Collective Knowledge System (PICK)

Software Design Document

Version 1.5

3/31/2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.5 |
| Indicator of Last Page in Document: | $$ |
| Date of Last Review: | 03/31/2020 |
| Date of Next Review: | 04/07/2020 |
| Target Date for Next Update: | 04/14/2020 |

**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. Gates

Dr. Salamah

Dr. Roach

Elsa Tai Ramirez

Peter Hanson

Customer: Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members: Charlie Juarez

Miriam Juarez

Angelica Marquez

Andrew Munoz

Aaron Rodriguez

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 3/10/2020 | Andrew Munoz | Creation of Document |
| 1.1 | 3/24/2020 | Aaron Rodriguez | Section 1.2, Section 1.3 (partial), Section 1.4 (partial), Section 2.2 (partial), Section 4, Section 11 |
| 1.2 | 3/25/2020 | Charlie Juarez | Section 2(description), 2.1, 2.2 (Configuration, startup), Section 5, Section 6 |
| 1.3 | 3/26/2020 | Angelica Marquez | Section 1.1 |
| 1.4 | 3/29/2020 | Andrew Munoz | Section 1.4 update, Section 1.5, Section 2.1 update, Section 2.2 (Regex, Vec. Node. Mod.), Section 7, Section 8 |
| 1.5 | 3/31/2020 | Angelica Marquez | Section 9 (9.1-9.4), Section 10 (10.1-10.2) |
| 1.5 | 3/31/2020 | All | Section 2.1 diagram update |

Table of Contents

[**DOCUMENT CONTROL II**](#_heading=h.30j0zll)

[Approval ii](#_heading=h.2nusc19)

[Document Change Control ii](#_heading=h.1302m92)

[Distribution List ii](#_heading=h.3mzq4wv)

[Change Summary ii](#_heading=h.2250f4o)

[**1.**](#_heading=h.haapch) **INTRODUCTION 1**

[1.1.](#_heading=h.319y80a) Purpose and Intended Audience 1

[1.2.](#_heading=h.1gf8i83) Scope of Product 1

[1.3.](#_heading=h.40ew0vw) References 1

[1.4.](#_heading=h.17dp8vu) Definitions, Acronyms, and Abbreviations 1

[*1.4.1.*](#_heading=h.2fk6b3p) *Definitions 1*

[*1.4.2.*](#_heading=h.upglbi) *Acronyms 1*

[*1.4.3.*](#_heading=h.3ep43zb) *Abbreviations 1*

[1.5.](#_heading=h.1tuee74) Overview 2

[**2.**](#_heading=h.4du1wux) **DECOMPOSITION DESCRIPTION 3**

[2.1.](#_heading=h.z337ya) System Collaboration Diagram 3

[2.2.](#_heading=h.2szc72q) Subsystem and Component Descriptions 3

[2.3.](#_heading=h.1y810tw) Dependencies 4

[**3.**](#_heading=h.184mhaj) **DETAILED DESCRIPTION OF COMPONENT VALIDATION 5**

[3.1.](#_heading=h.3s49zyc) Component Description 5

[3.2.](#_heading=h.3whwml4) Class Description Script Handler 5

[*3.2.1.*](#_heading=h.279ka65) *Contract Run Script 5*

[3.3.](#_heading=h.meukdy) Class Description Log File 5

[*3.3.1.*](#_heading=h.36ei31r) *Contract Log Ingestion 5*

[3.4.](#_heading=h.1pxezwc) Class Description ValidationIngestionWindow 5

[*3.4.1.*](#_heading=h.49x2ik5) *Contract Ingest Data Selection 5*

[3.5.](#_heading=h.2p2csry) Class Description Enforcement Action Report 6

[*3.5.1.*](#_heading=h.147n2zr) *Contract Create Error Report 6*

[3.6.](#_heading=h.3o7alnk) Class Description DirDialog 6

[*3.6.1.*](#_heading=h.23ckvvd) *Contract Directory Chooser 6*

[**4.**](#_heading=h.ihv636) **DETAILED DESCRIPTION OF COMPONENT INGESTION 7**

[4.1.](#_heading=h.32hioqz) Component Description 7

[4.2.](#_heading=h.1hmsyys) Class Description ValidationIngestionWindow 7

[*4.2.1.*](#_heading=h.41mghml) *Contract Ingest Data Selection 7*

[4.3.](#_heading=h.2grqrue) Class Description Log Entry 7

[*4.3.1.*](#_heading=h.vx1227) *Contract Save Log Entry 7*

[4.4.](#_heading=h.3fwokq0) Class Description MongoDB 7

[*4.4.1.*](#_heading=h.1v1yuxt) *Contract Create Log Entry DB Table 7*

[**5.**](#_heading=h.4f1mdlm) **DETAILED DESCRIPTION OF COMPONENT CONFIGURATION 8**

[5.1.](#_heading=h.2u6wntf) Component Description 8

[5.2.](#_heading=h.19c6y18) Class Description ProjectConfigWindow 8

[*5.2.1.*](#_heading=h.3tbugp1) *Contract Display Directory 8*

[*5.2.2.*](#_heading=h.28h4qwu) *Contract Display Start Date 8*

[*5.2.3.*](#_heading=h.nmf14n) *Contract Display End Date 8*

[5.3.](#_heading=h.37m2jsg) Class Description CalendarDialog 8

[*5.3.1.*](#_heading=h.1mrcu09) *Contract Select Date 8*

[5.4.](#_heading=h.46r0co2) Class Description DirDialog 9

[*5.4.1.*](#_heading=h.2lwamvv) *Contract Select Directory 9*

[**6.**](#_heading=h.111kx3o) **DETAILED DESCRIPTION OF COMPONENT STARTUP 10**

[6.1.](#_heading=h.3l18frh) Component Description 10

[6.2.](#_heading=h.206ipza) Class Description Main 10

[**7.**](#_heading=h.4k668n3) **DETAILED DESCRIPTION OF COMPONENT REGEX 11**

[7.1.](#_heading=h.2zbgiuw) Component Description 11

[7.2.](#_heading=h.1egqt2p) Class Description Filter 11

[*7.2.1.*](#_heading=h.3ygebqi) *Contract Process Filter Request 11*

[7.3.](#_heading=h.2dlolyb) Class Description Log Entry 11

[*7.3.1.*](#_heading=h.sqyw64) *Contract Save Log Entry 11*

[7.4.](#_heading=h.3cqmetx) Class Description NavigatorWindow 11

[*7.4.1.*](#_heading=h.1rvwp1q) *Contract Display Results 11*

[**8.**](#_heading=h.4bvk7pj) **DETAILED DESCRIPTION OF COMPONENT VECTOR NODE MODIFICATION 12**

[8.1.](#_heading=h.2r0uhxc) Component Description 12

[8.2.](#_heading=h.1664s55) Class Description Vector 12

[*8.2.1.*](#_heading=h.3q5sasy) *Contract Create Vector 12*

[8.3.](#_heading=h.25b2l0r) Class Description Node 12

[*8.3.1.*](#_heading=h.kgcv8k) *Contract Create Node 12*

[8.4.](#_heading=h.34g0dwd) Class Description Relationship 14

[*8.4.1.*](#_heading=h.1jlao46) *Contract Create Relationship 14*

[8.5.](#_heading=h.43ky6rz) Class Description VectorTableWindow 14

[*8.5.1.*](#_heading=h.2iq8gzs) *Contract Display Updated Table View 14*

[**9.**](#_heading=h.xvir7l) **DETAILED DESCRIPTION OF COMPONENT GRAPH CONFIGURATION 15**

[9.1.](#_heading=h.3hv69ve) Component Description 15

[9.2.](#_heading=h.1x0gk37) Class Description Graph 15

[*9.2.1.*](#_heading=h.1x0gk37) *Contract Create Graph 15*

[9.3.](#_heading=h.4h042r0) Class Description GraphWindow 15

[*9.3.1.*](#_heading=h.2w5ecyt) *Contract Graph Window 15*

[9.4.](#_heading=h.1baon6m) Class Description Node 15

[*9.4.1.*](#_heading=h.3vac5uf) *Contract Node 15*

[**10.**](#_heading=h.2afmg28) **DETAILED DESCRIPTION OF COMPONENT GRAPH EXPORT 16**

[10.1.](#_heading=h.pkwqa1) Component Description 16

[10.2.](#_heading=h.39kk8xu) Class Description Graph 16

[*10.2.1.*](#_heading=h.39kk8xu) *Contract Graph Export 16*

[**11.**](#_heading=h.1opuj5n) **DATABASE 17**

[11.1.](#_heading=h.48pi1tg) Database Schema 17

# Introduction

## Purpose and Intended Audience

The purpose of creating the software design document is to report how the design is based and meets the client requirements. A software design document also helps assess the quality of the software during creation. The intended audience are the software engineers that will be implementing the design and the analysts that will be using the software, for future verification and maintenance.

## Scope of Product

The PMR Insight Collective Knowledge (PICK) System will facilitate the current node correlation process for analysts to complete their cyber security attack graphs. Currently, the analysis process takes months to complete and the system is needed to shorten the completion time to a more reasonable length. The system will save the analysts time with the automatization of log file processing along with filtering/searching functionality. The system’s attack graph creator component will further save analysts time and provide a flexible structure for analysts to work off of. The purpose of PICK is to retrieve and layout log information provided by the adversarial assessment. The software will ingest the log files to the analyst’s configuration and allow the analyst to produce attack graphs containing the events that transpired to provide a report for LSH.

## References

[1] S. Roach et al, Software Requirements Specification, Lethality, Survivability, and HSI Directorate (LSH),

2019.

## Definitions, Acronyms, and Abbreviations

### Definitions

|  |  |
| --- | --- |
| Log Entry | Splunk takes the validated log files and convert them into normalized data. The normalized data are called log entries. Users of the system can filter and edit log entries. |
| Timestamp | Denotes time in hours:minutes, date in month:date:year, and section in am/pm. |

### Acronyms

|  |  |
| --- | --- |
| PICK | PMR Insight Collective Knowledge |
| SDD | Software Design Document |

### Abbreviations

|  |  |
| --- | --- |
| DB | Database |
|  |  |
|  |  |

## Overview

The Software Design Document (SDD) is divided into 4 major sections: Introduction (Section 1), Decomposition Description (Section 2), Detailed Description of Components (Sections 3-10), and the Database (Section 11).

The Introduction section includes four subsections. Section 1.1 provides the purpose and intended audience of the SDD. Section 1.2 provides the scope of the product. Section 1.3 lists the references used in this SDD. Section 1.4 provides the definitions, acronyms and abbreviations used. Section 1.5 provides an overview of the sections within this SDD.

The Decomposition Description section includes three subsections. Section 2.1 provides the Wirffs-Brock Collaboration Diagram. Section 2.2 provides a description of the subsystems and components in the diagram. Section 2.3 describes how the component dependencies will impact development.

The Detailed Description of Components is made of Sections 3-10. Each Detailed Description of Component section includes the component description, and its class descriptions alongside its contracts. Section 3-10 detail the following components: Validation, Ingestion, Configuration, Startup, Regex, Vector Node Modification, Graph Configuration, and Graph Export.

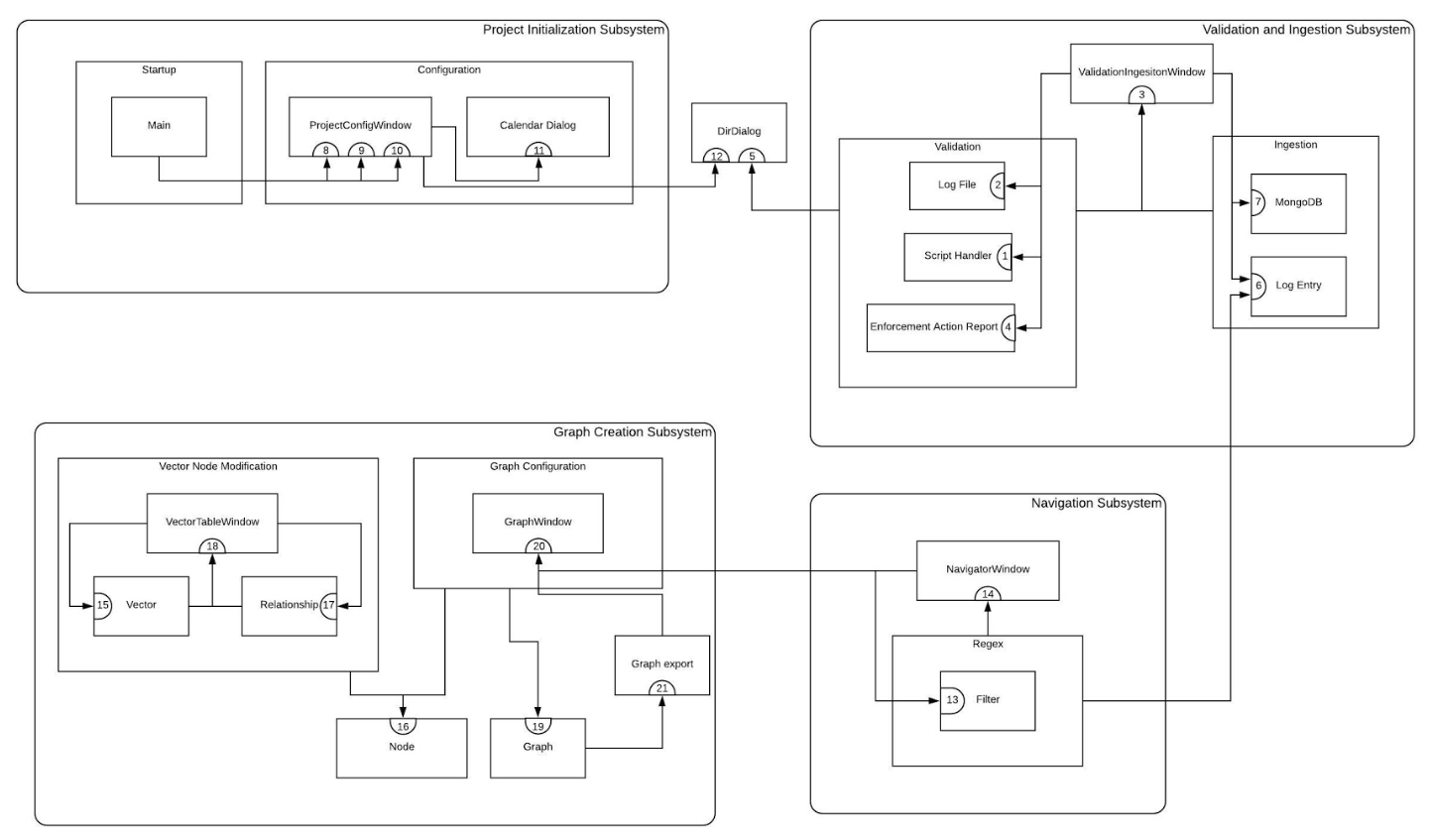
Lastly, the Database section contains the database schema which provides the description of the data layout within the database.

# Decomposition Description

The system is composed of four major subsystems that work together to go through the initialization, validation/ingestion, navigation and graph creation of the system. The first subsystem presented when starting up the software will be the initialization subsystem which takes care of creating a new project or opening an existing project. In which the creation of a project will take care of the details in configuring a new project otherwise if an existing project is chosen then it will be taken by the navigation subsystem. After the configuration is done when creating a new project it will be taken care of by the validation and ingestion subsystem which will take care of validating files after which it will be ingested into MongoDB. Once the ingestion process is complete, the information will be taken care of by the navigation subsystem which takes care of the organization of the ingested log files as well as any details that may be appended to the log entries. The information in the navigation subsystem will also be shared by the graph creation subsystem which takes care of visually representing the log entries. In this graph creation subsystem, aside from representing information visually it can also take care of any additional information that is missing or needs adjustment. Once the graph has been created, the graph creation subsystem will be able to export the information into a file whether it be a picture or csv file.

## System Collaboration Diagram

Below is the Wriffs-Brock Collaboration Diagram of the PICK system. It shows the collaborations between the major subsystems along with their components and classes.



## Subsystem and Component Descriptions

**Validation and Ingestion Subsystem:**

**Validation Component -** Within the Validation and Ingestion subsystem is the Validation component which is in charge of validating all the log file data to ensure it is ready to continue to the ingestion process. The contracts supported by the Validation component are contract numbers 1[run script], 2[log ingestion], 3[ingest data selection], 4[create error report], and 5[directory chooser]. The detailed description of the Validation component is further discussed in section 3.

**Ingestion Component -** The Ingestion component is also within the Validation and Ingestion subsystem. This component ensures that cleansed log files are stored within the PICK system’s MongoDB database. A log file is considered fully ingested once it has been cleansed, validated, and turned into multiple log entries into the database. The contracts supported by the Ingestion component are contract numbers 6[Save Log Entry], 7[Create Log Entry DB Table], and 3[ingest data selection] The detailed description of the Ingestion component is further discussed in section 3.

**Project Initialization Subsystem:**

**Configuration Component -** Within the configuration there are three classes ProjectConfigWindow, Calendar Dialog and Event Configuration. The purpose of the Event Configuration is to set the general description of the project that is being created. The purpose of the ProjectConfigWindow is to get the directories from which the raw file content will be extracted from. The purpose of the Calendar Dialog is to set the time frame from which the raw files will be extracted from.

**Startup Component -** The purpose of the startup component is to delegate the direction the software will go based on the users actions; which are going to the project configuration or the navigation subsystem

**Navigation Subsystem:**

**Regex Component –** Within the Navigation subsystem is the Regex component which is in charge of the information retrieval process within the log entry content in the system. The contracts supported by the Regex component are contract numbers 13[process filter request], 6[save log entry], and 14[display results]. The detailed description of the Regex component is further discussed in section 7.

**Graph Creation Subsystem:**

**Graph Configuration -** In the Graph Creation System, the contract supported by the graph configuration is 20 [graphWindow]. The purpose of graphWindow is to display the builded graph, edit it, and export it.

**Graph Export -** In the Graph creation system, graph export has the purpose of creating a JPG image of the graph. There is also the option of having the graph information exported into a csv file.

**Vector Node Modification Component –** Within the Graph Creation subsystem is the Vector Node Modification which handles the modification process of the vectors, nodes, and relationships. The contracts supported by the Vector Node Modification component are contract numbers 15[create vector], 16[create node], 17[create relationship], and 18[display updated table view]. The detailed description of the Vector Node Modification component is further discussed in section 8.

## Dependencies

As the components rely heavily on information flowing correctly without errors, there will be points when there needs to be checks on information to ensure the information does not interfere in further components causing cascading errors. Therefore such checks will alter how the information is handled during the development of the software causing the alteration of components to be common during development.

# Detailed Description of Component Validation

## Component Description

The validation component is in charge of validating all the log file data to ensure it is ready to continue to the ingestion process. It runs a cleansing script on the log files and checks if they meet the requirements specified by the user. Any encountered errors are shown to the user for the user to address. The validation component utilizes the Script handler, Log file, ValidationIngestionWindow, Enforcement Action Report, and DirDialog.

## Class Description Script Handler

The Script Handler class represents the cleansing script used on the log files during the validation process. It has the responsibility of knowing the cleansing script directory path to be able to run it.

### Contract Run Script

Contract number: 1

Contract name: run script

Contract description: runs the cleansing script on the given log file.

Protocol 1:

Signature: runScript(filePath)

input filePath- log file directory path to run script on, type string.

output - none

Pre-Cond: the given filePath is the correct directory path and exists.

Post-Cond: the cleansing script is executed on given log file.

## Class Description Log File

The Log File represents a file of any type, for example, csv, jpeg, pdf, etc., that have been ingested by the system. It has the responsibility of knowing its own file type, knows ingestion errors, and know all its own content.

### Contract Log Ingestion

Contract number: 2

Contract name: Log Ingestion

Contract description: Given a log file, the class will return the given log type, the log content and if there are any ingestion errors.

Protocol 1:

Signature: logIngest(anyTypeLog)

input anyTypeLog - Log file of any type.

output fileType - log file type (i.e. csv, jpeg, pdf, etc…)

output logContent - data information from the log file.

output ingestionErrors - information of an error if it exists.

PreCond: Log file has to exist.

Post-Cond: Log file stays unchanged.

## Class Description ValidationIngestionWindow

The ValidationIngestionWindow class is the visual representation of the handler of the validation and ingestion process of the log file data. The superclass will be ProjectConfigWindow, and will handle the transition to NavigationWindow.

### Contract Ingest Data Selection

Contract number: 3

Name: Ingest Data Selection

Contract description:

Protocol 1:

Signature: ingestDataSelection(validLogFile)

input validLogFile- files that are validated.

output: none

Pre-Condn: the given LogFile has to be previously validated.

Post-Condition: LogFiles are now LogEntries.

## Class Description Enforcement Action Report

The Enforcement Action Report class represents the report that will be generated with each invalidated log file. This class will include the error(s) associated with the log file that makes it invalidated, and thus convey to the analysts the reasons as to why a log file is invalidated.

### Contract Create Error Report

Contract number: 4

Contract name: Create error report

Contract description: Creates an enforcement action report at the detection of an error in a log file.

Protocol: 1

Signature: createErrorReport(logFileInfo, errorType)

input logFileInfo: Log file information to display in the enforcement action report, so that the analyst can know which log file is invalidated and must be changed or approved as is.

input errorType: The error type(s) associated with the invalidated log file. The error type(s) enables the analyst to see which validation rules the associated log file broke.

output: Enforcement Action Report

Pre-Cond: Log file must not be null and error type must be a valid error for generating an enforcement action report.

Post-Cond: Log file must remain unchanged.

## Class Description DirDialog

The DirDialog class will handle the directory choosing function whenever the software requires it. The superclass will be ProjectConfigWindow as that is where all the information will be returning to once a directory has been chosen.

### Contract Directory Chooser

Contract number: 5

Contract name: Directory Chooser

Contract description: Display a file chooser window that will select the directory for log data.

Protocol 1:

Signature: dir\_dialog(self, dialogtitle)

self-identify method as an instance of the class DirDialog

dialogtitle- name for the dialog that will be created.

output - none

Pre-Cond: a dialogtitle has been given

Post-Cond: the instance saves the file directory path

# Detailed Description of Component Ingestion

## Component Description

The Ingestion component will ensure that log files are stored into the PICK system database as log entries for referencing and editing by the analyst. Log files are not fully ingested if they have errors in the validation component, and are only converted into multiple log entries through ingestion when they have been cleansed and validated. The Ingestion component utilizes the ValidationIngestioWindow, Log Entry, and MongoDB.

## Class Description ValidationIngestionWindow

The ValidationIngestionWindow class is the visual representation of the handler of the validation and ingestion process of the log file data. The superclass will be ProjectConfigWindow, and will handle the transition to NavigationWindow.

### Contract Ingest Data Selection

Contract number: 3

Name: Ingest Data Selection

Contract description: Refer to Section 3.4.1.

## Class Description Log Entry

The LogEntry class represents each part a log file will be broken down into after being ingested into the PICK system. Log entries allow the analyst to see all log file data in a single type of format regardless of the log file filetype (i.e., PDF, PNG). Every Log Entry is the converted format of a log file in the PICK system. The Log Entry class also represents a piece of log file information that can be eventually stored in a Vector and Graph image output.

### Contract Save Log Entry

Contract number: 6

Name: Save Log Entry

Contract description: Saves the log entry into the MongoDB

Protocol 1:

Signature: store\_logentry(logentry)

logentry: the Log Entry to store in the MongoDB

Output: None

Pre-Condition: Log Entry cannot be null and the log entry table in MongoDB cannot be null

Post -Condition: Log Entry is stored in the MongoDB table log entry

## Class Description MongoDB

### Contract Create Log Entry DB Table

Contract number: 7

Name: Create Log Entry DB Table

Contract description: Creates a table in the MongoDB called log entry where all Log Entry instances will be stored.

Protocol 1:

Signature: create\_table\_logentry()

Output: a boolean that is True if the DB table creation was successful and false if it was not successful

Pre-Condition: The MongoDB must not be null

Post-Condition: The table log entry in MongoDB is not null

# Detailed Description of Component Configuration

## Component Description

Within the configuration there are three components ProjectConfigWindow, Calendar Dialog and Event Configuration. The purpose of the Event Configuration is to set the general description of the project that is being created. The purpose of the ProjectConfigWindow is to get the directories from which the raw file content will be extracted from. The purpose of the Calendar Dialog is to set the time frame from which the raw files will be extracted from.

## Class Description ProjectConfigWindow

This class is responsible for visually representing the different entries for providing input for the event title and event description. It is also responsible for displaying the directories from which the raw log files will be extracted from. Afterwards it is also responsible for displaying the time range from which the selected raw log files will be retrieved. The superclass of ProjectConfigWindow is main.

### Contract Display Directory

Contract number: 8

Contract description: Display the directory chosen

Protocol 1:

Signature: on\_pathdir\_button\_clicked(self)

Output: None

Pre-Condition: A directory has been chosen by DirDialog.

Post -Condition: Display the directory that was chosen.

### Contract Display Start Date

Contract number: 9

Contract description: Display start date

Protocol 1:

Signature: on\_startdate\_button\_clicked(self)

Output: None

Pre-Condition: A date has been chosen by CalendarDialog.

Post -Condition: Display the date that has been chosen.

### Contract Display End Date

Contract number: 10

Contract description: Display end date

Protocol 1:

Signature: on\_enddate\_button\_clicked(self)

Output: None

Pre-Condition: A date has been chosen by CalendarDialog.

Post -Condition: Display the date that has been chosen.

## Class Description CalendarDialog

This class is responsible for displaying a calendar from which the user is able to select a date.

### Contract Select Date

Contract number: 11

Contract description: Select date

Protocol 1:

Signature: on\_confirm\_button\_clicked(self)

Output: None

Pre-Condition: A date has been selected in CalendarDialog.

Post -Condition: Set the date picked for its caller to retrieve from.

## Class Description DirDialog

This class is responsible for displaying a file chooser from which the user is able to select a directory.

### Contract Select Directory

Contract number: 12

Contract description: Select directory

Protocol 1:

Signature: dir\_dialog(self, dialogtitle)

Output: filename

Pre-Condition: A date has been selected in DirDialog.

Post -Condition: Set the directory picked for its caller to retrieve from.

# Detailed Description of Component Startup

## Component Description

The purpose of the startup component is to delegate the direction the software will go based on the users actions; which are going to the project configuration or the navigation subsystem.

## Class Description Main

The purpose of this class is to display the options of which will determine which window to display next. This class offers the user the ability to choose from creating a new project and opening an existing project. Of which the new project is handled by the Configuration component and the Validation and Ingestion Subsystem. While opening an existing project is handled by the Navigation Subsystem.

# Detailed Description of Component Regex

## Component Description

The Regex component is in charge of the information retrieval process within the log entry content in the system. It processes the given keywords, filters, and/or search expressions the user gives and displays the search results for the user to see. The Regex component utilizes the Filter, Log Entry, and NavigatorWindow.

## Class Description Filter

The Filter class represents the chosen filtering requirements the user has requested to be processed. It has the responsibility of knowing and processing the filtering request the user has chosen.

### Contract Process Filter Request

Contract number: 13

Contract Name: Process Filter Request

Contract Description: Processes the user’s filter request on the log entries.

Protocol 1:

Signature: ProcessFilter(filterRequest)

filterRequest: list of the user’s requested filtering requirements.

Output: list of all log entries which pass the filtering requirements.

Pre-Condition: At lease one log entry exists in the system.

Post-Condition: The current full list of log entries is filtered to the user’s request.

## Class Description Log Entry

The Log Entry class represents each part a log file will be broken down into after being ingested into the PICK system. Log entries allow the analyst to see all log file data in a single type of format regardless of the log file filetype (i.e., PDF, PNG). Every Log Entry is the converted format of a log file in the PICK system. The Log Entry class also represents a piece of log file information that can be eventually stored in a Vector and Graph image output.

### Contract Save Log Entry

Contract number: 6

Name: Contract Save Log Entry

Contract Description: Refer to Section 4.3.1.

## Class Description NavigatorWindow

The NavigatorWindow class is the visual representation of the handler of the log entry navigation/filtering process. It is responsible for displaying all the log entry content within the system and provide filtering options to the user for navigation of the log entries. The superclass will be ValidationIngestionWindow, and will handle the transition to VectorTableWindow.

### Contract Display Results

Contract number: 14

Contract Name: Display Results

Contract Description: Displays the user’s filter/search request results.

Protocol 1:

Signature: ResultsButtonClicked(self)

Output: None

Pre-Condition: At least one filtering request has been chosen by the user.

Post-Condition: Display the resulting list of log entries from the filtering request.

# Detailed Description of Component Vector Node Modification

## Component Description

The Vector Node Modification component is in charge of the modification process of the vectors, nodes, and relationships. This includes making changes in their attributes to allow the user to rework the vectors, nodes or relationships. The Vector Node Modification component utilizes the Vector, Node, Relationship, and VectorTableWindow.

## Class Description Vector

The Vector class represents a sequence of actions built up by nodes that detail the happenings of an event. The vector is responsible for knowing its vector name, nodes, and vector description.

### Contract Create Vector

Contract number: 15

Contract Name: Create Vector

Contract Description: Creates a vector.

Protocol 1:

Signature: createVector(Nodes, Name ,Description)

Nodes: list of the vector’s nodes.

Name: String of the vector name.

Description: String of vector’s description.

Output: none

Pre-Condition: Vector name is unique, and vector is comprised of at least one significant log entry.

Post-Condition: Vector with given attributes is created.

Protocol 2:

Signature: createVector(Nodes, Name)

Nodes: list of the vector’s nodes.

Name: String of the vector name.

Output: none

Pre-Condition: Vector name is unique, and vector is comprised of at least one significant log entry.

Post-Condition: Vector with given attributes is created and the vector description is an empty string.

## Class Description Node

The Node class represents a visual representation of a significant event that was marked for a vector. The node is responsible for knowing its node ID, name, timestamp, description, log entry reference, log creator, event type, icon type, source, node visibility, and also know each of its attribute visibility Boolean flags to be shown on the graph.

### Contract Create Node

Contract number: 16

Contract Name: Create Node

Contract Description: Creates Node

Protocol 1:

Signature: createNode(id, name, time, description, ref, creator, event, icon, source, visibl, visFlags)

ID: String of the Nodes’s unique ID

Name: String of the node name.

Time: Timestamp of when the log entry took place.

Description: String of nodes description

Ref: The node’s log entry reference

Creator: String of node’s creator type showing the team who created the log (White, blue, or red).

Event: String of node’s event type showing the team of who executed the activity (White, blue, or red)

Icon: String of node’s icon type showing path to an image used to reflect the nature of the activity.

Source: String of the node’s source.

Visibl: Boolean of the node’s visibility, stating if the node is visible on the graph.

visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source

Output: none

Pre-Condition: The id is unique and the given creator and event are either “white”, ”red”, or ”blue”. Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.

Post-Condition: The node is created with its given attributes.

Protocol 2:

Signature: createNode(id, name, time, description, creator, event, icon, visibl, visFlags)

ID: String of the Nodes’s unique ID

Name: String of the node name.

Time: Timestamp of when the log entry took place.

Description: String of nodes description

Creator: String of node’s creator type showing the team who created the log (White, blue, or red).

Event: String of node’s event type showing the team of who executed the activity (White, blue, or red)

Icon: String of node’s icon type showing path to an image used to reflect the nature of the activity.

Visibl: Boolean of the node’s visibility, stating if the node is visible on the graph.

visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source

Output: none

Pre-Condition: The id is unique and the given creator and event are either “white”, ”red”, or ”blue”. Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.

Post-Condition: The node is created with its given attributes and both ref and source are left empty.

Protocol 3:

Signature: createNode(id, name, time, description , creator, event, icon, source, visibl, visFlags)

ID: String of the Nodes’s unique ID

Name: String of the node name.

Time: Timestamp of when the log entry took place.

Description: String of nodes description

Creator: String of node’s creator type showing the team who created the log (White, blue, or red).

Event: String of node’s event type showing the team of who executed the activity (White, blue, or red)

Icon: String of node’s icon type showing path to an image used to reflect the nature of the activity.

Source: String of the node’s source.

Visibl: Boolean of the node’s visibility, stating if the node is visible on the graph.

visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source

Output: none

Pre-Condition: The id is unique and the given creator and event are either “white”, ”red”, or ”blue”. Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.

Post-Condition: The node is created with its given attributes and ref is left empty.

Protocol 4:

Signature: createNode(id, name, time, description, ref, creator, event, icon, visibl, visFlags)

ID: String of the Nodes’s unique ID

Name: String of the node name.

Time: Timestamp of when the log entry took place.

Description: String of nodes description

Ref: The node’s log entry reference

Creator: String of node’s creator type showing the team who created the log (White, blue, or red).

Event: String of node’s event type showing the team of who executed the activity (White, blue, or red)

Icon: String of node’s icon type showing path to an image used to reflect the nature of the activity.

Visibl: Boolean of the node’s visibility, stating if the node is visible on the graph.

visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source

Output: none

Pre-Condition: The id is unique and the given creator and event are either “white”, ”red”, or ”blue”. Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.

Post-Condition: The node is created with its given attributes and source is left empty.

## Class Description Relationship

The Relationship class represents the visual representation of a connection between nodes in a vector which is shown on the graph. The relationship is responsible for knowing its relationship ID, parent ID, child ID, and its label.

### Contract Create Relationship

Contract number: 17

Contract Name: Create Relationship

Contract Description: Creates a Relationship

Protocol 1:

Signature: createRelationship(id, parent, child, label)

id: String of the relationship’s unique ID.

Parent: String of the parent ID, node ID of the source node of the relationship.

Child: String of the child ID, node ID of the destination node of the relationship.

Label: String of the label, a description of the relationship between the source and destination nodes.

Output: none.

Pre-Condition: Parent ID cannot be the same as child ID and relationship ID must be unique.

Post-Condition: The relationship is created with the given attributes.

## Class Description VectorTableWindow

The VectorTableWindow class is the visual representation of the handler of the vector, node and relationship modification process. It is responsible for displaying all the vector, node, and relationship table views which the user can interact with to modify the data. The superclass will be NavigatorWindow, and will handle the transition back to GraphWindow and NavigatorWindow.

### Contract Display Updated Table View

Contract number: 18

Contract Name: Display Updated Table View

Contract Description: Displays the user’s filter/search request results.

Protocol 1:

Signature: updateTableView(self)

Output: None

Pre-Condition: At least one modification has been made by the user.

Post-Condition: Display the updated table views with the modification made.

# Detailed Description of Component Graph Configuration

## Component Description

The component Graph configuration has responsibility of creating the graph, configuring the graph, and to then display the graph on the window. Graph knows Orientation, Interval Units, Interval, Position of Nodes, Position of Relationships, and Export Format.

## Class Description Graph

Graph is a visual representation of a vector. Graph knows attributes of Orientation, Export Format, Interval Units, Interval, Position of Nodes, and Position of Relationships.

### Contract Create Graph

Contract number: 19

Contract Name: Create Graph

Contract Description: Creates Graph

Protocol 1:

Signature: createGraph(vectorOrientation, nodesPosition, relationPosition, interval, intervalUnits)

vectorOrientation: String with orientation, ID of the node to which it points to.

nodesPosition: String with node position by grid.

relationPosition: String of ID related nodes.

interval: String with interval of marks on a timeline.

intervalUnits: seconds, minutes, hours, days, weeks.

Output: graph

Pre-Condition: Information on data base.

Post-Condition: Graph is created with relationships.

## Class Description GraphWindow

It is responsible for displaying the graph, it has the responsibility to know the graph, and each one of the nodes.

### Contract Display Graph Window

Contract number: 20

Contract Name: Display Graph Window

Contract Description: Display Graph Window

Protocol 1:

Signature: displayGraphWindow(graph)

graph: Structure with graph information.

Output: graph displayed in window.

Pre-Condition: A pre-build graph to be displayed.

Post-Condition: Graph is created with relationships.

## Class Description Node

The Node class represents a visual representation of a significant event that was marked for a vector. The node is responsible for knowing its node ID, name, timestamp, description, log entry reference, log creator, event type, icon type, source, node visibility, and also know each of its attribute visibility Boolean flags to be shown on the graph.

### Contract Create Node

Contract number: 16

Contract Name: Create Node

Contract Description: Creates Node

Protocol 1: Reference 8.3.1

# Detailed Description of Component Graph Export

## Component Description

The responsibilities of Graph Export is to have a finished graph report. This could be in a JPG or a csv file. The intention of Graph Export is to enable the users to export and use the information outside of the system

## Class Description Graph

Graph Export utility will be a button on the Window GUI, where you can select the format and it will start downloading the end product of the graph.

### Contract Export Graph

Contract number: 21

Contract Name: Export Graph

Contract Description: Export Graph

Protocol 1:

Signature: exportGraph(graph, exportFormat)

graph: Structure with graph information.

exportFormat: String with export format information, options are JPG, csv.

Output: JPG image.

Pre-Condition: A pre-build graph to be displayed.

Post-Condition: None

# Database

## Database Schema

The database will be a MongoDB database. It will have a total of 5 tables (log entries, vectors, graphs, time filters, and nodes). Because there is no real-life sample data, the columns/sections of these tables are based on partial information from meetings with the clients and the SRS.

The table of log entries will have the following columns/sections: list number, log entry timestamp, log entry event, vector, log entry location, log entry team.

The table of vectors will have the following columns/sections: vector log entries, vector name, vector creator, vector graph.

The table of graphs will have the following columns/sections: graph icons, graph nodes, graph vector, graph export format.

The table of nodes will have the following columns/sections: Node ID, Node name, Node timestamp, Node description, Log entry reference, Log creator, Event type, Icon type, Source, Node visibility, Node relationship

The table of time filters will have the following columns/sections: name, start datetime, end datetime.

$$