caBIG Compatibility Guidelines

National Cancer Institute Center for Bioinformatics (NCICB)

Last Updated: June 16, 2004

Draft

TABLE OF CONTENTS

1. INTRODUCTION

Purpose
The purpose of this document is to provide guidelines for caBIG participants and interested stakeholders on caBIG compatibility. This document provides a set of definitions that caBIG participants can use to measure the maturity level of potential caBIG applications.

This is a living document that will evolve to keep in line with the emerging caBIG compatibility guidelines and definitions as determined by the caBIG community.

Audience
It is expected that all stakeholders familiarize themselves with the requirements of caBIG compatibility. For the purpose of this document, stakeholders include caBIG participants (Developers, Adopters and Users) and any other groups or individuals interested in contributing to caBIG activities.

Document Organization
This document primarily defines and describes caBIG compatibility in terms of specified maturity levels - Legacy, Bronze, Silver, and Gold. Requirements for attaining each of the maturity levels are described in detail in subsequent sections, along with additional illustrative examples.

How to use these guidelines
These guidelines can be used as a baseline to derive the software maturity level for a given application with regards to caBIG compliance and should be used to guide future software development efforts.

The overall goal of the caBIG initiative is to have all caBIG compliant applications to eventually attain the Gold level of maturity. It is anticipated that new development activities undertaken as part of caBIG in the near term will meet the Silver level of compliance, until Gold standards are more clearly defined.

Bronze level maturity will be applicable for meeting the minimum caBIG compliance level of maturity. This level of maturity is a transitional state along the pathway to the Silver and Gold levels of compliance. In the immediate term, it may be necessary to retrofit existing legacy systems to achieve the Bronze level of compliance.

As the Gold level compliance requirements are defined by the caBIG community, the natural progression will be to ensure that those existing applications meet these finalized requirements.

Updates to the caBIG Compatibility Guidelines
All updates to the caBIG Compatibility Guidelines will be reflected in the evolving document and will be released on an ongoing basis.

2. COMPATABILITY MATRIX

This Compatibility Matrix is a reference guide to measure caBIG Compatibility of Software Applications.

Maturity Model	Legacy	Bronze	Silver	Gold
Interface Integration	*- No Programming interfaces to the system are available. Only local data files in a custom format can be read* *- Some ad hoc data transfer mechanism such as FTP*	*- Provide baseline* programmatic access to data. Data can be read from remote electronic sources or from commonly used file formats Data can be pushed out to from applications to other external data sources*	*- Well-described API's that provide access to data objects.* *- System architecture separated into tiers and interoperable components* *- Data read in from standards-based electronic sources that support standard or commonly used interchange formats* *- Documented component description of the underlying data structures that are accessible* *- Standard messaging systems where appropriate*	*- All features of Silver, plus:* *- Interoperable with data grid architecture to be defined by caBIG* *- Fully componentized provide access to individual resources in the form of grid services*
Vocabularies / Terminologies & Ontologies	*- Free text used throughout for data collection*	*- Use of publicly accessible standardized controlled vocabularies as well as local terminologies*	*- Standard terminologies approved by public standards bodies or the caBIG Vocabulary/CDE Workspace are used for all relevant data collection fields.*	*- All features of Silver, plus:* *- Fully compliant with caBIG recommended standards for vocabulary terminology services and content sources*
Data Elements	*- No Structured metadata is recorded*	*- Some type of metadata describing the information in the system is used for data collection and external reporting. Metadata is retrieved from external repository shared by multiple applications.* *- Common Data Elements should be built using controlled terminology*	*- Use common standard electronic representation for CDE's such as ISO 11179 or comparable standard* *- CDEs are harmonized and re-used from across the Domain Workspace* *- Common Data Elements are built using standard controlled terminologies approved by public standards bodies or the caBIG Vocabulary/CDE Workspace*	*- All features of Silver, plus:* *- Programmatic access to all metadata, including data class descriptions, site and source information, and any other caBIG-defined metadata requirements and use information models* *- Use the caBIG standard or electronic representation of metadata and Common Data Elements*
Information Models	*- No particular information model is used to represent data*	*- Some type of diagrammatic model describing the data relationship is available in electronic format*	*- Information models defined in a standard modeling language such as UML*	*- All features of Silver, plus:* *- Information models are harmonized with other s across the caBIG Domain Workspace*

** Baseline includes the data needed to be shared with other apps in order to eliminate duplicate entry and manual processing.*

3. caBIG – *"Legacy"* MATURITY LEVEL

General Description:

caBIG – "*Legacy*" Maturity Level would not be considered caBIG compliant.

1. Interface Integration: There are no programming interfaces to the system. It has a proprietary data repository with no existing means for external systems to access to data repository. Only local data files in a custom format can be read by the internal applications, i.e., data access is proprietary.

2. Vocabularies/Terminologies & Ontologies: The data is collected verbatim in free text format. No external vocabularies or utilities are used to assist in medical coding.

3. Data Elements: No structured metadata is recorded.

4. Information Models: No particular information model is used to represent the data architecture.

The illustration below shows two applications that have no programmatic access to data. Having an FTP method of data access or transfer does not meet with the Interface Integration requirements.

Fig. 1 below illustrates a non-caBIG compliant environment

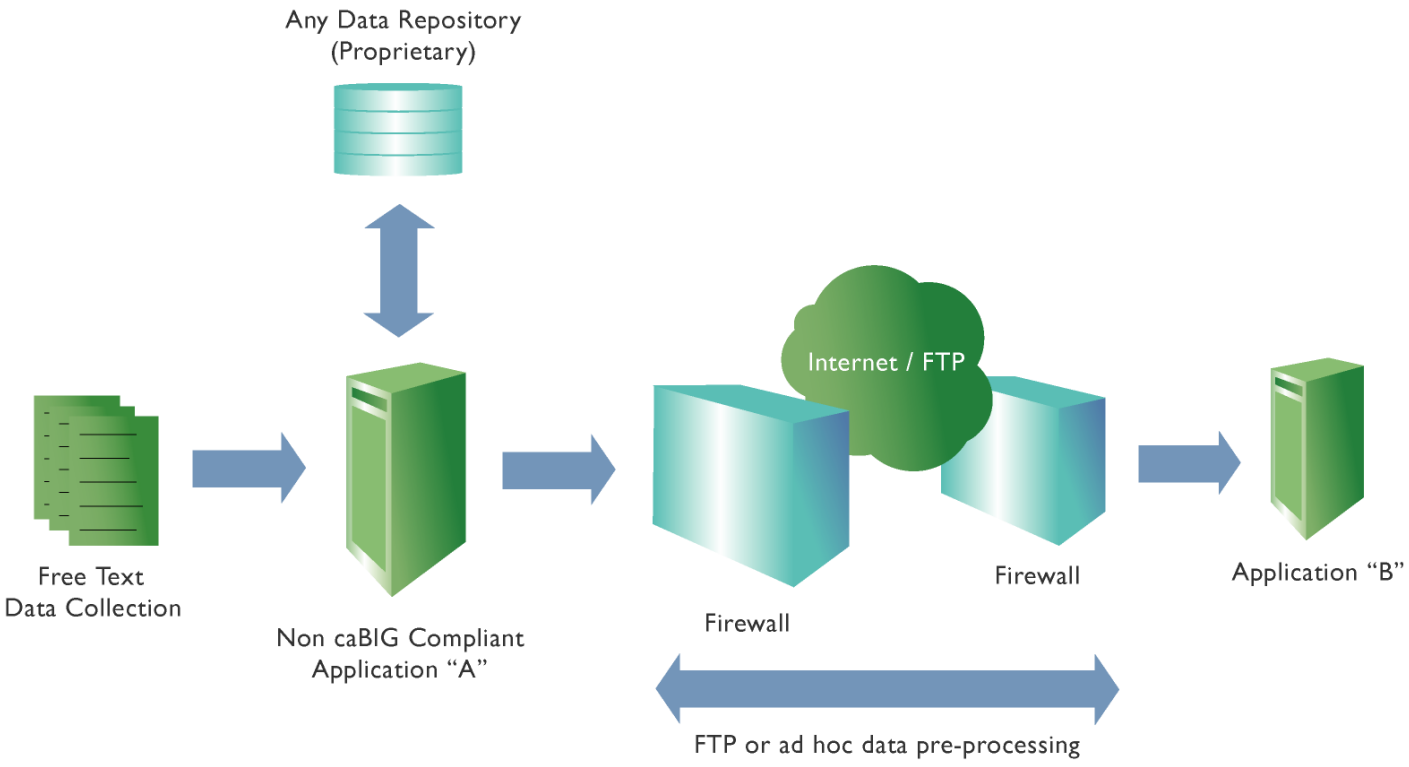


Fig. 1

4. caBIG – *"Bronze"* MATURITY LEVEL

General Description:

caBIG – *"Bronze"* Maturity Level demonstrates that a given application satisfies the minimum set of requirements to be caBIG compliant. At this Level, the given system satisfies all of the following requirements:

1. Interface Integration: The application would provide baseline* programmatic access to data. There should be a way for applications at external sites to access the data electronically by using industry standard file formats. A consumer program should be able to access data remotely using industry standard API's such as ODBC; JDBC or Web Services technologies.

* Baseline includes the data needed to be shared with other apps in order to eliminate duplicate entry and manual processing.

Examples:

JDBC Example: http://www.eas.asu.edu/~cse494db/IonJDBC/JDBCExample.html

Web Services Example: http://www.w3schools.com/wsdl/wsdl_documents.asp
http://ncicb.nci.nih.gov/core/caBIO/SOAP

CGI Bin Script Example: http://my.execpc.com/~keithp/bdlogcgi.htm

2. Vocabularies/Terminologies: Use of public standardized controlled vocabularies (such as MedDRA, Snomed, NCI Thesaurus) as well as local terminologies is required.

3. Data Elements: The application should have some type of metadata describing the information in the system, recorded in electronic format of some kind. Metadata should be retrieved from an external repository shared by multiple applications. Metadata should encompass those fields used for data collection and external reporting. Common Data Elements should be built using controlled terminology.

4. Information Models: There is some type of diagrammatic model describing the data relationship available in electronic format, such as an Entity Relationship Diagram.

Examples:

Entity Relationship Diagram: http://members.iinet.net.au/~lonsdale/docs/erd.pdf

Please refer to Appendix: I

Fig. 2 below illustrates a caBIG - Bronze Level compliant environment

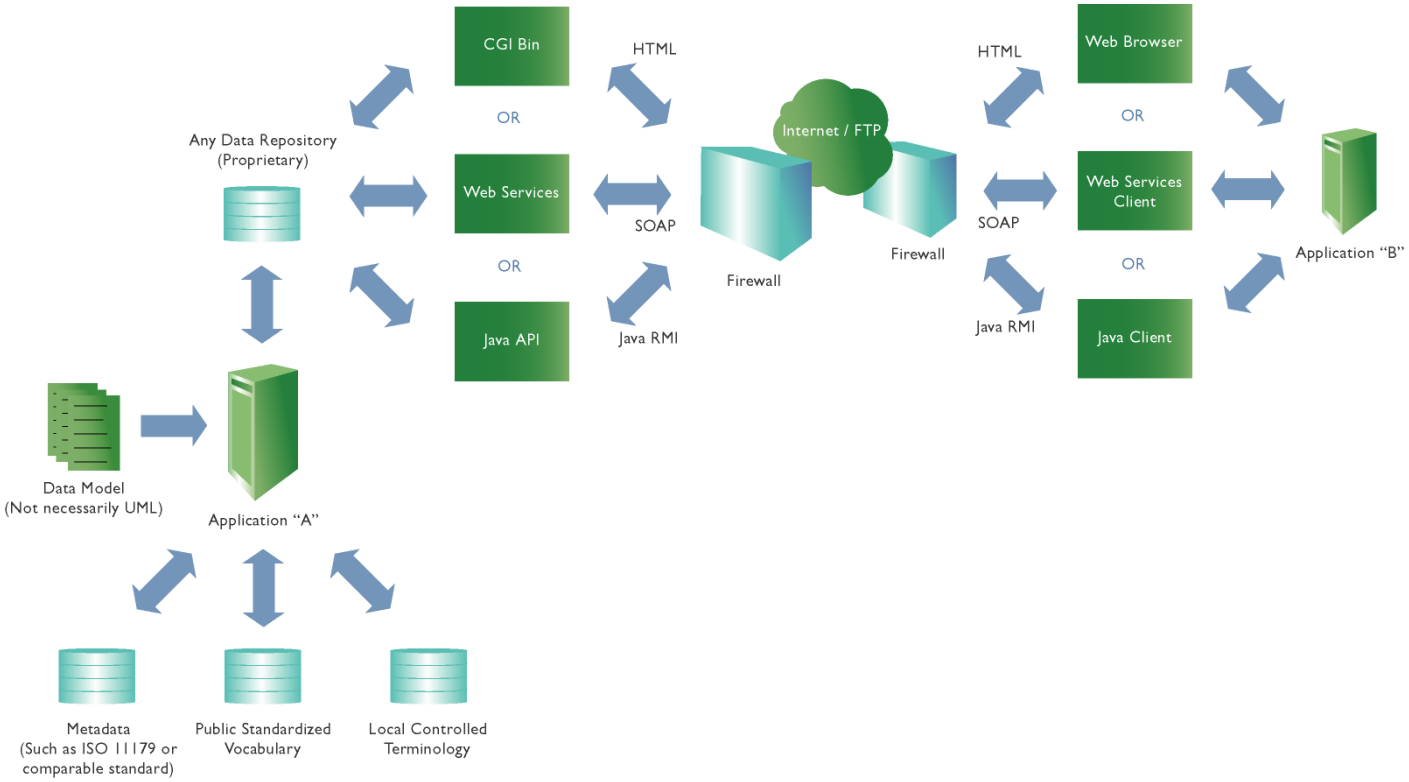


Fig. 2

5. caBIG – *"Silver"* MATURITY LEVEL

General Description:

caBIG – "*Silver*" Maturity Level demonstrates that a given application satisfies the moderate set of requirements to be caBIG compliant. At this Level, the given system satisfies all the Bronze requirements plus all of the following requirements:

1. Interface Integration: The application will provide a well-documented Application Programming Interfaces (API) on the caBIG platform. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code. An API can also provide an interface between a high level language and lower level utilities and services which were written without consideration for the calling conventions supported by compiled languages. In this case, the API's main task may be the translation of parameter lists from one format to another and the interpretation of call-by-value and call-by-reference arguments in one or both directions. API's provided will expose methods to external systems for accessing system data or performing system functionality.
The API's provided could be a Java API

Examples:

Java API: http://ncicb.nci.nih.gov/NCICB/content/ncicblfs/caBIO2-1_JavaDocs
Please refer to Appendix: A
The system architecture will be separated into tiers and interoperable components. A standard J2EE application will have the Presentation Tier, Business Tier, Integration Tier and Data Tier.
Please refer to Appendix: B
The data will be read in from standards-based electronic sources that support standard or commonly used interchange formats.
Please refer to Appendix: C
Documented component description of underlying data structure where appropriate will be used.
Please refer to Appendix: D
Standard messaging systems are used where appropriate.
Examples: The following is the URL for the HL7.
HL7 : http://www.hl7.org/
Please refer to Appendix: F

2. Vocabularies/Terminologies: The application will use Standard terminologies approved by public standards bodies or the caBIG Vocabulary/CDE Workspace are used for all relevant data collection fields.

Examples: The following CHI example is a set of standardized terminologies to be used for specific purposes.

CHI: http://www.whitehouse.gov/omb/egov/gtob/health_informatics.htm
Please refer to Appendix: E

3. Data Elements: The application would use common standard electronic representation for Common Data Elements such as ISO 11179 or comparable standard or comparable standard.

Examples:
ISO11179: http://ncicb.nci.nih.gov/NCICB/core/caDSR/ISO11179
Please refer to appendix: G

The CDE's are harmonized and re-used from across the Domain Workspaces. Common Data Elements are built using standard controlled terminologies approved by public standards bodies or the caBIG Vocabulary/CDE Workspace.

4. Information Models: The Object Model of the application is represented in a standard modeling language such as Universal Modeling Language (UML). The Object Model will represent the automated processes, human interactions, and application/data associations. These Object Model may not necessarily be programmatically available. The Object Model will use standard notation for defining various human and application activities, using:

- Sequence diagrams
- Collaboration diagrams
- Use case diagrams
- State diagrams
- Activity diagrams
- Static structure diagrams (class and object diagrams)
- Component diagrams
- Deployment diagrams

Examples:

UML Example: http://www.geocities.com/SiliconValley/Network/1582/uml-example.htm

Please refer to Appendix: H

Fig. 3 below illustrates a caBIG - Silver Level compliant environment

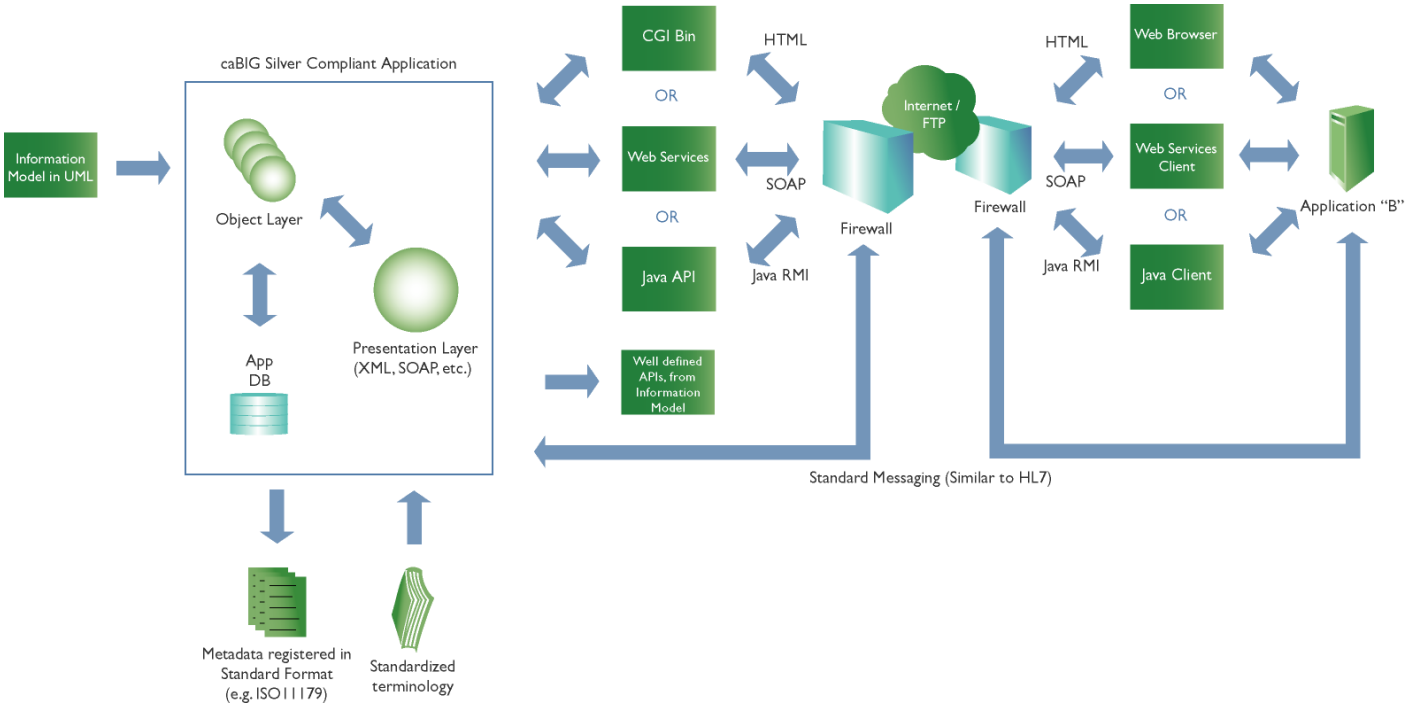


Fig. 3

6. caBIG – *"Gold"* MATURITY LEVEL (PLACE HOLDER)

General Description:

caBIG – *"Gold"* Maturity Level, the maximum set of requirements to be caBIG compliant, will be defined as caBIG guidelines and definitions evolve through the work of the caBIG community.

1. Interface Integration: The application will comply with all of the Silver requirements, plus the following set of requirements:

The application will be Interoperable with "data grid architecture" yet to be defined by the caBIG consortium.

The application will also be fully componentized and provide access to individual resources in the form of grid services.

2. Vocabularies/Terminologies: The application should be fully compliant with caBIG recommended standards for vocabulary terminology services and content sources.

3. Data Elements: There should be programmatic access to all metadata, including data class descriptions, site and source information, and any other caBIG-defined metadata requirements and use information models.

The application should use the caBIG standard or electronic representation of metadata and Common Data Elements.

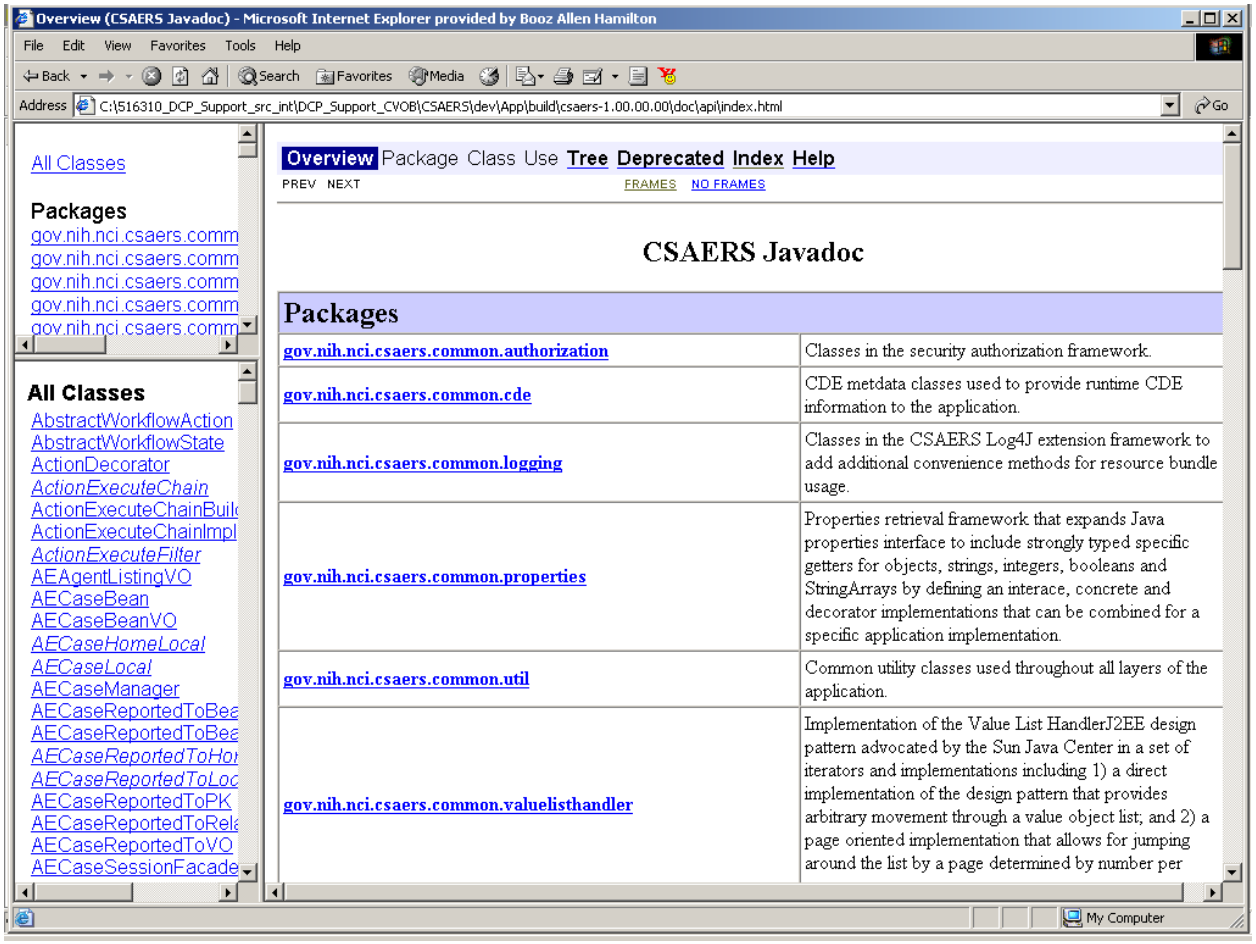
4. Information Models: The Information models are harmonized across the various contexts as defined in the caBIG Domain Workspaces.

Please NOTE: *CSAERS (Cancer Serious Adverse Event Reporting System) is a system developed by Booz Allen Hamilton for the Division for Cancer prevention. We have leveraged some of the artifacts that are part of the CSAERS application as examples in the following Appendices to show how it maps to the caBIG Compatibility Matrix.*

APPENDIX: A

Well-described API's that provide access to data objects

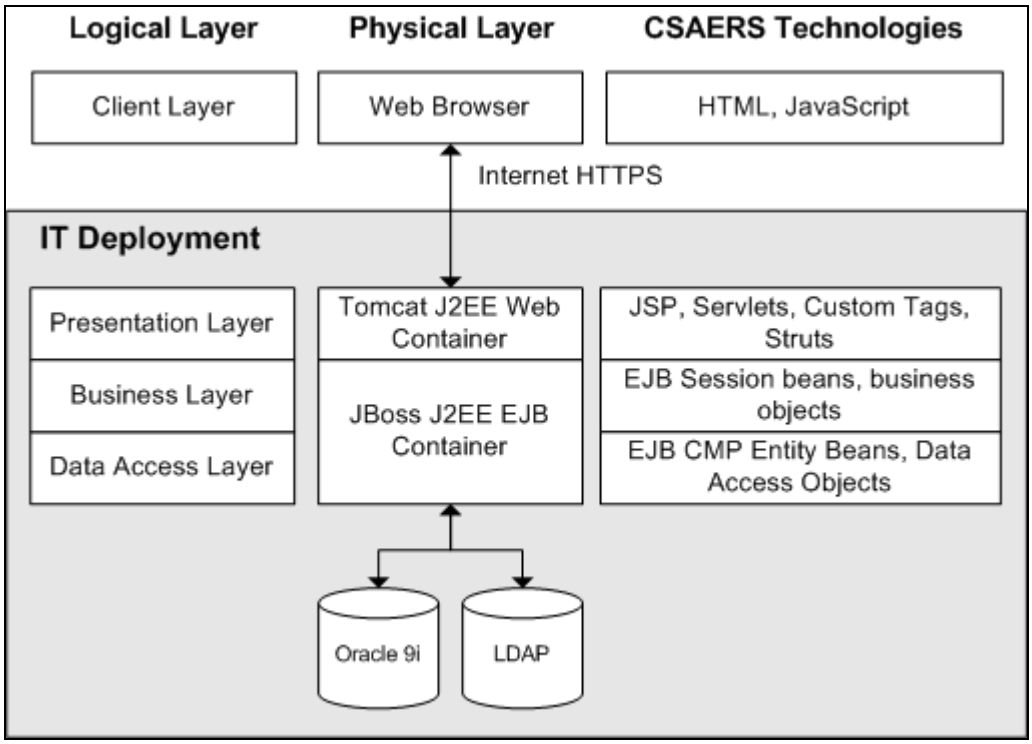
For this example we are showing a screen shot of the CSAERS Javadoc.



APPENDIX: B

System architecture separated into tiers and interoperable components

The following illustration shows how CSAERS is divided into multiple tiers and components within each tier.



CSAERS Architecture Tiers

APPENDIX: C

Data read in from standards-based electronic sources that support standard or commonly used interchange formats.

In the following example we have shown a SAX parser that reads-in an XML Configuration file. This code is used in the CSAERS workflow module.

```
/*
 * WorkflowDocumentHandler.java
 *
 * Chemoprevention Serious Adverse Event Reporting System
 * Division of Cancer Prevention
 * National Cancer Institute
 *
 * created by Booz Allen Hamilton
 * Created on Dec 9, 2003
 */

package gov.nih.nci.csaers.model.workflow.fsm.xml;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import gov.nih.nci.csaers.model.workflow.fsm.FSMAction;
import gov.nih.nci.csaers.model.workflow.fsm.FSMActionEnumeratedType;
import gov.nih.nci.csaers.model.workflow.fsm.FSMState;
import gov.nih.nci.csaers.model.workflow.fsm.FSMStateEnumeratedType;
import gov.nih.nci.csaers.model.workflow.fsm.FiniteStateMachineWorkflowModel;

import org.apache.log4j.Logger;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.helpers.DefaultHandler;

/**
 * This document handler can handle parsing XML representations of
 * finite state machine workflow models.
 *
 * Note that it does not need to do a lot of data validation and
 * throw exceptions, because the parser validates the data against the
 * schema.
 *
 * @author Tajh Taylor
 * @see gov.nih.nci.csaers.mode.workflow.fsm.FiniteStateMachineWorkflowModel
 */
public class FSMWorkflowDocumentHandler extends DefaultHandler {
    private static final Logger logger =
        Logger.getLogger(FSMWorkflowDocumentHandler.class);

    // constants for the tags we need to know about
    public static final String PROPERTIES_TAG = "properties";
    public static final String WORKFLOW_NAME_TAG = "workflow-name";
    public static final String WORKFLOW_VERSION_TAG = "version";
    public static final String DESCRIPTION_TAG = "description";
    public static final String STATE_CODE_TAG = "state-code";
    public static final String APPROVAL_CODE_TAG = "approval-code";
    public static final String STATE_TAG = "state";
    public static final String OWNER_ROLE_NAME_TAG = "owner-role-name";
```

```
    public static final String DESIGNATED_ROLE_NAME_TAG = "designated-role-name";
    public static final String ENTRY_POINT_TAG = "entry-point";
    public static final String EXIT_POINT_TAG = "exit-point";
    public static final String ACTIONS_TAG = "actions";
    public static final String ACTION_TAG = "action";
    public static final String ACTION_NAME_TAG = "action-name";
    public static final String REFLEXIVE_TAG = "reflexive";
    public static final String EXTERNAL_TAG = "external";
    public static final String TARGET_USER_METHOD_TAG = "target-user-method";
    public static final String REFLECTIONS_TAG = "reflections";
    public static final String REFLECTION_TAG = "reflection";
    public static final String TRANSITIONS_TAG = "transitions";
    public static final String TRANSITION_TAG = "transition";
    public static final String SOURCE_STATE_CODE_TAG = "source-state-code";
    public static final String TARGET_STATE_CODE_TAG = "target-state-code";

    // parsing status and temporary content data
    private StringBuffer charDataBuffer;
    private String       charContainerName;
    private boolean      finishedParsing;
    private Map          elementContainerMap;
    private List         actionList;
    private List         stateList;

    private boolean inStateTag;
    private boolean inActionTag;
    private boolean inReflectionTag;
    private boolean inTransitionTag;
    private boolean inPropertiesTag;

    // the currently worked on workflow model
    private FiniteStateMachineWorkflowModel workflowModel;

    // all completed workflow models stored in here
    private Map workflowModelMap;

    public FSMWorkflowDocumentHandler() {
        charDataBuffer = new StringBuffer();
        workflowModelMap = new TreeMap();
    }

    public void startDocument()
    throws SAXException {
        logger.debug("Starting to parse a FSMWorkflowModel document");
        workflowModel = new FiniteStateMachineWorkflowModel();
        setFinishedParsing(false);
        setInActionTag(false);
        setInStateTag(false);
        setInReflectionTag(false);
        setInTransitionTag(false);
        setInPropertiesTag(false);
        elementContainerMap = new TreeMap();
        actionList = new ArrayList();
        stateList = new ArrayList();
        charContainerName = null;
    }

    public void startElement(String uri, String localName, String qName,
                             Attributes attributes)
    throws SAXException {
        String tagName = localName;
//        if (logger.isDebugEnabled()) {
//            logger.debug("Starting an element, localName is " + tagName);
//        }
//        logger.debug("   uri: " + uri);
//        logger.debug("   localName: " + localName);
```

```
//        logger.debug("   qName: " + qName);

        // make a note of tags that contain char data
        if (WORKFLOW_NAME_TAG.equals(tagName)
            || WORKFLOW_VERSION_TAG.equals(tagName)
            || DESCRIPTION_TAG.equals(tagName)
            || STATE_CODE_TAG.equals(tagName)
            || APPROVAL_CODE_TAG.equals(tagName)
            || OWNER_ROLE_NAME_TAG.equals(tagName)
            || DESIGNATED_ROLE_NAME_TAG.equals(tagName)
            || ACTION_NAME_TAG.equals(tagName)
            || TARGET_USER_METHOD_TAG.equals(tagName)
            || SOURCE_STATE_CODE_TAG.equals(tagName)
            || TARGET_STATE_CODE_TAG.equals(tagName)) {

            charContainerName = tagName;
            charDataBuffer.setLength(0);
        } else {
            charContainerName = null;
        }

        // make a note of our basic container tags
        if (ACTION_TAG.equals(tagName)) {
            setInActionTag(true);
            elementContainerMap.clear();
        } else if (STATE_TAG.equals(tagName)) {
            setInStateTag(true);
            elementContainerMap.clear();
        } else if (REFLECTION_TAG.equals(tagName)) {
            setInReflectionTag(true);
            elementContainerMap.clear();
        } else if (TRANSITION_TAG.equals(tagName)) {
            setInTransitionTag(true);
            elementContainerMap.clear();
        } else if (PROPERTIES_TAG.equals(tagName)) {
            setInPropertiesTag(true);
        }

        // check for contained element tags w/o content
        if (isInActionTag()) {
            if (REFLEXIVE_TAG.equals(tagName)) {
                elementContainerMap.put(REFLEXIVE_TAG, "");
            } else if (EXTERNAL_TAG.equals(tagName)) {
                elementContainerMap.put(EXTERNAL_TAG, "");
            }
        } else if (isInStateTag()) {
            if (ENTRY_POINT_TAG.equals(tagName)) {
                logger.debug("Encountering entry point tag");
                elementContainerMap.put(ENTRY_POINT_TAG, "");
            } else if (EXIT_POINT_TAG.equals(tagName)) {
                logger.debug("Encountering exit point tag");
                elementContainerMap.put(EXIT_POINT_TAG, "");
            }
        }
    }

        /**
         * @see org.xml.sax.ContentHandler#endElement(java.lang.String, java.lang.String, java.lang.String)
         */
        public void endElement(String uri, String localName, String qName)
        throws SAXException {
         String tagName = localName;
         String charData = "";
//        if (logger.isDebugEnabled()) {
//            logger.debug("Ending an element, localName is " + localName);
//        }
```

```
//        logger.debug("   uri: " + uri);
//        logger.debug("   localName: " + localName);
//        logger.debug("   qName: " + qName);

        // get chardata from buffer, if any
        if ((charContainerName != null)
            && (charContainerName.equals(tagName))) {
            // get the contents
            charData = charDataBuffer.toString();

            // reset
            charContainerName = null;
        }

        // check for closing basic container tags
        if (ACTION_TAG.equals(tagName)) {
            setInActionTag(false);
            createActionComponent();
        } else if (STATE_TAG.equals(tagName)) {
            setInStateTag(false);
            createStateComponent();
        } else if (REFLECTION_TAG.equals(tagName)) {
            setInReflectionTag(false);
            createReflectionComponent();
        } else if (TRANSITION_TAG.equals(tagName)) {
            setInTransitionTag(false);
            createTransitionComponent();
        }

        // check for closing contained elements with content
        if (isInActionTag()) {
            if (ACTION_NAME_TAG.equals(tagName)) {
                elementContainerMap.put(ACTION_NAME_TAG, charData);
            } else if (DESCRIPTION_TAG.equals(tagName)) {
                elementContainerMap.put(DESCRIPTION_TAG, charData);
            } else if (TARGET_USER_METHOD_TAG.equals(tagName)) {
                elementContainerMap.put(TARGET_USER_METHOD_TAG, charData);
            }
        } else if (isInStateTag()) {
            if (STATE_CODE_TAG.equals(tagName)) {
                elementContainerMap.put(STATE_CODE_TAG, charData);
            } else if (APPROVAL_CODE_TAG.equals(tagName)) {
                elementContainerMap.put(APPROVAL_CODE_TAG, charData);
            } else if (DESCRIPTION_TAG.equals(tagName)) {
                elementContainerMap.put(DESCRIPTION_TAG, charData);
            } else if (OWNER_ROLE_NAME_TAG.equals(tagName)) {
                elementContainerMap.put(OWNER_ROLE_NAME_TAG, charData);
            } else if (DESIGNATED_ROLE_NAME_TAG.equals(tagName)) {
                elementContainerMap.put(DESIGNATED_ROLE_NAME_TAG, charData);
            }
        } else if (isInReflectionTag()) {
            if (STATE_CODE_TAG.equals(tagName)) {
                elementContainerMap.put(STATE_CODE_TAG, charData);
            } else if (ACTION_NAME_TAG.equals(tagName)) {
                elementContainerMap.put(ACTION_NAME_TAG, charData);
            }
        } else if (isInTransitionTag()) {
            if (SOURCE_STATE_CODE_TAG.equals(tagName)) {
                elementContainerMap.put(SOURCE_STATE_CODE_TAG, charData);
            } else if (ACTION_NAME_TAG.equals(tagName)) {
                elementContainerMap.put(ACTION_NAME_TAG, charData);
            } else if (TARGET_STATE_CODE_TAG.equals(tagName)) {
                elementContainerMap.put(TARGET_STATE_CODE_TAG, charData);
            }
        } else if (isInPropertiesTag()) {
            if (WORKFLOW_NAME_TAG.equals(tagName)) {
```

```
                logger.debug("workflow name is " + charData);
                workflowModel.setName(charData);
            } else if (WORKFLOW_VERSION_TAG.equals(tagName)) {
                logger.debug("workflow version is " + charData);
                workflowModel.setVersion(charData);
            } else if (DESCRIPTION_TAG.equals(tagName)) {
                workflowModel.setDescription(charData);
            }
        }
        }


        /**
     * Note that this method would normally maintain a stack of open
     * CDATA buffers, but we won't need that for the FSM workflow file.
     *
     * @see org.xml.sax.ContentHandler#characters(char[], int, int)
     */
    public void characters(char[] ch, int start, int length)
    throws SAXException {
        // only save it if we're inside a container we care about
        if (charContainerName != null) {
            charDataBuffer.append(ch, start, length);
        }
    }

    public void ignorableWhitespace(char[] ch, int start, int length)
    throws SAXException {
        // only save it if we're inside a container we care about
        if (charContainerName != null) {
            charDataBuffer.append(ch, start, length);
        }
    }

    public void endDocument()
    throws SAXException {
        setFinishedParsing(true);

        // construct the action enumerated type
        workflowModel.setActionEnumType(
            FSMActionEnumeratedType.createFSMStateType(
                workflowModel.getName() + " actions", actionList));

        // construct the state enumerated type
        workflowModel.setStateEnumType(
            FSMStateEnumeratedType.createFSMStateType(
                workflowModel.getName() + " states", stateList));

        // store the completed workflow in the workflow map
        workflowModelMap.put(workflowModel.getKey(), workflowModel);
    }

        /* (non-Javadoc)
         * @see org.xml.sax.ErrorHandler#error(org.xml.sax.SAXParseException)
         */
        public void error(SAXParseException e) throws SAXException {
                // TODO Auto-generated method stub
                super.error(e);
        }

        /* (non-Javadoc)
         * @see org.xml.sax.ErrorHandler#fatalError(org.xml.sax.SAXParseException)
         */
        public void fatalError(SAXParseException e) throws SAXException {
                // TODO Auto-generated method stub
                super.fatalError(e);
```

```
    }

private FSMState findState(String stateCode) {
    Iterator iter = stateList.iterator();
    while (iter.hasNext()) {
        FSMState state = (FSMState)iter.next();
        if (state.getStateCode().equals(stateCode)) {
            return state;
        }
    }
    return null;
}

private void createTransitionComponent() {
    try {
        // build transition in state from contents in map
        String sourceStateCode = (String)elementContainerMap.get(SOURCE_STATE_CODE_TAG);
        String actionName = (String)elementContainerMap.get(ACTION_NAME_TAG);
        String targetStateCode = (String)elementContainerMap.get(TARGET_STATE_CODE_TAG);
        FSMState state = findState(sourceStateCode);
        state.createTransition(actionName, targetStateCode);
    } catch (ClassCastException e) {
        //TODO throw a better exception here
    } catch (NullPointerException e) {
        //TODO throw a better exception here
    }
}

private void createReflectionComponent() {
    try {
        // build self-transition in state from contents in map
        String stateCode = (String)elementContainerMap.get(STATE_CODE_TAG);
        String actionName = (String)elementContainerMap.get(ACTION_NAME_TAG);
        FSMState state = findState(stateCode);
        state.createTransition(actionName, stateCode);
    } catch (ClassCastException e) {
        //TODO throw a better exception here
    } catch (NullPointerException e) {
        //TODO throw a better exception here
    }
}

private void createStateComponent() {
    try {
        // build state object from contents in map
        FSMState state = new FSMState();
        state.setStateCode((String)elementContainerMap.get(STATE_CODE_TAG));
        state.setApprovalCode((String)elementContainerMap.get(APPROVAL_CODE_TAG));
        state.setDescription((String)elementContainerMap.get(DESCRIPTION_TAG));
        state.setOwnerRoleName((String)elementContainerMap.get(OWNER_ROLE_NAME_TAG));
        state.setDesignatedRoleName((String)elementContainerMap.get(DESIGNATED_ROLE_NAME_TAG));
        state.setEntryPoint(elementContainerMap.containsKey(ENTRY_POINT_TAG));
        state.setExitPoint(elementContainerMap.containsKey(EXIT_POINT_TAG));

        // store it
        stateList.add(state);
    } catch (ClassCastException e) {
        //TODO throw a better exception here
    } catch (NullPointerException e) {
        //TODO throw a better exception here
    }
}

private void createActionComponent() {
    try {
        // build action object from contents in map
```

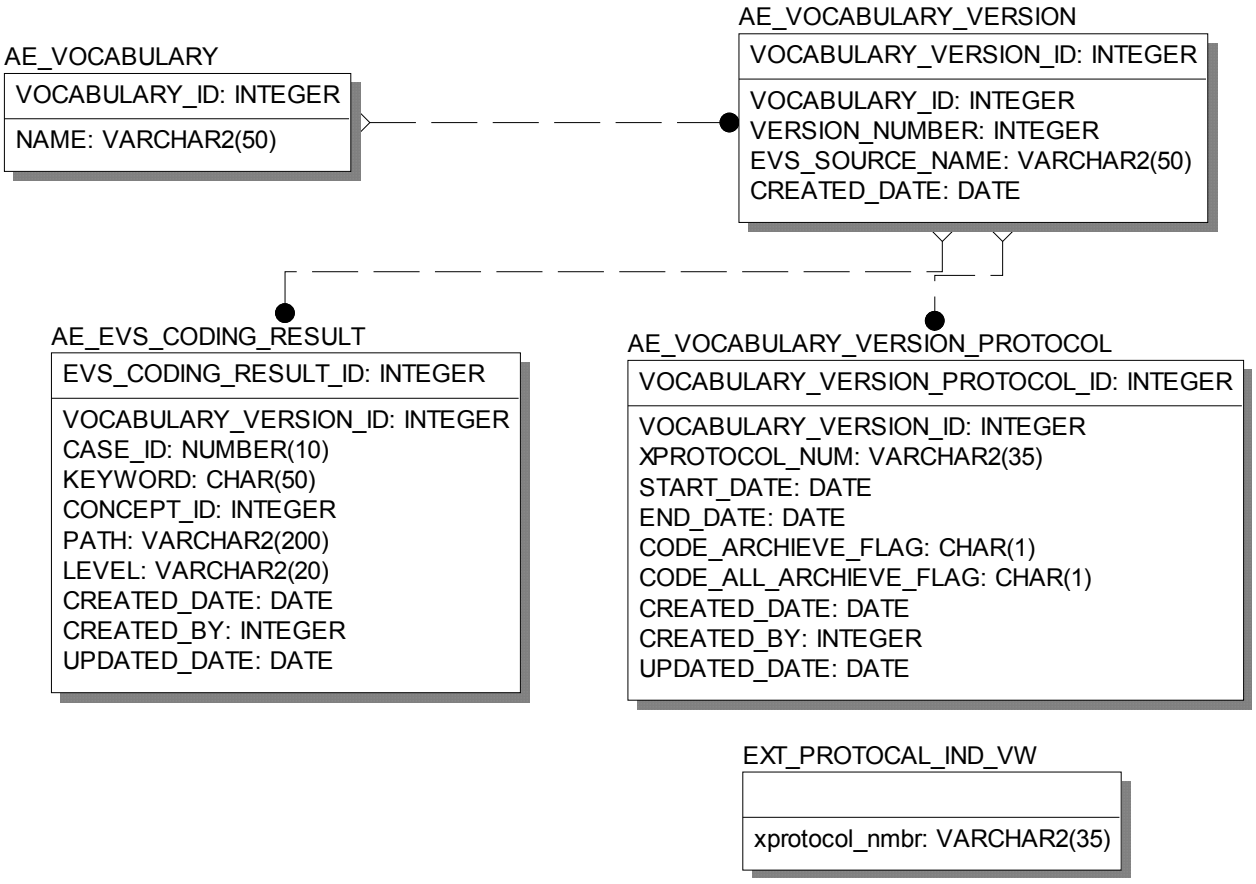
```
            FSMAction action = new FSMAction();
            action.setName((String)elementContainerMap.get(ACTION_NAME_TAG));
            action.setDescription((String)elementContainerMap.get(ACTION_NAME_TAG));
            action.setReflexive(elementContainerMap.containsKey(REFLEXIVE_TAG));
            action.setExternal(elementContainerMap.containsKey(EXTERNAL_TAG));
            if ("system".equals(elementContainerMap.get(TARGET_USER_METHOD_TAG))) {
                action.setTargetedBySystem(true);
            } else if ("user".equals(elementContainerMap.get(TARGET_USER_METHOD_TAG))) {
                action.setTargetedByUser(true);
            }

            // store it
            actionList.add(action);
        } catch (ClassCastException e) {
            //TODO throw a better exception here
        } catch (NullPointerException e) {
            //TODO throw a better exception here
        }
    }


    /**
     * Are we done parsing a document yet?
     *
         * @return
         */
        public boolean isFinishedParsing() {
                return finishedParsing;
        }

        /**
     * Are we done parsing a document yet?
         * @param b
         */
        private void setFinishedParsing(boolean b) {
                finishedParsing = b;
        }

        /**
         * @return
         */
        public boolean isInActionTag() {
                return inActionTag;
        }

        /**
         * @return
         */
        public boolean isInReflectionTag() {
                return inReflectionTag;
        }

        /**
         * @return
         */
        public boolean isInStateTag() {
                return inStateTag;
        }

        /**
         * @return
         */
        public boolean isInTransitionTag() {
                return inTransitionTag;
        }
```

```
        /**
         * @param b
         */
        public void setInActionTag(boolean b) {
                inActionTag = b;
        }

        /**
         * @param b
         */
        public void setInReflectionTag(boolean b) {
                inReflectionTag = b;
        }

        /**
         * @param b
         */
        public void setInStateTag(boolean b) {
                inStateTag = b;
        }

        /**
         * @param b
         */
        public void setInTransitionTag(boolean b) {
                inTransitionTag = b;
        }

        /**
         * @return
         */
        public boolean isInPropertiesTag() {
                return inPropertiesTag;
        }

        /**
         * @param b
         */
        public void setInPropertiesTag(boolean b) {
                inPropertiesTag = b;
        }

        /**
     * Returns an unmodifiable map of all the workflow
     * models parsed by this handler instance.
     *
         * @return a map of workflow names to FSMWorkflowModel instances
         */
        public Map getWorkflowModelMap() {
                return Collections.unmodifiableMap(workflowModelMap);
        }

}
```

APPENDIX: D

Documented component description of the underlying data structures.

The following example illustrates the Object Model for the CSAERS Medical Coding Module. This shows the underlying data structures that are part of the overall CSAERS schema.



APPENDIX: E

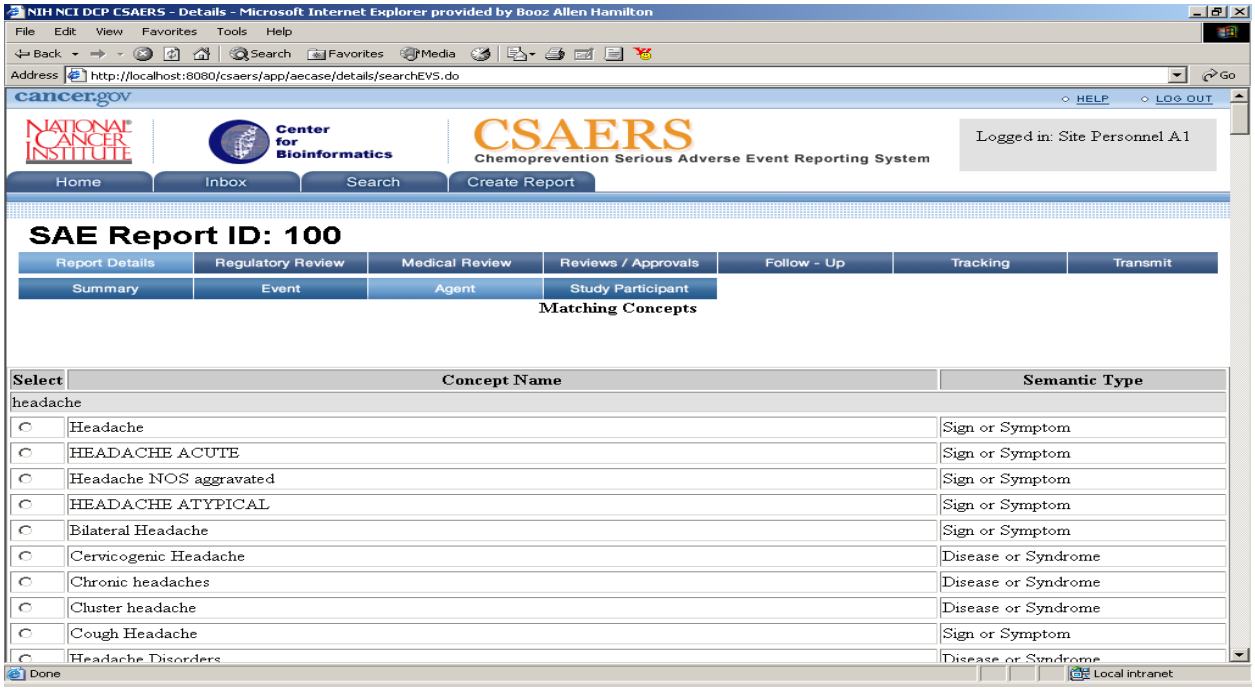
Use of public standardized controlled vocabularies.

In this example a list of concepts from EVS MedDRA is obtained for each verbatim term.

```
/*
 * Chemoprevention Serious Adverse Event Reporting System
 * Division of Cancer Prevention
 * National Cancer Institute
 *
 * Created by Booz Allen Hamilton
 * Created $date$
 *
 */

package gov.nih.nci.csaers.model.facade.ejb.aecasecodingfacade;

import gov.nih.nci.EVS.bean.Concept;
import gov.nih.nci.EVS.bean.MetaThesaurusConcept;
import gov.nih.nci.EVS.bean.Source;
import gov.nih.nci.EVS.search.MetaThesaurusConceptSearchCriteria;
import gov.nih.nci.csaers.model.vo.ConceptVO;
import gov.nih.nci.csaers.model.vo.KeywordSearchVO;
import gov.nih.nci.csaers.model.vo.MatchedKeywordConceptsVO;
import gov.nih.nci.csaers.model.vo.VocabularyVO;

import org.apache.log4j.Logger;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class MetaPhrasAdapter implements EVSAdapter {
    private static final Logger logger =
        Logger.getLogger(MetaPhrasAdapter.class);

    public List findMatchingConcepts(KeywordSearchVO vo) {
        List matchings = new ArrayList();

        Iterator it = vo.getKeywords().iterator();

        while(it.hasNext()) {
            String keyword = (String) it.next();
            MatchedKeywordConceptsVO matchedVO =
                getConcepts(keyword, vo.getVocabulary());
            matchings.add(matchedVO);
```

```
        }
        return matchings;
    }
    public ConceptVO getConceptDetails(String ConceptUID) {
        // TODO Auto-generated method stub
        return null;
    }

    private MatchedKeywordConceptsVO getConcepts(
        String keyword, VocabularyVO vocabularyVO) {
        MetaThesaurusConceptSearchCriteria mtcsc =
            new MetaThesaurusConceptSearchCriteria();
        MetaThesaurusConcept mtc = new MetaThesaurusConcept();
        MatchedKeywordConceptsVO matchedVO = new MatchedKeywordConceptsVO();
        matchedVO.setKeyword(keyword);
        mtcsc.setSearchTerm(keyword);
            logger.debug("Keyword:" + keyword);
        Source src = new Source(); //Set source in the search criteria
        src.setAbbreviation(vocabularyVO.getSourceName());
        mtcsc.setSource(src);
            mtcsc.setShortResult(false);
            mtcsc.setLimit(200);

        //call search
        Concept[] conceptArray = mtc.search(mtcsc);

        if(conceptArray != null) {
            for(int i = 0; i < conceptArray.length; i++) {
                ConceptVO conceptVO = new ConceptVO();
                conceptVO.setSource(vocabularyVO.getSourceName());
                conceptVO.setName(conceptArray[i].getName());

                        logger.debug("Concept Name:" + conceptArray[i].getName());
                StringBuffer semanticTypes = new StringBuffer();

                for(int j = 0; j < conceptArray[i].getSemanticTypeCount();
                        j++) {
                    semanticTypes.append(
                        conceptArray[i].getSemanticTypes()[j].getName() + ",");
                }

                // take out the last , for semanticTypes
                String semantics = semanticTypes.toString();
                semantics = semantics.substring(0, semantics.length() - 1);
                conceptVO.setSemanticType(semantics);
                matchedVO.addToList(conceptVO);
            }
        }

        return matchedVO;
    }
}
```

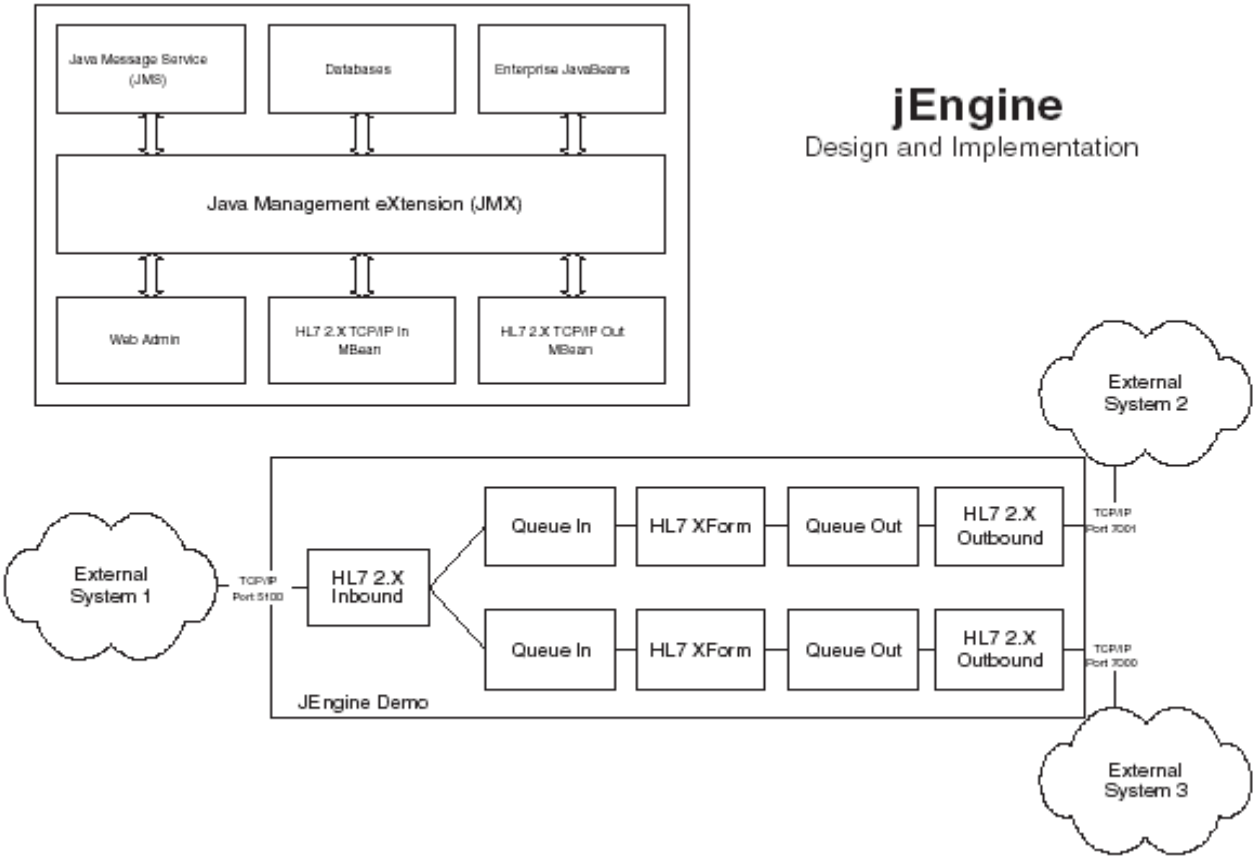
The following screen shot shows a list of EVS MedDRA concepts for verbatim term “Headache”. The code above is only part of the middle-tier classes that deal with Medical Coding.



APPENDIX: F

Messages need to be based on a common messaging model.

The following diagram illustrates the Design of jEngine. Here the messaging model used is HL7.



APPENDIX: G

Use standard electronic representation for CDE's

In the following example from the CSAERS application, various CDEs for the elements on the form are being retrieved and made available to the JSP's to be displayed.

```
package gov.nih.nci.csaers.web.util;

import gov.nih.nci.csaers.common.cde.*;
import gov.nih.nci.csaers.common.logging.*;

/**
 * Form bean used to contain CDEs
 */
public class CDEForm extends MappedActionForm {

    private static final AppLogger logger =
        AppLoggerImpl.getAppLogger(
            CDEForm.class);
       private int cdeImplementationSetId;

    public CDEForm() {
        initUsingCachedValues();
    }

       public int getCdeImplementationSetId() {
               return cdeImplementationSetId;
       }

       public void setCdeImplementationSetId(int aCdeImplementationSetId) {
               cdeImplementationSetId = aCdeImplementationSetId;
       }

    private void initUsingCachedValues() {

        logger.entry("initUsingCachedValues()");

        CDECache cdeCache = CDECache.getInstance(new Integer(100));
        CDEVO cde = null;

        cde = cdeCache.findByImplementationSetSubscriber(
            "address");

        this.setProperty("customerEditForm_address", cde);

        cde = cdeCache.findByImplementationSetSubscriber(
            "state");

        this.setProperty("customerEditForm_state", cde);

        this.setProperty("multiState", cde);

        // Loading CDEs based on ScreenFormKeys

        if (logger.isDebugEnabled()){
                       logger.debug("Loading ScreenFormKeyCDEs");
        }

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.COMMON__PROTOCOL_NMBR);
        this.setProperty(ScreenFormKeys.COMMON__PROTOCOL_NMBR, cde);
```

```
cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__STUDY_PARTICIPANT_NMBR);
this.setProperty(ScreenFormKeys.COMMON__STUDY_PARTICIPANT_NMBR, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__SALUTATION_CODE);
this.setProperty(ScreenFormKeys.COMMON__SALUTATION_CODE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__FIRST_NAME);
this.setProperty(ScreenFormKeys.COMMON__FIRST_NAME, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__MIDDLE_NAME);
this.setProperty(ScreenFormKeys.COMMON__MIDDLE_NAME, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__LAST_NAME);
this.setProperty(ScreenFormKeys.COMMON__LAST_NAME, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__NAME);
this.setProperty(ScreenFormKeys.COMMON__NAME, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__ADDRESS1);
this.setProperty(ScreenFormKeys.COMMON__ADDRESS1, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__ADDRESS2);
this.setProperty(ScreenFormKeys.COMMON__ADDRESS2, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__CITY);
this.setProperty(ScreenFormKeys.COMMON__CITY, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__STATE);
this.setProperty(ScreenFormKeys.COMMON__STATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__ZIP);
this.setProperty(ScreenFormKeys.COMMON__ZIP, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__COUNTRY);
this.setProperty(ScreenFormKeys.COMMON__COUNTRY, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__TELEPHONE);
this.setProperty(ScreenFormKeys.COMMON__TELEPHONE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__FAX);
this.setProperty(ScreenFormKeys.COMMON__FAX, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__EMAIL);
this.setProperty(ScreenFormKeys.COMMON__EMAIL, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__ORGANIZATION);
this.setProperty(ScreenFormKeys.COMMON__ORGANIZATION, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.COMMON__OCCUPATION);
```

```
        this.setProperty(ScreenFormKeys.COMMON__OCCUPATION, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.COMMON__CREATED_DATE);
        this.setProperty(ScreenFormKeys.COMMON__CREATED_DATE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.COMMON__FDA_REPORT_DATE);
        this.setProperty(ScreenFormKeys.COMMON__FDA_REPORT_DATE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.COMMON__AGENT_NAME);
        this.setProperty(ScreenFormKeys.COMMON__AGENT_NAME, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.SUMMARY__PROTOCOL_NAME);
        this.setProperty(ScreenFormKeys.SUMMARY__PROTOCOL_NAME, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.SUMMARY__IND_SPONSOR);
        this.setProperty(ScreenFormKeys.SUMMARY__IND_SPONSOR, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.SUMMARY__IND_NMBR);
        this.setProperty(ScreenFormKeys.SUMMARY__IND_NMBR, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.SUMMARY__STUDY_DESIGN);
        this.setProperty(ScreenFormKeys.SUMMARY__STUDY_DESIGN, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__EVENT_STATUS_TYPE);
        this.setProperty(ScreenFormKeys.EVENT__EVENT_STATUS_TYPE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__AWARENESS_DATE);
        this.setProperty(ScreenFormKeys.EVENT__AWARENESS_DATE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__ONSET_DATE);
        this.setProperty(ScreenFormKeys.EVENT__ONSET_DATE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__ONSET_TIME);
        this.setProperty(ScreenFormKeys.EVENT__ONSET_TIME, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__LOCATION);
        this.setProperty(ScreenFormKeys.EVENT__LOCATION, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__SPONSOR_NTFCTN_RESPONSE);
        this.setProperty(ScreenFormKeys.EVENT__SPONSOR_NTFCTN_RESPONSE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__SPONSOR_NTFCTN_DATE);
        this.setProperty(ScreenFormKeys.EVENT__SPONSOR_NTFCTN_DATE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__SPONSOR_NTFCTN_METHOD);
        this.setProperty(ScreenFormKeys.EVENT__SPONSOR_NTFCTN_METHOD, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.EVENT__IRB_NTFCTN_RESPONSE);
        this.setProperty(ScreenFormKeys.EVENT__IRB_NTFCTN_RESPONSE, cde);
```

```
cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__BLIND_BROKEN_RESPONSE);
this.setProperty(ScreenFormKeys.EVENT__BLIND_BROKEN_RESPONSE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__EVENT_DESC);
this.setProperty(ScreenFormKeys.EVENT__EVENT_DESC, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__TREATMENT_DATE);
this.setProperty(ScreenFormKeys.EVENT__TREATMENT_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__TREATMENT_DESC);
this.setProperty(ScreenFormKeys.EVENT__TREATMENT_DESC, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__INITIAL_REPORTER);
this.setProperty(ScreenFormKeys.EVENT__INITIAL_REPORTER, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__HEALTH_PROFESSIONAL);
this.setProperty(ScreenFormKeys.EVENT__HEALTH_PROFESSIONAL, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__WHY_SERIOUS);
this.setProperty(ScreenFormKeys.EVENT__WHY_SERIOUS, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__ATTRIBUTION);
this.setProperty(ScreenFormKeys.EVENT__ATTRIBUTION, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__EXPECTEDNESS);
this.setProperty(ScreenFormKeys.EVENT__EXPECTEDNESS, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__GRADE);
this.setProperty(ScreenFormKeys.EVENT__GRADE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__OUTCOME);
this.setProperty(ScreenFormKeys.EVENT__OUTCOME, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__RESOLVED_DATE);
this.setProperty(ScreenFormKeys.EVENT__RESOLVED_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__FATAL_DATE);
this.setProperty(ScreenFormKeys.EVENT__FATAL_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__CAUSE_OF_DEATH);
this.setProperty(ScreenFormKeys.EVENT__CAUSE_OF_DEATH, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__AUTOPSY_DATE);
this.setProperty(ScreenFormKeys.EVENT__AUTOPSY_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.EVENT__AUTOPSY_PERFORMED);
this.setProperty(ScreenFormKeys.EVENT__AUTOPSY_PERFORMED, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.AGENT__START_DATE);
```

```
        this.setProperty(ScreenFormKeys.AGENT__START_DATE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__UNDER_INVESTIGATION);
        this.setProperty(ScreenFormKeys.AGENT__UNDER_INVESTIGATION, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__FORMULATION);
        this.setProperty(ScreenFormKeys.AGENT__FORMULATION, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__LOT_NMBR);
        this.setProperty(ScreenFormKeys.AGENT__LOT_NMBR, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__PLND_AMT);
        this.setProperty(ScreenFormKeys.AGENT__PLND_AMT, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__PLND_UNIT);
        this.setProperty(ScreenFormKeys.AGENT__PLND_UNIT, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__PLND_FREQUENCY);
        this.setProperty(ScreenFormKeys.AGENT__PLND_FREQUENCY, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__PLND_ROUTE);
        this.setProperty(ScreenFormKeys.AGENT__PLND_ROUTE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__ACT_AMT);
        this.setProperty(ScreenFormKeys.AGENT__ACT_AMT, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__ACT_UNIT);
        this.setProperty(ScreenFormKeys.AGENT__ACT_UNIT, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__ACT_FREQUENCY);
        this.setProperty(ScreenFormKeys.AGENT__ACT_FREQUENCY, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__ACT_ROUTE);
        this.setProperty(ScreenFormKeys.AGENT__ACT_ROUTE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__INTERRUPTED);
        this.setProperty(ScreenFormKeys.AGENT__INTERRUPTED, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__DECHALLENGE);
        this.setProperty(ScreenFormKeys.AGENT__DECHALLENGE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__RECHALLENGE);
        this.setProperty(ScreenFormKeys.AGENT__RECHALLENGE, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__CONTINUED);
        this.setProperty(ScreenFormKeys.AGENT__CONTINUED, cde);

        cde = cdeCache.findByImplementationSetSubscriber(
               ScreenFormKeys.AGENT__LAST_TAKEN_DATE);
        this.setProperty(ScreenFormKeys.AGENT__LAST_TAKEN_DATE, cde);
```

```
cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.AGENT__MOD_DATE);
this.setProperty(ScreenFormKeys.AGENT__MOD_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.AGENT__MOD_AMT);
this.setProperty(ScreenFormKeys.AGENT__MOD_AMT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.AGENT__MOD_UNIT);
this.setProperty(ScreenFormKeys.AGENT__MOD_UNIT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.AGENT__DURATION_NTAKEN);
this.setProperty(ScreenFormKeys.AGENT__DURATION_NTAKEN, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__YOB);
this.setProperty(ScreenFormKeys.PARTICIPANT__YOB, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__GENDER);
this.setProperty(ScreenFormKeys.PARTICIPANT__GENDER, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__WEIGHT);
this.setProperty(ScreenFormKeys.PARTICIPANT__WEIGHT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__WEIGHT_UNIT);
this.setProperty(ScreenFormKeys.PARTICIPANT__WEIGHT_UNIT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__HEIGHT);
this.setProperty(ScreenFormKeys.PARTICIPANT__HEIGHT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__HEIGHT_UNIT);
this.setProperty(ScreenFormKeys.PARTICIPANT__HEIGHT_UNIT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__MED_HST_DATE);
this.setProperty(ScreenFormKeys.PARTICIPANT__MED_HST_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__MED_HST_TXT);
this.setProperty(ScreenFormKeys.PARTICIPANT__MED_HST_TXT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__TST_TAKEN);
this.setProperty(ScreenFormKeys.PARTICIPANT__TST_TAKEN, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__TST_DATE);
this.setProperty(ScreenFormKeys.PARTICIPANT__TST_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__TST_NAME);
this.setProperty(ScreenFormKeys.PARTICIPANT__TST_NAME, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__TST_UNIT);
this.setProperty(ScreenFormKeys.PARTICIPANT__TST_UNIT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__TST_RESULT);
```

```
this.setProperty(ScreenFormKeys.PARTICIPANT__TST_RESULT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__TST_NORM_HIGH);
this.setProperty(ScreenFormKeys.PARTICIPANT__TST_NORM_HIGH, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__TST_NORM_LOW);
this.setProperty(ScreenFormKeys.PARTICIPANT__TST_NORM_LOW, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_RESPONSE);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_RESPONSE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_AGENT);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_AGENT, cde);


cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_AMT);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_AMT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_UNIT);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_UNIT, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_ROUTE);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_ROUTE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_FREQUENCY);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_FREQUENCY, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_START_DATE);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_START_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_END_DATE);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_END_DATE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_CONTINUING);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_CONTINUING, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.PARTICIPANT__CONCOMITANT_IND_USE);
this.setProperty(ScreenFormKeys.PARTICIPANT__CONCOMITANT_IND_USE, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.RR__WHY_SERIOUS);
this.setProperty(ScreenFormKeys.RR__WHY_SERIOUS, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.RR__ATTRIBUTION);
this.setProperty(ScreenFormKeys.RR__ATTRIBUTION, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.RR__EXPECTEDNESS);
this.setProperty(ScreenFormKeys.RR__EXPECTEDNESS, cde);

cde = cdeCache.findByImplementationSetSubscriber(
       ScreenFormKeys.RR__COMMENT);
this.setProperty(ScreenFormKeys.RR__COMMENT, cde);
```

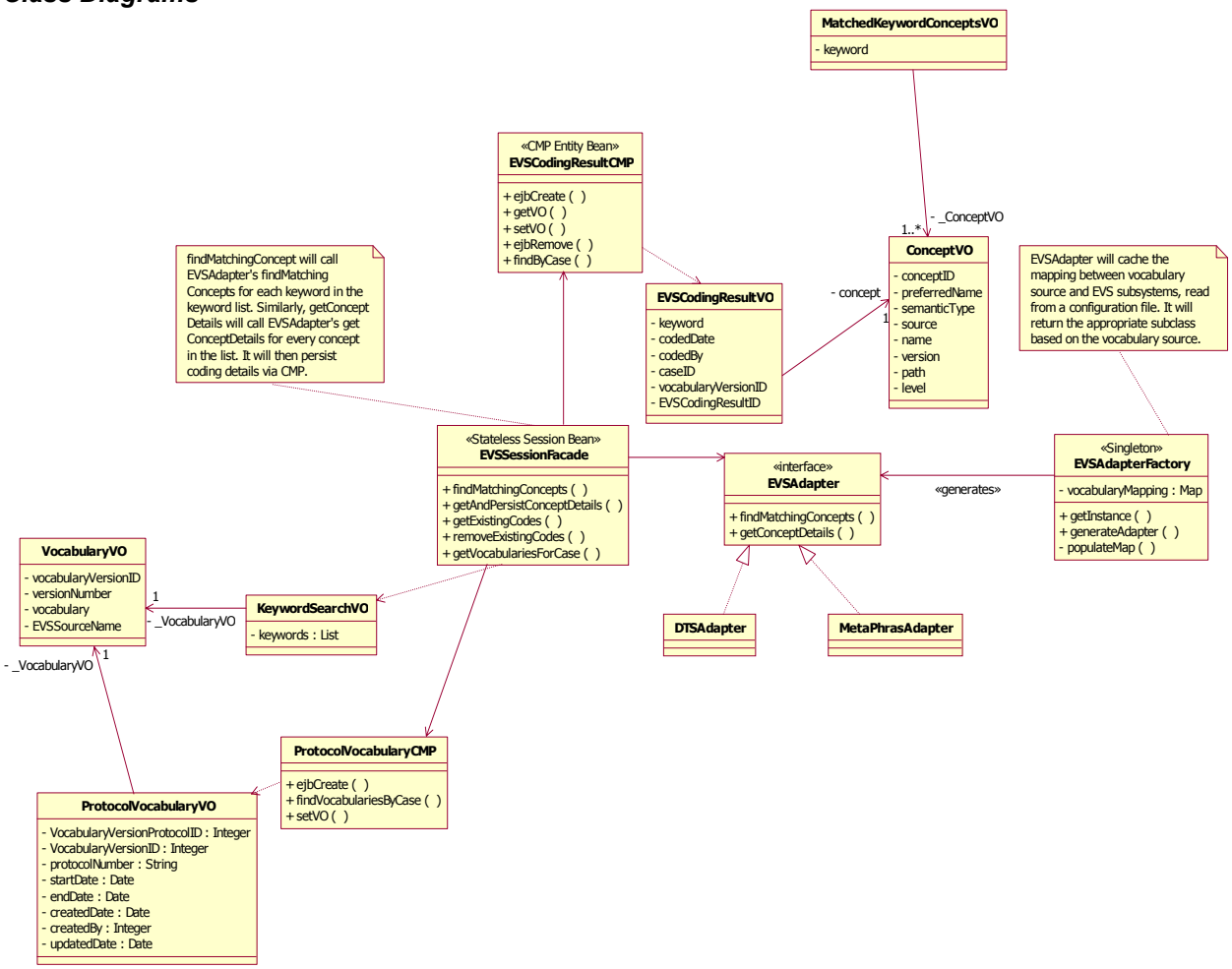
```
                cde = cdeCache.findByImplementationSetSubscriber(
                        ScreenFormKeys.MM__WHY_SERIOUS);
                this.setProperty(ScreenFormKeys.MM__WHY_SERIOUS, cde);

                cde = cdeCache.findByImplementationSetSubscriber(
                        ScreenFormKeys.MM__ATTRIBUTION);
                this.setProperty(ScreenFormKeys.MM__ATTRIBUTION, cde);

                cde = cdeCache.findByImplementationSetSubscriber(
                        ScreenFormKeys.MM__EXPECTEDNESS);
                this.setProperty(ScreenFormKeys.MM__EXPECTEDNESS, cde);

                cde = cdeCache.findByImplementationSetSubscriber(
                        ScreenFormKeys.MM__COMMENT);
                this.setProperty(ScreenFormKeys.MM__COMMENT, cde);

                cde = cdeCache.findByImplementationSetSubscriber(
                        ScreenFormKeys.MM__FDA_REPORT_DUE);
                this.setProperty(ScreenFormKeys.MM__FDA_REPORT_DUE, cde);

                cde = cdeCache.findByImplementationSetSubscriber(
                        ScreenFormKeys.MM__FDA_REPORT_DUE_EXPL);
                this.setProperty(ScreenFormKeys.MM__FDA_REPORT_DUE_EXPL, cde);

        logger.exit("initUsingCachedValues()");
    }
}
```

APPENDIX: H

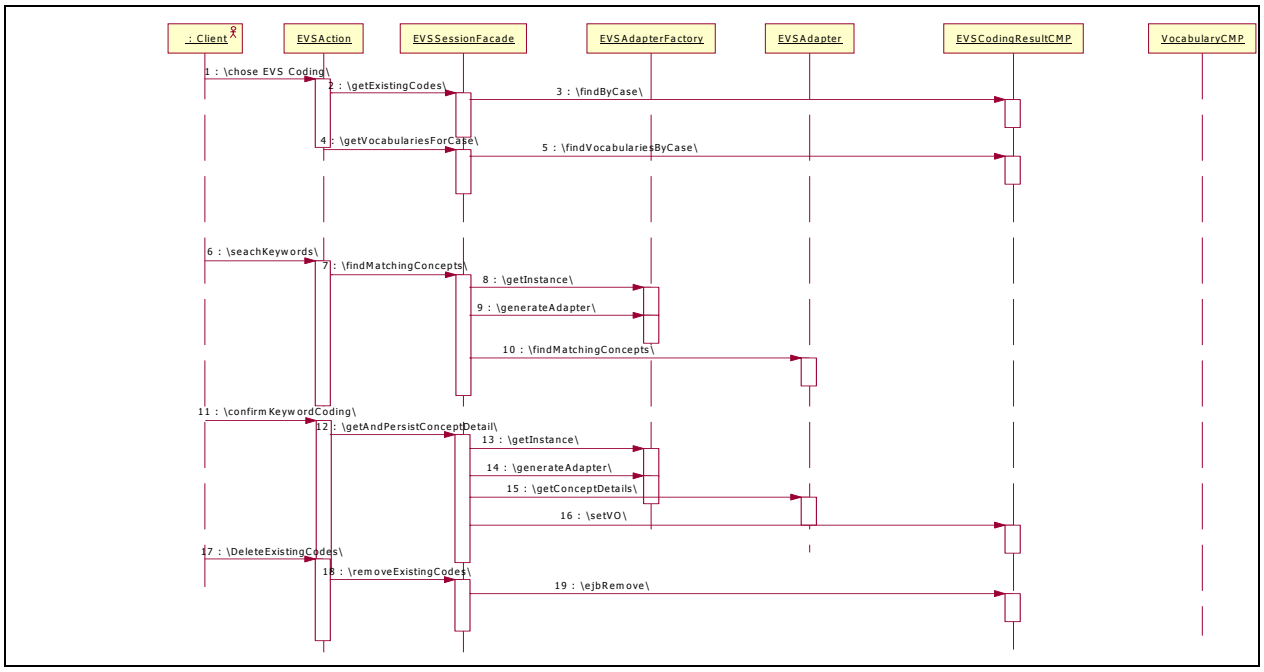
Information Models are defined in a standard modeling language such as UML.

The following examples of the Class Diagrams & the Sequence Diagrams are from the CSAERS Medical Coding Module.

Class Diagrams



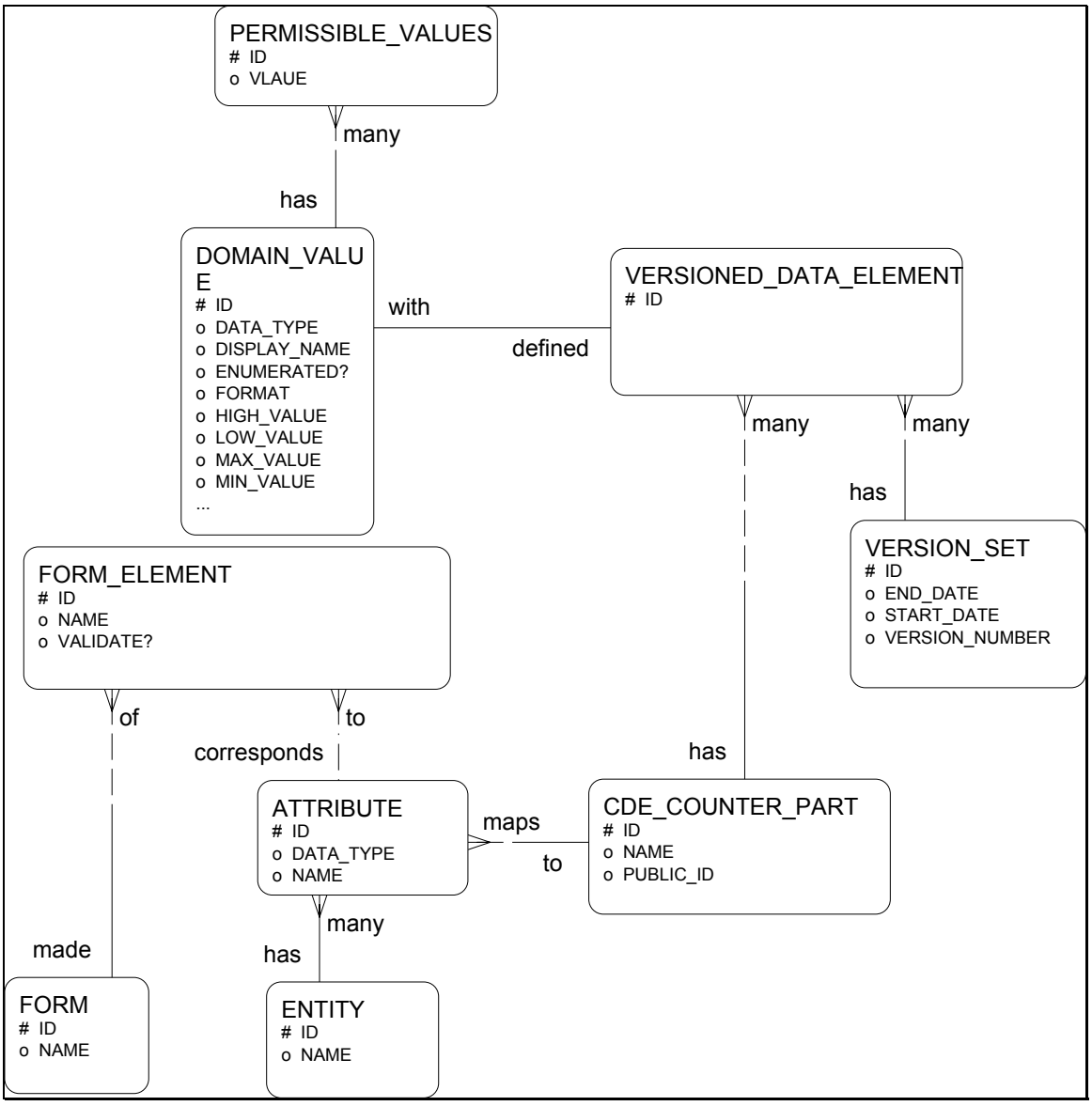
Sequence Diagrams



APPENDIX: I

Entity Relationship Diagram

The following Entity Relationship diagram is from “The Cancer Clinical Central Participant Registry” (C3PR) – CDE Integration Design Document.



Appendix: J

CaBIG Compatibility Score Card Example

The caBIG Compatibility Score Card similar to the one shown below would be used to map the degree to which a given application is caBIG compliant.

caBIG Compatibility Score Card — **Application Name:** *CSAERS (Cancer Serious Adverse Event Reporting System)*
Date: — **Assessment Performed by:**

Maturity Model	Basic (Legacy)	Bronze		Silver		Gold	
Interface Integration	- No Programming interfaces to the system are available. Only local data files in a custom format can be read	- Provide baseline* programmatic access to data. Data can be read from remote electronic sources or from commonly used file formats Data can be pushed out...		- Well-described API's that provide access to data objects.	✓	- All features of Silver, plus	✓
				- System architecture separated into tiers and interoperable components	✓	- Interoperable with data grid architecture to be defined by caBIG	
				- Data read in from standards-based electronic sources that support standard or commonly used interchange formats	✓	- Fully componentized provide access to individual resources in the form of grid services	
				- Documented component description of the underlying data structures that are accessible	✓		
				- Standard messaging systems where appropriate			
Vocabularies / Terminologies & Ontologies	- Free text used throughout for data collection	- Use of public standardized controlled vocabularies as well as local terminologies				- Fully compliant with caBIG recommended standards for vocabulary terminology services and content sources	
				- Messages need to be based on a common messaging model (such as HL7)			
Data Elements	- No Structured metadata is recorded	- Some type of metadata describing the information in the system is recorded in electronic format of some kind				- Programmatic access to all metadata, including data class descriptions, site and source information, and any other caBIG-defined metadata requirements and use information models	
				- Use common standard electronic representation for CDE's such as ISO 11179 or comparable standard	✓	- Use the caBIG standard or electronic representation of metadata and Common Data Elements	✓
				- CDEs are harmonized and re-used from across the Domain Workspace	✓		
Information Models	- No particular information model is used to represent data classes	- Some type of diagrammatic model describing the data relationship is available in electronic format		- Information models defined in a standard modeling language such as UML	✓	- All features of Silver, plus	✓
						- Information models are harmonized with others across the caBIG Domain Workspace	

* Baseline includes what data is needed to be shared with other apps in order to eliminate duplicate entry, manual processing.