Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)

Software Design Document

Version 0.5

10 March 2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 3/3/2020 |
| Current Release: | 3/10/2020 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: | 3/9/2020 |
| Date of Next Review: | 3/11/2020 |
| Target Date for Next Update: | 3/11/2020 |

**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

ElsaTai Ramirez

Jake Lasley

Customer: Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members:

Eduardo Herrera – Systems Architect

Micheal Sansone – Lead Programmer

Jazmin Paz – Designer

Leslie Gomez – V&V

Jorge Flores – System Analyst

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 0.1 | 02/28/2020 | Leslie Gomez | Creation of Document |
| 0.2 | 03/10/2020 | Eduardo Herrera has no soul | Creation of Section 3; the Intake Component. |
| 0.3 | 03/10/2020 | Micheal John Sansone | added to2.2 |
| 0.4 | 03/10/2020 | Jorge Flores | Made contributions to 2.2 |
| 0.5 | 03/10/2020 | Leslie Gomez | Added sections 1.4, 1.5, 2.1 |
| 0.6 | 03/10/2020 | Jazmin Paz | Added sections 1.1, 1.2, 6.5, 6.6 |
| 0.7 | 03/31/2020 | Micheal Sansone | Added 4.4, 4.5, 4.6, 4.7 |
| 0.8 | 03/31/2020 | Jorge Flores | Contributed Section 5 and to Section 6.2 |
| 0.9 | 03/31/2020 | Leslie Gomez | Added sections 6.3,6.4, and added dependencies, modified document |
| 1.0 | 03/31/2020 | Eduardo Herrera | Modified the diagram and Section 3 |

Table of Contents

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

[Approval ii](#_heading=h.3as4poj)

[Document Change Control ii](#_heading=h.1pxezwc)

[Distribution List ii](#_heading=h.49x2ik5)

[Change Summary ii](#_heading=h.2p2csry)

1.**Introduction**

1.1. Purpose and Intended Audience

1.2. Scope of Product

1.3. References

1.4. Definitions, Acronyms and Abbreviations

1.4.1. Definitions

1.4.2. Acronyms

1.4.3. Abbreviations

1.5. Overview

2.**Decomposition Description**

2.1. System Collaboration Diagram

2.2. Subsystem and Component Descriptions

2.3. Descriptions

3.**Detailed Description of Intake Component**

3.1. Component Description

3.2. Class Description-Intake Controller

3.3. Class Description-Transcription Service

3.4. Class Description-Configuration

3.5. Class Description-OCR Service

3.6. Class Description-Cleansing Service

3.7. Class Description-ETL Service

4.**Detailed Description of User Interface Component**

4.1. Component Description

4.2. Class Description-Main Window

4.3. Class Description-Node Modification

4.4. Class Description-Correlate

4.5. Class Description-Searching

4.6. Class Description-Sorting

4.7. Class Description-Filtering

5.**Detailed Description of Access Control Component**

5.1. Component Description

5.2. Class Description-Permission Service

6.**Detailed Description of Vector DB**

6.1. Component Description

6.2. Class Description-State Management Service

6.3. Class Description-Pushing Service

6.4. Class Description-Pulling Service

6.5. Class Description-Log Entry Modification

6.6. Class Description-Vector Modification

# Introduction

## Purpose and Intended Audience

The system to be developed is named the PMR Insight Collective Knowledge (PICK), and it is a tool used to perform analyses that has the capability of deriving a thorough story from log files having to do with a series of events that take place on a given network. The system is able to ingest raw log files of a multitude of formats, parse the information, and store within the system as log entries. Furthermore, the user can specify a vector to analyze, flagging significant events and introducing them as nodes of the vector. Once the vector is finalized, the system can export a visual representation of the vector, known as a graph, as well as the table representation of the vector. The purpose of the system is to facilitate and accelerate event analysis, which would otherwise take up to months to complete. The system will be dynamic in that it will allow for little user intervention, if desired, because most of the work will be performed by the system.

The purpose of creating the Software Design Document (SDD) is to provide a fully enough description of the PMR Insight Collective Knowledge (PICK) system, that will allow for a comprehensive system design. This design will allow the software development and implementation with a complete understanding of what is to be built and how it is expected to be built. The SDD will identify all the subsystems, components and classes pertaining to the PICK system. This document will help the team analyze more in-depth how the design of the system was chosen as well as leading the team as how it shall be implemented.

## Scope of Product

The development of this tool is essential because the time and complexity needed in analyzing logs and generating reports from given data is currently a setback for the client. In creating this tool, the clients will be able to quickly analyze relevant logs and artifacts, and from the gathered data, generate detailed reports from the adversarial assessment. PMR Insight Collective Knowledge will do most of the work, retracting sizable workload from the analyst.

## References

[1] Dr. Roach Tai et al, Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)

Software Requirements Specification, version 1.7, 2020.

## Definitions, Acronyms, and Abbreviations

### Definitions

|  |  |
| --- | --- |
| Contract | Set of cohesive responsibilities |
| Pre- condition | Capture the conditions that must be true in order of the method to execute correctly |
| Post- condition | Must clearly state what is true when the method completes execution. |
| Subsystem | Set of cohesive classes that collaborate among themselves to assist a set of contracts |

### Acronyms

|  |  |
| --- | --- |
| SDD | Software Design Document |
| UTEP | University of Texas at El Paso |
| PICK | Prevent Insight Collective Knowledge |
| GUI | Graphical User Interface |

### Abbreviations

|  |  |
| --- | --- |
| e.g. | For example |
| i.e. | That is |

## Overview

The Software Design Document comprises the following sections: Decomposition Description, Detailed Description of Component and Database.

The Decomposition Description provides a description of how the component descriptions can be used by designers and maintainers. It will identify major design entities, for purposes such as determining which entity is responsible for specific functions and tracing requirements to design entities.

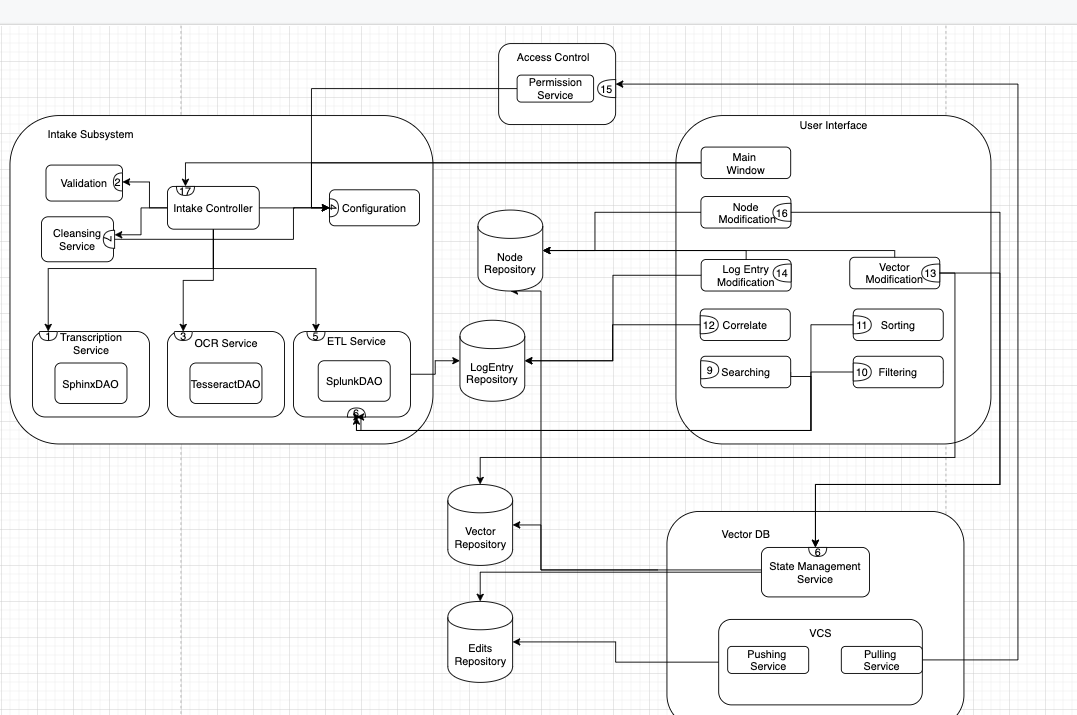
The Detailed Description of Component will provide a section with a detailed design description of each of the components listed in section 2.2, Subsystem and Component Description.

The Database section will describe the database schema and layout required by the system.

# Decomposition Description

Our system was designed using the following steps: first, we identified a set of classes, then we determined their responsibilities, and finally their collaborations. Each class description gives a description of the responsibilities that the class should have, as well as contains a list of contracts with their respective methods or functions that relate to those responsibilities. Each contract has a cohesive set of responsibilities it will implement and a list of the other contracts they will interact with to fulfill their task. Each contract is numbered and named so that it will be easier to identify and trace which contracts are interacting together. Once all the classes were properly identified, they were assembled into subsystems. These subsystems enclose groups of classes that collaborate with one another in order to support a set of contracts.

## System Collaboration Diagram



## Subsystem and Component Descriptions

**Subsystem Name:** Validation

**Subsystem Purpose:** The purpose is to validate the files based on rules before ingestion.

**Subsystem Contracts:**

**Subsystem Name:** Intake Controller

**Subsystem Purpose:** The purpose is to decide what to do with a file based on its type.

**Subsystem Contracts: 2,4,7**

**Subsystem Name:** Configuration

**Subsystem Purpose:** The purpose is to allow for the configuration of the event.

**Subsystem Contracts:**

**Subsystem Name:** Transcription Service

**Subsystem Purpose:** The purpose is to send the file for transcription.

**Subsystem Contracts:** 5

**Subsystem Name:** OCR Service

**Subsystem Purpose:** The purpose is to send the files to the ocr.

**Subsystem Contracts:** 5

**Subsystem Name:** Cleansing Service

**Subsystem Purpose:** The purpose is to remove empty files or lines.

**Subsystem Contracts:** 4

**Subsystem Name:** ETL Service

**Subsystem Purpose:** The purpose is to ingest files based on rules of ingestion.

**Subsystem Contracts:**

**Subsystem Name:** Permission Service

**Subsystem Purpose:** To be able to provide certain privileges to a user.

**Subsystem Contracts:** 4, 17

**Subsystem Name:** Searching

**Subsystem Purpose:** Be able to search by criteria

**Subsystem Contracts:** 6

**Subsystem Name:** Filtering

**Subsystem Purpose:** Be able to filter what is currently being shown to a user.

**Subsystem Contracts:** 6

**Subsystem Name:** Sorting

**Subsystem Purpose:** Be able to sort what is currently being shown to a user.

**Subsystem Contracts:** 6

**Subsystem Name:** Correlate

**Subsystem Purpose:** Be able to provide additional features for searching to the user.

**Subsystem Contracts:** 6

**Subsystem Name:** Vector Modification

**Subsystem Purpose:** Provide the user with capabilities to modify vector information.

**Subsystem Contracts:** 8

**Subsystem Name:**  Log Entry Modification

**Subsystem Purpose:** Provide the user with capabilities to modify a log entry.

**Subsystem Contracts:** N/A

**Subsystem Name:** Node Modification

**Subsystem Purpose:** Provide the user with capabilities to modify information on a node.

**Subsystem Contracts:** 8

**Subsystem Name:** VCS

**Subsystem Purpose:** Provide users with pushing and pulling capabilities with vectors.

**Subsystem Contracts:** N/A

## Dependencies

The coupling of the different subsystems in our system are depicted in figure 1 which shows the complete system with the main components.

For the PICK system to perform properly, additional software requirements must be fulfilled. Our system will utilize the help of third-party software, the system is highly coupled to other specific versions of external software required to perform its functionalities. The system requirements for the software are:

* Operating System: Kali Linux
* OCR: TesseractDAO
* Audio Transcription: SphinxDAO

# Detailed Description of Intake Component

## Component Description

Intake various log files and convert them into log entries; the log entries are then going to be validated. If there are Image files the OCR Service is going to convert the files into text, then using the ETL Service its going to convert the text into Log Entries. Similarly, If there is an audio Log File encountered, the Transcription is going to convert the audio file into text, then ETL Service is going to convert the text into Log Entries. Finally, the Intake Component is responsible for the validation of the Log Entries; for example, when a Log Entry is outside of the date range (set at the beginning of the workflow) it's going to be declared violated.

## Class Description - Intake Controller

The intake controller is going to be the medium between the different intake services and the presentation of the intake. From a path to a directory it's going to grab all the log files and convert them to log entries with the use of the ETL Service; similarly, the process is applied to a single file, if a path to a file is given. However, before converting the Log File into a Log Entry, the Intake Controller is going to cleanse the file using the Cleansing Service. Then it's going to continue with the conversion of Log Files into Log Entries. Furthermore, It's going to call the Transcription or the OCR Services if the log file is an unreadable format; which extracts the text and returns the log entry, again with the use of the ETL Service. Finally, it allows an external class to the component to subscribe an Observer to the Intake to receive the log entries that failed the Validation Service.

|  |  |
| --- | --- |
| Class Name: Intake Controller | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Private Responsibilities· Provide an abstract interface over the intake process, being the conversion of raw log files into validated log entries. | |
| Contract 1: Ingestion of Log Files | |
| Responsibilities | Collaborations |
| · From the relative paths in the configuration, cleanse and validate all the log files within those paths. | · Configuration · Cleansing Service · Validation Service |
| Contract 2: Conversion of Multimedia Files | |
| Responsibilities | Collaborations |
| · Convert audio or image files into text that could be interpreted by the system.· Once the text is extracted, the text is then going to be cleansed. | · OCRService · TranscriptionService · CleansingService |
| Contract 3: Extraction of Log Entries | |
| Responsibilities | Collaborations |
| · Extraction of Log Entries from the Log Files that were ingested. | · ETLService |

## Class Description - Transcription Service

The Transcription Service is going to grab the audio files that are being ingested and extract the text from those files. Once the text is extracted from the audio files, the Transcription Service is going to make use of the Cleansing Service to cleanse the data that was extracted from the audio files.

|  |  |
| --- | --- |
| Class Name: Transcription Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Audio File Conversion | |
| Responsibilities | Collaborations |
| · Transform the audio file into a string format. | · SphinxDAO |

### 

## Class Description - Configuration

Contains the general configuration of the project. This includes - the path of each of the team’s directory, the current working directory and any other information that affects the responsibilities within the Intake Component.

|  |  |
| --- | --- |
| Class Name: Configuration | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Private Responsibilities· Maintaining the global configuration variables that affect the behavior of the classes that depend on the state of information. | |
| Contract: 1. Cleansing Text | |
| Responsibilities | Collaborations |
| · Maintain the distinct relative paths of the Log Files (Red, Blue, and White Team directories). |  |
| Contract: 2. Audio File Conversion | |
| Responsibilities | Collaborations |
| · Maintain the cleansing scripts that are going to be used by other classes. |  |

### 

## Class Description - OCR Service

Just like the Transcription Service, the OCR Service is going to grab Log Files that are being processed during the ingestion phase that are images. The OCR is going to take those images and extract the text from the file.

|  |  |
| --- | --- |
| Class Name: Transcription Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Extraction of Image files | |
| Responsibilities | Collaborations |
| · Transform the image files into a string format. | · TesseractDAO |

### 

## Class Description - Cleansing Service

The Cleansing Service is going to remove many unwanted series of characters from the Log Files. Furthermore, it's going to take in a cleansing script (which is defined in the configuration file) that applies that cleansing to each of the Log Files.

|  |  |
| --- | --- |
| Class Name: Cleansing Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Cleanse Text | |
| Responsibilities | Collaborations |
| · Removing the series of unwanted characters in the given text.· Applying any cleansing scripts specified in the configuration to the given text. | · Configuration |

## Class Description - ETL Service

The ETL Service is going to communicate with the SplunkDAO which converts Log Files into Log Entries, Filters the Log Entries with a given pattern, and returns the Log Entries that matched a given pattern.

|  |  |
| --- | --- |
| Class Name: ETL Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Log Entry Conversion | |
| Responsibilities | Collaborations |
| · Converting the given Log Files into Log Entries. | · SplunkDAO |
| Contract: 2. Log Entry Search | |
| Responsibilities | Collaborations |
| · Search the ingested Log Entries with a given string, wildcard, or regex pattern. | · SplunkDAO |
| Contract: 3. Log Entry Filter | |
| Responsibilities | Collaborations |
| · Retrieve a set of Log Entries that have been filtered with a given timestamp or an absolute path. | · SplunkDAO |

## 

# Detailed Description of User Interface Component

## Component Description

This component is going to handle all the graphics and user interactions of the system. It contains the main window, which is the start of the system. The user specifies the configuration variables and commences the ingesting of the log files into the system. Furthermore, it contains the Sorting, Filtering, and Searching subsystems. These subsystems are going to allow the user to create smaller segments of data from all the Log Entries in the system; making it easier to discover what the user is searching for. Finally, there is the Node Modification, Vector Modification, Log Entry Modification, and Correlate subsystems. These final subsystems handle the modification of data from the user.

## Class Description - Main Window

The main window is the start of the system. This is going to handle the configuration variables that the user is declaring at the beginning of the system process. It's going to save those variables into the configuration subsystem in the Intake Component. Furthermore, it's going to delegate the ingestion process to the Intake Controller.

|  |  |
| --- | --- |
| Class Name: Main Window | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Private Responsibilities· Edit the configuration variables in the configuration subsystem with the values that the user provides.· Initialize the Intake phase by delegating it to the Intake Controller. | |

## Class Description - Node Modification

The node modification is going to handle the user request to change the data of a specific node.

|  |  |
| --- | --- |
| Class Name: Node Modification | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Edit Node Data | |
| Responsibilities | Collaborations |
| · Handle the request for the modification of data to a specific Node. | · Node Repository |
| Contract: 2. Save Edit | |
| Responsibilities | Collaborations |
| · Save the previous and current edit of a | · State Management Service |

## 

## Class Description - Correlate

The Correlate will allow users to correlate Log Entries to Vectors.

|  |  |
| --- | --- |
| Class Name: Correlate | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Correlation | |
| Responsibilities | Collaborations |
| · Correlate Log Entries to Vectors. | · LogEntry Repository |

## Class Description - Searching

The Correlate will allow users to search for Log Entries.

|  |  |
| --- | --- |
| Class Name: Searching | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Handle the searching of Log Entries | |
| Responsibilities | Collaborations |
| · Search for Log Entries. | · ETL Service |

## Class Description - Sorting

The Correlate will allow users to sort Log Entries.

|  |  |
| --- | --- |
| Class Name: Sorting | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Handle the sorting of Log Entries | |
| Responsibilities | Collaborations |
| · Sort Log Entries. | · ETL Service |

## Class Description - Filtering

The Correlate will allow users to filter Log Entries.

|  |  |
| --- | --- |
| Class Name: Filtering | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract: 1. Handle the filtering of Log Entries | |
| Responsibilities | Collaborations |
| · Filter Log Entries. | · ETL Service |

## 

# Detailed Description of Access Control Component

## Component Description

The access control component is going to contain several actions that are permitted or denied based on the user class. The Lead is going to be able to perform all the actions of the system, while the Analyst class is not going to be able to verify the sync from the analyst (an action performed in the VCS Subsystem).

## Class Description - Permission Service

The permission service is going to handle the request of other classes, in checking if the request they are doing is valid or denied based on the user's class. The user class is going to be retrieved from the configuration subsystem in the Intake Component. Additionally, the request other classes are sending is going to contain the actions the user is performing in their component. The classes or services are checking with the Permission Service that the actions performed by the user are possible under their current user class.

|  |  |
| --- | --- |
| Class Name: Permission Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Private Responsibilities· Identify the user’s role and accepted actions by retrieving their User Class (Analyst or Lead) from the Configuration Subsystem. | |
| Contract 1: Check Permission for Sync | |
| Responsibilities | Collaborations |
| · Checking if the user has the capabilities of synching with an Analyst. |  |
| Contract 2: Check Permission for Hosting | |
| Responsibilities | Collaborations |
| · Checking if the current user has the capabilities of hosting the project; making it available for other Analysts to connect. |  |

### 

# Detailed Description of Vector DB

## Component Description

The Vector DB component is going to deal with the synching of data among users and the being able to move through the history of edits. Provides a way for Analysts to push or pull their edits in the current project; this could be any type of modification to the data. Furthermore, it's going to provide the current users to navigate their own edits locally. Finally, they have the availability to restore a previous state of the data.

## Class Description - State Management Service

The permission service is going to handle the request of other classes, in checking if the request they are doing is valid or denied based on the user's class. The user class is going to be retrieved from the configuration subsystem in the Intake Component. Additionally, the request other classes are sending is going to contain the actions the user is performing in their component. The classes or services are checking with the Permission Service that the actions performed by the user are possible under their current user class.

|  |  |
| --- | --- |
| Class Name: State Management Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Private Responsibilities· Tracking the changes in the Node Repository, Log Entry Repository, Vector Repository, and placing them as an entry in the Edits Repository. | |
| Contract 1: Display Edits | |
| Responsibilities | Collaborations |
| · Retrieve the chain of edits that the user has performed in the data. | · Edits Repository |
| Contract 2: Restore State | |
| Responsibilities | Collaborations |
| · Restoring to a previous state of the data in the repositories; specified by a specific edit that the caller wants to return to. | · Node Repository **·** LogEntry Repository  **·** Vector Repository  **·** Edits Repository |

## 6.3. Class Description - Pushing Service

The pushing service handles version control properties that lets the user update any current version from their own machine to the Vector DB for analyst to be in sync within each other

|  |  |
| --- | --- |
| Class Name: Pushing Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract 1: Syncs data to database | |
| Responsibilities | Collaborations |
| · Pushes any edits created by an analyst to the database for history control. |  |

## 6.4. Class Description - Pulling Service

The pushing service handles version control properties that lets the user update any current version from their own machine to the Vector DB for analyst to be in sync within each other

|  |  |
| --- | --- |
| Class Name: Pulling Service | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract 1: Pulls data | |
| Responsibilities | Collaborations |
| · Pulls any edits from the main database for analysts to keep track of any changes within analysts. | · Access Control |

### 

## 6.5. Class Description - Log Entry Modification

The log entry modification handles the modification of log entries that may have been changed, added or deleted, and updates it to the system.

|  |  |
| --- | --- |
| Class Name: Log Entry Modification | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract 1: Modifies Log Entries | |
| Responsibilities | Collaborations |
| · Modifies any log entries that need to be updated in the system. | · Node Repository· Log Entry Repository |

## 6.6. Class Description - Vector Modification

The log entry modification handles the modification of vectors and updates to the system.

|  |  |
| --- | --- |
| Class Name: Vector Modification | |
| Superclass: N/A | |
| Subclasses: N/A | |
| Contract 1: Modifies Vectors | |
| Responsibilities· Modifies any vectors that need to be updated in the system. | Collaborations· Node Repository· State Management Service |
|  |  |

### 