task due date	**Collaborations**: Task (9) Task(10) Task(11)
Method Signature:public static boolean taskServer(Request req){}	
Algorithmic Description: This contract will direct the request to the appropriate contract using the descriptor of the request.	
Pre-conditions: //@ requires req != null && [Associate tasks to subtask,Provide task,Provide task due date].contains(req) && taskList.length > 0	
Post-conditions: //@ensure \result(conditionMet) == true	

4. Detailed Description of Subsystem: Field Assignment Subsystem

This component of F.R.I.C will be where event, finding, report, analyst, progress, and archive will interact.

4.1 Field Assignment Description (subsystem cards)

The name of this subsystem will be called Field Assignment. The following subsystem would be in charge of handling the interactions between event, finding, report, analyst, progress, and archive.

4.2 Class Description <Event>

<table>
<tr><td colspan="2">Class Name: Event</td></tr>
<tr><td colspan="2">Description: The cyber engagement specified by CEAD in which this class will contain all information related to the engagement.</td></tr>
<tr><td>Private Responsibilities:
Knows No. of System
Knows No. of Findings
Knows No. of Tasks
Knows No. of Subtasks
Knows Analysts in Event
Create New Event
Edit Existing Event</td><td>Collaborations:
System(5)
Findings(15)
Tasks(10)
Subtasks(13)
Analyst(14)</td></tr>
<tr><td colspan="2">Superclass: N/A</td></tr>
<tr><td colspan="2">Subclass: Field Assignment</td></tr>
</table>

4.2.1 Contract < Provide event >

<table>
<tr><td colspan="2">Class name: Event</td></tr>
<tr><td colspan="2">Superclass: N/A</td></tr>
<tr><td colspan="2">Collaboration Diagram Reference: 1</td></tr>
<tr><td colspan="2">Class Description:The cyber engagement specified by CEAD in which this class will contain all information related to the engagement.</td></tr>
<tr><td colspan="2">--Contracts--</td></tr>
<tr><td colspan="2">Contract Name: Provide Event</td></tr>
<tr><td colspan="2">Contracts Description:This will provide certain information regarding the Event class.</td></tr>
<tr><td colspan="2">--Protocols--</td></tr>
<tr><td>Responsibilities:
1. Provide Event
a. Knows Event Information
b. Provides the current Event information</td><td>Collaborations:</td></tr>
<tr><td colspan="2">Method Signature: public static Event getEvent(int eventId){}</td></tr>
<tr><td colspan="2">Algorithm Description: The algorithm would receive the eventId and it would retrieve the event with the specified eventId.</td></tr>
<tr><td colspan="2">Pre-conditions: //@ requires \eventlist.length > 0</td></tr>
<tr><td colspan="2">Post-conditions: //@ ensures \result(findingInformation) != null</td></tr>
</table>

Comments: **Event Information:** Knows Event Type,Knows Version,Knows Assessment Date,Knows Organization Name,Knows Security Classification ,Knows Title guide, Knows Event Classification, Knows Declassification Date, and Knows Customer Name, Event Name, Number of Systems, Number of Findings, Number of Analysts

4.3 Class Description <Report>

Class Name: Report	
Description: A collection of all findings, artifacts found in a given event in a formatted document.	
Private Responsibilities: ● Provide a ERB report. ○ Obtain Event Information. ○ Obtain System Information. ○ Obtain Finding Information ○ Obtain Analyst Information ○ Generate ERB report. ● Provide a Technical report. ○ Obtain Event Information. ○ Obtain System Information. ○ Obtain Finding Information ○ Obtain Analyst Information ○ Generate ERB report. ● Provide a Risk Assessment Report ○ Obtain Event Information. ○ Obtain System Information. ○ Obtain Finding Information ○ Obtain Analyst Information ○ Generate ERB report. ● Provide Analyst Who Generated Report ○ Obtain Analyst who Generated Report	**Collaborations**: Event(1) System(5) Findings(15) Analyst(14) Event(1) System(5) Findings(15) Analyst(14) Event(1) System(5) Findings(15) Analyst(14) Analyst(14)
Superclass: N/A	

Subclass: Field Assignment

4.4 Class Description <Finding>

Class Name: Finding	
Description: This class will contain all information related for a finding	
Private Responsibilities: Know Analyst Information Know Task Information Know Subtask Information Know System Information Associate Finding(s) Create Finding Archive Finding Restore Archived Finding	Collaborations: Analyst(14) Task(10) Subtask(27) System(5) Archive(23) Archive(19)
Superclass: N/A	
Subclass: Field Assignment	

4.4.1 Contract < Provide Finding >

Class Name: Finding
Superclass: N/A
Collaboration Diagram Reference: 15
Class Description: This will provide information regarding Finding class.
---Contracts---
Contract Name: Provide Finding
Contract Description: This will provide certain information regarding the Finding class.
---Protocol---

Responsibilities: 15. Provide Finding a. Provide Finding Information	**Collaborations**:
Method Signature: public static Finding getFinding(int findingId){}	
Algorithm Description: Algorithm Description: The algorithm would receive the findingId and it would retrieve the finding with the specified findingId from the finding database.	
Pre-conditions: //@ requires \findinglist.length > 0	
Post-conditions: //@ ensures \result(findingInformation) != null	
Comments: Finding Information: Knows ID, Hostname, IP Port, Description, Long Description, Status, Type, Classification, Evidence, Confidentiality, Integrity, Availability, Posture, Mitigation Description, Threat Relevance, Effectiveness Rating, Impact Description, Impact Level, Severity Category Code, Severity Category Score, Vulnerability Severity, Quantitative Vulnerability Severity, Risk, Likelihood, Confidentiality Finding Impact on System, Integrity Finding Impact on System, Availability Finding Impact on System, Impact Score	

4.5 Class Description <Analyst>

Class Name: Analyst
Class Description: This class will contain information about analysts.
Description: This class will contain all information related to an analyst.

Private Responsibilities: ● Knows overall Task Progress ● Knows overall System Progress ● Add a collaborator to a specific task. a. Obtain the Task b. Obtain Analyst c. Assign Collaborator to Task ● Add a collaborator to a specific subtask. a. Obtain the Subtask b. Obtain Analyst c. Assign Collaborator to Subtask ● Add collaborator to a specific finding. a. Obtain the finding b. Knows Analyst c. Assign Collaborator to Finding ● Assign a Task to Analyst ○ Obtain Task ○ Knows Analyst ○ A link is set between task and analyst.	Collaborations: Task(10) Subtask(27) Finding(15) Task(10)
Superclass: N/A	
Subclass:Field Assignment	

4.5.1 Contract <Provide Analyst >

Class Name: Provide Analyst
Superclass: N/A
Collaboration Diagram Reference: 14
Class Description: This class will contain information about analysts.
--Contracts--
Contract Name: Provide Analyst
Contract Description: This will provide certain information regarding the Analyst class.

---Protocol---	
Responsibilities: 14. Provide Analyst b. Provide Analyst Information	**Collaborations**:
Method Signature: public static Analyst getAnalyst(int analystID){}	
Algorithm Description: The algorithm would receive the analystId and it would retrieve the analyst with the specified analystId from the analyst database.	
Pre-conditions: //@ requires \AnalystList.length > 0	
Post-conditions: //@ ensures \result(analystsInformation) != null	
Comments: Knows Analyst Information: (Knows Analyst Initials, Knows First Name, Knows Last Name, Knows Role, Knows IP,and Knows Posture)	

4.6 Class Description <Progress>

Class Name: Progress	
Description: This class will contain all information related to progress.	
Private Responsibilities:	Collaborations:
Superclass: Progress Server	
Subclass: N/A	

4.6.1 Contract <Calculate Tasks Progress >

Class Name: Progress
Superclass: Progress Server

Collaboration Diagram Reference: 27	
Class Description: This class provides details of progress class	
--Contracts--	
Contract Name: Calculate Tasks Progress	
Contract Description:This will provide certain information regarding the Progress class.	
---Protocol---	
Responsibilities: 2. Calculate Tasks Progress a. Obtain Tasks b. Average Task progress with existing Tasks progress c. Provides the calculated task progress	**Collaborations**: Tasks(10)
Method Signature: Public static Progress GetTaskProgress(int tasksID){}	
Algorithm Description: The algorithm would add the progress of all tasks and it would be divided by the number of tasks in the system in order to return the tasks progress.	
Pre-conditions: //@ requires \tasksList.length > 0	
Post-conditions: //@ ensures \result(ProgressOfAllTask / tasksList.length) == tasksTotalprogress	
Comments: The progress for each attribute will be displayed in increments of 10%. Progress of System SRS 16. Referenced from overview tables.	

4.6.2 Contract <Calculate Subtasks Progress >

Class Name: Calculate Subtasks Progress
Superclass: Progress Server
Collaboration Diagram Reference: 28
Class Description: This class provides details of progress class
--Contracts--

Contract Name: Calculate Subtasks Progress	
Contract Description:This will provide certain information regarding the Progress class.	
---Protocol---	
Responsibilities: 3. Calculate Subtasks Progress a. Obtain Subtask b. Average Subtask progress with existing Subtasks progress c. Provides the calculated subtasks progress	**Collaborations**: Subtasks(13)
Method Signature: Public static Progress GetSubtaskProgress(Subtasks subtasks){}	
Algorithm Description: The algorithm would add the progress of all subtasks and it would be divided by the number of subtasks in the system in order to return the subtasks progress.	
Pre-conditions: //@ requires \subtasksList.length > 0	
Post-conditions: //@ ensures \result(subtaskProgress / subtask.length) == subtaskTotalProgress	
Comments: The progress for each attribute will be displayed in increments of 10%. Progress of System SRS 16. Referenced from overview tables.	

4.6.3 Contract <Calculate System Progress >

Class Name: Progress
Superclass: Progress Server
Collaboration Diagram Reference: 5
Class Description: This class provides details of progress class
--Contracts--
Contract Name: Calculate System Progress
Description:This will provide certain information regarding the Progress class.

---Protocol---	
Responsibilities: 4. Calculate System Progress a. Obtain System b. Average System progress with existing Systems progress c. Provides the calculated System progress	**Collaborations**: System (5)
Method Signature: Public static Progress GetSystemProgress(System system){}	
Algorithm Description: The algorithm would add the progress of all systems and it would be divided by the number of systems in the system in order to return the system's progress.	
Pre-conditions: //@ requires \systemList.length > 0	
Post-conditions: //@ ensures \result(systemProgress / system.length) == systemsTotalProgress	
Comments: The progress for each attribute will be displayed in increments of 10%. Progress of System SRS 16. Referenced from overview tables.	

4.7 Class Description <Archive>

Class Name: Archive	
Description: This class will contain all information and functionality related to Archive.	
Private Responsibilities:	Collaborations:
Superclass: Archive Server	
Subclass: N/A	

4.7.1 Contract <Restore Task>

<table>
<tr><td colspan="2">Class Name: Archive</td></tr>
<tr><td colspan="2">Superclass: Archive Server</td></tr>
<tr><td colspan="2">Collaboration Diagram Reference: 16</td></tr>
<tr><td colspan="2">Class Description: Description: This class will contain all information and functionality related to Archive.</td></tr>
<tr><td colspan="2">---Contracts---</td></tr>
<tr><td colspan="2">Contract Name: Restore Task</td></tr>
<tr><td colspan="2">Contract Description: This class is responsible for restoring an archived task.</td></tr>
<tr><td colspan="2">---Protocols---</td></tr>
<tr><td>Responsibilities:
16. Restore Task
a. Knows archived task information
b. Identify Task as Active
c. Move archived Task into Task</td><td>Collaborations:</td></tr>
<tr><td colspan="2">Method Signature: public static boolean restoreTask(int taskID){}</td></tr>
<tr><td colspan="2">Algorithm Description: The algorithm would be in charge of receiving a taskId of a task and it would move the task with the specified taskId from the archived task database to the task database.</td></tr>
<tr><td colspan="2">Pre-conditions: //@ requires taskId > 0</td></tr>
<tr><td colspan="2">Post-conditions: //@ ensures \result(archivedTaskList.contains(taskID)) == false</td></tr>
</table>

4.7.2 Contract <Restore Subtask>

Class Name: Archive

Superclass: Archive Server	
Collaboration Diagram Reference: 28	
Class Description: This class will contain all information and functionality related to Archive.	
---Contracts---	
Contract Name: Restore Subtask	
Contract Description:This class is responsible for restoring an archived subtask.	
---Protocol---	
Responsibilities: 17. Restore Subtask a. Knows archived Subtask information b. Identify Subtaskask as Active c. Move archived Subtask into Subtask	**Collaborations**:
Method Signature: Public static boolean restoreSubtask(int subtaskId){}	
Algorithm Description: The algorithm would be in charge of receiving a subtaskId of a subtask and it would move the subtask with the specified subtaskId from the archived subtask database to the subtask database.	
Pre-conditions: //@ requires subtaskId > 0	
Post-conditions: //@ ensures \result(archivedsubtaskList.contains(subtaskId)) == false	

4.7.3 Contract <Restore System>

Class Name: Archive
Superclass: Archive Manager

Collaboration Diagram Reference: 18	
Class Description: This class will contain all information and functionality related to Archive.	
--Contracts--	
Contract Name: Restore System	
Contract Description: This class is responsible for restoring an archived system	
---Protocol---	
Responsibilities: 18. Restore System a. Knows archived System information b. Identify System as Active c. Move archived System into System	**Collaborations**:
Method Signature: Public static boolean restoreSystem(int systemId){}	
Algorithm Description: The algorithm would be in charge of receiving a systemId of a system and it would move the system with the specified systemId from the archived system database to the system database.	
Pre-conditions: //@ requires systemId > 0	
Post-conditions: //@ ensures \result(archivedSystemList.contains(systemId)) == false	
Comments:	

4.7.4 Contract <Restore Finding>

Class Name: Archive
Superclass: Archive Server
Collaboration Diagram Reference: 19

Class Description: This class will contain all information and functionality related to Archive.	
--Contracts--	
Contract Name: Restore Finding	
Contract Description: This class is responsible for restoring an archived finding.	
---Protocol---	
Responsibilities: 19. Restore Finding a. Knows archived Finding information b. Identify Finding as Active c. Move archived Finding into Finding	**Collaborations**:
Method Signature: Public static boolean restoreFinding(int findingId){}	
Algorithm Description: The algorithm would be in charge of receiving a FindingId of a finding and it would move the finding with the specified findingId from the archived finding database to the finding database.	
Pre-conditions: //@ requires findingId > 0	
Post-conditions: //@ ensures \result(archivedTaskList.contains(findingId)) == false	
Comments:	

4.7.5 Contract <Archive Subtask>

Class Name: Archive
Superclass: Archive Server
Collaboration Diagram Reference: 27

Class Description: This class will contain all information and functionality related to Archive.	
--Contracts--	
Contract Name: Archive Subtask	
Contract Description: This class is responsible for archiving subtasks.	
---Protocol---	
Responsibilities: 20. Archive Subtask a. Obtains Subtask Information b. Moves Subtask Information Into Archive c. Identifies Subtask as not active in FRIC	**Collaborations**: Subtask(27)
Method Signature: Public static boolean archiveSubtask(int subtaskId){}	
Algorithm Description: The algorithm would be in charge of receiving subtaskId and it would move the subtask with the specified subtaskId from the subtask database to the archived subtasks database.	
Pre-conditions: //@ requires subtaskList.length > 0	
Post-conditions: //@ensures \result(Arrays.asList(archivedSubtasks).contains(subtaskId)) == true	

4.7.6 Contract <Archive Task>

Class Name: Archive
Superclass: Archive Server
Collaboration Diagram Reference: 27
Class Description: This class will contain all information and functionality related to Archive.

--Contracts--	
Contract name: Archive Task	
Contract Description: This class is responsible for archiving tasks.	
---Protocol---	
Responsibilities: 21. Archive Task a. Obtains Task Information b. Moves Task Information Into Archive c. Identifies Task as not active in FRIC.	**Collaborations**: Task(10)
Method Signature: Public static boolean archiveTask(int taskId){}	
Algorithm Description: The algorithm would be in charge of receiving taskId and it would move the task with the specified taskId from the task database to the archived tasks database.	
Pre-conditions: //@ requires task.length > 0	
Post-conditions://@ensures \result(Arrays.asList(archivedTasks).contains(taskId)) == true	

4.7.7 Contract <Archive System>

Class Name: Archive
Superclass: Archive Server
Collaboration Diagram Reference: 5
Class Description: This class will contain all information and functionality related to Archive.
--Contracts--
Contract Name: Archive System
Description: This class is responsible for archiving systems.

---Protocol---	
Responsibilities: 22. Archive System a. Knows System Information b. Moves System Information Into Archive c. Identifies System as not active in FRIC	**Collaborations**: System(5)
Method Signature: Public static boolean archiveSystem(int systemid){}	
Algorithm Description: The algorithm would be in charge of receiving systemId and it would move the system with the specified systemkId from the system database to the archived system database.	
Pre-conditions: //@ requires systemList.length > 0	
Post-conditions://@ensures \result(Arrays.asList(archivedSystems).contains(systemId)) == true	

4.7.8 Contract <Archive Finding>

Class Name: Archive	
Superclass: Archive Server	
Collaboration Diagram Reference: 15	
Class Description: This class will contain all information and functionality related to Archive.	
--Contracts--	
Contract Name: Archive Finding	
Description: This class is responsible for archiving findings.	
---Protocl---	
Responsibilities: 23. Archive Finding a. Knows Finding Information b. Moves Finding Information Into Archive	**Collaborations**: Finding(15)

c. Removes Finding Information from Finding List	
Method Signature: Public static boolean archiveFinding(int findingId){}	
Algorithm Description: The algorithm would be in charge of receiving findingId and it would move the finding with the specified findingId from the subtask database to the archived finding database.	
Pre-conditions: //@ requires findingList.length > 0	
Post-conditions://@ensures \result(Arrays.asList(archivedFindings).contains(findingId)) == true	

4.8 Class Description <Progress Server>

Class Name: Progress server	
Description: This class will manage the responsibilities of the progress class.	
Responsibilities: • Handle all requests made to the progress class	**Collaborations**: Finding 2,3,4
Superclass: N/A	
Subclass: Progress	

4.8.1 Contract <HandleProgressRequest>

Class Name: HandleProgressRequest

Superclass: N/A	
Collaboration Diagram Reference:	
Class Description: This class will manage the responsibilities of the progress class	
---Contracts---	
Constract Name: Progress Server	
Contract Description: This contract manages the requests made to Progress.	
Responsibilities: Handle Progress Request ● Identify request ● Send request to appropriate contract: ○ Calculate Tasks Progress ○ Calculate Subtasks Progress ○ Calculate System Progress	**Collaborations**: Progress (2) Progress(3) Progress(4)
Method Signature:public static boolean progressServer(Request req){}	
Algorithmic Description: This contract will direct the request to the appropriate contract using the descriptor of the request.	
Pre-conditions: /@ requires req != null && [Provide Tasks Progress, Provide Subtasks Progress, Provide System Progress].contains(req) && (taskList.length > 0 || subtaskList.length > 0 || systemList.length > 0)	
Post-conditions: /@ensure \result(conditionMet) == true	

4.9 Class Description <Archive Server>

Class Name: Archive server	
Description: This class will manage the responsibilities of the archive class.	
Responsibilities: ● Handle all requests made to the archive class	**Collaborations**: Archive 16,17,18,19,20,21,22,23
Superclass: N/A	

Subclass: Archive

4.9.1 Contract <Archive Server>

Class Name: Archive Server	
Superclass: N/A	
Collaboration Diagram Reference: ?	
Class Description: This class will manage the responsibilities of the archive class	
---Contracts---	
Constract Name: Archive Server	
Contract Description: This contract manages the requests made to archive.	
Responsibilities: Handle Archive Request ● Identify request ● Send request to appropriate contract: ○ Restore Task ○ Restore Subtask ○ Restore System ○ Restore Finding ○ Archive Subtask ○ Archive Task ○ Archive System ○ Archive Finding	**Collaborations**: Archive(16) Archive(17) Archive(18) Archive(19) Archive(20) Archive(21) Archive(22) Archive(23)
Method Signature: public static boolean archiveServer(Request req){}	
Algorithmic Description: This contract will direct the request to the appropriate contract using the descriptor of the request.	
Pre-conditions: /@ requires req != null && [Restore Task , Restore Subtask, Restore System ,Restore Finding, Archive Subtask, Archive Task, Archive System, Archive Finding].contains(req) && taskList.length > 0 && () && (subtaskList.length > 0 || systemList.length > 0 || findingList.length > 0)	

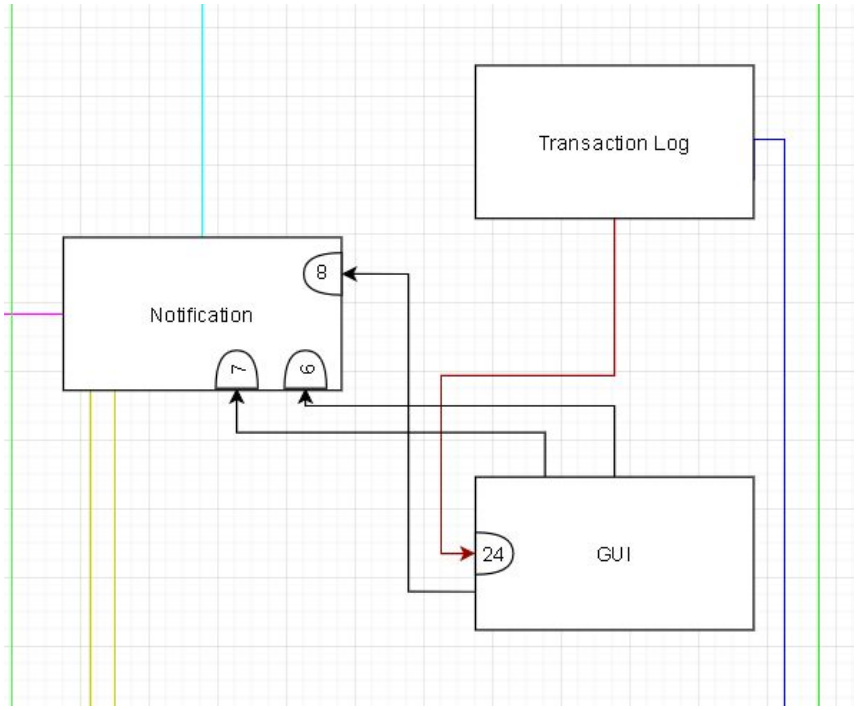
Post-conditions: /@ensure \result(conditionMet) == true

5. Detailed Description of Subsystem: Notification

This Subsystem only holds one class which is the Notification class itself. The notification subsystem will be responsible for delivering a message/alert to the analyst who is logged in. The message can differ and is currently set up to show up based on 3 different factors. The factors include: Task Due Date is upcoming, Task Progress, and Subtask Progress.

5.1 Notification Description (subsystem cards)

The name of the Subsystem is Notification. The purpose of this subsystem is to handle when a user, presumably an analyst, has an upcoming due date for a task/subtask, or progress on a task/subtask is behind the scheduled progress at that point in time. The subsystem is a subsystem of one and the only class that is present is the *Notification* class itself.



5.2 Class Description (CRC cards) <Notification>

Class Name: Notification	
Description: This class handles sending out pop up alerts to a user who has an upcoming due date or the progress of a task/subtask is behind schedule.	
Private Responsibilities: ● Create Notification ● Knows Notify Duration ● Knows Notify Frequency	**Collaborations**:
Comments: **Duration:** Duration refers to how long the notification will be active in FRIC. Once this time has expired the notification will disappear from the users interface. **Frequency:** Frequency refers to how often the notification will pop up on the Users page.	
Superclass: n/a	
Subclass: n/a	

5.2.1 Contract <Alert Analyst>

<table>
<tr><td colspan="2">Class Name: Notification</td></tr>
<tr><td colspan="2">Superclass: n/a</td></tr>
<tr><td colspan="2">Collaboration Diagram Reference: 6</td></tr>
<tr><td colspan="2">Class Description: This class handles sending out pop up alerts to a user who has an upcoming due date or the progress of a task/subtask is behind schedule.</td></tr>
<tr><td colspan="2">---Contracts---</td></tr>
<tr><td colspan="2">Contract Name: Alert Analyst</td></tr>
<tr><td colspan="2">Contract Description:This contract will obtain the Due Date for a given task and determine whether the due date is within range for the system to send a notification. If the Due Date is within range the notification is sent.</td></tr>
<tr><td colspan="2">---Protocols---</td></tr>
<tr><td>Responsibilities:
6. Alert Analyst
b. Check due date for Tasks in a System
c. If a Tasks due date is within next couple of days create Notification
d. Display notification to Analyst</td><td>Collaborations:

Task(11)</td></tr>
<tr><td colspan="2">Method Signature: public static boolean sendNotification_DueDate(int taskID, Date dueDate, int status){}</td></tr>
<tr><td colspan="2">Algorithm Description: The contract will make sure the task is existing and the due date is filled out. Next it will check the status and make sure that the task is not already marked as complete (which is only at 100 percent). Next it will check the number of days remaining before the current date and the due date. If this number is less than the specified date the notification is created and sent.</td></tr>
<tr><td colspan="2">Pre-conditions:
/* @requires taskID != null && dueDate != null && status != 100 && (currentDate - dueDate) < specified_date;
*/</td></tr>
<tr><td colspan="2">Post-conditions:
/* @ensures notification.message = dueDate, notification.isShown = True
*/</td></tr>
</table>

Comments: specified_date is the number amount of days before the analyst should be sent the notification. For example if set to 3 then the notification will check the task 3 days before the due date.

5.2.2 Contract <Alert Analyst of Progress of Task>

Class Name: Notification	
Superclass: n/a	
Collaboration Diagram Reference: 7	
Class Description: This class handles sending out pop up alerts to a user who has an upcoming due date or the progress of a task/subtask is behind schedule.	
---Contracts---	
Contract Name: Alert Analyst of Progress of Task	
Contract Description:This contract is responsible for creating the notification and displaying the notification when an analyst is behind schedule for the expected	
---Protocols---	
Responsibilities: 7. Alert Analyst of Progress of Task a. Display progress of assigned Task to Analyst	**Collaborations**: Progress(2)
Method Signature: public static boolean sendNotification_TaskProgress(int taskID, int progress){}	
Algorithm description: First make sure the task exists, then we check if the current progress on task is less than that expected progress at that point in time. If so, we create a notification notifying the analyst they are behind schedule	
Pre-conditions: /* **@requires** taskID != null && progress < expectedProgress */	
Post-conditions: /* **@ensures** notification.message = ((expectedProgress - progress)), notification.isShown = True	
Comments: expectedProgres - progress is how much the analyst is missing from where they should be (completion of task wise)	

5.2.3 Contract <Alert Analyst of Progress of Subtask>

<table>
<tr><td colspan="2">Class Name: Notification</td></tr>
<tr><td colspan="2">Superclass: n/a</td></tr>
<tr><td colspan="2">Collaboration Diagram Reference: 8</td></tr>
<tr><td colspan="2">Class Description: This class handles sending out pop up alerts to a user who has an upcoming due date or the progress of a task/subtask is behind schedule.</td></tr>
<tr><td colspan="2">---Contracts---</td></tr>
<tr><td colspan="2">Contract Name: Alert Analyst of Progress of Subtask</td></tr>
<tr><td colspan="2">Contract Description:This will provide certain information regarding the Event class.</td></tr>
<tr><td colspan="2">---Protocols---</td></tr>
<tr><td>Responsibilities:
7. Alert Analyst of Progress of Task
b. Display progress of assigned Task to Analyst</td><td>Collaborations:

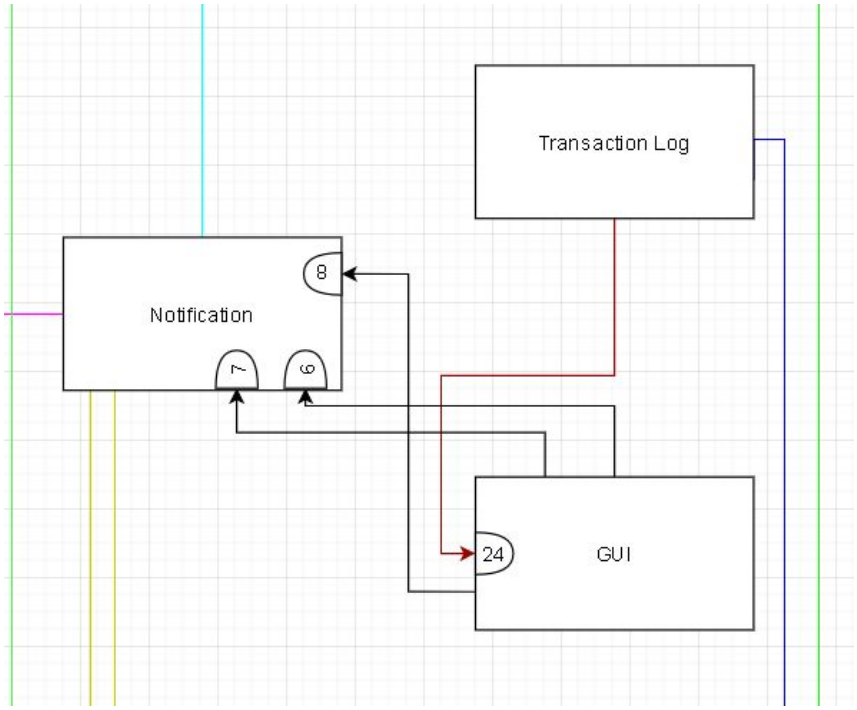
Progress(2)</td></tr>
<tr><td colspan="2">Method Signature: public static boolean sendNotification_SubtaskProgress(int subtaskID, int progress){}</td></tr>
<tr><td colspan="2">Algorithm Description: First makes sure the Subtask exists, then we check if the current progress on the Subtask is less than that expected progress at that point in time. If so, we create a notification notifying the analyst they are behind schedule</td></tr>
<tr><td colspan="2">Pre-conditions:
/* @requires subtaskID != null && progress < expectedProgress
*/</td></tr>
<tr><td colspan="2">Post-conditions:
/* @ensures notification.message = ((expectedProgress - progress)), notification.isShown = True</td></tr>
<tr><td colspan="2">Comments: expectedProgres - progress is how much the analyst is missing from where they should be (completion of Subtask wise)</td></tr>
</table>

6. Detailed Description of Subsystem: Transaction Log

The transaction log is the behind the scenes list of actions that were done by the user. This class logs every action and stores it for future reference. Logging is directly linked with the GUI subsystem in the sense that every action done on the GUI is logged. The log uses the same format provided by the SRS and logs can be seen on user to user basis or entire event basis.

6.1 Transaction Log Description (subsystem cards)

The name of the subsystem is Transaction Log. The transaction log is responsible for keeping a record of anything that is done on FRIC, this includes adding/editing/archiving an entity in FRIC. The only class that is present in this subsystem is the Transaction Log class.



6.2 Class Description (CRC cards) <Transaction Log>

Class Name: Transaction Log
Description:Handles the logging of analysts.

Private Responsibilities: ● Log Action ○ Obtain UI Action ○ Obtain Current Date ○ Obtain Current Analyst	**Collaborations**: GUI(24) Analyst(14)
Superclass: n/a	
Subclass: n/a	

6.2.1 Contract <>

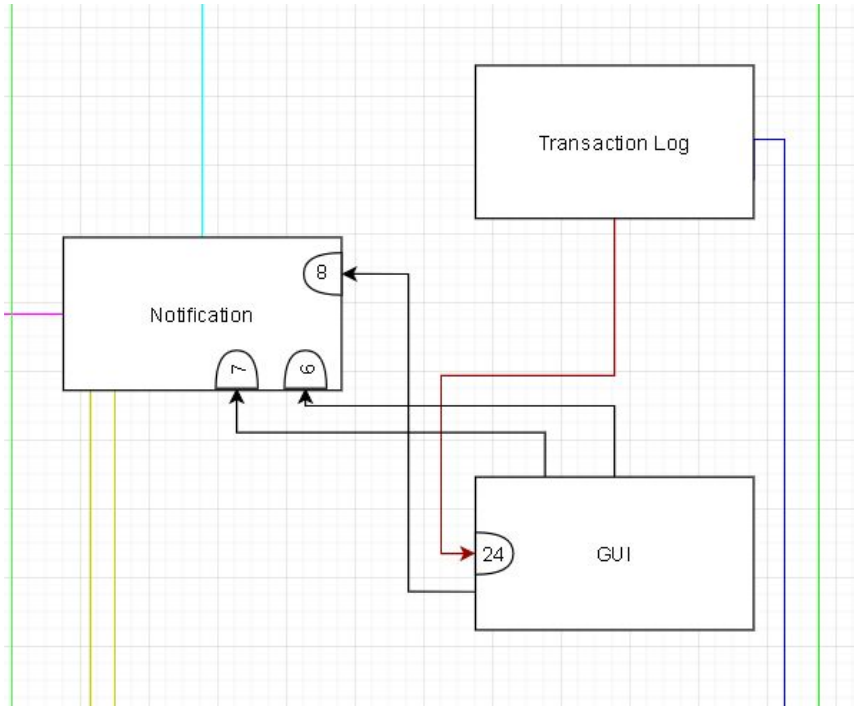
This class does not currently offer any contracts. The only things that this class is logging the actions that it receives.

7. Detailed Description of Subsystem: GUI

GUI entails the User Interface and the user actions associated with the interface. The GUI class will be responsible for providing the action done by the user, this could be creating/editing/archiving an entity in FRIC.

7.1 GUI Description (subsystem cards)

The name of the Subsystem is GUI. The main purpose of the GUI is simply to provide the User Interface however we do use the GUI to provide the action to other classes, such as the Transaction Log. GUI is responsible for showing notifications that are pushed through to the user that is logged in as well. The only class inside of the GUI subsystem is GUI itself.



7.2 Class Description (CRC cards) <GUI>

Class Name: GUI	
Description: The GUI class handles the logged in users interactions.	
Private Responsibilities: ● Provide UI ○ Display UI ● Provide Notification ○ Provide Due Date Notification ○ Provide Task Progress Notification ○ Provide Subtask Progress notification	**Collaborations**: Notification(6) Notification(7) Notification(8)
Superclass: :n/a	
Subclass:n/a	

7.2.1 Contract <Provide UI Action>

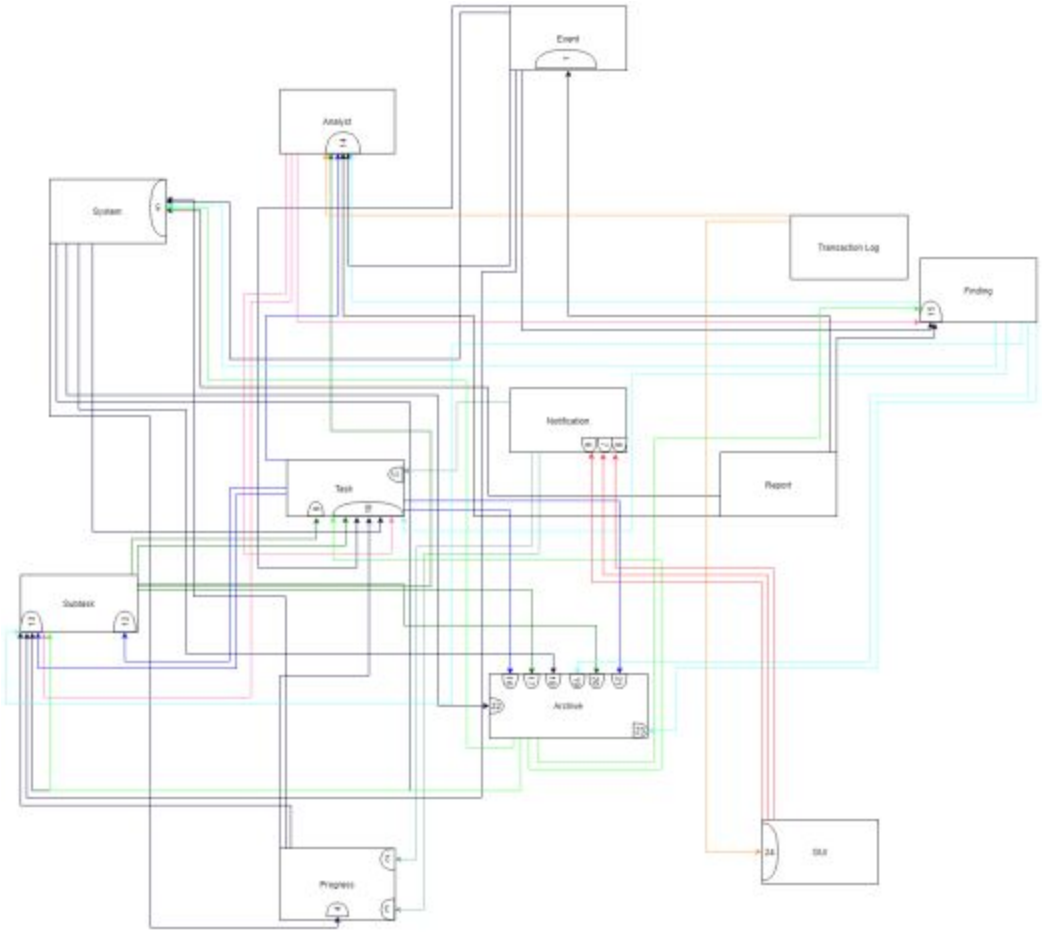
Class Name: GUI	
Superclass: N/A	
Collaboration Diagram Reference: 24	
Class Description: The GUI class handles the logged in users interactions.	
---Contracts---	
Contract Name: Provide UI Action	
Contract Description: This contract will take not of the action performed anytime a button is pressed inside of FRIC. This includes editing/adding/archiving any entity in FRIC along with the user who is responsible for this action.	
---Protocols---	
Responsibilities: 24. Provide UI Action a. Provide action details	**Collaborations**:
Method Signature: public string Event provideAction(){}	
Algorithm Description: The algorithm will record any button is pressed inside of FRIC and associate that action with a readable version of what the user did.	
Pre-conditions: //@ requires button_clicked = True	
Post-conditions: //@ensures \result("/*action occurred */") != null	
Comments: Result is going to be the string that comes out of which action occurred. If a task was created the action would be "String created" for example.	

Appendix

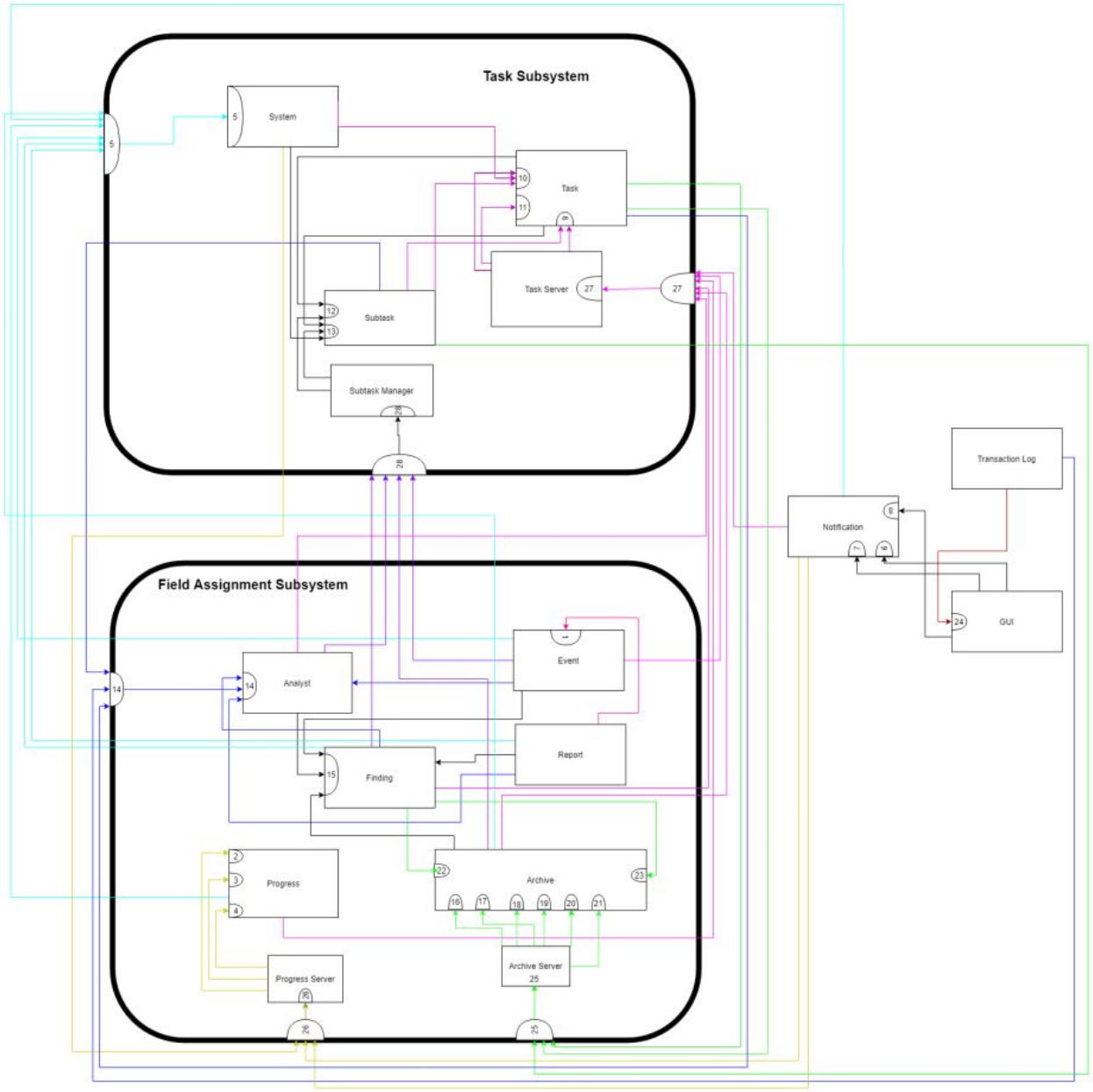
Please note: The collaboration graph requires a big image size to see the detailed lines.

SEE BELOW

Collaboration Graph:
https://minersutep-my.sharepoint.com/:u:/g/personal/ialeos_miners_utep_edu/Ebh4aURFbUtEkonmSeK4tFcBhe912FBLh7YY1ytG1mq6eg?e=5f13cR



SubSystem Collaboration Graph:



</3