Design Documents - caAERS

Design Documents that were created prior to use of this wiki are available here: https://ncisvn.nci.nih.gov/svn/caaersappdev/docs/DesignDocs

Active-Inactive status on People and Organization

This document briefly describe how the status of people and organizations are managed. The following objects will go through activation and deactivation.

1. StudyPersonel
2. StudyInvestigator
3. StudyOrganzation (includes StudySite, CoordinatingCenter and FundingSponsor)
4. Investigator
5. ResearchStaff

How it works?

The above mentioned objects will have the following attributes

```
termStartDate:java.util.Date
termEndDate:java.util.Date

termStartDate cannot be null, termEndDate could be null.
```

How active or inactive is determined?

- Active if, termStartDate <= TODAY and termEndDate is empty or termEndDate >=TODAY
- Inactive, if NOT active

```
@Transient
    public boolean isActive(){
     return (termStartDate != null && DateUtils.between(new Date(), termStartDate, termEndDate));
    }


    @Transient
    public boolean isInActive(){
     return (termStartDate == null || !DateUtils.between(new Date(), termStartDate, termEndDate));
    }
```

How cascade works?

- Whenever ResearchStaff, Investigator is saved/modified, the term start and end dates are cascaded to respective associations.

For example take the case of ResearchStaff

Entry point : gov.nih.nci.cabig.caaers.domain.repository.ResearchStaffRepository.**save**(ResearchStaff, String)

```
//ResearchStaff.java
public void cascadeTenure(){
    for(StudyPersonnel sp : getStudyPersonnels()){
     sp.setTermStartDate(this.termStartDate);
     sp.setTermEndDate(this.termEndDate);
    }
}
```

Archived Design Pages - caAERS

2008 AdEERS-caAERS Information

You need flash player installed to preview ppt and pdf files

2008 Development Guidelines

Contents

To create a page with a table of contents, add your headings and text. **Be sure to keep the column and section codes below at the end of your page.** Enter the maxLevel equal to the highest numbered heading you want to see in the table of contents. Delete this instruction and any extra headings.

Guidelines of using Extremecomponents

Important things to remember

1. Make sure formName and tableId are not same. For ex :

```
Table table = model.getTableInstance();
        table.setForm("ajaxTable");
        table.setTableId("ajaxTable"); //now pagination will not work becz tableid and formId are same
```

pagination will not work if tableId and formId are same
2. make sure tableId does not have any nested name (i.e. no '.' are allowed). For ex :

```
table.setTableId("aeReport.diseaseHistory.codedPrimaryDiseaseSite"); //now pagination will not work
becz tableid has .
```

First heading two

Some level two text.

Second heading two

Second level two text.

Second and last heading one

Second and last level one text.

2008 Drools upgrade notes

What is new in Drools 4.0

Language Expressiveness Enhancements

- New Conditional Elements: from, collect, accumulate and forall
- New Field Constraint operators: not matches, not contains, in, not in, memberOf, not memberOf
- New Implicit Self Reference field: this
- Full support for Conditional Elements nesting, for First Order Logic completeness.
- Support for multi-restrictions and constraint connectives && and ||
- Parser improvements to remove previous language limitations, like character escaping and keyword conflicts
- Support for pluggable dialects and full support for MVEL scripting language
- Complete rewrite of DSL engine, allowing for full l10n
- Fact attributes auto-vivification for return value restrictions and inline-eval constraints
- Support for nested accessors, property navigation and simplified collection, arrays and maps syntax
- Improved support for XML rules

Core Engine Enhancements

- Native support for primitive types, avoiding constant autoboxing
- Support for transparent optional Shadow Facts
- Rete Network performance improvements for complex rules
- Support for Rule-Flows
- Support for Stateful and Stateless working memories (rule engine sessions)
- Support for Asynchronous Working Memory actions
- Rules Engine Agent for hot deployment and BRMS integration
- Dynamic salience for rules conflict resolution
- Support for Parameterized Queries
- Support for halt command
- Support for sequential execution mode
- Support for pluggable global variable resolver

SRINI NOTES

Drools support JSR 94 which provides a vendor-neutral Java interface to access a rule engine. Rule language itself is not vendor-neutral. Rule language is drools specific and doesn't comply with any

generic standards (I don't see any standards available)

There are few API changes in 4.0 . Even rules XSD is also changed .

Basic steps involved in version upgrade.

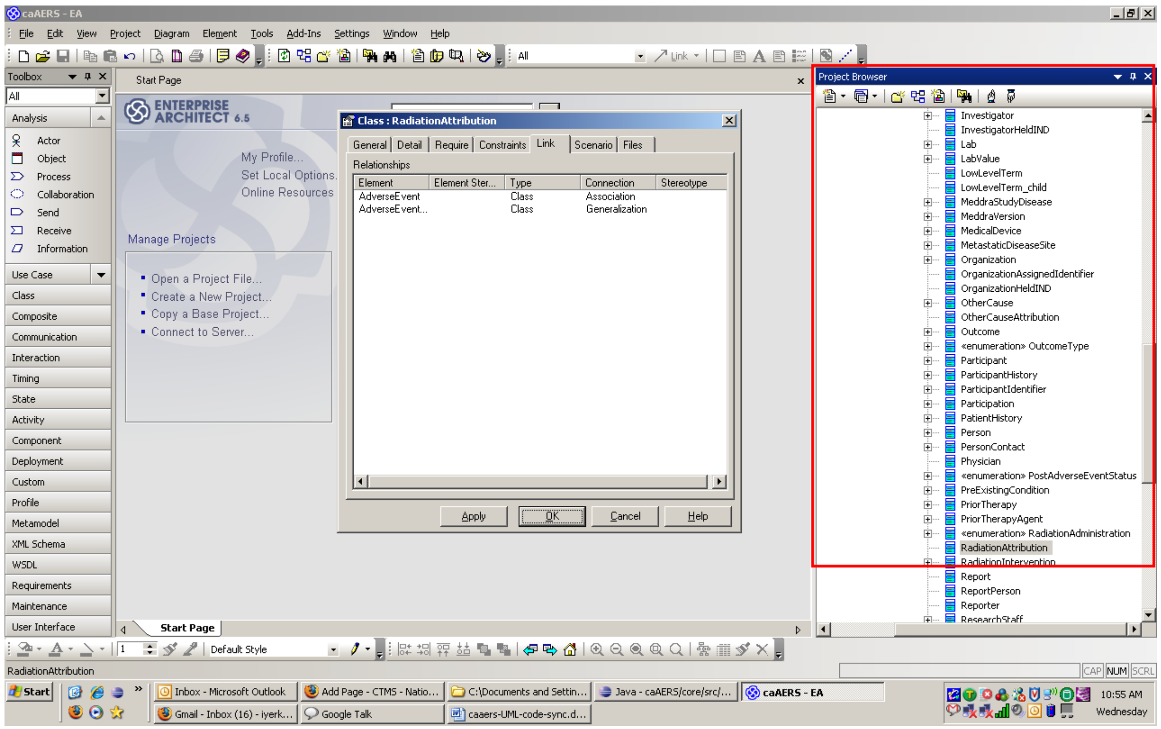
1. Change code with new APIs.

2. Create new xsl to transform to new XML .

XSD has major changes in 4.0 release. I am researching on the changes and effort to transform to new xsd.

2008 Guidelines for creating or adding domain objects to UML model

1. When adding a new domain object, ensure that such an object does not already exist in EA.
    - Do not check for a domain object in a diagram. Rather browse the list of domain objects through the Project browser

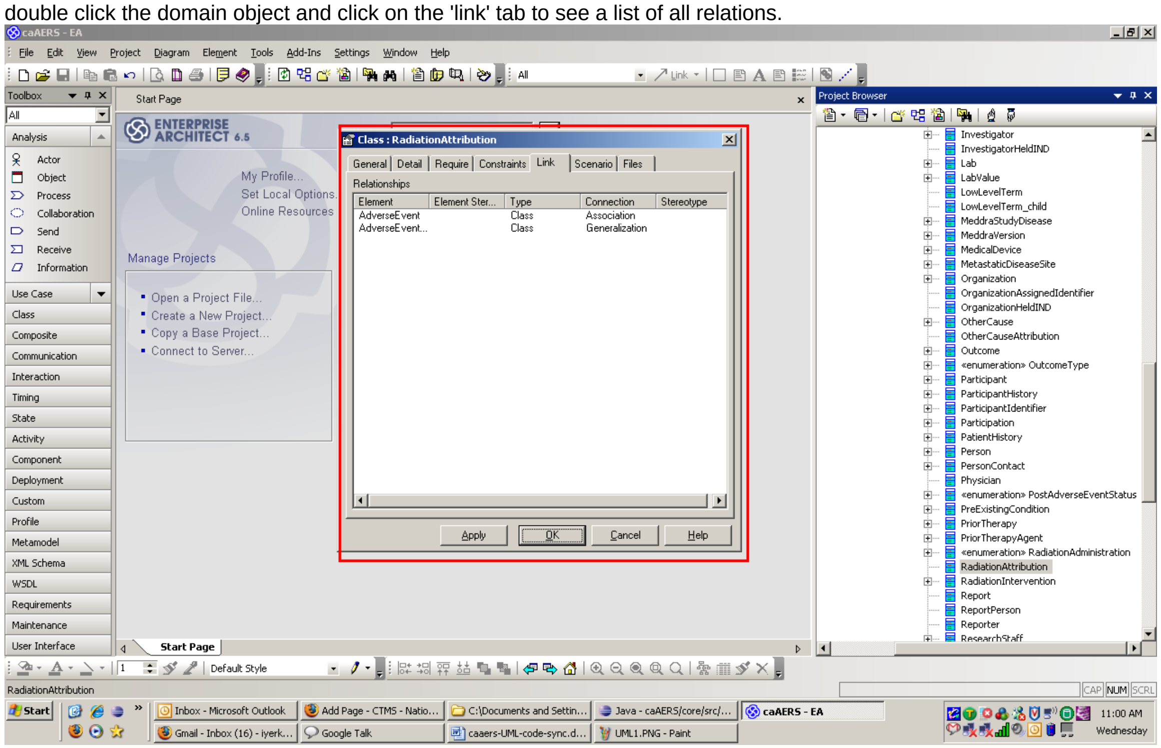

Class : RadiationAttribution
General | Detail | Require | Constraints | Link | Scenario | Files
Project Browser
Apply | OK | Cancel | Help

Relationships

Element	Element Ster...	Type	Connection	Stereotype
AdverseEvent		Class	Association	
AdverseEvent...		Class	Generalization	

-

2. When adding a new association between any two domain objects ensure that the relationship does not already exist in EA.
   - Do not check for the relationship in a diagram. Rather browse the list of domain objects through the Project browser. Then double click the domain object and click on the 'link' tab to see a list of all relations.


Class : RadiationAttribution
General | Detail | Require | Constraints | Link | Scenario | Files
Project Browser
Apply | OK | Cancel | Help

Relationships

Element	Element Ster...	Type	Connection	Stereotype
AdverseEvent		Class	Association	
AdverseEvent...		Class	Generalization	

-

3.
4. When adding a new relationship add the following information:
   a. Directionality: Should be one of : Bi-directional, Source -> destination, Destination -> source.
   b. Multiplicity: This should be mentioned for both ends of the relationship, irrespective of whether it is uni-directional or bi-directional.
   c. Rolenames: Rolename should be specified for the target end of the relationship [CTMS:if relation is uni-directional] or both ends of the relationship [CTMS:if relation is bi-directional]

2008 How to Install Globus

Downloading GLOBUS and WS-Core

download ws-core from following site

http://www-unix.globus.org/ftppub/gt4/4.0/4.0.3/ws-core/bin/ws-core-4.0.3-bin.zip

execute following command from ws-core directory to install globus to your tomcat.

```
ant -f share/globus_wsrf_common/tomcat/tomcat.xml deployTomcat -Dtomcat.dir=tomcat_home

where tomcat_home=it would be the directory where tomcat is installed.
```

2008 IE Notes

This document explains couple of notable issue
- short cut notation of closing the tags not allowed (eg: <span /> is wrong, <span></span> is correct)

- Ajax rendering issues: -

  - <a name="abcd">, if present in ajax response, there was display issues.

  - comments (HTML comments) in list editor was causing some issues.

  - <input type=image onClick=xxx /> was causing the form to submit twice

2008 Implementing Domain Driven Design in caAERS

Contents

Domain driven design basics

Layered Architecture

Any web application can be divided into mainly 4 layers.

User Interface:

it's the ui

Application Layer:

it is kept thin. Its responsibility is to only coordinates task and delegate work to domain layer. It must not have any domain logic

Domain Layer:

Its responsible for representing concepts of business. This layer is heart of business

Infrastructure Layer:

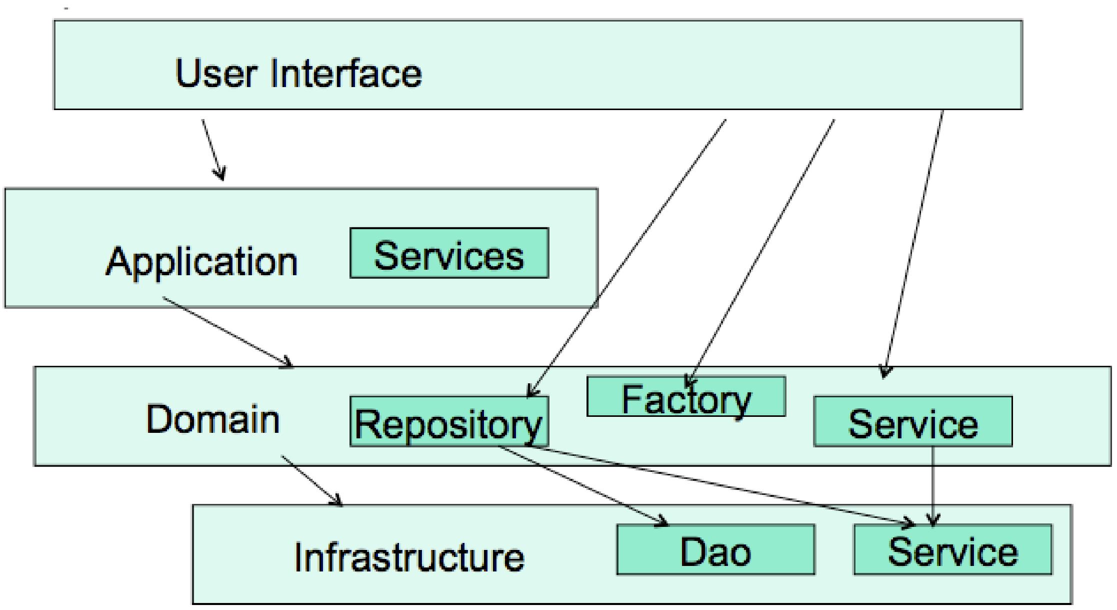
knows the infrastructure. Must not have any domain logic. Ex: sending message (EmailService), persistence for domain objects (DAO's).
Technical capabilities of this layer are generally offered as Services

The building blocks of DDD

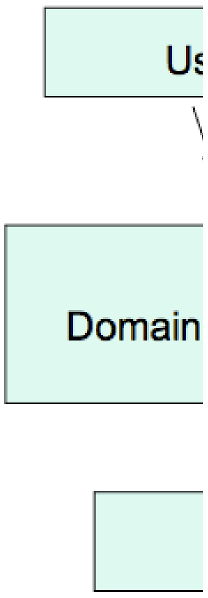
There are mainly 8 components of DDD

- Entities
- Value objects
- Services
- Modules
- Aggregates
- Factory
- Repository

Where does all this fits?



Code example



Entities

Value Object

Repository

Responsible for obtaining persistent objects and managing their lifecycle. Responsible for persisting objects also.

Ex: OrganizationRepository, ParticipantRepository

Factory

Responsible for creation of new objects (generally complex object); objects which does not exists in db. Factory is like object constructor. Ex: StudyFactory

Services

Some operations that are not part of an ENTITY or VALUE OBJECT. It is defined purely in terms of what it can do for a client.Its a verb not a noun. Ex:RegistrationService, ParticipantImportService, EmailService.

2008 IVY integration Notes

Ant commands to use

project-root> **ant publish-all**
This command will build and publish all the modules to local repository

project-root> **ant clean-all**
This command will clean all the modules

Switches

```
-Ddb=test            - Will load the test.properties instead of the default (datasource.properties)

-Dskip.test=true    - Will skip execution of testcases.
-DcctsWAR=true   - Will use the distGrid dependency when making the war.
-Dtest=StudyDaoTest - will run the test class (will only run in modules and on test target)
-Doffline=true       - Will run in offline mode.
-Drun.review.reports - Will run the cobertura reports
-Drecreate-db-hudson - Will execute the sql commands available in <datasource>-admin.properties
```

project-root/core>ant -f ivy-build.xml test -Ddb=test -Dtest=OrganizationRepositoryTest

project-root/core> **ant migrate -Ddb=test**
This command will run the migration scripts

project-root> **ant migrate -Ddb=test**
This command will run the migration scripts

project-root> **ant insert-csm-policy -Ddb=test**
This command will run the cms user migration scripts

Issues and fixes

For an introduction to ivy, click CTMS:here

Issue 1 : ivy:resolve

ivy:resolve in individual modules was not able to reslove using the ivysettings.xml in project root.

Fix: ivy:settings modified to use an explicit id attribute

```
<ivy:settings file="${proj.root.dir}/ivysettings.xml" id="caaers.ivy.instance"/>
Then refer every where caaers.ivy.instance using settingsRef
<ivy:resolve file="${ivy.file}" conf="default" haltonfailure="false" settingsRef="caaers.ivy.instance"
/>
<ivy:retrieve pattern="${lib.dir}/[artifact]-[revision].[ext]"
 haltonfailure="false" settingsRef="caaers.ivy.instance"/>
```

Issue 2 : Northwestern Repository POM issue
Certain of the CTMS related pom has wrong entries due to which the resolve task throw error saying "$[CTMS:groupId]#ctms-commons-core;0.8: not found"

Fix: haltonfailure="false"

```
<ivy:resolve file="${ivy.file}" conf="default" haltonfailure="false" settingsRef="........"/>
```

Issue 3 : Transitive dependency issue on Spring module jars

Wrong version of spring module jars (1.2.8),were getting loaded via acegi framework
Fix : In core ivy.xml, added explicitly spring modules and forced it to use 2.0.2 rev.

Wrong version of cglib was getting added
Fix: In core ivy.xml did a global exclude on cglib.

Issue 4 : Antlr jar conflict in Rules
antlr-2.7.6 and antlr-3.0.5b is required in rules, basically both the jar has different content. In the remote repository, both the jars are available in same location. The default conflict management strategy of IVY load only the latest, in this case antlr-3.0.5b, and evicts antlr-2.7.6.
Fix : antlr3.0.5b jar is pulled out from a different organization, via a different repository.

Eclipse Build Path Configuration

1. project-root> **ant sync-classpath**

FAQs

1. I want to build core or web only

execute following command

```
project-root/core>ant -f ivy-build.xml publish-local -Dskip.test=true
```

2008 Links to External Technical Articles, Blogs, Presentations

This page provide links to various technical blogs, interviews, presentations and articles. This is page is added to share the knowledge that could be useful for developers.

If you think it could be useful sharing any article with team, please add the link here. 🙂

Presentations

The Principles of Agile Design

one of the best presentation on agile design and patterns.

"Bob Martin known as "Uncle" Bob of Object Mentor presents the first of his five principles of agile design. Beginning with an explanation of the real purpose of object-oriented design - the management of dependencies - Bob walks through a code example to illustrate how dependencies can be managed with abstractions, and that good designs are those in which high-level abstractions do not depend on low-level details.

Here's a list of some of the Principles that he explained in this interview

- Open Closed Principle
- Liskov Substitution Principle
- Dependency Inversion Principle
- Interface Segregation Principle
- Packaging Principles
- Package Dependency Principle

Domain Driven Design with AOP and DI

"Domain Driven Design (DDD) suggests dealing with complex software system using a domain model and preserving the model in implementation. Since domain model entities have rich behavior, so should their software implementation artifacts. A direct mapping between domain model and software artifacts create simple-to- understand, inexpensive-to-implement, and easy-to-evolve systems.

While the idea behind DDD isn't new and the value is easily understood, many implementations do not adhere to its principles. This disconnection may be due to many obstacles in implementing it. Combining Dependency Injection (DI) with a full-fledged aspect-oriented programming (AOP) system such as AspectJ help overcome many obstacles.

The traditional DI mechanism allows injecting dependencies into coarse-grained objects such as services exposed to the application level. However, it cannot do the same for fine-grained domain objects, which are not exposed in the same manner. The DI and AOP combination overcomes this limitation allowing creation a web of domain objects mirroring the model. Now domain classes can implement rich behavior by collaborating with dependent objects, instead of acting as mere data carriers. Further, domain concepts such as security, change tracking, and business rules are crosscutting in nature. AOP allows expressing these concepts directly using aspects.

In this talk, Ramnivas Laddad will examine the need for domain driven design, obstacles in achieving it, the basics of enabling behavior-richness for domain objects, and patterns of usages. This session will also present several examples that show the power behind the techniques. The knowledge gained through this session will enable you to readily apply domain driven design in your systems."

Articles

Improving Performance of Healthcare Systems with Service Oriented Architecture

"The rapid rise of technology and its adoption into the healthcare field has caused healthcare organizations to collect an accumulation of non-interoperable systems that not only need to work together within the organization, but are also accessed from outside. The burden of integration usually falls on the users of the system, who are forced often to access many different systems to complete one task. The use of a service oriented architecture (SOA), however, can improve the delivery of important information and make the sharing of data across a community of care practical in cost, security, and risk of deployment."

Setup Eclipse Effectively

The series of article named "Effective Eclipse" at Eclipse Zone, explains how effectively we can configure Eclipse.

Interviews

Michael Stal on Architecture Refactoring

"In this interview, Michael Stal describes what architecture refactoring is about and how it relates to both code refactoring and patterns. He describes some architectural refactorings by giving real work examples from his work as Siemens, and he elaborates on some situations where you may want to avoid doing this kind of refactorings."

some External sites for nice technical articles, presentations and tutorials

http://www.infoq.com
http://www.parleys.com/display/PARLEYS/Home
DDD - Entity, ValueObject
What is SOA, really?

2008 Unit Testing in Agile Projects (or in caAERS or in C3PR)

> You can browse source code from https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/trunk/projects/?root=caaersappdev

1. What are different kind of test cases that developer write?

Developer write two types of Test cases

- Unit Test cases
- Integration Test cases (or Functional testing)

Unit testing

unit testing is a procedure used to validate that individual units of source code are working properly. A unit is the smallest testable part of an application (*or a method*). The goal is to test each part (unit) of code in isolation.

A test is not a unit test if:

- It talks to the database
- It communicates across the network
- It touches the file system
- You have to do special things to your environment (such as editing config files) to run it.

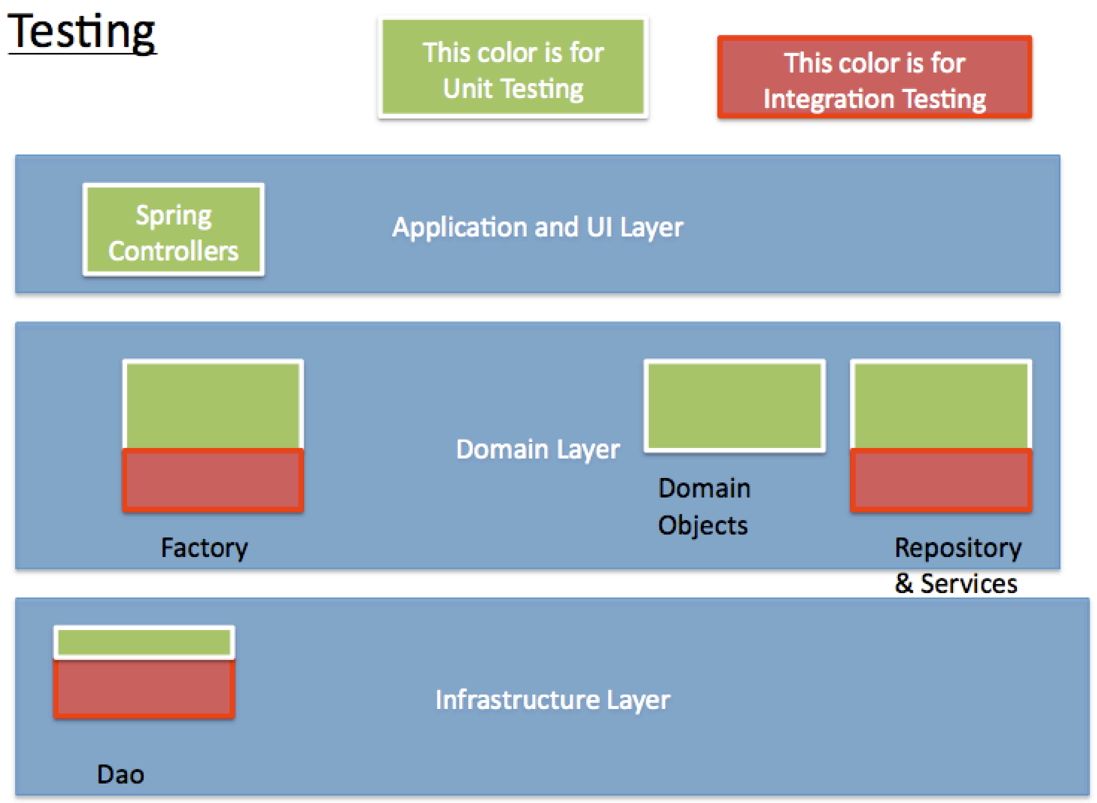
Integration testing

Integration testing is the phase of software testing in which individual software modules are combined and tested as a group..

Purpose: The purpose of integration testing is to verify functional, performance and reliability requirements.

Ex: Dao test cases which connects with DB.

Where does all this fits?



Here you can notice that-

- We can test almost 100% functionality of Domain object by simply writing unit test cases only
- We can test almost 70-80% functionality of Factories by unit testing
- almost complete functionality of Controllers can be tested by unit testing only

2. What are the benefits of Unit testing?

Facilitates change

Unit testing allows the programmer to re-factor code at a later date, and make sure the module still works correctly. The procedure is to write test cases for all methods so that whenever a change causes a fault, it can be quickly identified and fixed.

Simplifies integration

Unit testing helps to eliminate uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

Documentation

Unit testing provides a sort of living documentation of the system. Unit tests constitute design documentation that evolves **naturally** with a system. Read that again. This is the Holy Grail of software development, documentation that evolves ***naturally*** with a system. What better way to document a class than to provide a coded set of use cases. That's what these unit tests are: a set of coded use cases that document what a class does, given a controlled set of inputs. As such, this design document is always up-to-date because the unit tests always have to pass.

Very Easy and Simple To write and maintain

Unit testing are more simpler to write and maintain as compare of Integration testing. Unit test cases run very fast which makes them easier to develop.

A maintained suite of unit tests:

- Represents the most practical design possible
- Provides the best form of documentation for classes
- Determines when a class is "done"
- Gives a developer confidence in the code
- Is a basis for refactoring quickly

Lastly, unit tests provide you with a high degree of confidence, which translates into developer satisfaction. If you run unit tests whenever you make changes to code, you'll find out immediately if your changes broke something.

3. How to write Unit test cases?

It is easy to become overwhelmed when you start writing unit tests. Following are some best practices on writing unit test cases.

1. In general, **there should be a unit test for every public method of your class**. However, methods with very straightforward functionality, for example, getter and setter methods, don't need unit tests unless they do their getting and setting in some "interesting" way. A good guideline to follow is to **write a unit test whenever you feel the need to comment some behavior in the code**. If you're like many programmers who aren't fond of commenting code, unit tests are a way of documenting your code behavior.

So one of the easiest ways to find out what the test should be testing is to look at the comments in the existing code*. Any comment should be captured in a unit test. Translate block comments at the beginning of methods describing what the method does into unit tests.*

2. Put the unit tests in the same package as the associated classes being tested. This type of organization allows each unit test to call methods and reference variables that have access modifiers of package or protected in the class being tested.

3. These mechanics will serve you well, but a comprehensive suite of unit tests will not be worth anything if you don't run the tests. **Running the tests early and often gives** you absolute confidence in your code all the time. As the project proceeds, you will add features. Running the tests will tell you if the new features you've just implemented have broken something.

4. **Use mock objects** (see below on mock object concept) of dependencies

How to use Mock objects for unit testing?

mock objects are simulated objects that mimic the behavior of real objects in controlled ways. A mock object conforms to the interface of the real object, but has just enough code to fool the tested object and track its behavior. For example, a database connection for a particular unit test might record the query while always returning the same hardwired result. .

There are two open-source frameworks for creating mock objects- JMock and EasyMock.

The common coding style for testing with mock objects is to:

- Create instances of mock objects
- Set state and expectations in the mock objects
- Invoke domain code with mock objects as parameters
- Verify consistency in the mock objects

⚠ see external links and example in following sections for more information on how to work with mock objects.

Examples of Unit test cases

Unit test cases for Domain Objects

Following is an example of a `ExpeditedAdverseEventReport` domain object.

ExpeditedAdverseEventReport.java

```
public class ExpeditedAdverseEventReport extends AbstractMutableDomainObject {

@Transient
    public Map<String, String> getSummary() {
        Map<String, String> summary = new LinkedHashMap<String, String>();
        summary.put("Participant", summaryLine(getParticipant()));
        summary.put("Study", summaryLine(getStudy()));
        summary.put("Report created at", getCreatedAt() == null ? null : getCreatedAt().toString());
        String primaryAeLine = null;
        if (getAdverseEvents().size() > 0 &&
          getAdverseEvents().get(0).getAdverseEventTerm() != null &&
          getAdverseEvents().get(0).getAdverseEventTerm().getUniversalTerm() != null) {
            primaryAeLine = getAdverseEvents().get(0).getAdverseEventTerm().getUniversalTerm();
        }
        summary.put("Primary AE", primaryAeLine);
        summary.put("AE count", Integer.toString(getAdverseEvents().size()));
        summary.put("Public identifier", getPublicIdentifier());
        // TODO: placeholders
summary.put("Ticket number", null);
        summary.put("Next report due", null);

        return summary;
    }
```

Following is an example of a unit test case added for this domain object.

```java
public class ExpeditedAdverseEventReportTest extends AbstractTestCase {
    private static final Timestamp CREATED_AT = DateTools.createTimestamp(2006, Calendar.MAY, 8, 9,
                    8, 7);

    private ExpeditedAdverseEventReport report;
    private BeanWrapper wrappedReport;
    private CtcTerm ctcTerm;

    private AdverseEvent adverseEvent;

    @Override
    protected void setUp() throws Exception {
        super.setUp();
        report = new ExpeditedAdverseEventReport();
        report.setCreatedAt(CREATED_AT);

    }

    public void testSummaryIncludesStudy() throws Exception {
        Participant participant = Fixtures.createParticipant("Joe", "Shabadoo");
        Study study = Fixtures.createStudy("El Study");
        report.setAssignment(Fixtures.assignParticipant(participant, study, new Organization()));
        Map<String, String> summary = report.getSummary();
        assertEquals("El Study", summary.get("Study"));
    }

    public void testSummaryStudyIncludesPrimaryIdentifier() throws Exception {
        Participant participant = Fixtures.createParticipant("Joe", "Shabadoo");
        Study study = Fixtures.createStudy("El Study");
        study.addIdentifier(Identifier.createTemplate("1845"));
        study.getIdentifiers().get(0).setPrimaryIndicator(true);
        report.setAssignment(Fixtures.assignParticipant(participant, study, new Organization()));
        Map<String, String> summary = report.getSummary();
        assertEquals("El Study (1845)", summary.get("Study"));
    }

    public void testSummaryIncludesParticipant() throws Exception {
        Participant participant = Fixtures.createParticipant("Joe", "Shabadoo");
        Study study = Fixtures.createStudy("El Study");
        report.setAssignment(Fixtures.assignParticipant(participant, study, new Organization()));
        Map<String, String> summary = report.getSummary();
        assertEquals("Joe Shabadoo", summary.get("Participant"));
    }

    public void testSummaryParticipantIncludesPrimaryIdentifier() throws Exception {
        Participant participant = Fixtures.createParticipant("Joe", "Shabadoo");
        participant.addIdentifier(Identifier.createTemplate("MRN1138"));
        participant.getIdentifiers().get(0).setPrimaryIndicator(true);
        Study study = Fixtures.createStudy("El Study");
        report.setAssignment(Fixtures.assignParticipant(participant, study, new Organization()));
        Map<String, String> summary = report.getSummary();
        assertEquals("Joe Shabadoo (MRN1138)", summary.get("Participant"));
    }

    public void testSummaryIncludesFirstAETerm() throws Exception {
        Map<String, String> summary = report.getSummary();
        assertEquals("Term - Select", summary.get("Primary AE"));
    }

    public void testSummaryIncludesAECount() throws Exception {
        Map<String, String> summary = report.getSummary();
        assertEquals("1", summary.get("AE count"));
    }

    public void testSummaryIncludesCreatedAt() throws Exception {
        Map<String, String> summary = report.getSummary();
        assertEquals("2006-05-08 09:08:07.0", summary.get("Report created at"));
    }
}
```

You can browse the latest source code at
https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/trunk/projects/core/src/main/java/gov/nih/nci/cabig/caaers/domain/Expedite

Unit test cases for Repository

Repository needs dependencies so you need mock objects for writing test cases for repositories. Following is an example of repository

see external links and example in following sections for more information on how to work with mock objects.

```
package gov.nih.nci.cabig.caaers.domain.repository;
@Transactional(readOnly = true)
public class ParticipantRepository {

 private ParticipantDao participantDao;

public boolean checkIfParticipantExistsForGivenIdentifiers(List<Identifier> identifiers) {

for (Identifier identifier : identifiers) {
            Participant tempParticipant = participantDao.getByIdentifier(identifier);
            if (tempParticipant != null) {
                return true;

            }
        }
        return false;
}
}
```

Following is the unit test case for this repository

ParticipantRepositoryTest.java

```java
public class ParticipantRepositoryTest extends AbstractTestCase {

    private ParticipantRepository participantRepository;
    private ParticipantDao participantDao;
    private Participant participant;
    private SystemAssignedIdentifier systemAssignedIdentifier;

    protected void setUp() throws Exception {
        participantRepository = new ParticipantRepository();
        participantDao = registerDaoMockFor(ParticipantDao.class);
        participantRepository.setParticipantDao(participantDao);
        participant = Fixtures.createParticipant("first", "last");
        systemAssignedIdentifier = Fixtures.createSystemAssignedIdentifier("value");
    }

    public void testCheckIfParticipantExistsForGivenIdentifiersForNoIdentifiers() {
        //first check for no identifiers
boolean participantExists = participantRepository.
                checkIfParticipantExistsForGivenIdentifiers(participant.getIdentifiers());

        assertFalse("participant should not exists as there are no identifiers", participantExists);


    }

    public void testCheckIfParticipantExistsForGivenIdentifiersIfNoParticipantExits() {

        //check if participant does not exits
participant.addIdentifier(systemAssignedIdentifier);
        EasyMock.expect(participantDao.getByIdentifier(systemAssignedIdentifier)).andReturn(null);
        replayMocks();
        boolean participantExists = participantRepository.
                checkIfParticipantExistsForGivenIdentifiers(participant.getIdentifiers());
        verifyMocks();
        assertFalse("participant should not exists as there is are participants for given identifiers"
, participantExists);


    }

    public void testCheckIfParticipantExistsForGivenIdentifiersIfParticipantDoesExits() {

        //check if participant  exits
participant.addIdentifier(systemAssignedIdentifier);
        EasyMock.expect(participantDao.getByIdentifier(systemAssignedIdentifier)).andReturn(new
Participant());
        replayMocks();
        boolean participantExists = participantRepository.
                checkIfParticipantExistsForGivenIdentifiers(participant.getIdentifiers());
        verifyMocks();
        assertTrue("participant should  exists as there  are participants for given identifiers",
participantExists);

    }


}
```

Unit Test cases for Controllers

Unit test cases for Factory

What is a Fixture or Object Mother pattern for easing Test Object Creation in XP?

"One of the most time-consuming and unfulfilling activities in testing is coding and maintaining test objects. The ObjectMother pattern addresses this issue by proposing a simple, scalable framework whose sole purpose is to construct and facilitate the customization of test objects.

The pattern's primary responsibilities are: to create test objects, to tailor those objects at any point during the testing process. "(reference : http://www.agilealliance.com/system/article/file/910/file.pdf)

in caAERS, we use a class Fixture which provides methods for creating test objects.

```
package gov.nih.nci.cabig.caaers.domain;
public class Fixtures {
    public static <T extends DomainObject> T setId(final int id, final T target) {
        target.setId(id);
        return target;
    }

    public static Participant createParticipant(final String first, final String last) {
        Participant p = new Participant();
        p.setFirstName(first);
        p.setLastName(last);
        return p;
    }

    public static Study createStudy(final String shortTitle) {
        Study s = new Study();
        s.setShortTitle(shortTitle);
        s.setLongTitle(shortTitle);
        return s;
    }

    public static Organization createOrganization(final String name) {
        Organization organization = new Organization();
        organization.setName(name);
        organization.setDescriptionText("dec:" + name);
        organization.setNciInstituteCode("NCI333:" + name);
        return organization;
    }

    /**
     * Creates an assignment and the associated Study, Participant, StudySite, and Site objs
     */
    public static StudyParticipantAssignment createAssignment() {
        return assignParticipant(createParticipant("D", "C"), createStudy("DC"),
                createOrganization("N/A"));
    }

    public static StudyParticipantAssignment assignParticipant(final Participant participant,
                                                               final Study study, final Organization
organization) {
        StudySite studySite = new StudySite();
        studySite.setId(123);
        studySite.setOrganization(organization);
        study.addStudySite(studySite);
        organization.addStudySite(studySite);

        StudyParticipantAssignment assignment = new StudyParticipantAssignment();
        studySite.addAssignment(assignment);
        participant.addAssignment(assignment);

        return assignment;
    }


    public static ResearchStaff createResearchStaff(final Organization organization,
                                                    final List<UserGroupType> userGroupTypes, final
String name) {
        ResearchStaff researchStaff = new ResearchStaff();
        researchStaff.setFirstName("Jeff");
        researchStaff.setLastName("Someone");
        researchStaff.setEmailAddress(name + "@def.com");
        researchStaff.setPhoneNumber("123-5-789");
        researchStaff.setNciIdentifier("nci id");

        for (UserGroupType userGroupType : userGroupTypes) {
            researchStaff.addUserGroupType(userGroupType);
        }

        researchStaff.setOrganization(organization);
        return researchStaff;
```

```
    }
}
```

Test Coverage using Clover or Cobertura?

Our project has many hundreds of unit tests, and JUnit's green bar of goodness gives you a sense that your code base is well tested. But how well do those tests actually test the code base? What code are the tests actually testing (and which code isn't being tested at all?) Which particular tests are hitting a particular piece of code? Is the test suite getting out of date?

Clover or Cobertura gives you these answers. These are code coverage tools. Cobertura is an open source project. Clover is paid but free for open source projects.

4. How much time and efforts developer should put in writing test cases in agile projects?

As a developer, testing is so important that you should be doing it all of the time. It should not be relegated to a specific stage of the development cycle. It definitely shouldn't be the last thing done before giving your system to a customer. T**esting, both unit and integration, needs to be an integrated part of the development process.**

Typically, we write java code, jsp, html and javascript for implementing a functionality. Following are typical guidelines on how much time should put in writing test cases

	time spent in coding	time spent in writing test cases
Java code	50-60%	40-50%
jsp/html/css	100%	(you don't write any test cases for jsps)
javascript	100%	(you don't write any test cases for javascripts)

So developer should consider these factors also while estimating time for a given requirement.

External Resources

unit testing vs functional testing

Testing Persistent Domain Model

unit testing with mock objects
http://www.realsolve.co.uk/site/tech/easymock.php

how to use Easy Mock objects

Object Mother Pattern

Conclusion

Unit tests are written from the developer's perspective and focus on particular methods of the class under test. Use these guidelines when writing unit tests:

- Write the unit test before writing code for class it tests.
- Capture code comments in unit tests.
- Test all the public methods that perform an "interesting" function (that is, not getters and setters, unless they do their getting and setting in some unique way).
- Put each test case in the same package as the class it's testing to gain access to package and protected members.

Functional tests are written from the user's perspective and focus on system behavior that users are interested in. Find a good functional testing framework, or develop one, and use these functional tests to identify what the user really wants. In this way, the functional tester gains an automated tool and has a starting point for using the tool.

Make unit testing and functional testing central to your development process. If you do, you will have confidence that your system works and can grow. If you don't, you can't be sure. Testing may not be fun, but having working unit and functional tests makes development a lot more fun.

IDE - Unit testing Notes

(databaseConfigurationName)

Problem In caAERS applicationContext-core-db.xml, applicationContext-core-spring.xml, applicationContext-rules-jcr.xml files, the database/property configuration is an ant filter, which will only get replaced if built/compiled using ant.

```
<bean id="mainPropertyConfigurer" class=
"gov.nih.nci.cabig.ctms.tools.spring.PropertyPlaceholderConfigurer">
        <property name="properties">
            <bean class="gov.nih.nci.cabig.caaers.tools.CaaersDataSourcePropertiesFactoryBean">
                <property name="databaseConfigurationName"
><value>@databaseConfigurationName@</value></property>
            </bean>
        </property>
        <!-- have to use a separate prefix/suffix to avoid maven filtering in everything
             at build time -->
        <property name="placeholderPrefix"><value>s[</value></property>
        <property name="placeholderSuffix"><value>]</value></property>
        <property name="useNullForUnresolvablePlaceholders"><value>true</value></property>
    </bean>
```

Fix Add a new applicationContext-test.xml, which contains the same beans re-written, but will replace (databaseConfigurationName) with "test". Then the CaaersTestCase#getConfigLocations() method will include applicationContext-test.xml.

CaaersTestCase.java

```java
public  String[] getConfigLocations() {
        return new String[] {
            "classpath*:gov/nih/nci/cabig/caaers/applicationContext-*.xml",
            "classpath*:applicationContext-test.xml"
        };
}
```

But this mandates that you should have test.properties available in your TOMCAT_HOME/conf/caaers or USER_HOME/.caaers or /etc/caaers folder.

✓ **Handy Hint**
To avoid conflict, **ant build** will ignore applicationContext-test.xml

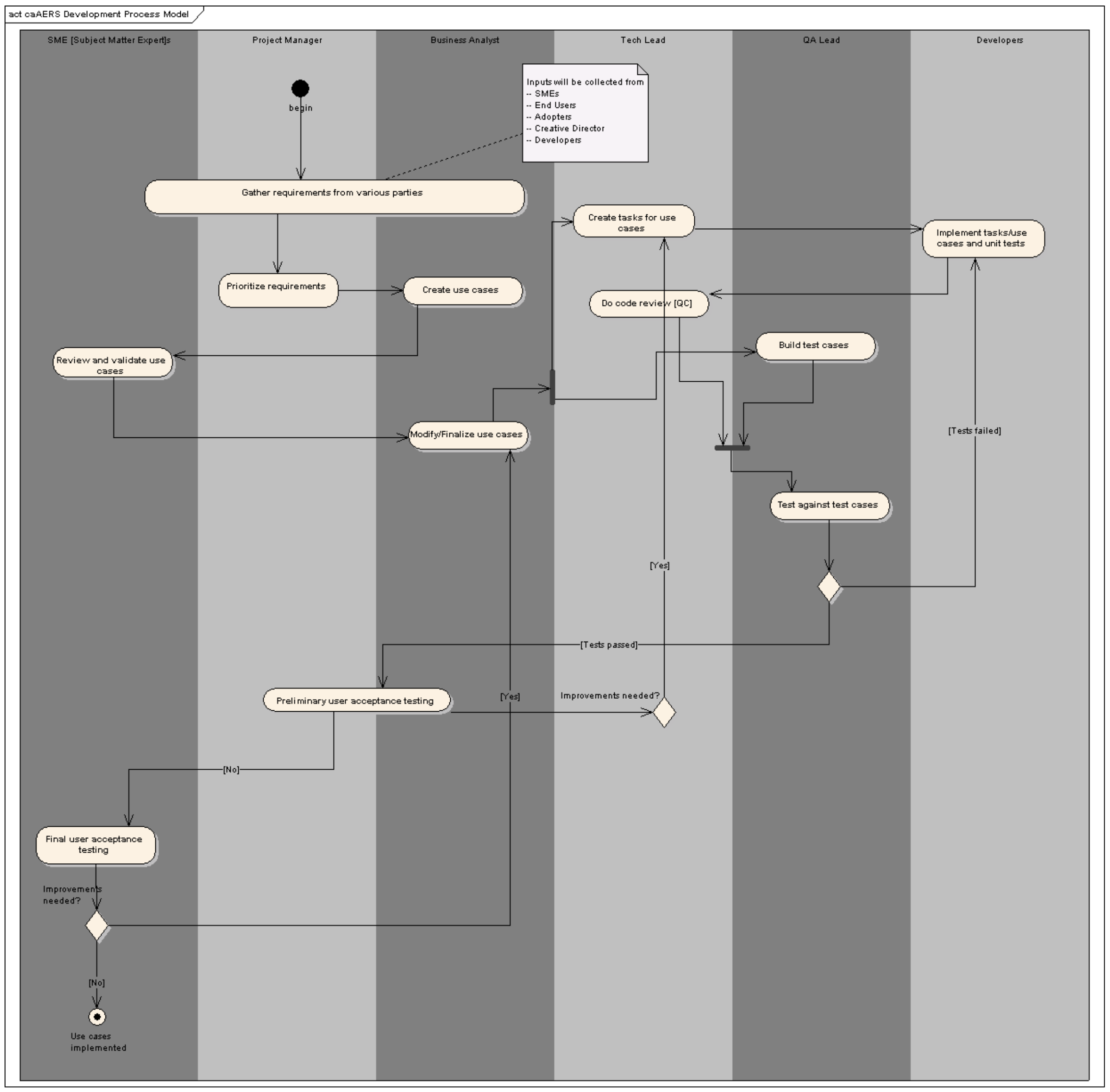
Static applicationContextDeployment

ProblemCaaersTestCase, defined static getDeployedApplicationContext(), which loads all applicationContext-*.xml files. The issue with this is, some of the Security related beans needs compile time aspect weaving, without which the testcases will not be able to load application context. It is cumbersome to run through AspectJ compiler in IDE

Fix Make the getDeployedApplicationContext() an instance level method, as such this will not cause any issue, as the applicationContext loading function makes sure that the context is loaded/try to load only once. Advantage this provide is, the child testcase if (required) can instruct the specific set of application contexts to be loaded.

caAERS 1.x development process model

The following is the activity chart for caAERS development.



caAERS 1.x running instances

Details about the various running instances of caAERS.

System Level

Properties	Values	
Machine / VM name	Duncan.herndon.semanticbits.com	
IP Number	10.10.10.41	
Machine login	caaers	
Machine password	caaers	
Type of DB [CTMS:Postgres / Oracle]	Postgres	

Location of DB folder	/usr/local/pgsql	
DB Schema name	caaers_dev	
DB username	caaers	
DB password	caaers	
DB URL		
Build type [CTMS:ant / maven]	ant	
CATALINA_HOME	/usr/local/tomcats/caaers/dev	
USER_HOME	/home/caaers_dev	
caAERS URL		
servicemix URL		
Java version	1.5_16	
Ant version	1.7.1	
Tomcat version	5.5.23	

Ports allocated to caAERS instance.

Port	Client
5432	Postgres DB
8080	Tomcat standard
8443	Tomcat secure
8005	Tomcat shutdown
default	Tomcat debug
61616	Servicemix

Application Level

Properties	Values	
caAERS name	**caaers_dev**	
caAERS version	trunk	
codebase location	/usr/local/codebase	
SERVICEMIX_HOME	/usr/local/servicemix/caaers/dev	
RULES_REPO folder	/usr/local/caaers/rules_repo	

CRA caAERS walk-through feedback 6-11-08

Paul to send out follow-up email with link to demo and user name/password, study/subject combo

- need to make sure to explain that the info provide on patient is what is standard for the center. some centers only use initials and identifiers. Need to make sure that the application supports the use of initials only; name is not included in any reports though
- Need to prevent changing of data of subjects in caAERS when tied to a local ctms
- Birthday needs to require just month and year, not day
- radiation, surgery, and device should only show up if they are allowed to be included in a report (if they're associated with the TAC)

- Data coordinators generally review per study, not per subject, so need a report that will allow them to review AEs on a study to look for patterns, trends, etc
- thinks it would be cumbersome to have to enter the same info in for each cycle (not sure if she was talking just subject info, or with AE)
- Need to be able to print anything they do, generate a 1-page form showing what was entered for a reporting period for a patient (filled in version of the CRA - wants to be able to print a blank version of that form as well)
    - worksheet to take notes, is that considered part of the study, or could it be discarded?
    - It would not be considered a source document
    - to consider it source, you'd end up creating study specific
    - Generic worksheet for use as a tool would be fine, but not everyone will use
    - could develop, but need to be considered throw-away
- Need a dashboard that shows how many patients on a study, status of all patients at their site, etc (ann thought it was requirement for later phase), useful for the PI
- what is going to be provided to auditors? back to auditors, what do they want, how much detail do you need? Barbara Barrett provided info
    - green thing would be to be able to log in and look.
    - many institutions don't have internet (or even cell)
    - Need paper copies for sites that don't have medical records
    - would be good to include start/end dates for AEs on manage reports
- Begin screen "search by patient identifier", "search by study #"
- need to continue to show ctc version/medra version for clarification
- Wants instructions on AE page so they only report what they need to (if study only wants grades 3-5 reported)
- would like message/pop-up that says "study only requires reporting of AEs 3 or higher" if they enter an AE with grade of 1 or 2 and it's not required
- don't want the cras to be entering the expectedness field
- Ann doesn't like all the white space
- don't care how many pages online they have to go through, but care about having the fewest # of printed pages - Debbie

CSM Tutorial with caAERs 1.x Examples

This document explains how CSM security module handles the security aspects of any web application. These security aspects are authentication, authorization, method level security, instance level security etc.

⚠ All example and code snapshots are taken from caAERS application. You can checkout caAERs codebase from https://gforge.nci.nih.gov/svnroot/caaersappdev/trunk/

We uses CSM's Authentication and Authorization under the Acegi Security Framework. So let me first explain the Acegi security concepts.

Acegi security framework

Acegi Security is widely used within the Spring community for comprehensive security services for Spring-powered applications. It comprises a set of interfaces and classes that are configured through a Spring IoC container. The design of Acegi Security allows many applications to implement the common enterprise application security requirements via declarative configuration settings in the IoC container. Acegi Security is heavily interface- driven, providing significant room for customization and extension. **Important:** Acegi Security, like Spring, emphasizes pluggability.

What Acegi can Do?

- Prompt the user for login before accessing a secure resource.
- Authenticate the user by checking a security token such as a password.
- Check whether an authenticated user has the privilege to access a secure resource.
- Redirect a successfully authenticated and authorized user to the secure resource requested.
- Display an Access Denied page to a user who does not have the privilege to access a secure resource.
- Remember a successfully authenticated user on the server and set a secure cookie on the user's client. The next authentication can then be performed using the cookie and without asking the user to log in.
- Store authentication information in server-side session objects to securely serve subsequent requests for resources.
- Build and maintain a cache of security information in server-side objects to optimize performance.
- When the user signs out, destroy server-side objects maintained for the user's secure session.
- Communicate with a variety of back-end data storage services (like a directory service or a relational database) that are used to store users' security information

Acegi Architecture and Components

Filters
Managers
Providers
Handlers

Four major components

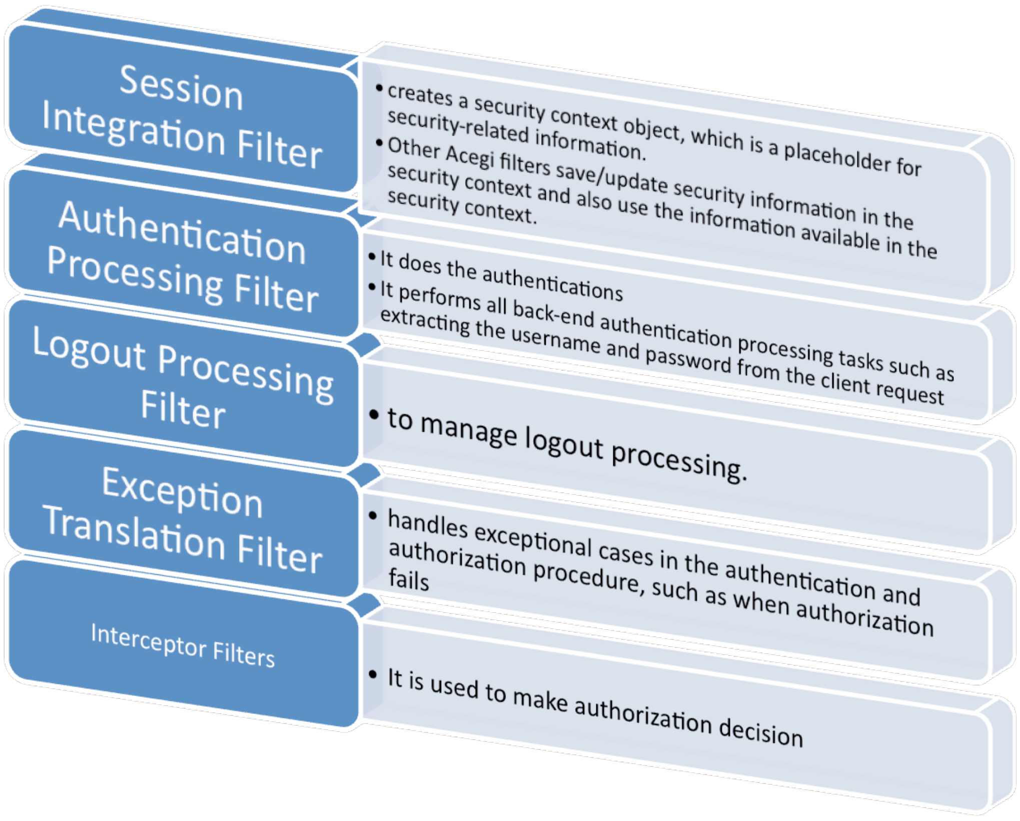
Filters: These most high-level components provide common security services like authentication processing, session handling, and logout.

Managers: Filters are only a high-level abstraction of security-related functionality: managers and providers are used to actually implement authentication processing and logout services. Managers manage lower level security services offered by different providers.

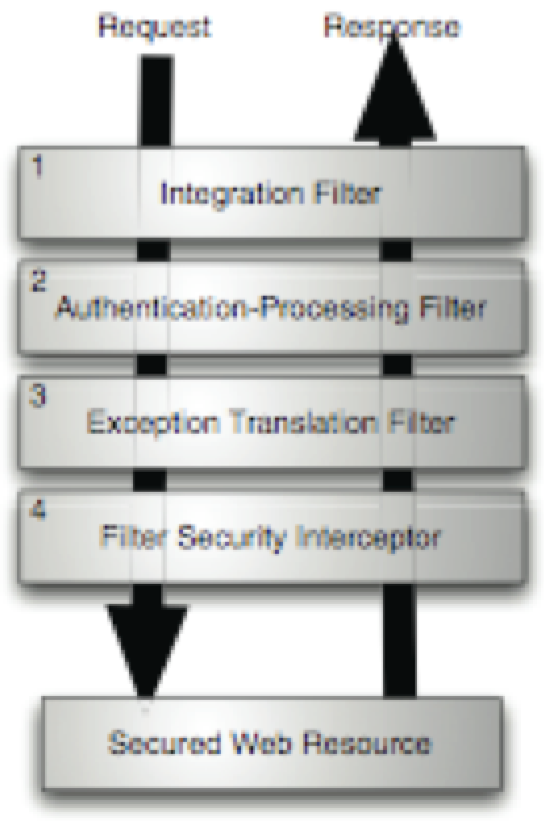
Providers: A variety of providers are available to communicate with different types of data storage services, such as directory services, relational databases, or simple in-memory objects

Handlers: Tasks are sometimes broken up into multiple steps, with each step performed by a specific handler. For example, Acegi's logout filter uses two handlers to sign out an HTTP client.
One handler invalidates a user's HTTP session and another handler destroys the user's cookie.

Filters


Session Integration Filter
• creates a security context object, which is a placeholder for security-related information.
• Other Acegi filters save/update security information in the security context and also use the information available in the security context.
Authentication Processing Filter
• It does the authentications
• It performs all back-end authentication processing tasks such as extracting the username and password from the client request
Logout Processing Filter
• to manage logout processing.
Exception Translation Filter
• handles exceptional cases in the authentication and authorization procedure, such as when authorization fails
Interceptor Filters
• It is used to make authorization decision

How do filters work?


Request
Response
1 Integration Filter
2 Authentication-Processing Filter
3 Exception Translation Filter
4 Filter Security Interceptor
Secured Web Resource

1. Due to the stateless nature of HTTP, Spring Security needs a way to preserve a user's authentication between web requests. An integration filter is responsible for retrieving a previously stored authentication (most likely stored in the HTTP session) at the beginning of a request so that it will be ready for Spring Security's other filters to process.

2 Next, one of the authentication-processing filters will determine if the request is an authentication request. If so, the pertinent user information (typically a username/ password pair) is retrieved from the request and passed on to the authentication manager to determine the user's identity. If this is not an authentication request, the filter performs no processing and the request flows on down the filter chain.

3 The next filter in line is the exception translation filter. The exception translation filter's sole purpose is to translate **AccessDeniedExceptions** and **AuthenticationExceptions** that may have been thrown into appropriate HTTP responses. If an AuthenticationException is detected, the request will be sent to an authentication entry point (e.g., login screen). If an AccessDeniedException is thrown, the default behavior will be to return an HTTP 403 error to the browser.

4 The last of the required filters is the filter security interceptor. This filter plays the part of the security interceptor for web applications. It examines the request and determines whether the user has the necessary privileges to access the secured resource. It doesn't work alone. It leans hesavily on the authentication manager and the access decision manager to help it grant or restrict access to the resource. If the user makes it past the filter security interceptor, the user will be granted access to the secured web resource. Otherwise, an AccessDeniedException will be thrown and the exception translation filter will handle it appropriately.

Configuring Filters

1. Add following configuration in web.xml (see web.xml of caAERS web application)

```
<filter>
  <filter-name>Acegi Filter Chain Proxy</filter-name>
  <filter-class>
   org.acegisecurity.util.FilterToBeanProxy
  </filter-class>
  <init-param>
   <param-name>targetClass</param-name>
   <param-value>
    org.acegisecurity.util.FilterChainProxy
   </param-value>
  </init-param>
 </filter>

 <filter-mapping>
  <filter-name>Acegi Filter Chain Proxy</filter-name>
  <url-pattern>/*</url-pattern>
 </filter-mapping>
```

2. define `filterChainProxy` bean in `applicationContext-acegi-security.xml` file (of caAERS for example)

```
<bean id="filterChainProxy" class="org.acegisecurity.util.FilterChainProxy">
        <property name="filterInvocationDefinitionSource">
            <value>
                CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
                PATTERN_TYPE_APACHE_ANT
                /**=channelProcessingFilter,httpSessionContextIntegrationFilter,
LogoutFilter,s[authenticationMode]AuthenticationProcessingFilter,
securityContextHolderAwareRequestFilter, anonymousProcessingFilter,exceptionTranslationFilter,
filterInvocationInterceptor
            </value>
        </property>
    </bean>
```

here you can see that we have added these 4 filters. We have some more filters. Please refer the java docs of these filters for their functionality.

caAERS uses two authentication mode; local and webSSO (its a single signon authentication). In local authentication mode, you authenticate users by using local database/ldap. The s[CTMS:authenticationMode]
property refers this authentication mode. For all understanding purpose, please consider value of this property equals to "local"

Session Integration Filter (SIF)

1. SIF checks whether it has already processed this Web request or not. If it finds that it has, it does no further processing .
2. SIF checks whether a session object exists and contains a security context. It retrieves the security context from the session object and places it in a temporary placeholder called security context holder.
3. SIF calls the next filter in the filter chain.
4. Other filters may edit the security context.
5. SIF receives control after the filter chain processing completes.
6. SIF checks whether any other filter changed the security context during its processing (for example, APF may have stored user details in the security context). If so, it updates the security context in the session object. This means that any changes made to the security context during filter chain processing now reside in the session object.

```
<bean id="httpSessionContextIntegrationFilter" class=
"org.acegisecurity.context.HttpSessionContextIntegrationFilter"/>
```

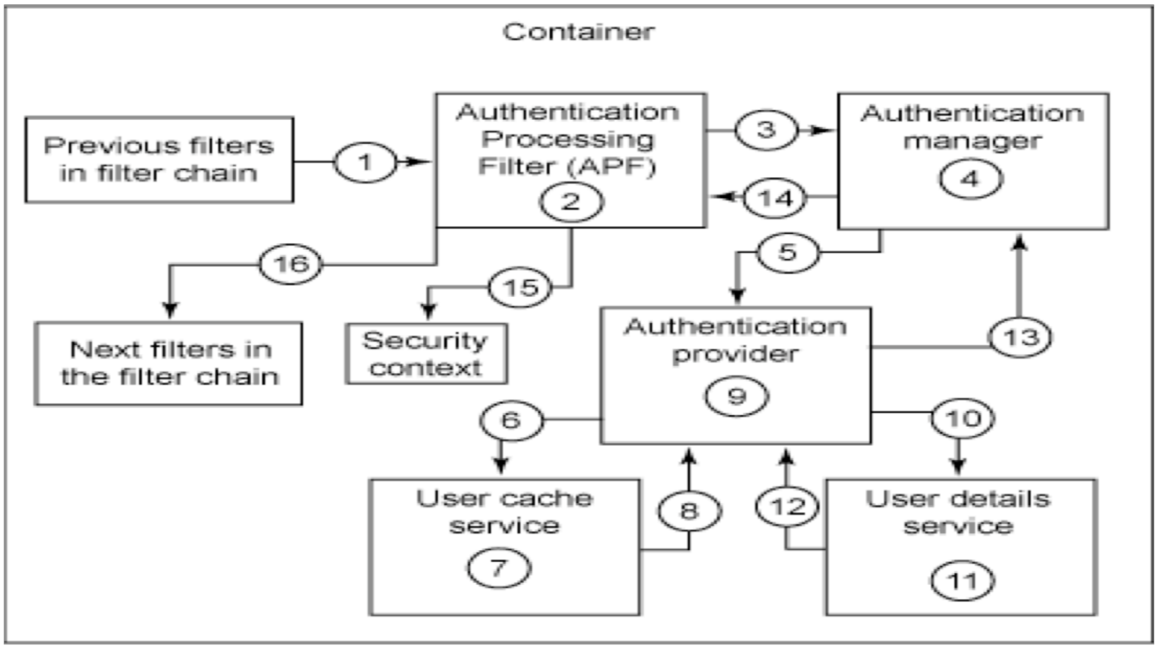
Authentication Processing Filter (APF)

1. Acegi uses the Authentication Processing Filter (APF) for authentication. APF uses an authentication (or login) form, in which a user enters a username and password and triggers authentication.
2. APF performs all back-end authentication processing tasks such as extracting the username and password from the client request, reading the user's parameters from the back-end user base, and using the information to authenticate the user.
3. APF fetches the username, password, and other information from the user's request object. It passes this information to the

authentication manager. The authentication manager uses an appropriate provider to read detailed user information (such as username, password, e-mail address, and the user's access rights or privileges) from the back-end user base, authenticates the user, and stores the information in an Authentication object.

4. Finally, APF saves the Authentication object in the security context created earlier by SIF. The Authentication object stored in the security context will be used later to make authorization decisions.

```
<bean id="localAuthenticationProcessingFilter" class=
"org.acegisecurity.ui.webapp.AuthenticationProcessingFilter">
        <property name="authenticationManager" ref="authenticationManager"/>
        <property name="authenticationFailureUrl" value="/public/login?login_error=1"/>
        <property name="defaultTargetUrl" value="/"/>
        <property name="filterProcessesUrl" value="/j_acegi_security_check"/>
    </bean>
```



Exception Translation Filter (ETF)

It handles exceptional cases in the authentication and authorization procedure, such as when authorization fails. In these exceptional cases, ETF decides what to do.

Example,

1. in case of AuthenticationException, ETF serves the login page inviting the user to authenticate.
2. Similarly, in case of authorization failure, you can configure ETF to serve an Access Denied page

```
<bean id="exceptionTranslationFilter" class="org.acegisecurity.ui.ExceptionTranslationFilter">
        <property name="authenticationEntryPoint" ref=
"s[authenticationMode]AuthenticationProcessingFilterEntryPoint"/>
        <property name="accessDeniedHandler">
            <bean class="org.acegisecurity.ui.AccessDeniedHandlerImpl">
                <property name="errorPage" value="/accessDenied.jsp"/>
            </bean>
        </property>
    </bean>
```

value of `s[CTMS:authenticationMode]` is equals to "local"

Interceptor Filters

This filter takes authorization decision.

```
<bean id="filterInvocationInterceptor" class=
"org.acegisecurity.intercept.web.FilterSecurityInterceptor">
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="accessDecisionManager">
   <bean class="org.acegisecurity.vote.AffirmativeBased">
    <property name="allowIfAllAbstainDecisions" value="false"/>
    <property name="decisionVoters">
     <list>
      <bean class="gov.nih.nci.security.acegi.csm.vote.AuthorizationSwitchVoter">
       <property name="authorizationSwitch" ref="authorizationSwitch"/>
      </bean>
      <bean class="gov.nih.nci.security.acegi.csm.vote.CSMAuthorizationCheckVoter">
       <property name="processConfigAttribute" value="CSM_FILTER_CHECK"/>
       <property name="authorizationCheck" ref="filterAuthorizationCheck"/>
      </bean>
     </list>
    </property>
   </bean>
  </property>
  <property name="objectDefinitionSource">
   <value>
    CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
    \A/pages/.*\Z=CSM_FILTER_CHECK
   </value>
  </property>
 </bean>
```

- The **authenticationManager** component is the same as the authentication manager discussed in the Authentication Processing Filter.
- The **accessDecisionManager** component manages the process of authorization
- The **objectDefinitionSource** component contains access control definitions according to which authorization will take place

accessDecisionManager

This manages the process of authorization . It uses the access control definitions provided by the `objectDefinitionSource` to make authorization (or access control) decisions. Spring Security's access decision managers are ultimately responsible for determining the access rights for an authenticated user but they they do not work on their own. Instead, they poll one or more objects that vote on whether or not a user is granted access to a secured resource. Once all votes are in, the decision manager tallies the votes and arrives at a final decision.

Spring Security comes with three implementations of AccessDecisionManager, as listed in table below. Each takes a different approach to tallying votes.

Access decision manager	How it decides to grant/deny access
org.acegisecurity.vote.AffirmativeBased	Allows access if at least one voter votes to grant access
org.acegisecurity.vote.ConsensusBased	Allows access if a consensus of voters vote to grant access
org.acegisecurity.vote.UnanimousBased	Allows access if all voters vote to grant access

Following is the configuration of `objectDefinitionSource` with one Voter

```
<bean id="accessDecisionManager" class="org.acegisecurity.vote.AffirmativeBased">
                <property name="allowIfAllAbstainDecisions"  value="false"/>
                <property name="decisionVoters"
                   <list>
                        <bean class="gov.nih.nci.security.acegi.csm.vote.AuthorizationSwitchVoter">
                            <property name="authorizationSwitch"  ref="authorizationSwitch"/>
                        </bean>
   <bean class="gov.nih.nci.security.acegi.csm.vote.CSMAuthorizationCheckVoter">
                            <property name="processConfigAttribute"  value="CSM_FILTER_CHECK"/>
                            <property name="authorizationCheck"  ref="filterAuthorizationCheck"/>
                        </bean>
       </list>
                </property>
            </bean>
```

The `accessDecisionManager` passes information about an authenticated user (including the user's business roles) and the `objectDefinitionSource` object to a voter

Example of Voter

```
public class CSMAuthorizationCheckVoter implements AccessDecisionVoter {

 public boolean supports(ConfigAttribute config) {
  return config.getAttribute().equals(getProcessConfigAttribute());
 }


 public int vote(Authentication authentication, Object object,
   ConfigAttributeDefinition configDef) {

  if(getAuthorizationCheck().checkAuthorization(authentication, getRequiredPrivilege(), object)){
   return AccessDecisionVoter.ACCESS_GRANTED;
  }else{
   return AccessDecisionVoter.ACCESS_DENIED;
  }
 }
}
```

⚠ Although access decision voters don't have the final say on whether access is granted to a secured resource (that job belongs to the access decision manager), they play an important part in the access decision process. An access decision voter's job is to consider the user's granted authorities alongside the authorities required by the configuration attributes of the secured resource. Based on this information, the access decision voter casts its vote for the access decision manager to use in making its decision.

External tutorials

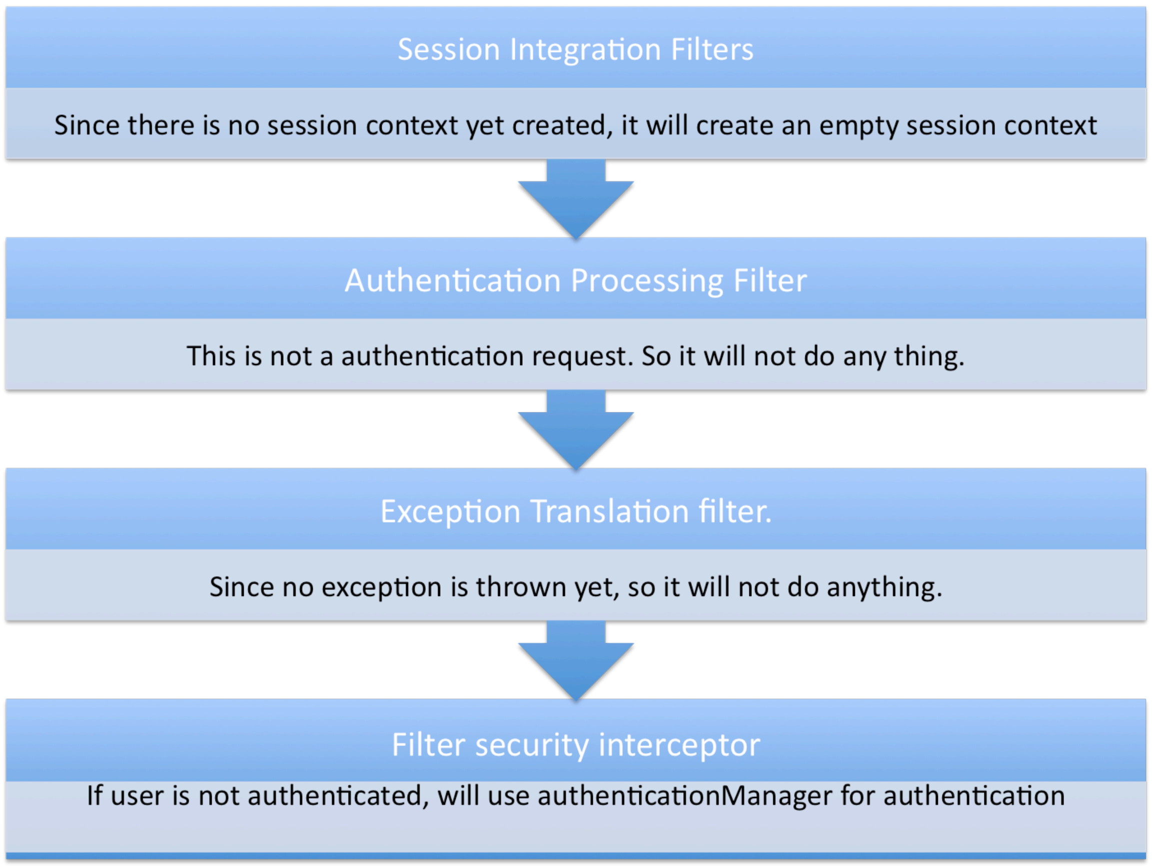
1. http://www.ibm.com/developerworks/java/library/j-acegi1/ : This link has sets of really very good tutorials on Acegi security framework.
2. Chapter 7 from Spring in Action book.

Work flows

Use case 1- How does authentication happens?

For example: when a user is trying to directly hitting a url (ex: http://localhost:8080/caaers/pages/createStudy) without login to application

1. The following diagram explains how filters will decide if authentication needs to be done


Session Integration Filters
Since there is no session context yet created, it will create an empty session context
Authentication Processing Filter
This is not a authentication request. So it will not do any thing.
Exception Translation filter.
Since no exception is thrown yet, so it will not do anything.
Filter security interceptor
If user is not authenticated, will use authenticationManager for authentication

2. The following work flow explains how `authenticationManager` does the authentication

```
<bean id="authenticationManager" class="org.acegisecurity.providers.ProviderManager">
  <property name="providers">
   <list>
    <ref bean="s[authenticationMode]AuthenticationProvider"/>
    <bean class="org.acegisecurity.providers.anonymous.AnonymousAuthenticationProvider">
     <property name="key" value="anonymousAuthKey"/>
    </bean>
   </list>
  </property>
 </bean>
```

```
<bean id="localAuthenticationProvider" class=
"gov.nih.nci.security.acegi.csm.authentication.CSMAuthenticationProvider">
  <property name="csmAuthenticationManager">
   <bean class="gov.nih.nci.security.SecurityServiceProvider" factory-method=
"getAuthenticationManager">
    <constructor-arg ref="csmApplicationContextName"/>
   </bean>
  </property>
  <property name="userDetailsService" ref="csmUserDetailsService">
  </property>
  <property name="userCache" ref="userCache"/>
 </bean>
```

```
<bean id="csmUserDetailsService" class=
"gov.nih.nci.security.acegi.csm.authorization.CSMUserDetailsService">
  <property name="csmUserProvisioningManager" ref="csmUserProvisioningManager"/>
 </bean>
```

`csmUserDetailsService` loads the `UserDetails` from database. If you want to user own authentication implementation, implement

`UserDetailsService` interface and implement `loadUserByUserName` method. See acegi security tutorials for more details.

Use case 2- How does authorization happens? How does url security work?

For example: when the user is trying to directly hitting a url (ex: http://localhost:8080/caaers/pages/createStudy) without login to application.

1. First the user must be authenticated. See previous use case which explains how authentication happens.

2. Now once user is authenticated, `org.acegisecurity.intercept.web.FilterSecurityInterceptor` intercepts request and takes authorization decision. We have configured this security interceptor in caAERS as following

```
<bean id="filterInvocationInterceptor" class=
"org.acegisecurity.intercept.web.FilterSecurityInterceptor">
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="accessDecisionManager">
   <bean class="org.acegisecurity.vote.AffirmativeBased">
    <property name="allowIfAllAbstainDecisions" value="false"/>
    <property name="decisionVoters">
     <list>
      <bean class="gov.nih.nci.security.acegi.csm.vote.AuthorizationSwitchVoter">
       <property name="authorizationSwitch" ref="authorizationSwitch"/>
      </bean>
      <bean class="gov.nih.nci.security.acegi.csm.vote.CSMAuthorizationCheckVoter">
       <property name="processConfigAttribute" value="CSM_FILTER_CHECK"/>
       <property name="authorizationCheck" ref="filterAuthorizationCheck"/>
      </bean>
     </list>
    </property>
   </bean>
  </property>
<property name="objectDefinitionSource">
   <value>
    CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
    \A/pages/.*\Z=CSM_FILTER_CHECK
   </value>
  </property>
 </bean>
```

3. This `accessDecisionManager` property of `FilterSecurityInterceptor` actually responsible for authorization decision. As explained above in the section of acegi security, it uses voters for its authorization decision. We are using `org.acegisecurity.vote.AffirmativeBased` as our `accessDecisionManager` which means- it will that grants access if any voter returns an affirmative response.

So **<u>authorization decision depend totally upon how voters are used and configured</u>** since `accessDecisionManager` used in caAERS will that grants access if any voter returns an affirmative response. So lets look at each voter and see how it actually works

1.
`gov.nih.nci.security.acegi.csm.vote.AuthorizationSwitchVoter`

```
<bean class="gov.nih.nci.security.acegi.csm.vote.AuthorizationSwitchVoter">
 <property name="authorizationSwitch" ref="authorizationSwitch"/>
</bean>
```

```
public class AuthorizationSwitchVoter implements AccessDecisionVoter {

 private AuthorizationSwitch authorizationSwitch;

 private boolean requiresAuthentication = true;

 public boolean isRequiresAuthentication() {
  return requiresAuthentication;
 }

 public boolean supports(ConfigAttribute arg0) {
  return true;
 }
 public boolean supports(Class arg0) {
  return true;
 }

 public int vote(Authentication arg0, Object arg1,
   ConfigAttributeDefinition arg2) {
  int retVal = AccessDecisionVoter.ACCESS_ABSTAIN;
  if(!getAuthorizationSwitch().isOn()){

   logger.warn("###### AuthorizationSwitch is OFF #####");
   if(!isRequiresAuthentication() ||
     isRequiresAuthentication() &&
     SecurityContextHolder.getContext().getAuthentication() != null){
    logger.warn("###### AuthorizationSwitch ACCESS_GRANTED #####");
    retVal = AccessDecisionVoter.ACCESS_GRANTED;
   }

  }
  return retVal;
 }
}
```

The two main important methods are `supports` and `vote`. The first method Indicates whether Voter is able to vote or not and second method Indicates whether or not access is granted(see the java docs of `org.acegisecurity.vote.AccessDecisionVoter`

So if `authorizationSwitch` is "off"- it means this `AuthorizationSwitchVoter` will always return true and user can access the url. But if the `authorizationSwitch` is "on", this voter will always return false (or ACCESS_ABSTAIN). We have configured `authorizationSwitch` as following in caAERS

```
<bean id="authorizationSwitch"
          class="gov.nih.nci.security.acegi.csm.authorization.AuthorizationSwitch"
          factory-method="getInstance">
        <property name="on" value="true"/>
    </bean>
```

Since the switch is on, this voter will always return false (or ACCESS_ABSTAIN)

2.
`gov.nih.nci.security.acegi.csm.vote.CSMAuthorizationCheckVoter`
We have configured this voter as following in caAERS

```
<bean class="gov.nih.nci.security.acegi.csm.vote.CSMAuthorizationCheckVoter">
 <property name="processConfigAttribute" value="CSM_FILTER_CHECK"/>
        <property name="authorizationCheck" ref="filterAuthorizationCheck"/>
</bean>
```

```
public class CSMAuthorizationCheckVoter implements AccessDecisionVoter {

 private String processConfigAttribute;

 private CSMAuthorizationCheck authorizationCheck;

 private String requiredPrivilege;

 public String getRequiredPrivilege() {
  return requiredPrivilege;
 }

 public void setRequiredPrivilege(String requiredPrivilege) {
  this.requiredPrivilege = requiredPrivilege;
 }

 public boolean supports(ConfigAttribute config) {
  return config.getAttribute().equals(getProcessConfigAttribute());
 }

 public boolean supports(Class arg0) {
  return true;
 }

 public int vote(Authentication authentication, Object object,
   ConfigAttributeDefinition configDef) {

  if(getAuthorizationCheck().checkAuthorization(authentication, getRequiredPrivilege(), object)){
   return AccessDecisionVoter.ACCESS_GRANTED;
  }else{
   return AccessDecisionVoter.ACCESS_DENIED;
  }
 }

 public String getProcessConfigAttribute() {
  return processConfigAttribute;
 }

 public void setAuthorizationCheck(CSMAuthorizationCheck authorizationCheck) {
  this.authorizationCheck = authorizationCheck;
 }

}
```

Now lets explore how does the {{supports(ConfigAttribute config) }} method work

```
public boolean supports(ConfigAttribute config) {
  return config.getAttribute().equals(getProcessConfigAttribute());
 }
```

The value of `getProcessConfigAttribute()` is `CSM_FILTER_CHECK`. So if `ConfigAttribute` returns the same value (`CSM_FILTER_CHECK`), this voter will check for authorization otherwise this voter will not vote. This `ConfigAttribute` is created by the `objectDefinitionSource` property of web security interceptor. We have configured `objectDefinitionSource` as following

```
<property name="objectDefinitionSource">
   <value>
    CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
    \A/pages/.*\Z=CSM_FILTER_CHECK
   </value>
  </property>
```

See the second line `\A/pages/.*\Z` is java expression which means "all the Strings which has '/pages' ". So `objectDefinitionSource` creates `ConfigAttribute` with the value of `CSM` for all the urls which have '/pages' string. See the java docs for more information.

Now lets explore how `vote(Authentication authentication, Object object, ConfigAttributeDefinition configDef)` method works

```
public int vote(Authentication authentication, Object object,
   ConfigAttributeDefinition configDef) {

  if(getAuthorizationCheck().checkAuthorization(authentication, getRequiredPrivilege(), object)){
   return AccessDecisionVoter.ACCESS_GRANTED;
  }else{
   return AccessDecisionVoter.ACCESS_DENIED;
  }
 }
```

You can notice that it uses the `CSMAuthorizationCheck#.checkAuthorization()` method for voting.

Summary of how Url security works

`filterInvocationInterceptor` is responsible for securing url access. It uses `authenticationManager` to authenticate the users and `accessDecisionManager` for authorization decision. It uses `objectDefinitionSource` to decide which 'security check' should get applied for which url. For ex: in caAERS, it will apply `CSM_FILTER_CHECK` for all urls having '/pages' string.

⚠ Applying `CSM_FILTER_CHECK` security check means- `objectDefinitionSource` will create an object of `ConfigAttribute` with the value as `CSM_FILTER_CHECK`. This 'security check' or `ConfigAttribute` is used by the voters to decide if voter should vote or not.

Following is the configuration of web security interceptor

```
<bean id="filterInvocationInterceptor" class=
"org.acegisecurity.intercept.web.FilterSecurityInterceptor">
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="accessDecisionManager">
   <bean class="org.acegisecurity.vote.AffirmativeBased">
    <property name="allowIfAllAbstainDecisions" value="false"/>
    <property name="decisionVoters">
     <list>
      <bean class="gov.nih.nci.security.acegi.csm.vote.AuthorizationSwitchVoter">
       <property name="authorizationSwitch" ref="authorizationSwitch"/>
      </bean>
      <bean class="gov.nih.nci.security.acegi.csm.vote.CSMAuthorizationCheckVoter">
       <property name="processConfigAttribute" value="CSM_FILTER_CHECK"/>
       <property name="authorizationCheck" ref="filterAuthorizationCheck"/>
      </bean>
     </list>
    </property>
   </bean>
  </property>
  <property name="objectDefinitionSource">
   <value>
    CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
    \A/pages/.*\Z=CSM_FILTER_CHECK
   </value>
  </property>
 </bean>
```

We are using `org.acegisecurity.vote.AffirmativeBased` as our `accessDecisionManager` which means - it will that grants access if any voter returns an affirmative response. We have configured two voters `AuthorizationSwitchVoter` and `CSMAuthorizationCheckVoter`. `AuthorizationSwitchVoter` will always return `ACCESS_ABSTAIN` because security switch is 'on'. The second voter `CSMAuthorizationCheckVoter` will vote only if `ConfigAttribute` has value equals to `CSM_FILTER_CHECK`. So this voter actually votes for all the urls in caAERS which has '/pages' string because `objectDefinitionSource` will create an object of `ConfigAttribute` with the value as `CSM_FILTER_CHECK`.

This `filterAuthorizationCheck` is configured as following. I will explain how `filterAuthorizationCheck` and `CSMAuthorizationCheck#.checkAuthorization()` works in next use case.

```
<bean id="filterAuthorizationCheck" autowire="byType" class=
"gov.nih.nci.security.acegi.csm.authorization.DelegatingObjectPrivilegeCSMAuthorizationCheck">
  <property name="csmAuthorizationCheck">
   <bean class="gov.nih.nci.security.acegi.grid.authorization.CSMGridGroupAuthorizationCheck">
    <property name="csmUserProvisioningManager" ref="csmUserProvisioningManager"/>
    <property name="authorizationSwitch" ref="authorizationSwitch"/>
   </bean>
  </property>
  <property name="objectIdGenerator" ref="filterPrivilegeAndObjectIdGenerator"/>
  <property name="privilegeGenerator" ref="filterPrivilegeAndObjectIdGenerator"/>
 </bean>
```

Use case 3: How does the `CSMAuthorizationCheck#.checkAuthorization()}` method works

For Example: when avuser is trying to directly hit a url (example: http://localhost:8080/caaers/pages/createStudy) without logging in to application.

1. First the user must be authenticated. See previous use cases which explains how authentication happens.

2. Now once user is authenticated, `org.acegisecurity.intercept.web.FilterSecurityInterceptor` intercepts request and takes authorization decision. `accessDesionManager` will ask voters to vote. `AuthorizationSwitchVoter` will return `ACCESS_ABSTAIN`.

So user can access the url only if `CSMAuthorizationCheckVoter` returns `ACESS_GRANTED`. This `CSMAuthorizationCheckVoter` uses `filterAuthorizationCheck` property for voting. So lets explore how this `filterAuthorizationCheck` property of `CSMAuthorizationCheckVoter` take authorization decision.

The following is the configuration of this property

```
<bean id="filterAuthorizationCheck" autowire="byType" class=
"gov.nih.nci.security.acegi.csm.authorization.DelegatingObjectPrivilegeCSMAuthorizationCheck">
  <property name="csmAuthorizationCheck">
   <bean class="gov.nih.nci.security.acegi.grid.authorization.CSMGridGroupAuthorizationCheck">
    <property name="csmUserProvisioningManager" ref="csmUserProvisioningManager"/>
    <property name="authorizationSwitch" ref="authorizationSwitch"/>
   </bean>
  </property>
  <property name="objectIdGenerator" ref="filterPrivilegeAndObjectIdGenerator"/>
  <property name="privilegeGenerator" ref="filterPrivilegeAndObjectIdGenerator"/>
 </bean>
```

This bean uses `csmAuthorizationCheck`, `objectIdGenerator`, `privilegeGenerator`. So let first see what are these `objectIdGenerator` and `privilegeGenerator`.

Following is the configuration of `objectIdGenerator` and `privilegeGenerator` (both uses the same class)

```
<bean id="filterPrivilegeAndObjectIdGenerator" class=
"gov.nih.nci.cabig.caaers.web.security.FilterInvocationPrivilegeAndObjectIdGenerator">
  <property name="objectPrivilegeMap" ref="filterObjectPrivilegeMap"/>
 </bean>
```

`FilterInvocationPrivilegeAndObjectIdGenerator` is used to generate privileges required to access a particular secured object. This secured object could be a url if you are securing the urls or domain object if you are securing domain objects. But the idea is to generate the privilege to access a particular secured object.

`FilterInvocationPrivilegeAndObjectIdGenerator` uses `filterObjectPrivilegeMap` for generating privileges.

```
<util:map id="filterObjectPrivilegeMap" map-class="java.util.LinkedHashMap">
        <entry key="/pages/participant/create.*" value=
"gov.nih.nci.cabig.caaers.domain.Participant:CREATE"/>
        <entry key="/pages/participant/view.*" value=
"gov.nih.nci.cabig.caaers.domain.Participant:READ"/>
        <entry key="/pages/participant/assign.*" value=
"gov.nih.nci.cabig.caaers.domain.Participant:UPDATE"/>
        <entry key="/pages/study/create.*" value="gov.nih.nci.cabig.caaers.domain.Study:CREATE"/>
        <entry key="/pages/study/view.*" value="gov.nih.nci.cabig.caaers.domain.Study:READ"/>
        <entry key="/pages/study/search.*" value="gov.nih.nci.cabig.caaers.domain.Study:READ"/>
        <entry key="/pages/study/edit.*" value="gov.nih.nci.cabig.caaers.domain.Study:UPDATE"/>
        <entry key="/pages/ae/create.*" value=
"gov.nih.nci.cabig.caaers.domain.ExpeditedAdverseEventReport:CREATE"/>
        <entry key="/pages/ae/edit.*" value=
"gov.nih.nci.cabig.caaers.domain.ExpeditedAdverseEventReport:UPDATE"/>
        <entry key="/pages/ae/list.*" value=
"gov.nih.nci.cabig.caaers.domain.ExpeditedAdverseEventReport:READ"/>
        <entry key="/pages/ae/createRoutine.*" value=
"gov.nih.nci.cabig.caaers.domain.ExpeditedAdverseEventReport:CREATE"/>
        <entry key="/pages/ae/captureRoutine.*" value=
"gov.nih.nci.cabig.caaers.domain.ExpeditedAdverseEventReport:CREATE"/>
        <entry key="/pages/task" value="gov.nih.nci.cabig.caaers.Task:ACCESS"/>
        <entry key="/pages/admin/.*" value="gov.nih.nci.cabig.caaers.Admin:ACCESS"/>
        <!-- TODO: set up correct default -->
  <entry key="/pages/.*" value="gov.nih.nci.cabig.caaers.domain.Study:READ"/>
 </util:map
```

So if a user is trying to acsess '/pages/participant/create.**', the user must have 'CREATE' privileges on `gov.nih.nci.cabig.caaers.domain.Participant` protection element/group. Also, if a user is trying to acsess '/pages/participant/view.**', the user must have 'READ' privileges on `gov.nih.nci.cabig.caaers.domain.Participant` protection element/group

Use case 4: allowing all logged-in users for accessing a particular url

Example: all logged-in users should be able to access '/pages/task' url

The authorization decision totally depends on how voters vote, so use `org.acegisecurity.vote.AuthenticatedVoter` to implement this functionality.

```
<bean id="filterInvocationInterceptor" class=
"org.acegisecurity.intercept.web.FilterSecurityInterceptor">
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="accessDecisionManager">
   <bean class="org.acegisecurity.vote.AffirmativeBased">
    <property name="allowIfAllAbstainDecisions" value="false"/>
    <property name="decisionVoters">
     <list>
      <bean class="gov.nih.nci.security.acegi.csm.vote.AuthorizationSwitchVoter">
       <property name="authorizationSwitch" ref="authorizationSwitch"/>
      </bean>
      <bean class="gov.nih.nci.security.acegi.csm.vote.CSMAuthorizationCheckVoter">
       <property name="processConfigAttribute" value="CSM_FILTER_CHECK"/>
       <property name="authorizationCheck" ref="filterAuthorizationCheck"/>
      </bean>
<bean class="org.acegisecurity.vote.AuthenticatedVoter">

      </bean>
     </list>
    </property>
   </bean>
  </property>
  <property name="objectDefinitionSource">
   <value>
    CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
    /pages/task=LOGIN_CHECK
\A/pages/.*\Z=CSM_FILTER_CHECK
   </value>
  </property>
 </bean>
```

Now all logged in user can access '/pages/task' url.

Check how PSC is using `org.acegisecurity.vote.RoleVoter` for implementing their own Role based authorization

CSM module

Please refer to the CSM user guide (CSM Guide For Application Developers)available at
https://gforge.nci.nih.gov/docman/index.php?group_id=12&selected_doc_group_id=2496&language_id=1

The CSM-Acegi integration guide is also available at (see caCORE_CSM_4 0_Technical Guide_113007.pdf)

Q&A

Does any application use its on authentication mechanism (or does not use CSMUserDetailsService)?

Ans: YES, PSC uses its own authentication mechanism. You can checkout code from
https://svn.bioinformatics.northwestern.edu/studycalendar/trunk . See `PSCUserDetailsService.java` class for more details.

Does any application use its on authorization mechanism (or does not use CSMAuthorizationCheckVoter)?

Ans: YES, PSC uses its own authorization mechanism. You can checkout code from
https://svn.bioinformatics.northwestern.edu/studycalendar/trunk . See `accessDecisionManager` in
`applicationContext-acegi-security.xml` for more details.

Documentation Plan - caAERS 1.x

Note
This page is an archive.

Documentation Category	Doc name	Type	Shared Source with?	Source/Location	Priority	Delivery Date
Application data						
	error messages			text files	high	Aug 29
	page instructions			spreadsheet	high	Aug 29
Help						
	field level			spreadsheet	med	
	AE module	robohelp online	User Guide?		high	
	Admin module	robohelp	Admin Guide		med	draft done
	Rules module	robohelp	Admin Guide?			
	Subject module	robohelp	Admin Guide?			
	Studies module	robohelp	Admin Guide?			
User Guides						
	Install Master doc	online			med	
	Install Guides (3)				high	drafts done
	Admin (Configuration) Guide	robohelp	Help files (Admin, rules, subject, studies modules)		high	draft done in word doc
	User Guide		Help files (AE module)		high	
Misc Documents						
	Architectural Doc					
	Requirement Document				high	Aug 15
Training						
	caAERS overview					
	AE module					

caAERS 1.x Documentation Updates needed

Note
This page is an archive.

Empty dynamic table removed; not supported by new version of Confluence.

Installation Guide for Developers - caAERS 1.x

Contents

Preliminary Considerations

Installing caAERS on a system already setup with the proper versions of the JDK, database, and Tomcat should require only about 30 minutes; installation from scratch could take several hours.
For installation on Windows, an account with administrator privileges must be used

Installing JDK

Step	Action
1	Download Java Standard Edition 5.0 or higher from http://java.sun.com/javase/downloads/index.jsp.
2	Install Java. Make sure neither the directory where you install java or the parent directories contain a white space in between any characters in the name of the directory.
3	Add the `${java.dir}\bin` to your system PATH.
4	Set a new environment variable named `JAVA_HOME`, whose value equals `${java.dir}`

Throughout this guide the java installation directory will be referred as

```
${java.dir}
```

Installing Tomcat

Step	Action
1	Download Apache Tomcat version 5.5.23 or higher from http://tomcat.apache.org/download-55.cgi. Add a new environment variable CATALINA_HOME, whose value must be set to `${tomcat.dir}`
2	Install the Tomcat application in a convenient location.

3	
4	

⚠ Throughout this guide the Tomcat installation directory will be referred as

`${tomcat.dir}.`

Installing the Database

⚠ caAERS supports two databases, Oracle and PostgreSQL. To successfully deploy caAERS, you need to have one of these databases installed on your system.
The instructions for installing an Oracle or Postgres database product are beyond the scope of this document. Refer to the installation documents that accompany these databases for step-by-step instructions.

In your database management system, create a new database named caaers. To do so, follow the steps outlined here.

Step	Action
1	Install the preferred database product (PostgreSQL 8.1.9 or Oracle 10g)
2	Create an empty database for caAERS to use. You may give your database any valid name, but this guide will use "caaers" as the name in each exam

Installing an SVN Client

To checkout the latest code, you need to have a Subversion (SVN) client installed on your system. Download and install the Subversion client provided below.

- Windows stand alone: http://tortoisesvn.tigris.org/
- Mac stand alone: http://www.apple.com/downloads/macosx/development_tools/svnx.html

Downloading the Latest Source Code

Step	Action
1	Once the SVN client is installed, create a directory in a convenient location to checkout the caAERS source code. \\\\
2	Right-click `WORKSPACE`, select SVN Checkout and enter the following repository URL: https://gforge.nci.nih.gov/svnroot/caaersappdev/trunk/

⚠ Throughout this guide the source code directory will be referred as `WORKSPACE`

Installing and Configuring the Database

caAERS supports two databases, Oracle and PostgreSQL. To successfully deploy caAERS, you need to have one of these databases installed on your system.

Step	Action
1	Install the preferred database product (PostgreSQL 8.1.9 or Oracle 10g)\\\\
2	Create an empty database for caAERS to use. You may give your database any valid name, but this guide will use "caaers" as the name in each example
3	Create a schema named "caaers" in the database

4	Create a superuser with privileges to access the database created in step 2

There is a readme.database in \caaers\projects\core that provides more details on how to do the configuration with Postgres and Oracle databases.

Configuring the Database for Use with caAERS

The following steps explain how to set up an etc/caaers directory, for the Postgres database

Steps	Action
1	Create the directory: c:\etc
2	Create a subfolder: caaers
3	Copy caaers.properties.sample into c:\etc\caaers\ and rename it datasource.properties
4	Edit datasource.properties attributes to match the "caaers" database. • Set `datasource.url` to the JDBC URL for your database • Set the username and password to use to access this URL • Uncomment the database configuration block that corresponds to your database type

How to setup caAERS on Eclipse and build project?

see following link Ivy Integration

Once you reach here, please ask Biju or Karthik on how to use automatic installer for installing caAERs. All the steps mentioned above were pre-requisite for installing the caAERS

Important ANT commands

1. To create a war file

go to the project root

```
ant publish-all -Dskip.test=true
```

This command will create a caaers.war file in web/build/dist folder. You can copy this war file and deploy it to tomcat/webapps directory.

2. To insert data in the database

execute following command

```
ant migrate
```

3. to add default users for caAERS web application

execute following command

```
ant insert-csm-policy
```

important mvn commands

1. To create a war file

go to web project and execute following command

```
mvn install -Dmaven.test.skip=true
```

This command will create a caaers.war file in web/target folder. You can copy this war file and deploy it to tomcat/webapps directory.

2. To insert data in the database

execute following command from core project

```
mvn initialize bering:migrate
```

3. to add default users for caAERS web application

execute following command from core project

```
mvn -P csm-policy initialize
```

4. To create a test-jar file of caaers-core

go to core module and execute following command

```
mvn -Ptest install -Dmaven.test.skip=true
```

This command will create a test-jar file of caaers-core module.

caAERS default users

username	password
SYSTEM_ADMIN	system_admin

caAERS Rules Design

Technical Specification

Version History		
Version	**Date Modified**	**Author**
0.1	10/18/07	Vinay Kumar
0.2	10/18/07	Srinivasa Akkala
0.3	10/18/07	Karthik Iyer
0.4	10/19/07	Ram Chilukuri
0.5	10/22/07	Karthik Iyer
Contacts and Support	
Vinay Kumar
vinay.kumar@semanticbits.com|NA|
Srinivasa Akkala

srini.akkala@semanticbits.com|NA|
LISTSERV facilities pertinent to software teams		
LISTSERV	URL	Name
Caaersappdev-technical	caaersappdev-technical@gforge.nci.nih.gov.	caAERS technical forum
Table of Contents

List of Figures

h1.Scope and Purpose
This document gives the details of the rules module from architectural and design perspective. The scope of this document is limited to the design and implementation of the rules module. The document will also discuss the testing approaches for this module.
h1.Overview
Rules that govern SAE reporting are complex and could vary from sponsor to sponsor, study to study and healthcare site to healthcare site. When certain criteria are satisfied a specific type of report should be submitted to variety of entities like the funding sponsor, coordinating center etc. That is, the type of SAE report that needs to be submitted to a sponsor or to any other interested agency may depends on a number of criteria. These criteria may also differ from study to study. Some studies may have much more strict criteria and some others may define relaxed criteria to reduce the number of SAE reports. Attributes from domain objects such as Study, AdverseEvent, etc. form the basis for defining the criteria. A rule is a set of criteria with one or more actions, which get executed when all the criteria in that rule are satisfied.
There are numerous use cases in caAERS, which dictate some kind of notification to be sent to interested parties by the system. In other cases, the system might be required to invoke some action/actions when certain pre-defined conditions are met. These conditions are based on the state of the domain objects. When a domain object achieves a predefined state then system has to perform preconfigured action(s). Definition of these conditions may change from time to time and/or from adopter to adopter, which introduces a lot of agility in business logic. Implementing the business rules using the classical approach of coding them as part business logic layer will necessitate changing the code and re-deployment of the application when rules change, which is not desirable and will lead to a maintenance nightmare.
Given the above problem statement, the caAERS team chose to implement the rules engine by integrating an open source rules engine called Java Rules (http://jcp.org/en/jsr/detail?id=94). The adoption of this approach gives the desired flexibility to caAERS adopters in defining rules, condition and pluggable actions. This will ease the process of introducing new rules or retiring old rules without redeploying the entire application. Rules and actions will be accessible to users in addition to system administrators. As a part of this effort, a Business Rule Management System (BRMS) has been developed and deployed as a part of caAERS application.
h2.Features
Following is the list of the features introduced with rules module
# A web-based user interface for provisioning rules. This provides functionality to create, update, enable , disable and delete rules.
# Run time environment for execution of rules
# Grouping the rules based on their types (SAE Reporting Rules vs. Mandatory Sections Rules)
# Categorization of rules (Sponsor level rules vs. Institution level rules)
# Rules for assessing the adverse event
# Rules for report scheduling and notification

h2.Assumptions
h1.Design Approach
We define some terms and concepts before discussing the design approach. These terms and concepts will be used throughout the document.

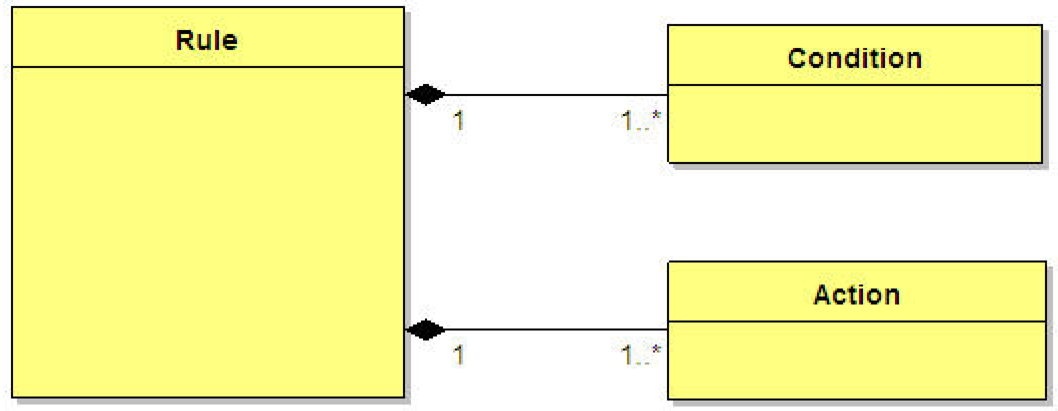
h2.Terms and Concepts
h3.Rule
A rule is a definition that is applied to data to produce outcomes. A rule in caAERS has a two part structure consisting of a set of conditions and a set of actions. Conditions are defined using the definition of domain objects.
Following is an example of a rule:
Rule R1
if condition C1 and C2 are true
Fire Action A1, A2 and A3
.


Figure 1: Rules are composed of conditions and invoke actions when triggered.
h3.Rule Set
A rule set is a collection of related rules. For example one can define a rule set called "**Adverse Event Assessment Rule Set**" which has all the rules related to assessing the adverse event.
11...* **Rule RuleSet**
Figure 2: A ruleset is a collection of related rules.
h2.High Level Architecture
The rules module consist of following components:
* Web interface to provision the rules
* Business service layer
* Rules run time environment
* Rules persistence

The diagram below shows the above mentioned components
Rules Auditing ModuleWeb Interface **JBoss Rules** Rules run time Database
Figure 3: High level architecture of rules module

Domain Objects

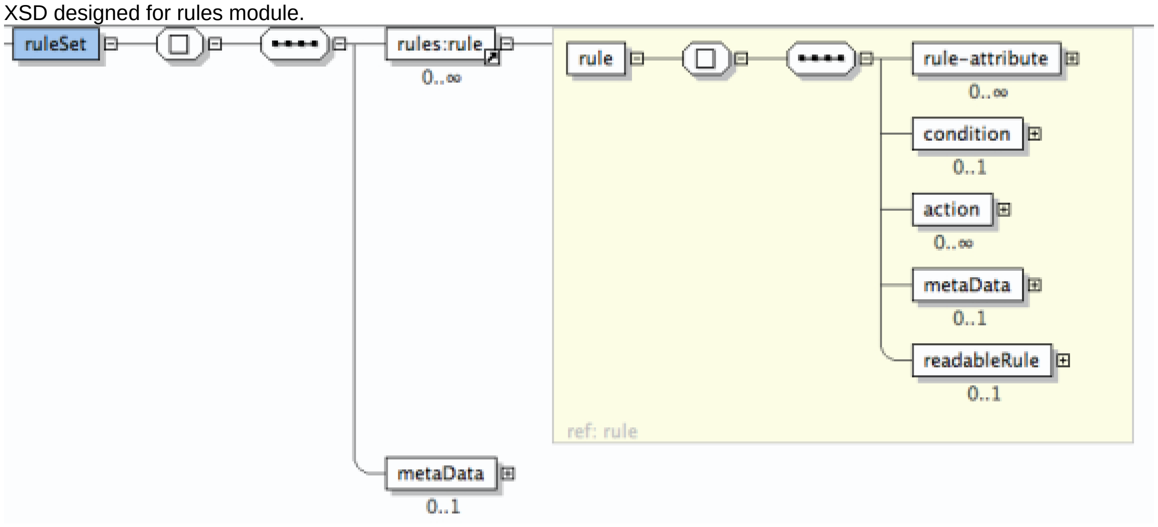
An XSD for rules is defined as part of rules module. Domain objects are generated from this XSD using JAXB (Java XML Binding). XML data binding refers to the process of representing the information in an XML document as an object. The following figure shows the high level view of XSD designed for rules module.


Figure 4: Rule set XML schema
The rules domain model is an abstraction layer which sits on top of the JBoss Rules object model. This not only eliminates the complexities involved in using the JBoss object model but also helps in providing a layer of indirection.
caAERS_rules_base
gov/nih/nci/cabig/caaers/rulesSponsor

/sponsorInstitution
/institutionSponsor 1
/sponsor1Sponsor 2
/sponsor2
Study
/study
Ruleset 1
/ruleset1Ruleset 2
/ruleset2Ruleset 1
/ruleset1
Ruleset 2
/ruleset2Study 1
/study1Study 2
/study2Sponsor 1
/sponsor1Sponsor 2
/sponsor2
Institution 1
/instituition1Institution 2
/instituition2
Study
/study
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Study 1
/study1Study 2
/study2Institution 1
/instituition1
Institution 2
/instituition2
Institution 1
/instituition1
Institution 2
/instituition2
Sponsor 1
/sponsor1Ruleset 1
/ruleset1Ruleset 2
/ruleset2
Sponsor 2
/sponsor2
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Ruleset 1
/ruleset1
Ruleset 1
/ruleset1
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Ruleset 1
/ruleset1
Ruleset 2
/ruleset2
Sponsor level rules
Institution level rules

Sponsor level study specific rules
Institution level study specific rules
Figure 5: Categorization of rules based on whether they are specific to a sponsor, institution, or study

Persistence Layer

Rules in caAERS are stored in a content repository. Apache Jackrabbit - a fully conformant implementation of the Content Repository for Java Technology API (JCR) - is being used in caAERS.
Jackrabbit provides persistence manager components that facilitate the storage of content nodes and properties. Rule sets represent the content in caAERS.
The **org.apache.jackrabbit.core.persistence.db.SimpleDbPersistenceManager** class implements a generic Java Database Connectivity (JDBC) based persistence manager. Content is stored in a set of simple non-normalized tables that are created automatically.

Service Layer

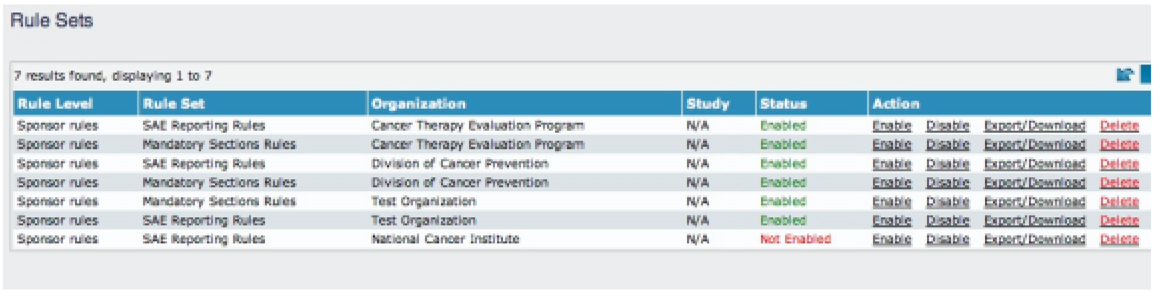
The service layer consists of set of business services, each of which contains a number of coarse-grained methods that implement a group of related use cases.
Service layer is comprised of following services.
RulesEngineService - This service implements use cases such as

- Create Rule set
- Update Rule set
- Enable and Disable Rule set
- Delete Rule set
- Export/Import Rule set

AdverseEventEvaluationService - This service evaluates adverse events for seriousness, fires the configured rules and invokes the required actions.

Presentation/UI Layer

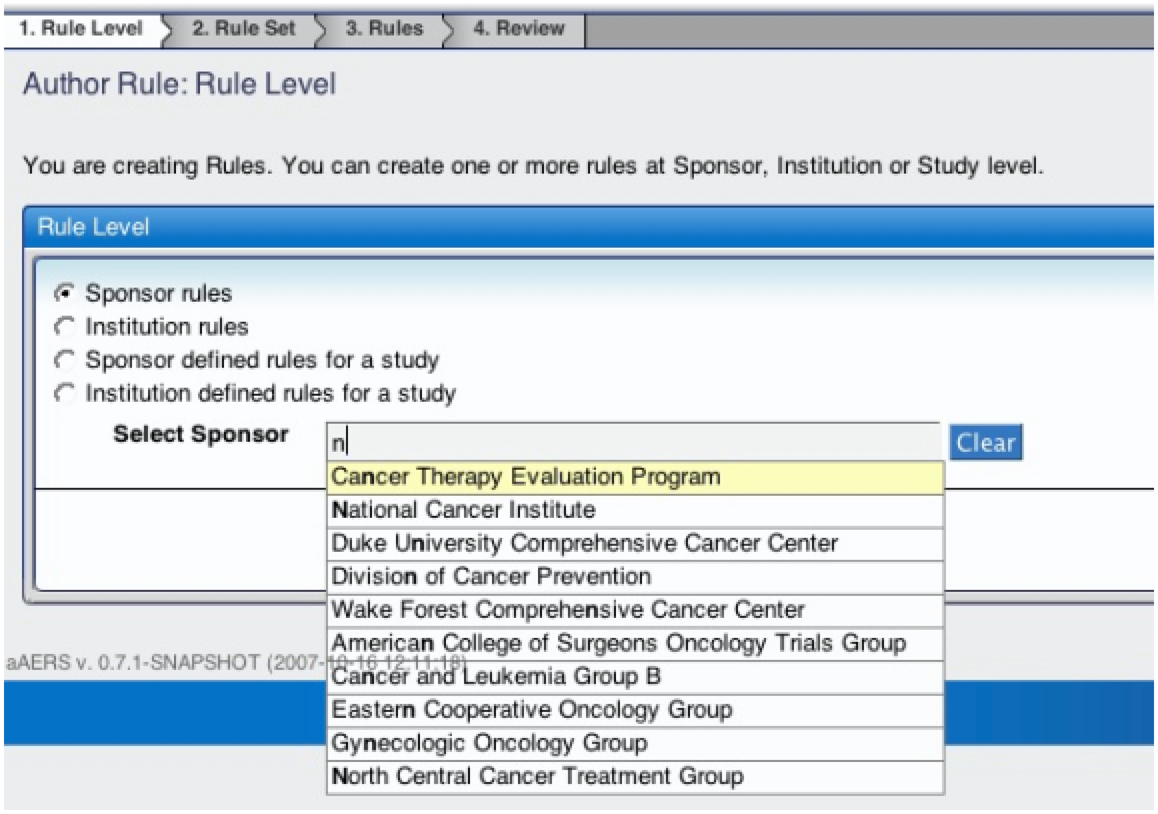
The presentation layer contains web components such as Java Server Pages (JSP) and Java servlets that can generate dynamic HTML content. The components in this layer are responsible for providing rule creation flow, deployment, import / export of rule sets. This layer is developed using Spring MVC framework and AJAX.
Below are the few screen shots of Rule Create/Edit Flow.


Rule Sets

7 results found, displaying 1 to 7

Rule Level	Rule Set	Organization	Study	Status	Action
Sponsor rules	SAE Reporting Rules	Cancer Therapy Evaluation Program	N/A	Enabled	Enable Disable Export/Download Delete
Sponsor rules	Mandatory Sections Rules	Cancer Therapy Evaluation Program	N/A	Enabled	Enable Disable Export/Download Delete
Sponsor rules	SAE Reporting Rules	Division of Cancer Prevention	N/A	Enabled	Enable Disable Export/Download Delete
Sponsor rules	Mandatory Sections Rules	Division of Cancer Prevention	N/A	Enabled	Enable Disable Export/Download Delete
Sponsor rules	Mandatory Sections Rules	Test Organization	N/A	Enabled	Enable Disable Export/Download Delete
Sponsor rules	SAE Reporting Rules	Test Organization	N/A	Enabled	Enable Disable Export/Download Delete
Sponsor rules	SAE Reporting Rules	National Cancer Institute	N/A	Not Enabled	Enable Disable Export/Download Delete

Figure 6: List rulesets page.



Author Rule: Rule Level

You are creating Rules. You can create one or more rules at Sponsor, Institution or Study level.

Rule Level

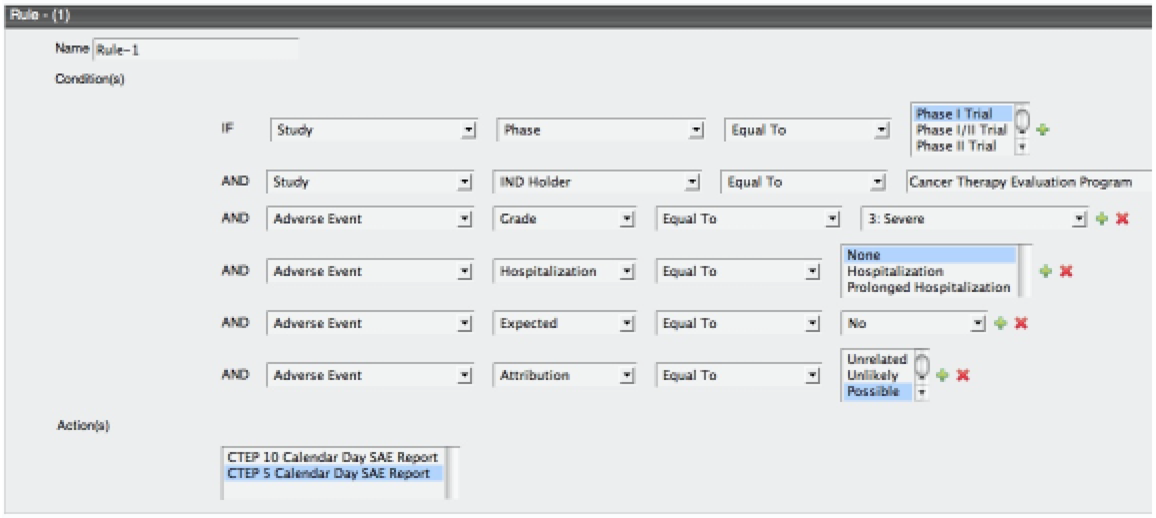
[✓ Sponsor rules]
Institution rules
Sponsor defined rules for a study
Institution defined rules for a study

Select Sponsor: n

Cancer Therapy Evaluation Program
National Cancer Institute
Duke University Comprehensive Cancer Center
Division of Cancer Prevention
Wake Forest Comprehensive Cancer Center
American College of Surgeons Oncology Trials Group
Cancer and Leukemia Group B
Eastern Cooperative Oncology Group
Gynecologic Oncology Group
North Central Cancer Treatment Group

aAERS v. 0.7.1-SNAPSHOT (2007-10-16 12:11:18)

Figure 7: Select rule level page.


Rule - (1)

Name: Rule-1

Condition(s)

IF	Study	Phase	Equal To	Phase I Trial, Phase I/II Trial, Phase II Trial
AND	Study	IND Holder	Equal To	Cancer Therapy Evaluation Program
AND	Adverse Event	Grade	Equal To	3: Severe
AND	Adverse Event	Hospitalization	Equal To	None, Hospitalization, Prolonged Hospitalization
AND	Adverse Event	Expected	Equal To	No
AND	Adverse Event	Attribution	Equal To	Unrelated, Unlikely, Possible

Action(s)

CTEP 10 Calendar Day SAE Report
CTEP 5 Calendar Day SAE Report

Figure 8: Create/edit rules page.

Implementation

Domain Model

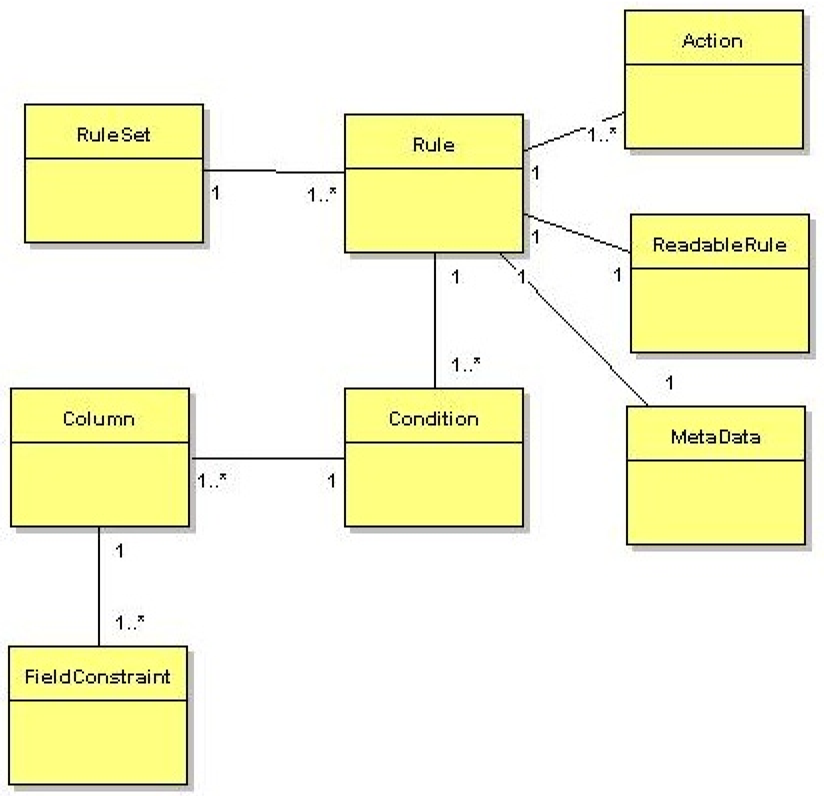


Figure 9: Domain model for rules.

Sequence Diagram

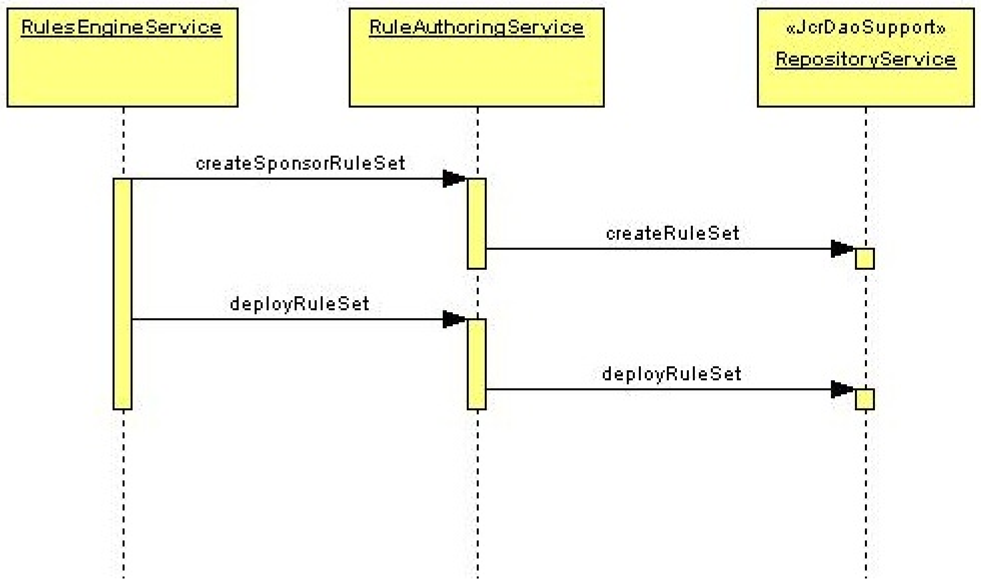
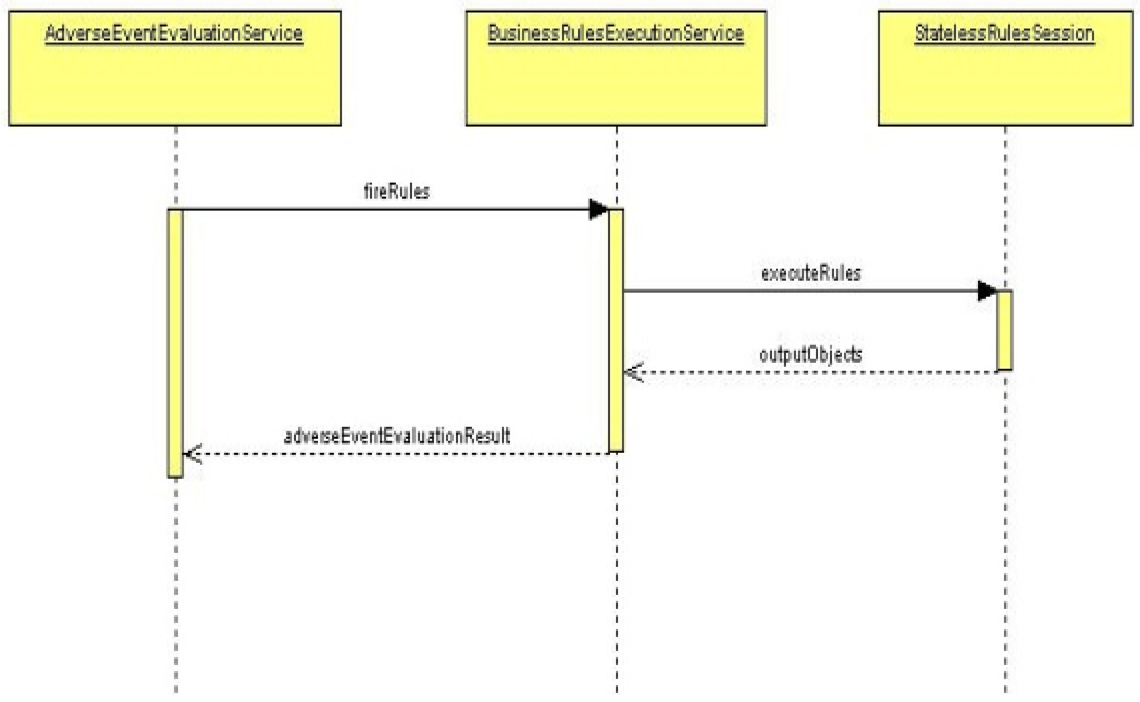


Figure 10: Sequence diagram showing rule creation and deployment.


AdverseEventEvaluationService
BusinessRulesExecutionService
StatelessRulesSession
fireRules
executeRules
outputObjects
adverseEventEvaluationResult
Figure 11: Sequence diagram showing rule execution.

Activity Diagram

caAERS supports two type of rule sets: SAE Reporting Rules and Mandatory Sections Rules. Following sections provide more details about these rule set types.

SAE Reporting Rules

This type of rules is being used to determine if an adverse event qualifies as a serious adverse event and thus requires expedited reporting. They indicate the appropriate reports that need to be submitted to various agencies such as the sponsor of the study, coordinating center for the study etc. SAE reporting rules are invoked in the Document AE's and Enter expedited report workflows. Report definitions are specified as actions for rules of this type.
Following table enumerates the list of domain objects and their attributes that can be leveraged in defining the SAE reporting rules:

Domain Object	**Attributes**
Adverse Event	• Grade • Hospitalization • Expected • Attribution • CTC Category • CTC Term • MedDRA Term
Study	• Phase • IND holder • Therapy • Treatment Assignment Code

Table 1: Domain objects and their attributes used in SAE reporting rules

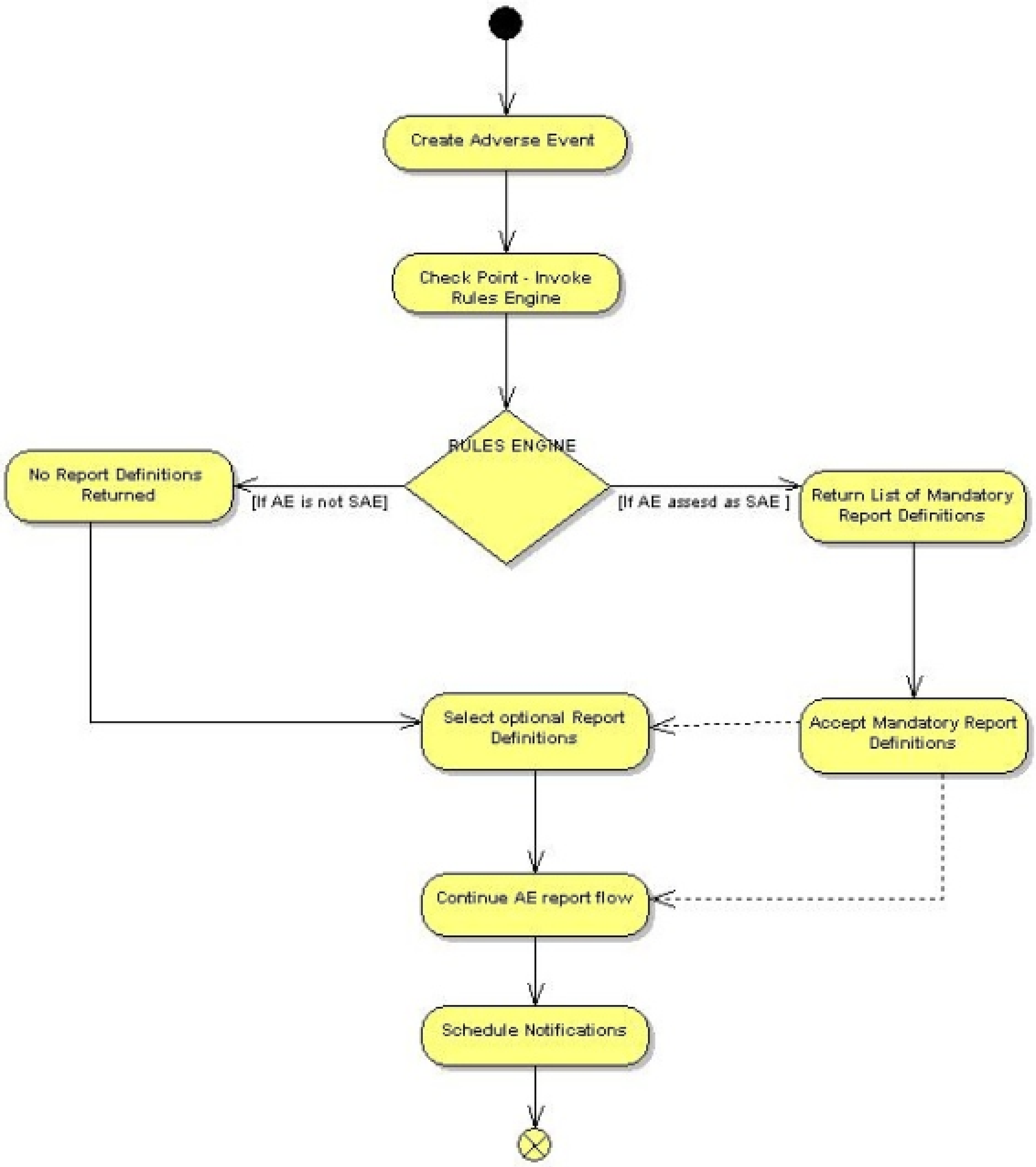

Create Adverse Event
Check Point - Invoke Rules Engine
RULES ENGINE
No Report Definitions Returned
[If AE is not SAE]
[If AE assesd as SAE]
Return List of Mandatory Report Definitions
Select optional Report Definitions
Accept Mandatory Report Definitions
Continue AE report flow
Schedule Notifications

Figure 12: Flowchart for the SAE reporting rules workflow.

Mandatory Section Rules

This type rules aid in determining what sections are mandatory for a given report definition. These are invoked during the "Enter expedited report" workflow just after the reporter selects the report definitions. caAERS will permit the submission of these reports only after all the mandatory sections are completed.
Following table enumerates the list of domain objects and their attributes that can be leveraged in defining the mandatory section rules:

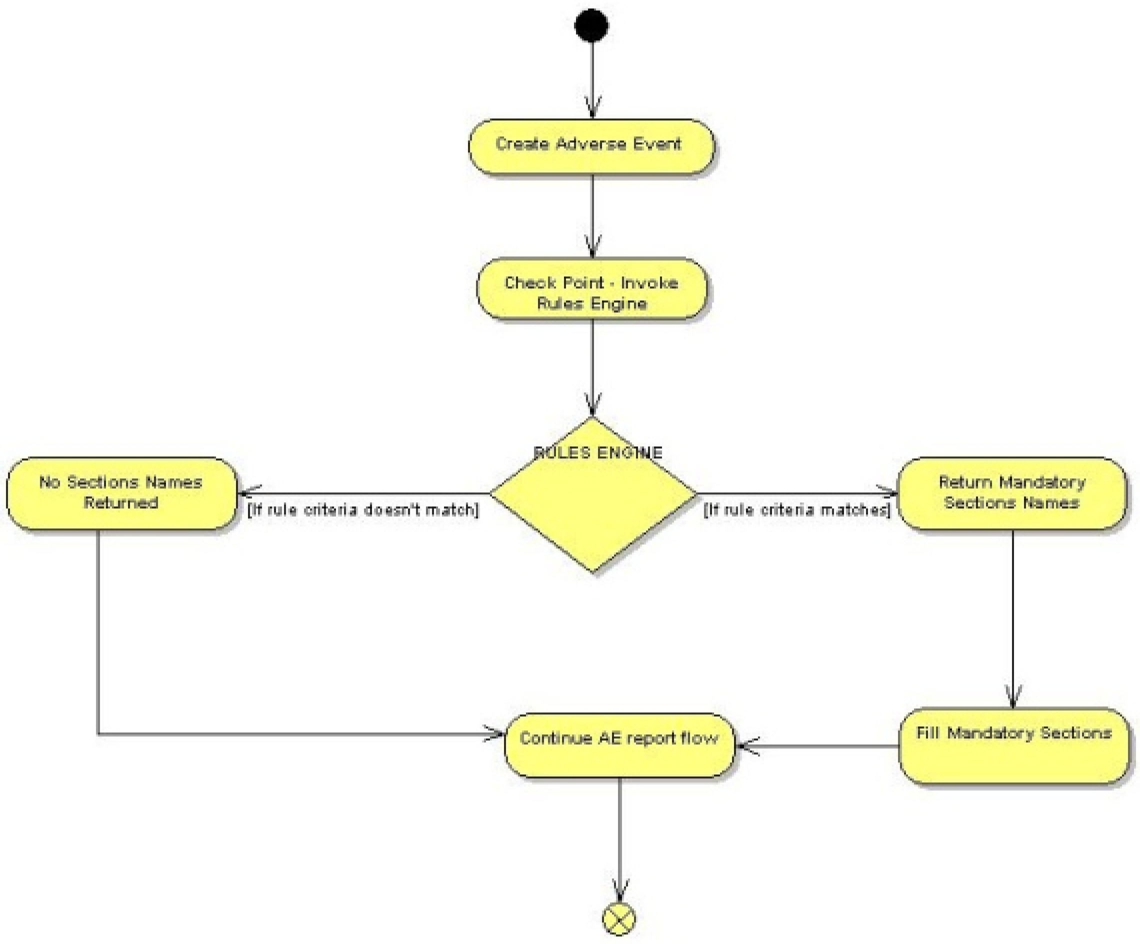

Create Adverse Event
Check Point - Invoke Rules Engine
RULES ENGINE
No Sections Names Returned
[If rule criteria doesn't match]
[If rule criteria matches]
Return Mandatory Sections Names
Continue AE report flow
Fill Mandatory Sections

Figure 13: Flowchart for the Mandatory sections rules workflow.

Domain Object	Attributes
Adverse Event	• Grade • Hospitalization • Expected • Attribution
Study	• Phase • IND holder • Therapy • Treatment Assignment Code
Report Definition	• Name

Table 2: Domain objects and their attributes used in mandatory sections rules

Coding Practice - Handling Hibernate Embeded Objects

Recently while we were on Bang-a-Thon, a lot of us has faced error saying "You cannot perform this operation, as another user is modifying the same details".

See http://jira.semanticbits.com/browse/CAAERS-2878

Reason:

The ResearchStaff domain object marked dirty at the load.This made hibernate to update the ResearchStaff, and that cascaded child entities.

Why:

A new instance of Address embedded object was getting instantiated in the research staff loaded from DB, whose address was empty.

See the code below:

```
public abstract class ResearchStaff extends User {
    ......
    ......

 @Embedded
 public Address getAddress() {
     if(address == null) address = new Address();
  return address;
 }
 public void setAddress(Address address) {
  this.address = address;
 }
}
```

The Fix :

Remove the "address == null" comparison from the getAddress() method.

```
public abstract class ResearchStaff extends User {
    ......
    ......

 @Embedded
 public Address getAddress() {
  return address;
 }
 public void setAddress(Address address) {
  this.address = address;
 }
}
```

Coding Practice - Transparently handling Hibernate Session, Security & Auditing

Transparently handle Hibernate Session , Security Authorization & Auditing

In some places in our code base
- Hibernate session is not alive, we used to explicitly open/close the hibernate session.
- Security information if not present, we explicitly logged-in as SYSTEM_ADMIN and turned of authorization.
- If auditing is not present, we explicitly created auditing context.

Example

```
<!-- My bean definition-->
<bean id="mybean" class="mypackage.MyBeanClass"/>
```

```
public class MyBeanClass {


     public void myMethod(){

       //1. Open the session
preProcess();
       //2. Initialize Security
enableAuthorization(false);
        switchUser("SYSTEM_ADMIN", "ROLE_caaers_super_user");
       //3. Initialize Auditing
......................
         ......................
        //EXECUTE MY LOGIC
.......................
        .......................

        //4. remove auditing from thread context
//5. remove security from thread context
enableAuthorization(.....);
          switchUser(null);
        //6. remove session from thread context
postProcess(.....);
     }
     //Method will open the hibernate session.
private WebRequest preProcess() {

        WebRequest stubWebRequest = new StubWebRequest();
        openSessionInViewInterceptor.preHandle(stubWebRequest);

        return stubWebRequest;
    }

    //Method will close the session
private void postProcess(WebRequest stubWebRequest) {
        openSessionInViewInterceptor.afterCompletion(stubWebRequest, null);
    }
 }
```

Try to leverage what we have configured

Well those are cross-cutting concerns and we should leverage what is configured at container level.

1. Configure all your cross-cutting concerns as AOP interceptors
   caAERS already has the following interceptors
   hibernateInterceptor - Takes care of opening/closing hibernate session
   runAsAuthenticationPopulatorInterceptor - Takes care of changing the authentication/roles
   auditInfoPopulatorInterceptor - Takes care of populating the auditing information.

2. Use Spring Proxy factory bean

```
<!-- New My bean definition-->
<bean id="mybeanTarget" class="mypackage.MyBeanClass"/>
<bean id="mybean" class="org.springframework.aop.framework.ProxyFactoryBean">
  <property name="proxyTargetClass"><value>true</value></property>
  <property name="target"><ref local="mybeanTarget"/></property>
  <property name="interceptorNames">
   <list>
    <value>runAsAutenticationProviderInterceptor</value>
    <value>auditInfoPopulatorInterceptor</value>
    <value>hibernateInterceptor</value>
   </list>
  </property>
 </bean>
```

```
public class MyBeanClass {

     public void myMethod(){
         ......................
         ......................
        //EXECUTE MY LOGIC
.......................
        .......................
     }

 }
```

Coding Practice - Use sequence in groovy

Recently while upgrading the Mayo Pilot demo box at SemanticBits to caAERS 2.1, we faced issues while bering update.Here after we should be cautious when using positive primary key values in groovy.

Seehttp://jira.semanticbits.com/browse/CAAERS-3882

Reason:

The a groovy inserting row into a table, was using a fixed positive primary key ID.

See the code below:

```
class AddOrganizationProtectionElement extends edu.northwestern.bioinformatics.bering.Migration {
    void up() {

   insert('csm_pg_pe',[pg_pe_id: 9012, protection_group_id: -5, protection_element_id: -7942],
      primaryKey: false);
    }

    void down(){
        ........
    }
}
```

The Fix :

Remove the positive primary key from insert value list and primaryKey:false attribute.

```
class AddOrganizationProtectionElement extends edu.northwestern.bioinformatics.bering.Migration {
    void up() {

   insert('csm_pg_pe',[ protection_group_id: -5, protection_element_id: -7942]);
    }

    void down(){
        ........
    }
}
```

Coding Style

Note :- The original document about coding style is available in GForge wiki.

Code Style for caAERS

In most cases, Java code for caAERS should follow the style outlined by_*Code Conventions for the Java Programming Language*_ from Sun.

In some areas, Sun does not make a recommendation. This document supplements the Sun conventions for those areas. It also emphasizes particularly important conventions.

Indenting and brackets

Code must be indented using spaces, four spaces per indent (no tabs). Lines longer than 80 characters should be avoided.

Brackets must be placed K&R-style:

```
if(operands.size() == 0) {
        throw new IllegalArgumentException("No operands specified");
    }
```

Brackets are required for all blocks, including single statement blocks:

```
if(operands.size() > 3)  // PROHIBITED
return null;
```

... unless the statement is on the same line as the condition:

```
if(operands.size() > 3) return null;  // OKAY
```

Note also section 7 of the Sun conventions.

Imports

Every class imported should get its own import statement -- avoid wildcard imports for classes. For static imports, wildcards are acceptable.

Comments

Readable code is better than a lot of comments. When the purpose or behavior of a local variable, field, method, or inner class is unclear from its name or context, the developer should consider how to make the code itself clearer. This might include renaming variables or refactoring into multiple methods or classes.

When the code is compact but complex, a brief comment describing its intent should be included. In no case should the entire approach be duplicated in comment(s) and code.

Javadoc

Every top-level class should have a javadoc comment with, at minimum, a list of the authors of that class:

```
/**
* @author Ram Chilukuri
* @author Rhett Sutphin
**/
```

When you make a substantial change to a class, add your name to the list of authors.

Individual fields, constructors, and methods do not need javadoc by default. In particular, disable any automatic javadoc generation in your IDE. If a method or field needs javadoc, add it, but don't clutter your code with automatically generated documentation stubs.

For JSP and TAG files.....

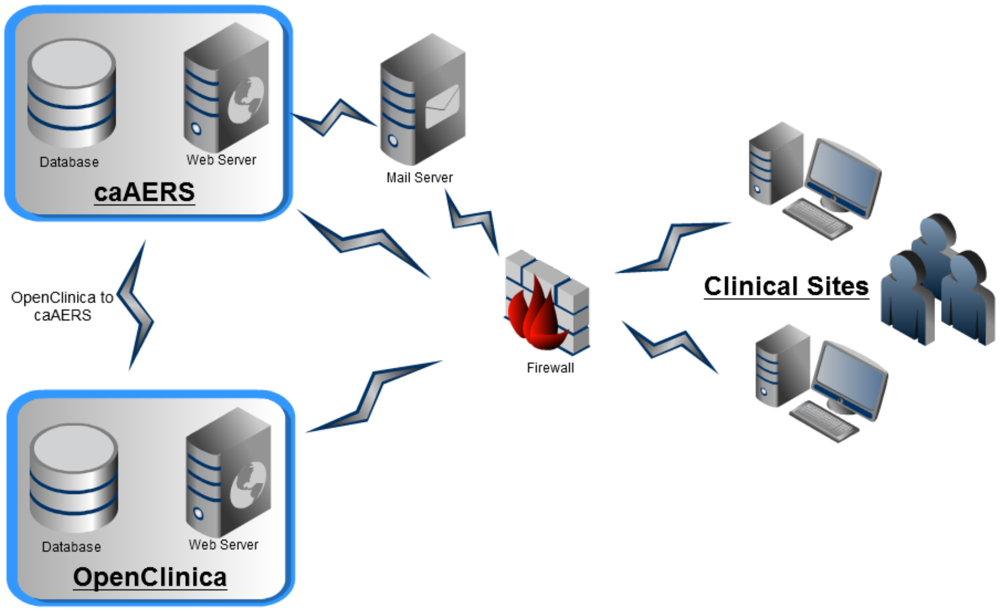
```
<%--
 The purpose of this page.
 Author : Vinay Kumar
--%>
```

Formatter -

Eclipse Codestyle

Note:- A code formating template for eclipse is available in
http://gforge.nci.nih.gov/svnroot/caaersappdev/docs/codestyle/caAERS_Code_Conventions.xml

Deployment



Design Document - Hosted Mode (caAERS)

The purpose of this page is to document the approach as to how a particular requirement or use case is being implemented. The owner of this document is the developer leading the implementation.

NOTE: This page is to serve as an ongoing record. It should be used as a whiteboard for brainstorming ideas and strategies and for documenting minutes from meetings and discussions. It should clearly show the evolution of the implementation approach.

1. Background / Introduction

In first phase

Implement page level and section level access based on rights and roles.

Implement instance based security in all searches and auto completers.

2. Use Cases Affected

All Searches

All Auto Completers

3. Design Approach

Security is implemented using **Acegi** Security framework . please refer to this documentation https://wiki.nci.nih.gov/display/CTMS/CSM+Tutorial+with+caAERs+Examples\\

Instance level Security :

Filter security interceptor intercepts all the search calls. It filters the search results and returns the filtered result set back to the caller.

4. Design Details

This section will contain the design details at different layers. This section should be reviewed by the Tech Lead or Architect.

5. Tools & Technologies

6. Configuration Requirements

Configuring security interceptor for any searches.

Configuration File : applicationContext-core-security.xml

Step 1 : Add the Object and methods to the defined point cut

```
<aop:pointcut id="daoSaveOrUpdatePointcut"
```

Example : If you want to apply security filter on all find methods in ResearchStaffDao

```
||execution(public *        gov.nih.nci.cabig.caaers.dao.ResearchStaffDao+.find*(..))
```

Step 2 : Add a security role name to access above mentioned methods.

Example : <entry key="execution(* gov.nih.nci.cabig.caaers.dao.ResearchStaffDao.find*(..))" value="gov.nih.nci.cabig.caaers.Admin:ACCESS"/> . User with role name "gov.nih.nci.cabig.caaers.Admin:ACCESS" .

Step 3 : Write a Security Filter class to filter your Domain Objects . In this case , filter ResearchStaff

Developing Security Filter Class:

1. Interface to implement gov.nih.nci.cabig.caaers.accesscontrol.DomainObjectSecurityFilterer
2. method : Object filter(org.acegisecurity.Authentication.Authentication authentication, String permission,**Object returnObject**);

   Object returnObject : Original Result Set.

This method filters out the original result set and returns the filtered one.

```
public class ResearchStaffSecurityFilterer implements DomainObjectSecurityFilterer {
    public Object filter(Authentication authentication, String permission, Object returnObject) {

        // some code ....

        Filterer filterer = (Filterer)returnObject;
        Iterator collectionIter = filterer.iterator();
        while (collectionIter.hasNext())
                Object domainObject = collectionIter.next();
                 ResearchStaff researchStaffResultObject = (ResearchStaff)domainObject;
                 if (.....) filterer.remove(researchStaffResultObject);
                  -------
        return filterer.getFilteredObject();
}
```

Step 4 : Add your implementation to domainObjectSiteSecurityAuhthorizationCheckProvidersMap in configuration file.

key : Domain Object , value : bean reference.

```
 <util:map id="domainObjectSiteSecurityAuhthorizationCheckProvidersMap"
   map-class="java.util.LinkedHashMap">
         <entry key="gov.nih.nci.cabig.caaers.domain.ajax.StudySearchableAjaxableDomainObject"
                value-ref="studySiteSecurityFilterer"/>

          -------

          -------
         <entry key="gov.nih.nci.cabig.caaers.domain.ResearchStaff"
                value-ref="researchStaffSecurityFilterer"/>

</util:map>

   <bean id="researchStaffSecurityFilterer"
           class="gov.nih.nci.cabig.caaers.accesscontrol.ResearchStaffSecurityFilterer"/>
```

THATS IT .....

Notes and Meetings

12/23/08 (Srini, Paul, Kruttik, Vinay G); 2-3:30pm
Overview: Purpose of the meeting was to discuss the hosted mode implementation approach within C3PR.

Open Issues:

- The main issue which prompted this meeting was the identification of bugs during QA regarding the ability of Site users to access the edit study flow and edit the study, including adding/removing sites, investigators, and research personnel.

Actions and Decisions:

1. Paul will contact CALGB and confirm who adds staff and investigators to a study, the site or CALGB.
2. The C3PR design was discussed where the study definition (plan study) and study personnel (conduct study) are split into two separate tabs. The reason for this was due to some roles needing access only to certain tabs and this resulted in eventually splitting this out.
    a. splitting out plan study and conduct study may be needed in caAERS, but currently, we will fix the Hosted Mode issues identified and release for user acceptance testing and feedback.
        i. Conduct study would likely have:
            1. Study Sites
            2. Study Personnel (and site investigators)
            3. Study/Site Routing and Review workflow configurations
        ii. Plan study would have all protocol level study information.
3. Srini will fix Karthik's bugs not involving the study/organization access issue
4. This may not be an issue for CALGB since they won't be using roles

Design Document - Routing and Review

The purpose of this page is to document the approach as to how a particular requirement or use case is being implemented. The owner of this document is the developer leading the implementation.

NOTE: This page is to serve as an ongoing record. It should be used as a whiteboard for brainstorming ideas and strategies and for documenting minutes from meetings and discussions. It should clearly show the evolution of the implementation approach.

1. Background / Introduction

The design aims to implement the following workflows into caAERS application.

- Expedited report flow:
  1. (domestic) Site CRA (entry) -> Physician (review) -> Site CRA (submit) -> CALGB Central Office (review) -> Submit to AdEERS
  2. ~~(international) Site CRA (entry) -> Physician (review) -> Monitor (review) -> CALGB Central Office (review) -> Submit to AdEERS~~
- Routine AE flow:
  1. ~~Site CRA -> CALGB Data Coordinator -> Finalize~~   Site CRA -> CALGB Data Coordinator ->Approved --> [CTMS:back] CALGB Data Coordinator ....
  2. ~~Site CRA -> Main Member Data Coordinator -> CALGB Data Coordinator -> Finalize~~

2. Use Cases Affected

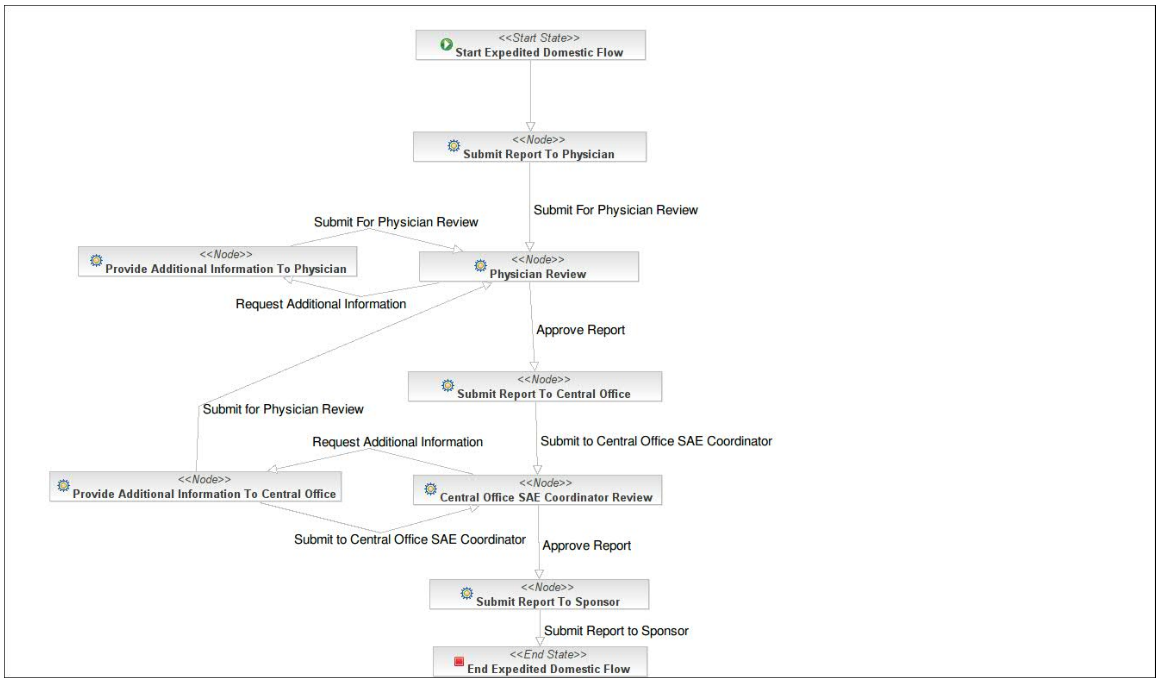
This design is to realize the requirements mentioned in 5. Internal Routing and Review. The implementation of this has a direct impact on the hosted mode(Design Document - Hosted Mode (caAERS)) and security usecases.

3. Design Approach

This section will serve as a summary of how the use case or requirement will be implemented. This section should be reviewed by the Tech Lead or Architect.

The workflow configuration for the above is mentioned below :-

Expedited Flow (domestic)



<<Start State>>
Start Expedited Domestic Flow
<<Node>>
Submit Report To Physician
Submit For Physician Review
Submit For Physician Review
<<Node>>
Provide Additional Information To Physician
<<Node>>
Physician Review
Request Additional Information
Approve Report
<<Node>>
Submit Report To Central Office
Submit for Physician Review
Request Additional Information
Submit to Central Office SAE Coordinator
<<Node>>
Provide Additional Information To Central Office
<<Node>>
Central Office SAE Coordinator Review
Submit to Central Office SAE Coordinator
Approve Report
<<Node>>
Submit Report To Sponsor
Submit Report to Sponsor
<<End State>>
End Expedited Domestic Flow

| **Task Name** | **Submit Report To Physician** |

Description	This task is created and assigned to the CRA (reporter) immediately after an expedited report is created in the system. Selection of the dropdown "Send to Physician" will advance the workflow to the next step "Physician Review".
Assignee	Reporter (The user associated to this Role is picked up from Expedited Flow Reporter page).
Notification	Subject : The task Submit Report to Physician is assigned to you Message :
Review Status	Draft / Incomplete
Drop down Option(s)	Please Select Submit for Physician Review

Task Name	**Physician Review**
Description	This task is assigned to the Physician immediately after the CRA (Reporter) publish the report. If additional information is required to complete the review, "Request Additional Information" is opted by the physician. If the information in the report is adequate, "Sign Off Report" option is opted by the physician.
Assignee	Physician (The user associated to this Role is picked up from Expedited Flow Reporter page)
Notification	Subject : The task Physician Review is assigned to you Message :
Review Status	Physician Review
Drop Down Option(s)	Please Select Request Additional Information Approve Report

Task Name	**Provide Additional Information To Physician**
Description	This task is assigned to the Reporter (CRA), when Physician during the review process requests for additional information. Selection of the dropdown "Send to Physician" will advance the workflow to the next step "Physician Review".
Assignee	Reporter (CRA)
Notification	Subject : The task Provide Additional Information To Physician is assigned to you Message :
Review Status	Additional Info Requested by Physician
Drop Down Option(s)	Please Select Submit for Physician Review

Task Name	**Submit Report To Central Office**

Description	This task is created and associated to the CRA (Reporter) immediately after the physician approves the expedited adverse event report. Selection of the drop down "Send to Central Office" will advance the workflow state to "Central Office Review"

Assignee	Reporter (CRA)
Notification	Subject : The task Submit Report To Central Office is assigned to you Message :
Review Status	Approved by Physician
Drop Down Option(s)	Please Select Submit to Central Office SAE Coordinator

Task Name	**Central Office SAE Coordinator Review**
Description	This task is created immediately after, the CRA (reporter) sends the report to central office for review. Selection of Approve Report will advance the workflow to "Submit Report To AdEERS" task Selection of "Request Additional Information" will advance the workflow to "Provide Additional Information To Central Office".
Assignee	AE Coordinator. This task is assigned to all the AE coordinators associated to the study.
Notification	Subject : The task Central Office Review is assigned to you. Message :
Review Status	Central Office SAE Coordinator Review
Drop Down Option(s)	Please Select Approve Report Approve and Submit Request Additional Information

Task Name	**Provide Additional Information To Central Office**
Description	This task is created and assigned to the Reporter (CRA), when the central office (AE coordinator) requests for additional Information. Selection of the drop down "Send to Central Office" will advance the workflow state to "Central Office Review"
Assignee	Reporter (CRA)
Notification	Subject : The task Provide Additional Information To Central Office is assigned to you. Message :
Review Status	Additional Info Request by Central Office
Drop Down Option(s)	Please Select Submit to Central Office SAE Coordinator Submit for Physician Review

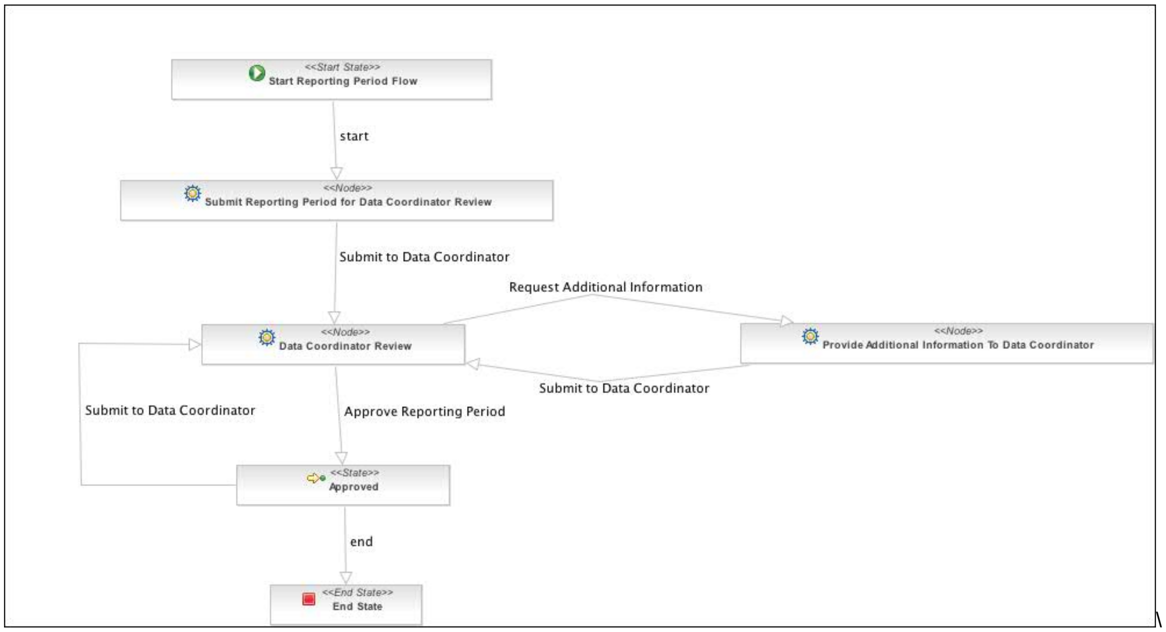
Task Name	**Submit Report To Sponsor**
Description	This task is created and assigned to the central office personnel (AE coordinators), when an AE coordinator approves the report.
Assignee	Central Office SAE Coordinators
Notification	Subject : The task submit report to AdEERS is assigned to you Message :
Review Status	Ready for Submission to Sponsor

Drop Down Options	Please select Submit Report to Sponsor

Evaluation Period Workflow (Coordinating Center)

A sample usecase scenario:-

1. **Site CRA** - creates a reporting period.
2. **Workflow Engine** - assigns the task "Submit Reporting Period for Data Coordinator Review" to the Subject Coordinators at the site.
3. **Site CRA** - completes the reporting period and pushes it for review.
4. **Workflow Engine** - assigns the task "Data Coordinator Review" to the Adverse Event Coordinators(is-the data coordinator) at the coordinating center.
5. **Data Coordinator** - review the reporting period, then approves it.
6. **Site CRA** - one fine day changes the reporting period, and submits it for review again. Steps from 4 continues...

- Alternate flow:*
- 5. **Data Coordinator** - review the reporting period, and request for additional information.
- 6. **Workflow Engine** - assigns the task "Provide Additional Information To Data Coordinator" to the Subject Coordinators at the site. Steps from 3 continues...
- 

<<Start State>> Start Reporting Period Flow
start
<<Node>> Submit Reporting Period for Data Coordinator Review
Submit to Data Coordinator
Request Additional Information
<<Node>> Data Coordinator Review
<<Node>> Provide Additional Information To Data Coordinator
Submit to Data Coordinator
Submit to Data Coordinator
Approve Reporting Period
<<State>> Approved
end
<<End State>> End State

Task Name	**Submit Reporting Period For Coordinating Center Review**
Description	This task is created and assigned to the CRA, immediately after a reporting period is created in the system.
Assignee	CRA (Reporting Period's -~~>ParticipantAssignment~~->Site's --> CRA)
Notification	Subject : The task Submit Reporting Period For Coordinating Center Review is assigned to you. Message :
Review Status	Draft / Incomplete
Drop Down Option(s)	Please Select Submit to Data Coordinator

Task Name	**Data Coordinator Review**
Description	This task is created and assigned to the AE coordinator at the coordinating center. Request Additional Information selection will advance the workflow to Provide Additional Information To Data Coordinator task Approve Reporting Period will advance the workflow to Finalize Reporting Period task.

Assignee	Coordinating Center's - Data Coordinator.
Notification	Subject : The task Central Office Review is assigned to you Message :
Review Status	Data Coordinator Review
Drop Down Option(s)	Please Select Approve Reporting Period Request Additional Information

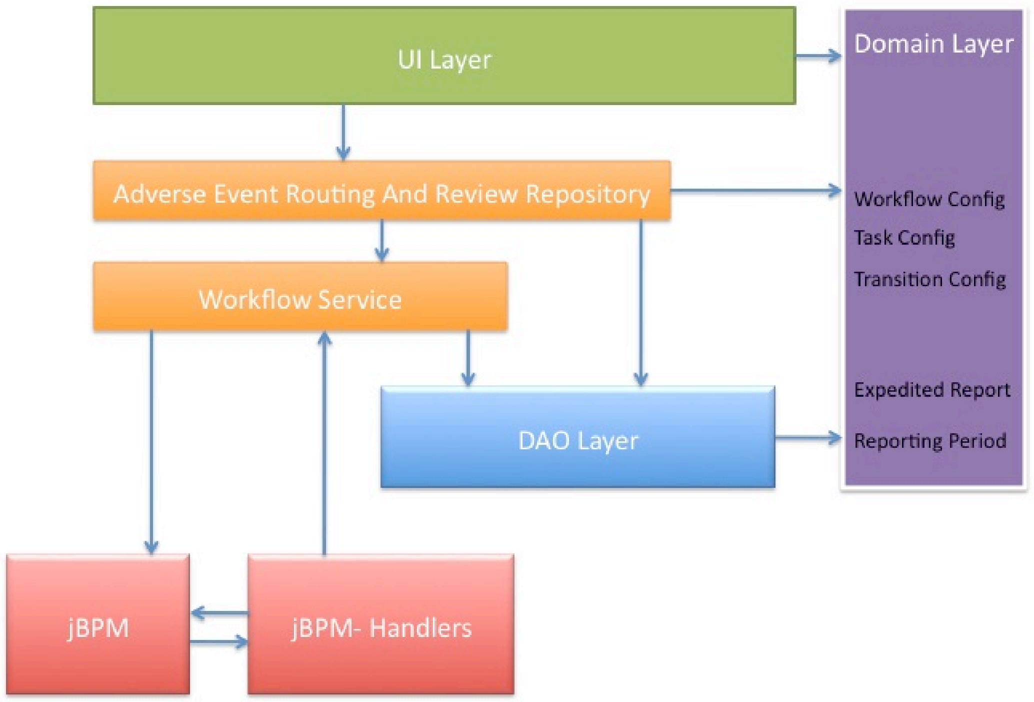
Task Name	**Provide Additional Information To Data Coordinator**
Description	This task is created and assigned to the CRA, when the central office (Data coordinator) requests for additional Information. Selection of the drop down Submit To Data Coordinator will advance the workflow state to Data Coordinator Review
Assignee	Site CRA
Notification	Subject : The task Provide Additional Information To Data Coordinator is assigned to you. Message :
Review Status	Additional Info Requested By Data Coordinator
Drop Down Option(s)	Please Select Submit To Data Coordinator

Task Name	**Approved <<State>>**
Description	Workflow moves into the approved state, from here, the Site CRA can drive the workflow further
Assignee	NONE
Notification	Subject : Message :
Review Status	Approved
Drop Down Option(s)	Please Select Submit to Data Coordinator (Only for Site CRA)

4. Design Details

This section will contain the design details at different layers. This section should be reviewed by the Tech Lead or Architect.

4.1 Design Stack



At a very high level the implementation should have mainly the following layers

Workflow Domain Layer

All the domain objects that is created by caAERS app, to enable the workflow feature should go in here.

Workflow UI Layer

The UI classes that implements workflow goes in this layer.

Workflow Service Layer

This layer consists of service methods that facilitates Routing & Review use-case. There are method for creating Process Instances, creating Task Instances, fetching them and closing task Instances.

Routing and Review Repository Layer

Any thing to do with the supporting domain objects of the workflow (eg: creation of comments etc.) will go in this layer.

Workflow Dao Layer

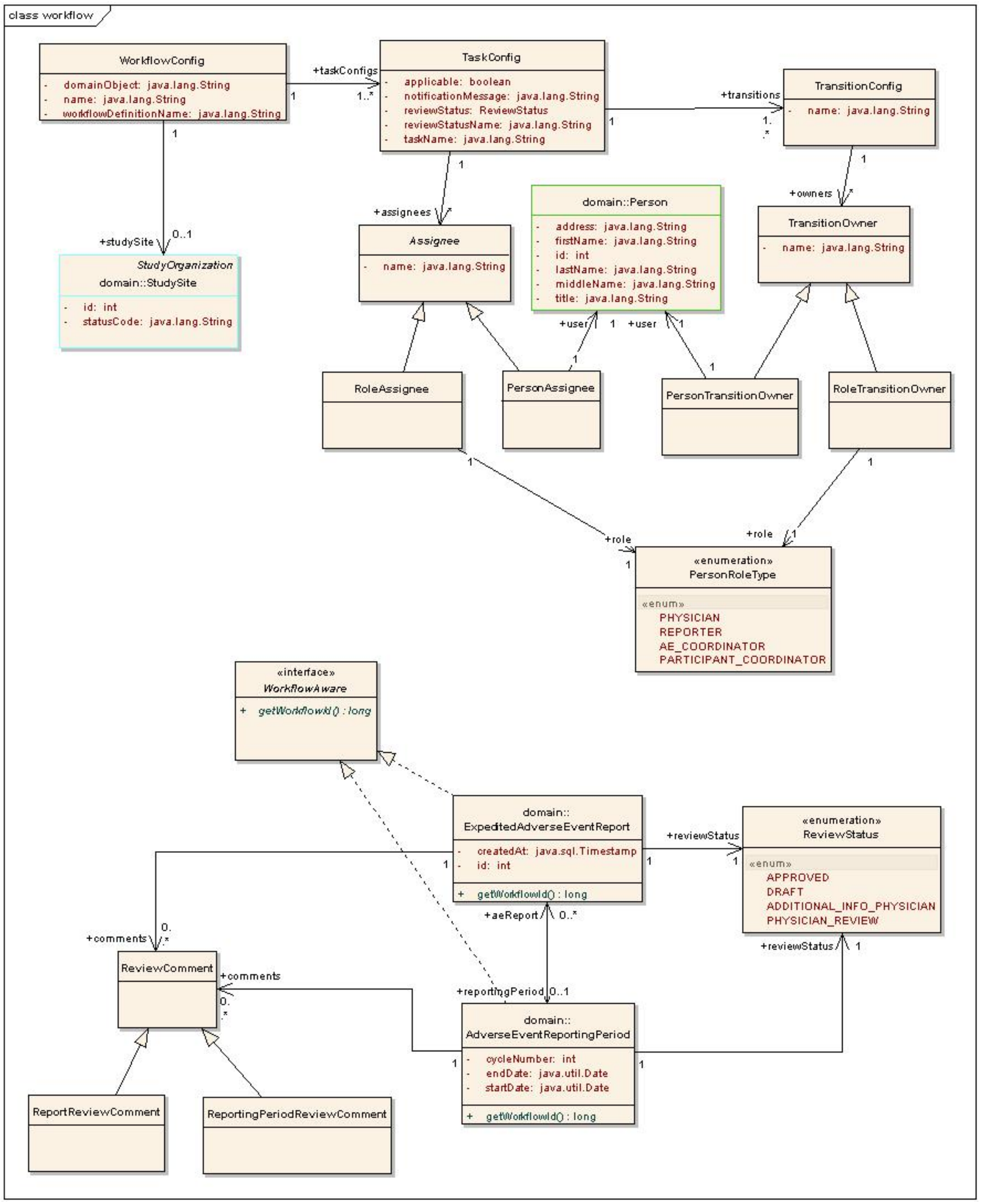
The code that manages the persistant state of the workflow domain objects go in this layer. (e.g: WorkflowConfigDao)

jBPM Handler Layer

The call back handlers, from workflow engine go in this layer. (eg :NodeSkipActionHandler, TaskCloseActionHandler)

4.2 Domain Layer in Detail

The below high level class diagram shows the classes and their relationships in workflow domain layer. The intention of each concept is explained below the diagram.



The domain classes can be classified into two. The classes that represents information related to process definition(eg:- who can perform a task, whether a task is enabled or not etc.), and the classes that represents information related to a process instance (eg: comments).

WorkflowConfig

Workflow process can be tailored to user needs, by customizing the base workflow definition loaded in the system. This class captures those details.

TaskConfig

The customizations applied on a task in the process definition such as, whether this task is enabled, the assignees etc is captured by this concept.

Assignee

The task can be assigned to a user or a role, the details is represented by this concept.

TransitionConfig

From a task, there could be transitions going out from it. Every such transition can be configured in such a way that, they could be only be performed by a certain user or a role. The details about that is represented by this concept.

TransitionOwner

This concept captures the owner of the transition. The owner could be a role or a user.

WorkflowAware

Any domain object that is workflow enabled, should implement this interface. The WorkflowAware implementations are ExpeditedAdverseEventReport and AdverseEventReportingPeriod, as currently we support only routing and review process for expedited reporting and course.

ReviewStatus

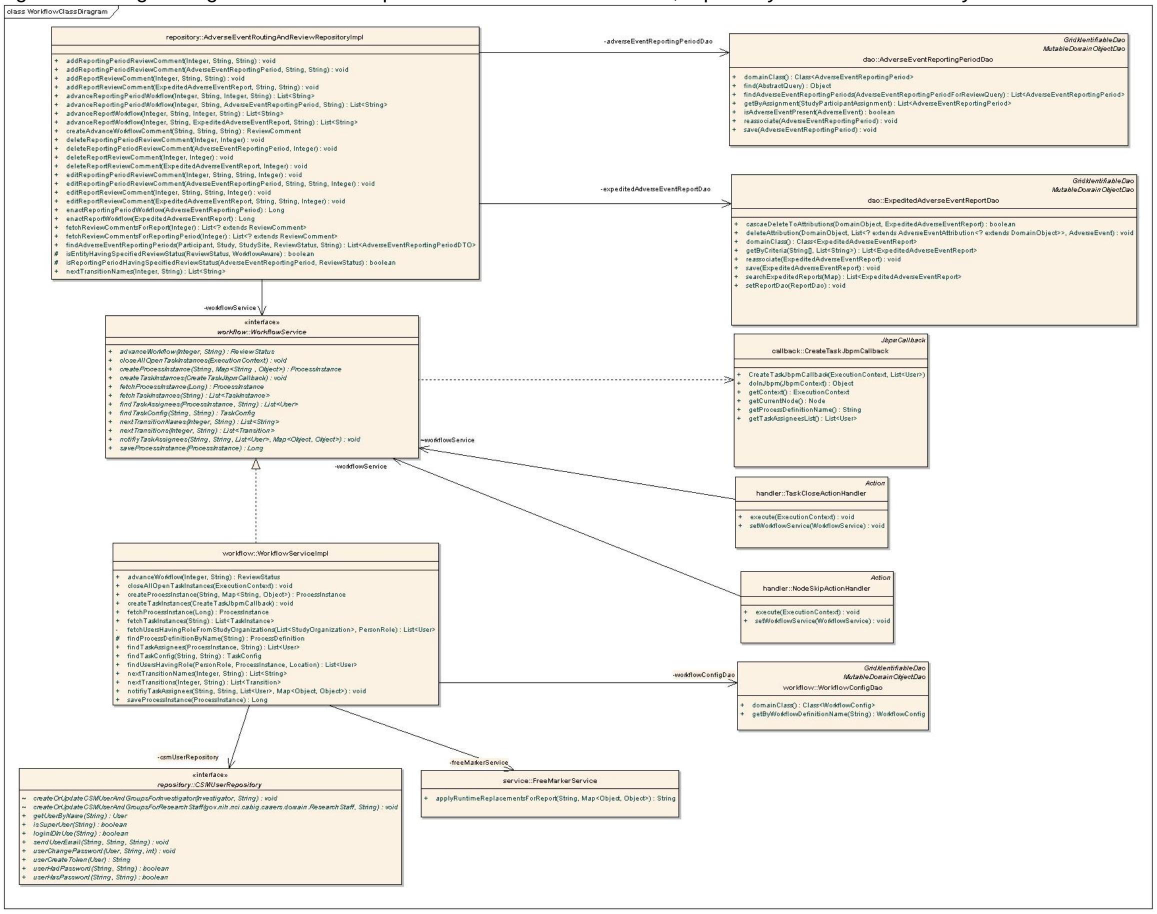
The possible review statuses are represented in this enumeration.

ReviewComment

A comment that is added at each step in the routing and review workflow is represented by this concept.

4.3 Workflow Service, Repository & Dao Layers In Detail

The class diagram showing the high level relationships of the classes in the service, repository and data access layer is below:-


class WorkflowClassDiragram
repository::AdverseEventRoutingAndReviewRepositoryImpl
+ addReportingPeriodReviewComment(Integer, String, String) : void
+ addReportingPeriodReviewComment(AdverseEventReportingPeriod, String, String) : void
+ addReportReviewComment(Integer, String, String) : void
+ addReportReviewComment(ExpeditedAdverseEventReport, String, String) : void
+ advanceReportingPeriodWorkflow(Integer, String, Integer, String) : List<String>
+ advanceReportingPeriodWorkflow(Integer, String, AdverseEventReportingPeriod, String) : List<String>
+ advanceReportWorkflow(Integer, String, Integer, String) : List<String>
+ advanceReportWorkflow(Integer, String, ExpeditedAdverseEventReport, String) : List<String>
+ createAdvanceWorkflowComment(String, String, String) : ReviewComment
+ deleteReportingPeriodReviewComment(Integer, Integer) : void
+ deleteReportingPeriodReviewComment(AdverseEventReportingPeriod, Integer) : void
+ deleteReportReviewComment(Integer, Integer) : void
+ deleteReportReviewComment(ExpeditedAdverseEventReport, Integer) : void
+ editReportingPeriodReviewComment(Integer, String, String, Integer) : void
+ editReportingPeriodReviewComment(AdverseEventReportingPeriod, String, String, Integer) : void
+ editReportReviewComment(Integer, String, String, Integer) : void
+ editReportReviewComment(ExpeditedAdverseEventReport, String, String, Integer) : void
+ enactReportingPeriodWorkflow(AdverseEventReportingPeriod) : Long
+ enactReportWorkflow(ExpeditedAdverseEventReport) : Long
+ fetchReviewCommentsForReport(Integer) : List<? extends ReviewComment>
+ fetchReviewCommentsForReportingPeriod(Integer) : List<? extends ReviewComment>
+ findAdverseEventReportingPeriods(Participant, Study, StudySite, ReviewStatus, String) : List<AdverseEventReportingPeriodDTO>
# isEntityHavingSpecifiedReviewStatus(ReviewStatus, WorkflowAware) : boolean
# isReportingPeriodHavingSpecifiedReviewStatus(AdverseEventReportingPeriod, ReviewStatus) : boolean
+ nextTransitionNames(Integer, String) : List<String>
-adverseEventReportingPeriodDao
GridIdentifiableDao
MutableDomainObjectDao
dao::AdverseEventReportingPeriodDao
+ domainClass() : Class<AdverseEventReportingPeriod>
+ find(AbstractQuery) : Object
+ findAdverseEventReportingPeriods(AdverseEventReportingPeriodForReviewQuery) : List<AdverseEventReportingPeriod>
+ getByAssignment(StudyParticipantAssignment) : List<AdverseEventReportingPeriod>
+ isAdverseEventPresent(AdverseEvent) : boolean
+ reassociate(AdverseEventReportingPeriod) : void
+ save(AdverseEventReportingPeriod) : void
-expeditedAdverseEventReportDao
GridIdentifiableDao
MutableDomainObjectDao
dao::ExpeditedAdverseEventReportDao
+ cascaeDeleteToAttributions(DomainObject, ExpeditedAdverseEventReport) : boolean
+ deleteAttribution(DomainObject, List<? extends AdverseEventAttribution<? extends DomainObject>>, AdverseEvent) : void
+ domainClass() : Class<ExpeditedAdverseEventReport>
+ getByCriteria(String[], List<String>) : List<ExpeditedAdverseEventReport>
+ reassociate(ExpeditedAdverseEventReport) : void
+ save(ExpeditedAdverseEventReport) : void
+ searchExpeditedReports(Map) : List<ExpeditedAdverseEventReport>
+ setReportDao(ReportDao) : void
-workflowService
«interface»
workflow::WorkflowService
+ advanceWorkflow(Integer, String) : ReviewStatus
+ closeAllOpenTaskInstances(ExecutionContext) : void
+ createProcessInstance(String, Map<String , Object>) : ProcessInstance
+ createTaskInstances(CreateTaskJbpmCallback) : void
+ fetchProcessInstance(Long) : ProcessInstance
+ fetchTaskInstances(String) : List<TaskInstance>
+ findTaskAssignees(ProcessInstance, String) : List<User>
+ findTaskConfig(String, String) : TaskConfig
+ nextTransitionNames(Integer, String) : List<String>
+ nextTransitions(Integer, String) : List<Transition>
+ notifiyTaskAssignees(String, String, List<User>, Map<Object, Object>) : void
+ saveProcessInstance(ProcessInstance) : Long
JbpmCallback
callback::CreateTaskJbpmCallback
+ CreateTaskJbpmCallback(ExecutionContext, List<User>)
+ doInJbpm(JbpmContext) : Object
+ getContext() : ExecutionContext
+ getCurrentNode() : Node
+ getProcessDefinitionName() : String
+ getTaskAssigneesList() : List<User>
~workflowService
-workflowService
Action
handler::TaskCloseActionHandler
+ execute(ExecutionContext) : void
+ setWorkflowService(WorkflowService) : void
Action
handler::NodeSkipActionHandler
+ execute(ExecutionContext) : void
+ setWorkflowService(WorkflowService) : void
workflow::WorkflowServiceImpl
+ advanceWorkflow(Integer, String) : ReviewStatus
+ closeAllOpenTaskInstances(ExecutionContext) : void
+ createProcessInstance(String, Map<String, Object>) : ProcessInstance
+ createTaskInstances(CreateTaskJbpmCallback) : void
+ fetchProcessInstance(Long) : ProcessInstance
+ fetchTaskInstances(String) : List<TaskInstance>
- fetchUsersHavingRoleFromStudyOrganizations(List<StudyOrganization>, PersonRole) : List<User>
# findProcessDefinitionByName(String) : ProcessDefinition
+ findTaskAssignees(ProcessInstance, String) : List<User>
+ findTaskConfig(String, String) : TaskConfig
+ findUsersHavingRole(PersonRole, ProcessInstance, Location) : List<User>
+ nextTransitionNames(Integer, String) : List<String>
+ nextTransitions(Integer, String) : List<Transition>
+ notifiyTaskAssignees(String, String, List<User>, Map<Object, Object>) : void
+ saveProcessInstance(ProcessInstance) : Long
-workflowConfigDao
GridIdentifiableDao
MutableDomainObjectDao
workflow::WorkflowConfigDao
+ domainClass() : Class<WorkflowConfig>
+ getByWorkflowDefinitionName(String) : WorkflowConfig
-csmUserRepository
«interface»
repository::CSMUserRepository
~ createOrUpdateCSMUserAndGroupsForInvestigator(Investigator, String) : void
~ createOrUpdateCSMUserAndGroupsForResearchStaff(gov.nih.nci.cabig.caaers.domain.ResearchStaff, String) : void
+ getUserByName(String) : User
+ isSuperUser(String) : boolean
+ loginIDInUse(String) : boolean
+ sendUserEmail(String, String, String) : void
+ userChangePassword(User, String, int) : void
+ userCreateToken(User) : String
+ userHadPassword(String, String) : boolean
+ userHasPassword(String, String) : boolean
-freeMarkerService
service::FreeMarkerService
+ applyRuntimeReplacementsForReport(String, Map<Object, Object>) : String

5. Tools & Technologies

This section will define the tools and technologies that will be utilized in the implementation. This section should be reviewed by the Tech Lead or

Architect.

6. Configuration Requirements

This section will detail the special modifications required to other elements in the application in order for this implementation to be successful. Examples of items to include in this section would be changes to Tomcat, increases in memory or disk space, etc.

Meetings & Notes

12/19/08 (Ram, Paul, Biju, Ion), 3-5pm
Overview: Discussion regarding feedback from the Routing and Review demos provided to the adopters on 12/18/08. Major impacts from feedback were that there were 2 different expedited workflows (domestic and international) and 2 routing AE workflows (main member review and no main member review). Essentially, these dictate that workflows will need to be configurable on a per site/per study basis.

Decisions and Actions:

1. A new "workflow" tab will be added to study that will list each study site and allow the selection of the routing and review workflow for expedited reports and evaluation periods for each study site.
2. The workflows that will be supported will initially be the domestic expedited report workflow and both the evaluation period workflows (update from 12/23 - only the "no main member review" workflow will initially be supported).
3. The ability to configure the workflows, assignees for each step, and the notification messages will be removed from the initial deployment. Thus, the "Admin" tab will not be needed initially.
4. The physician logging in to review the expedited report will need be be able to edit the AE flow (requires adding a Physician role into caAERS and assigning appropriate permissions). The changes they make should be logged as comments.
5. Biju to generate design doc
6. Paul to generate activity diagrams
7. Ion to begin refactoring study tab

Issues:

1. How will routing be used by CALGB if they are not using roles? Need a follow-up with CALGB to discuss.
2. Who is the physician that should be reviewing the expedited report? The treating physician for the AE, who could be non-study personnel (i.e. emergency room doc), or the site investigator responsible for the particular participant? If it's the former, how do we add them as a user to caAERS so they can perform their review?

12/23/08 (Ram, Paul, Biju), 10:30am-12:30pm
Overview: Purpose of this meeting was to discuss the design document Biju circulated on 12/22/08.

Decisions and Actions:

1. The decision was made to focus on only one (1) evaluation period flow (no main member review). The reason for this is that the main member review workflow depends upon an organization hierarchy which does not currently exist in caAERS.
    a. Adding a tab to the study flow to configure the workflow for each site will be on-hold until caAERS supports the organization hierarchy.
    b. Support for an organization hierarchy will be prioritized for Monish with respect to caAERS and his review of COPPA services.
2. The decision was made to utilize the alerts slider (initially developed to display messages from lab viewer) to display comments, allow the entry of comments, and change the review status of an evaluation period or expedited report. The slider will be displayed in all tabs of the capture AE flow and expedited flow.
3. Biju will update the design document
4. David will generate mock-up(s) of how the UI will enable the workflows.

12/29/08 (Paul, Biju, David); 11:30am-1:30pm
Overview: Purpose of this meeting was to discuss UI design and mock-ups needed for the slider and the review pages.

Decisions and Actions:

1. A slider will be utilized and is the first priority for the mock-up.
    a. The slider will have an expand/collapse link centered on the right hand side of the window.
    b. The slider link will retain it's position even when the user scrolls the page.
    c. When expanded, the slider will cover the top ~1/2 of the screen (i.e. a short and wide display). The goal was to allow users to have the slider open while editing the relavant information on the main screen.
        i. A tall and thin display of the expanded slider was rejected since it will cover too much content.
        ii. The top ~1/2 was initially chosen for expanded slider display since the navigation and page headers are not as critical for adding and reviewing comments as the lower page content.
        iii. Future implementation may allow for the user to choose if the slider expands in the top or bottom ~1/2 of the screen.
    d. The slider will be tabbed to support "Alerts" and "Review/Comment".
    e. The slider will be displayed in the capture AE flow and in the expedited flow.
2. The mock-up of the read-only reporting period view is a second priority. This will essentially look identical to the capture AE tab.

12/30/08 (Paul, Ram, Srini, Biju); 4:30pm-5:30pm
Overview: Purpose of this meeting was to discuss over the way to implement login for investigators.

Decisions and Actions:

1. Treating Physicians should be allowed to login to the system. The default role (the only role) to which they can be associated is 'Physician'
2. SAE flow, Reporter tab,
   a. the physician should be a dropdown list, showing all the investigators associated to the study site.
   b. There should be a create button beside the physician, that allow creation of an investigator.
   c. the research staff dropdown should show both study personnel and investigators
   d. the values (name, email phone etc.) of the research staff and investigator should not be editable.

1/21/09 (Paul, Ram, Sameer, Biju); 3:20pm-4:30pm
Overview: Purpose of this meeting was to refresh and discuss the workflow implementation.
Decisions and Actions:

1. Workflow should be tied up to expedited report
2. Role filtering on transitions should be in place
3. From the last node of reporting period workflow, there should be a transition to the beginning of the flow.
4. A new Expedited workflow should begin when the workflow is amended.

1/29/09 (Paul, Ram, Vinay, Sameer, Biju); 2:20pm-3:30pm

Overview: Purpose of this meeting was to have the first cut internal routing and review demo.
Decisions and Actions:

1. Reporting Period (Coordinating Center) workflow, changes suggested. (Remove 'Ready4Finalizing' node)
2. Routing and review page, should not show the Reporting Period , if the logged in user don't has any action to perform
3. Study Personnel page, "Summary" has "Personnels", which is wrong.

4/2/09 (Paul, Sameer, Biju); 10:00am-10:45am

Overview: Discussion regarding feedback reported in demo to CTEP/CBIIT yesterday
Decisions and Actions:

1. Physician review should be optional
   - Task: create a new workflow transition from CRA to SAE Coordinator Review prior to Physician review.
2. Change "Submit for Physician Review" to "Send to Physician for Review"
3. It seems like the "Next Action" is for having the comment reviewed
   - Task: Move "Next Action" to the Submit page to clarify that it is for the report.
4. If Physician approves report, it should populate the field in the Submit flow that asks if the physician has approved.
   - Task: move the submit flow question regarding physician approval into tab 13 (Submit) of the expedited flow.
   - Task: have workflow action "Physician Approved" fill in the corresponding filed in the submit flow.
   - Task: Have the "physician approved?" question in the expedited flow be driven by the mandatory fields section of the report.
   - Task: prevent the CRA from submitting a report to SAE coordinator unless all required fields are complete; sending an incomplete report to a physician for review should be allowed.

Issues raised for consideration:

- Since the routing and review is on the data collection, if there is more than on report in process on the data collection at a single time (i.e. an AdEERS report and an IRB report) there is no context for the specific report that is being reviewed and approved.
  - Example: The SAE coordinator at the coordinating center will have a submit button, however, since routing and review is on the data collection, there is not context for the report they are submitting if there is more than one report.
  - Possible solutions: one data collection per active report; provide a means for picking the report context for review.
- If routing and review is enabled in non-hosted mode, the SAE coordinator review will go to someone who doesn't have access to the system.
  - Need to add in a non-central processing workflow.

Design Document Template

The purpose of this page is to document the approach as to how a particular requirement or use case is being implemented. The owner of this document is the developer leading the implementation.

NOTE: This page is to serve as an ongoing record. It should be used as a whiteboard for brainstorming ideas and strategies and for documenting minutes from meetings and discussions. It should clearly show the evolution of the implementation approach.

Contents

1. Background / Introduction

This section should be used to summarize what will be implemented. It should also provide a link, if available, to the requirement or use case. This section should also provide a summary of the problem in the words of the developer. This section should be reviewed by the Business Analyst or Project Manager to ensure the developer understands what is to be implemented.

2. Use Cases Affected

This section should define all of the dependencies related to the implementation of this use case or requirement. Any component of the application or use of the application must be identified that will be impacted by this implementation or on which this implementation is dependent . This section should be reviewed by the Project Manager, Tech Lead, and/or Architect.

3. Design Approach

This section will serve as a summary of how the use case or requirement will be implemented. This section should be reviewed by the Tech Lead or Architect.

4. Design Details

This section will contain the design details at different layers. This section should be reviewed by the Tech Lead or Architect.

5. Tools & Technologies

This section will define the tools and technologies that will be utilized in the implementation. This section should be reviewed by the Tech Lead or Architect.

6. Configuration Requirements

This section will detail the special modifications required to other elements in the application in order for this implementation to be successful. Examples of items to include in this section would be changes to Tomcat, increases in memory or disk space, etc.

Human Interface Guidelines

In General:

- **Use conventions** (like boxes and divisions)
- **Look at what has been done in other parts of the application as a reference**
- **Look at what has been done in other popular applications** (Don't re-invent the wheel)
- **Use CSS rules that are already in place for layout of common elements**
- **A clean, consistent UI is YOUR responsibility. Your development is not complete until the UI looks good!**

Outline for future expansion of this document

- Chrome boxes and divisions
- flash messages
- Collapsible divisions
- Add row buttons ie. keep the button close to the element that it alters

Buttons

Behold...the omnipotent button

I have developed a button framework for our applications. It will satisfy all your button needs. With it you can easly code a button, select a size, color, and icon and give it your own text and it will do the rest for you. It will look sleek and cool and will work across browsers.

> Please report any bugs to David.

Example usage:

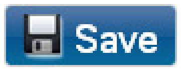
Use this JSP code:

```
<tags:button onclick="doSomething();" cssClass="foo" id="bar" color="blue" value="Save" icon="save" />
```

That will generate the following HTML:

```
<button id="bar" class="omnipotent-button blue foo" onclick="doSomething();">
 <table>
  <tbody>
   <tr>
      <td class="l"/>
      <td class="m">
         <img alt="" src="/ctcae/images/chrome/../buttons/button_icons/disk_icon.png"/>
         Save
      </td>
      <td class="r"/>
   </tr>
  </tbody>
 </table>
</button>
```

It will look like this:



Learn the attributes

icon

As of this writing there are 14 icons which you can choose from to add to your button. I can add more icons on a case by case basis as necessary.

```
icon="save"
```

Image	Value
💾	save

💾➡	Save & Continue
⬅💾	Save & Back
➡	continue
⬅	back
✖	x
➕	add
➕➕	add multiple
✏	edit
✓	check
🔍	search
📄	page
🗔	window
👤	subject

color

There are 4 colors you can choose from.

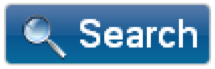
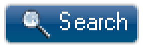
```
color="blue"
```

Look	Value	Use this color when:
🟩	green	Use green when the button follows the happy path, when the button is the next intended action for the user.
🟦	blue	Use blue for other low-risk, reversible actions like search, add & save etc.
🟥	red	Use red for buttons that destroy data, like delete, or cancel.
🟧	orange	Use orange for high-risk actions that are irreversible like a final submit button.

size

```
size="small"
```

All buttons are big by default. You can make them small by specifying size="small"

Look	Size
	Big
	Small

markupWithTag

The <tags:button /> will generate HTML using the <button> tag by default. This is useful when you want to submit a form. If this causes problems you can try markupWithTag="a". This will generate the HTML <a> tag but is styled just like the buttons and will look the same. <button> is the preferred use since it has a couple extra options with it.

markupWithTag="button"

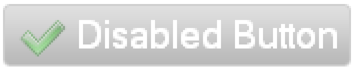
If you use markupWithTag="button" or don't specify markupWithTag, the following attributes are available to you:

Disabled

```
disabled="disabled"
```

You can disable a button by giving it disabled="disabled". This will gray out the button and make it unclickable. This can be changed through Javascript by calling the button by the **id** or **cssClass**.

It looks like this:



Type

```
type="button" <!--or--> type="submit"
```

You should always specify a type for browser compatibility reasons. Like regular HTML buttons, type="submit" will submit it's parent form, while type="button" will not.

markupWithTag="a"

When you markupWithTag="a", you cannot specify type or disabled, but you can give it an href.

href

Just like any link, put your href here.

value

```
 value="Click Me!"
```

This is where you define the text of your button that will be visible to the user.

onclick

```
onclick="doSomething();"
```

If you want the button to execute some Javascript, put it here like you would on any other element.

cssClass

```
cssClass="foobar"
```

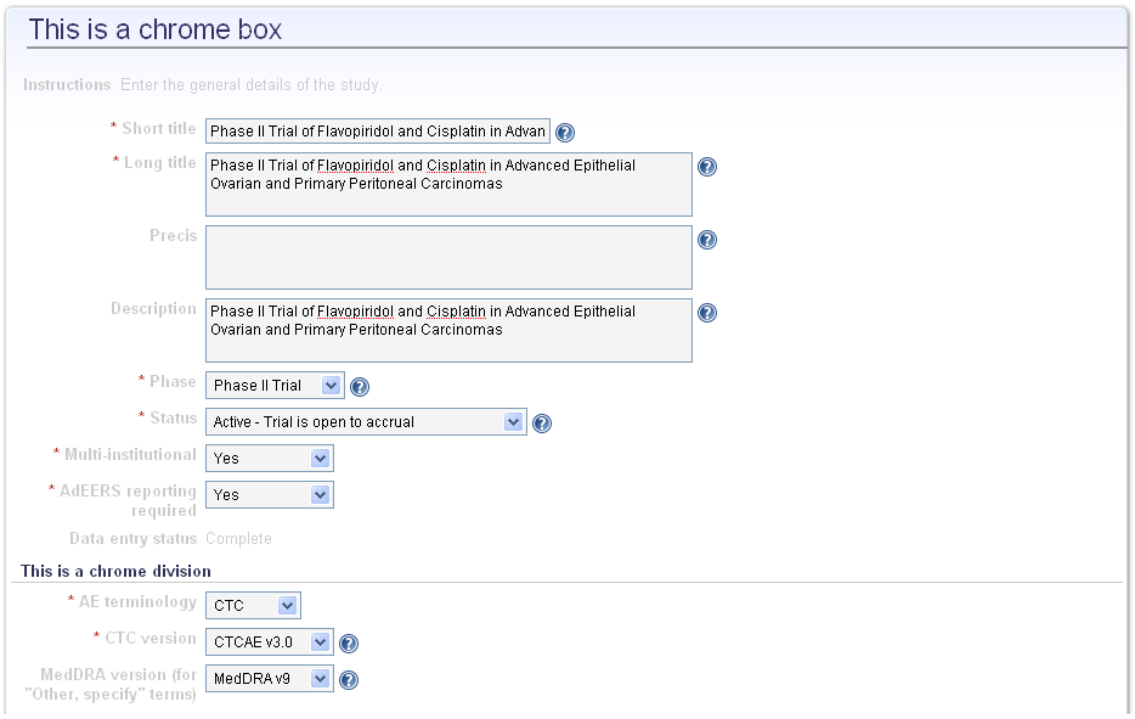
If you want to give your button an additional CSS class you can do so here, although you shouldn't need to most of the time.

id

```
id="submitButton"
```

If you want to give your button an id like any other element, do so here. This is useful for manipulating the button with Javascript.

Chrome Boxes and Divisions



This is a chrome box

Instructions Enter the general details of the study.

* Short title	Phase II Trial of Flavopiridol and Cisplatin in Advan
* Long title	Phase II Trial of Flavopiridol and Cisplatin in Advanced Epithelial Ovarian and Primary Peritoneal Carcinomas
Precis	
Description	Phase II Trial of Flavopiridol and Cisplatin in Advanced Epithelial Ovarian and Primary Peritoneal Carcinomas
* Phase	Phase II Trial
* Status	Active - Trial is open to accrual
* Multi-institutional	Yes
* AdEERS reporting required	Yes
Data entry status	Complete

This is a chrome division

* AE terminology	CTC
* CTC version	CTCAE v3.0
MedDRA version (for "Other, specify" terms)	MedDRA v9

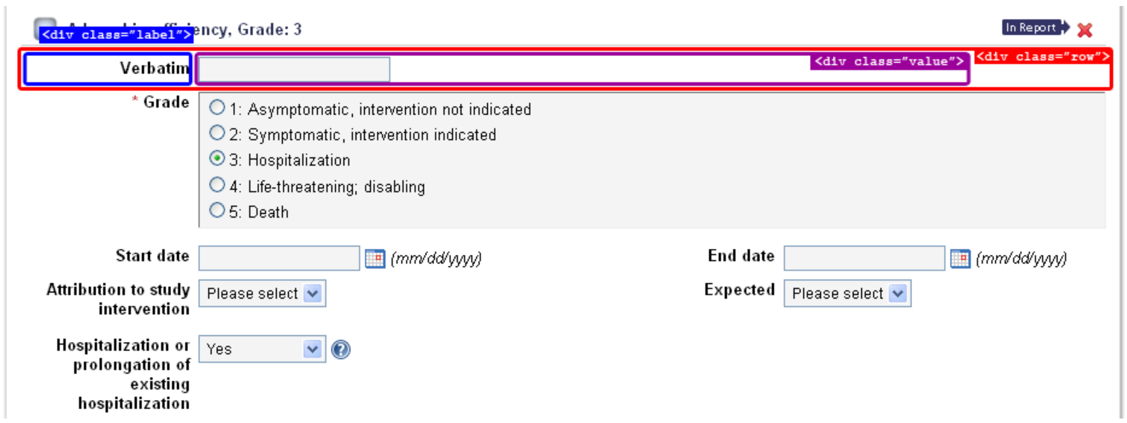
Fields and their Labels

One Column

When you want to lay out some fields with their associated labels, this is how to do it so it works consistently. The CSS classes are already set up.

```
<div class="row">
     <div class="label">
          <label for="mySelectBoxOrWhatever"> Label text goes here </label>
     </div>
     <div class="value">
          <select id="mySelectBoxOrWhatever"> or this can be text, or whatever you need </select>
     </div>
</div>
```

This is what is going on in your browser.



ency, Grade: 3

Verbatim

* Grade
- 1: Asymptomatic, intervention not indicated
- 2: Symptomatic, intervention indicated
- [✓ 3: Hospitalization]
- 4: Life-threatening; disabling
- 5: Death

Start date (mm/dd/yyyy)

End date (mm/dd/yyyy)

Attribution to study intervention: Please select

Expected: Please select

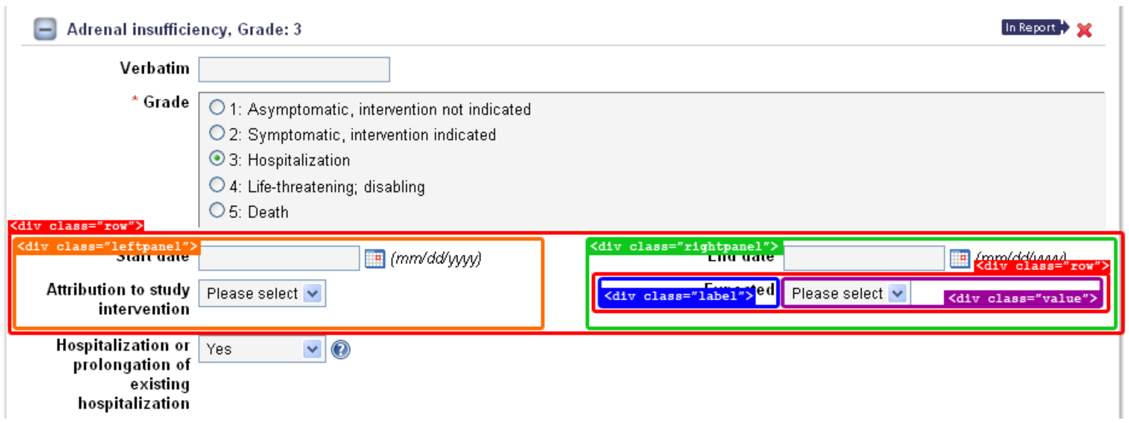
Hospitalization or prolongation of existing hospitalization: Yes

Two Columns

A left and right panel go inside a row, then inside that put your typical rows containing labels and values like above.

```
<div class="row">
     <div class="leftpanel">
          <div class="row">
               <div class="label">
                    <label for="mySelectBoxOrWhatever"> Label text goes here </label>
               </div>
               <div class="value">
                    <select id="mySelectBoxOrWhatever"> or this can be text, or whatever you need
</select>
               </div>
          </div>
     </div>
     <div class="rightpanel">
          <div class="row">
               <div class="label">
                    <label for="mySelectBoxOrWhatever"> Label text goes here </label>
               </div>
               <div class="value">
                    <select id="mySelectBoxOrWhatever"> or this can be text, or whatever you need
</select>
               </div>
          </div>
     </div>
</div>
```

This is what is going on in your browser.


Adrenal insufficiency, Grade: 3

Verbatim

* Grade
1: Asymptomatic, intervention not indicated
2: Symptomatic, intervention indicated
[✓ 3: Hospitalization]
4: Life-threatening; disabling
5: Death

Start date (mm/dd/yyyy)

Attribution to study intervention: Please select

End date (mm/dd/yyyy)

Expected: Please select

Hospitalization or prolongation of existing hospitalization: Yes

Interface Control Document - Organization Hierarchy

This document describes the interface caAERS expects for determining Organization hierarchy.

Requirement

- A Study Site should be able to determine who the main member (another Study Organization) is.

Interface

```
/**
 * Interface to determine organization hierarchy
 *
 */
public interface OrganizationHierarchyService {

 /**
  * Should return the parent (usually a main member) Organization.
  * @param site - Organization, a study site is representing.
  * @param study - OPTIONAL do we need this?
  * @return
  */
 Organization findParent(Organization site, Study study);
}
```

How it should work

A Spring bean (id : "**organizationHierarchyService**") must be defined that is an implementation of the below interface. caAERS, whenever/wherever needed will use "**organizationHierarchyService**" to determine the main member/parent.

There is no concept of organization hierarchy in caAERS, so the **default implementation** of OrganizationHierarchyService, will return null (OR may be the same organization in the input OR may be the coordinating center).

Adopters of caAERS, who maintain organization hierarchy must override the **organizationHierarchyService** bean with their own implementation.

Security - Content Filtering Design

Contents

1. Background / Introduction

Currently in order to achieve content filtering, caAERS uses the Acegi After Invocation technique as roughly outlined in hosted mode design document. In a nutshell what it dose is, fetch all entities matching the search criteria, then filter off the ones that the logged-in user should not see. As the data grew, this type of filtering has become a bottleneck and challenged the scalability of the system. Our new mantra is "*get only the chosen ones*". In this document we propose an approach that uses custom generated indexes to filter the data being fetched from backend.

2. Use Cases Affected

There is a direct impact on portions of below use cases that talks about content filtering.

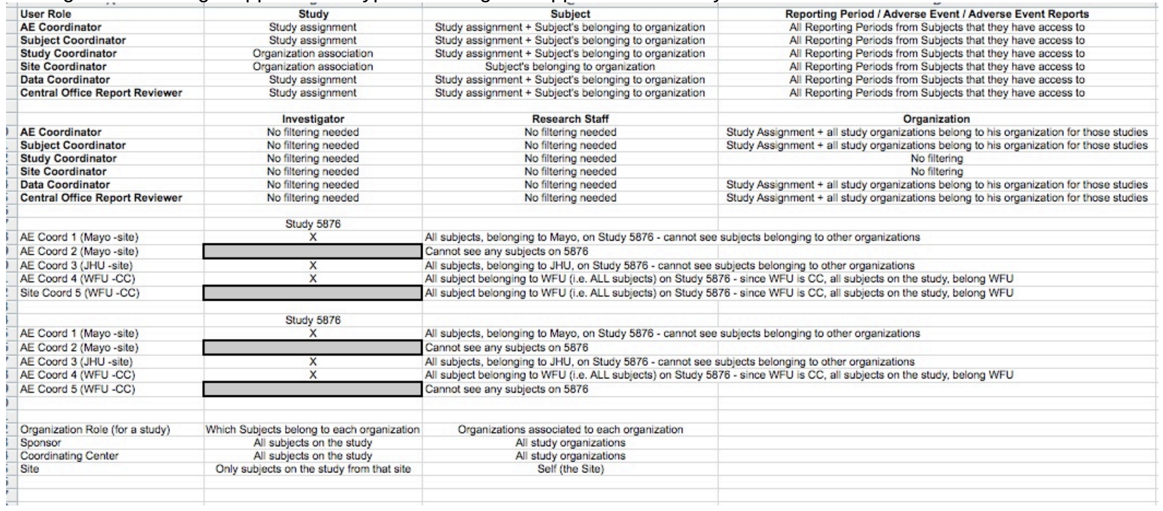
- Module 13 UC - Security
- 13.1 Using an Organization's system for Authorization - caAERS UC draft

3. Design Approach

Requirement

The requirement is that a user must only have access to content that they have been provisioned to access. Content provisioning is based upon the organization(s) of the user, the role(s) of the user's organization(s) on a study, the role of the user in the system, and the role of the user on the study.

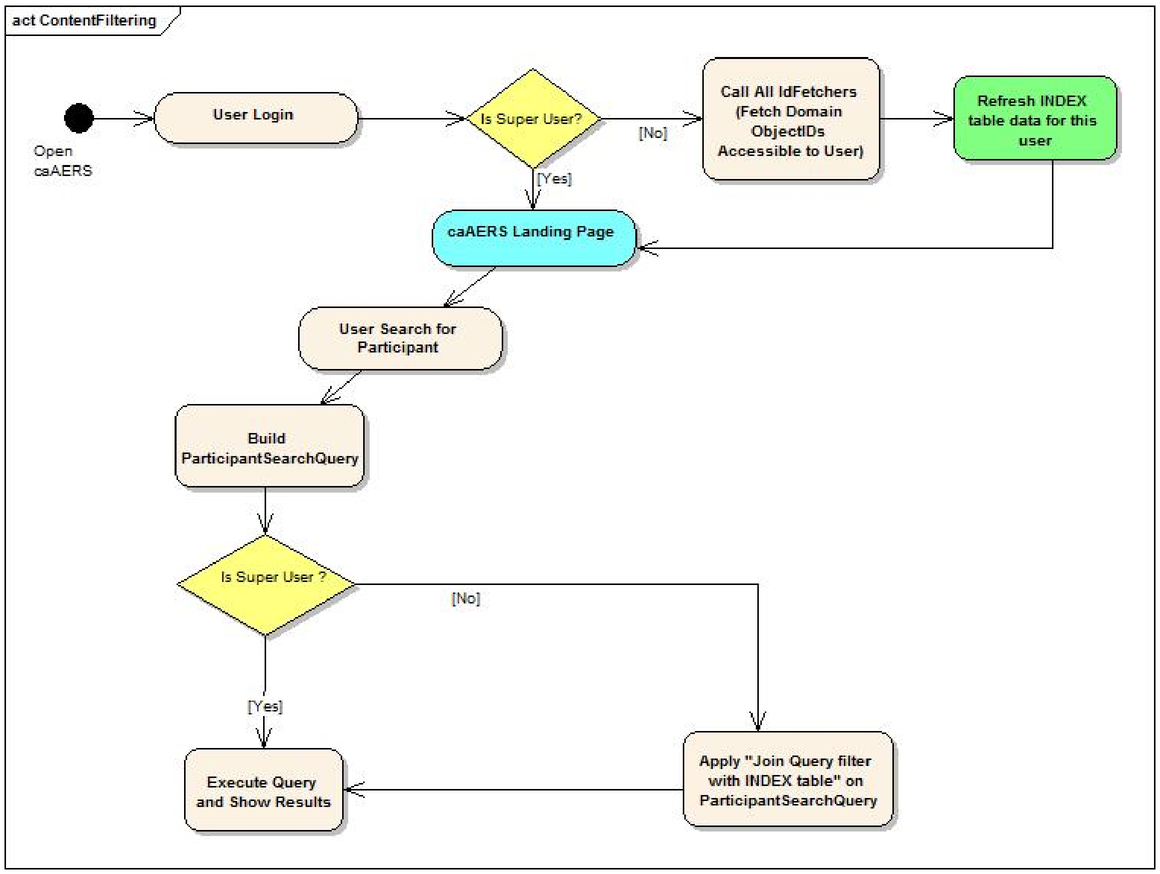
At runtime based on the logged-in person's role and kind of entity, either Organization filtering or Coordinating organization filtering or Study filtering or None will get applied. The typical filtering rule applied on an entity is mentioned below



User Role	Study	Subject	Reporting Period / Adverse Event / Adverse Event Reports
AE Coordinator	Study assignment	Study assignment + Subject's belonging to organization	All Reporting Periods from Subjects that they have access to
Subject Coordinator	Study assignment	Study assignment + Subject's belonging to organization	All Reporting Periods from Subjects that they have access to
Study Coordinator	Organization association	Study assignment + Subject's belonging to organization	All Reporting Periods from Subjects that they have access to
Site Coordinator	Organization association	Subject's belonging to organization	All Reporting Periods from Subjects that they have access to
Data Coordinator	Study assignment	Study assignment + Subject's belonging to organization	All Reporting Periods from Subjects that they have access to
Central Office Report Reviewer	Study assignment	Study assignment + Subject's belonging to organization	All Reporting Periods from Subjects that they have access to
	Investigator	Research Staff	Organization
AE Coordinator	No filtering needed	No filtering needed	Study Assignment + all study organizations belong to his organization for those studies
Subject Coordinator	No filtering needed	No filtering needed	Study Assignment + all study organizations belong to his organization for those studies
Study Coordinator	No filtering needed	No filtering needed	No filtering
Site Coordinator	No filtering needed	No filtering needed	No filtering
Data Coordinator	No filtering needed	No filtering needed	Study Assignment + all study organizations belong to his organization for those studies
Central Office Report Reviewer	No filtering needed	No filtering needed	Study Assignment + all study organizations belong to his organization for those studies
	Study 5876		
AE Coord 1 (Mayo -site)	X	All subjects, belonging to Mayo, on Study 5876 - cannot see subjects belonging to other organizations	
AE Coord 2 (Mayo -site)		Cannot see any subjects on 5876	
AE Coord 3 (JHU -site)	X	All subjects, belonging to JHU, on Study 5876 - cannot see subjects belonging to other organizations	
AE Coord 4 (WFU -CC)	X	All subject belonging to WFU (i.e. ALL subjects) on Study 5876 - since WFU is CC, all subjects on the study, belong WFU	
Site Coord 5 (WFU -CC)		All subject belonging to WFU (i.e. ALL subjects) on Study 5876 - since WFU is CC, all subjects on the study, belong WFU	
	Study 5876		
AE Coord 1 (Mayo -site)	X	All subjects, belonging to Mayo, on Study 5876 - cannot see subjects belonging to other organizations	
AE Coord 2 (Mayo -site)		Cannot see any subjects on 5876	
AE Coord 3 (JHU -site)	X	All subjects, belonging to JHU, on Study 5876 - cannot see subjects belonging to other organizations	
AE Coord 4 (WFU -CC)	X	All subject belonging to WFU (i.e. ALL subjects) on Study 5876 - since WFU is CC, all subjects on the study, belong WFU	
AE Coord 5 (WFU -CC)		Cannot see any subjects on 5876	
Organization Role (for a study)	Which Subjects belong to each organization	Organizations associated to each organization	
Sponsor	All subjects on the study	All study organizations	
Coordinating Center	All subjects on the study	All study organizations	
Site	Only subjects on the study from that site	Self (the Site)	

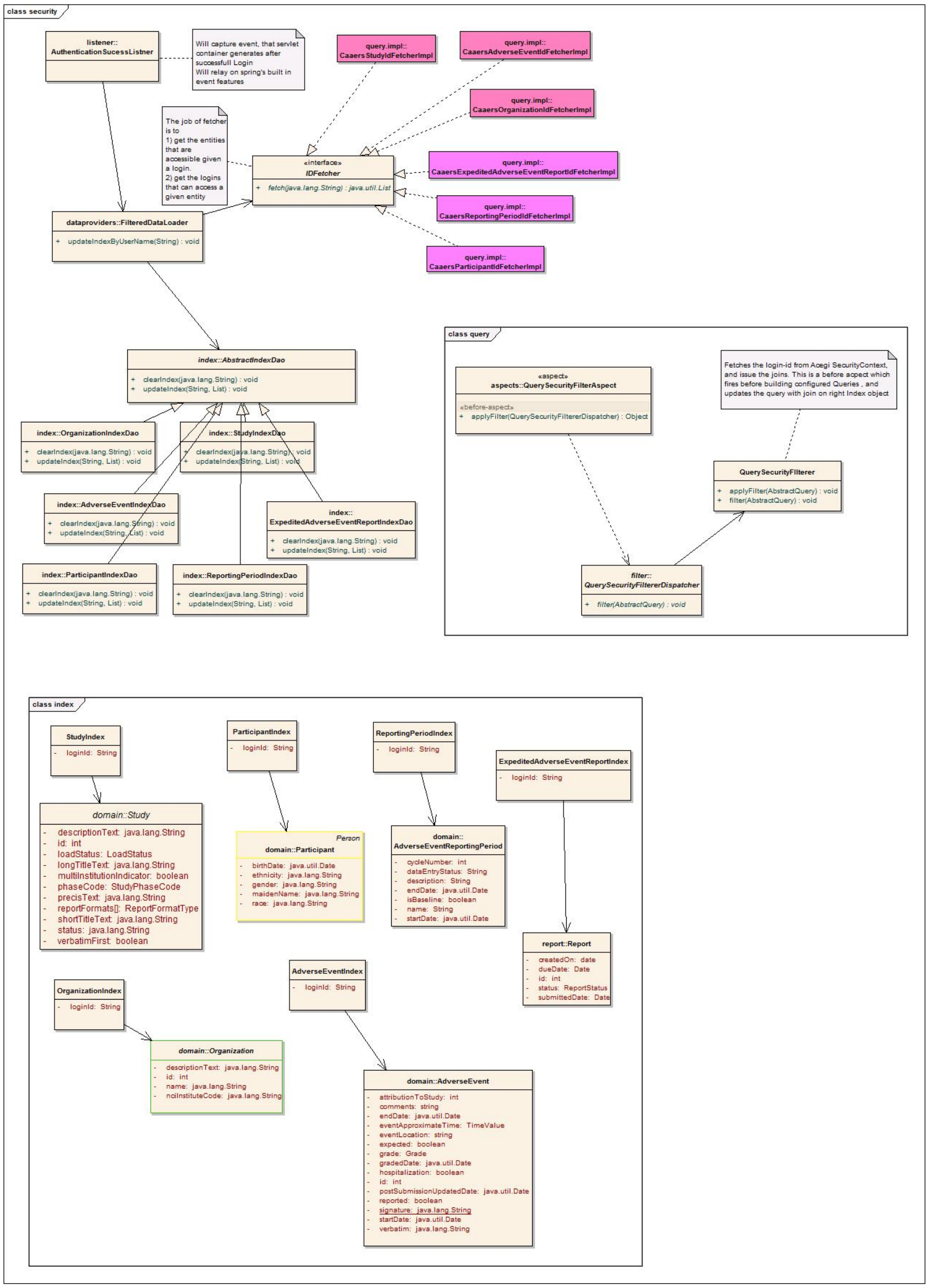
Activity Diagram

Activity diagram for Participant Query flow is outlined below.


act ContentFiltering
Open caAERS
User Login
Is Super User?
[No]
Call All IdFetchers (Fetch Domain ObjectIDs Accessible to User)
Refresh INDEX table data for this user
[Yes]
caAERS Landing Page
User Search for Participant
Build ParticipantSearchQuery
Is Super User ?
[No]
[Yes]
Execute Query and Show Results
Apply "Join Query filter with INDEX table" on ParticipantSearchQuery

Class diagram

A class diagram is outlined below.The classes in pink background needs to be overridden by sites whose authorization rules reside outside caAERS application.


class security
listener:: AuthenticationSucessListner
Will capture event, that servlet container generates after successfull Login Will relay on spring's built in event features
query.impl:: CaaersStudyIdFetcherImpl
query.impl:: CaaersAdverseEventIdFetcherImpl
query.impl:: CaaersOrganizationIdFetcherImpl
query.impl:: CaaersExpeditedAdverseEventReportIdFetcherImpl
query.impl:: CaaersReportingPeriodIdFetcherImpl
query.impl:: CaaersParticipantIdFetcherImpl
The job of fetcher is to 1) get the entities that are accessible given a login. 2) get the logins that can access a given entity
«interface» IDFetcher
+ fetch(java.lang.String) : java.util.List
dataproviders::FilteredDataLoader
+ updateIndexByUserName(String) : void
index::AbstractIndexDao
+ clearIndex(java.lang.String) : void
+ updateIndex(String, List) : void
index::OrganizationIndexDao
index::StudyIndexDao
index::AdverseEventIndexDao
index:: ExpeditedAdverseEventReportIndexDao
index::ParticipantIndexDao
index::ReportingPeriodIndexDao
+ clearIndex(java.lang.String) : void
+ updateIndex(String, List) : void
class query
«aspect» aspects::QuerySecurityFilterAspect
«before-aspect»
+ applyFilter(QuerySecurityFiltererDispatcher) : Object
Fetches the login-id from Acegi SecurityContext, and issue the joins. This is a before acpect which fires before building configured Queries , and updates the query with join on right Index object
QuerySecurityFilterer
+ applyFilter(AbstractQuery) : void
+ filter(AbstractQuery) : void
filter:: QuerySecurityFiltererDispatcher
+ filter(AbstractQuery) : void
class index
StudyIndex
- loginId: String
domain::Study
- descriptionText: java.lang.String
- id: int
- loadStatus: LoadStatus
- longTitleText: java.lang.String
- multiInstitutionIndicator: boolean
- phaseCode: StudyPhaseCode
- precisText: java.lang.String
- reportFormats[]: ReportFormatType
- shortTitleText: java.lang.String
- status: java.lang.String
- verbatimFirst: boolean
ParticipantIndex
- loginId: String
Person
domain::Participant
- birthDate: java.util.Date
- ethnicity: java.lang.String
- gender: java.lang.String
- maidenName: java.lang.String
- race: java.lang.String
ReportingPeriodIndex
- loginId: String
domain:: AdverseEventReportingPeriod
- cycleNumber: int
- dataEntryStatus: String
- description: String
- endDate: java.util.Date
- isBaseline: boolean
- name: String
- startDate: java.util.Date
ExpeditedAdverseEventReportIndex
- loginId: String
report::Report
- createdOn: date
- dueDate: Date
- id: int
- status: ReportStatus
- submittedDate: Date
OrganizationIndex
- loginId: String
domain::Organization
- descriptionText: java.lang.String
- id: int
- name: java.lang.String
- nciInstituteCode: java.lang.String
AdverseEventIndex
- loginId: String
domain::AdverseEvent
- attributionToStudy: int
- comments: string
- endDate: java.util.Date
- eventApproximateTime: TimeValue
- eventLocation: string
- expected: boolean
- grade: Grade
- gradedDate: java.util.Date
- hospitalization: boolean
- id: int
- postSubmissionUpdatedDate: java.util.Date
- reported: boolean
- signature: java.lang.String
- startDate: java.util.Date
- verbatim: java.lang.String

4. Design Details

This section will contain the design details at different layers. This section should be reviewed by the Tech Lead or Architect.

5. Tools & Technologies

This section will define the tools and technologies that will be utilized in the implementation. This section should be reviewed by the Tech Lead or Architect.

6. Custom Authorization step by step instructions.

This section will detail the special modifications required to other elements in the application in order for this implementation to be successful. Examples of items to include in this section would be changes to Tomcat, increases in memory or disk space, etc.

Sites need to override Id Fetchers implementation if authorization rules reside outside caAERS

Implementing an Id Fetcher .

- Implementation class should implement List fetch(String loginId) in com.semanticbits.security.contentfilter.IdFetcher.
- fetch method should return domain object IDS for provided loginId (userName). Domain object IDs should be caAERS DB primary keys .
- fetch method must return list of gov.nih.nci.cabig.caaers.domain.index.IndexEntry data type .
- IndexEntry data type supports accessible IDs based on roles . role can be null.

Fetcher	caAERS Database Table	Column
organizationIdFetcher	ORGANIZATIONS	ID
studyIdFetcher	STUDIES	ID
participantIdFetcher	PARTICIPANTS	ID
adverseEventIdFetcher	ADVERSE_EVENTS	ID
expeditedAdverseEventReportIdFetcher	AE_REPORTS	ID
reportingPeriodIdFetcher	AE_REPORTING_PERIODS	ID
researchStaffIdFetcher	researchstaff_index	ID
investigatorIdFetcher	investigator_index	ID

- Custom authorization systems
- need to implement only organizationIdFetcher and studyIdFetcher .
- no need to implement other fetchers as they depend on organization and study indicies .
- need to configure all other fetchers without any roles (in application context file).

Custom Authorization implementation steps .

STEP 1 : Implement Organization and Study Id Fetchers .
Sample Code - Organization IdFetcher Implementation

```
package edu.duke.calgb.caaers.fetchers.impl
import com.semanticbits.security.contentfilter.IdFetcher;
import gov.nih.nci.cabig.caaers.accesscontrol.query.impl.AbstractIdFetcher;
import gov.nih.nci.cabig.caaers.domain.UserGroupType;
import gov.nih.nci.cabig.caaers.domain.index.IndexEntry;
import java.util.List;

public class CalgbOrganizationIdFetcherImpl  extends  AbstractIdFetcher implements IdFetcher {

_     //Will return a list accessible organization IDs (_ORGANIZATIONS table ID column values) for given user

     @Override
     public  List fetch(String loginId) {
              List<Integer> organizationIdList = null;
             //call to your local authorization system which returns organization  IDS for given UserID

             organizationIdList = calgbAuthorizationProvider.getAccessibleOrganizationIds(loginId);
             List<IndexEntry> list = new ArrayList<IndexEntry>();

             IndexEntry ie = new IndexEntry(null);
             ie.setEntityIds(organizationIdList)
             list.add(ie)
             return list;
    }

}
```

Sample Code - Study IdFetcher Implementation

```
package edu.duke.calgb.caaers.fetchers.impl
import com.semanticbits.security.contentfilter.IdFetcher;
import gov.nih.nci.cabig.caaers.accesscontrol.query.impl.AbstractIdFetcher;
import gov.nih.nci.cabig.caaers.domain.UserGroupType;
import gov.nih.nci.cabig.caaers.domain.index.IndexEntry;
import java.util.List;

public class CalgbStudyIdFetcherImpl  extends  AbstractIdFetcher implements IdFetcher {

_     //Will return a list accessible study IDs (STUDIES table ID column values) for given user

     @Override
     public  List fetch(String loginId) {
              List<Integer> studyIdList = null;
             //call to your local authorization system which returns study  IDS for given UserID

             studyIdList = calgbAuthorizationProvider.getAccessibleStudyIds(loginId);
             List<IndexEntry> list = new ArrayList<IndexEntry>();

             IndexEntry ie = new IndexEntry(null);
             ie.setEntityIds(studyIdList)
             list.add(ie)
             return list;
    }

}
```

STEP 2 : Configure Study and Organization fetchers in *gov/nih/nci/cabig/caaers/applicationContext-core-indexing.xml*

```
File to update : gov/nih/nci/cabig/caaers/applicationContext-core-indexing.xml
point Id Fetcher beans to your implementation .

Existing entries
    <!-- ORGANIZATION , STUDY  FETCHERS IMPLEMENTATIONS -->
    <bean id="organizationIdFetcher" class="gov.nih.nci.cabig.caaers.accesscontrol.query.impl.CaaersOrganizationIdFetcherImpl"

          parent="abstractIdFetcher"/>
    <bean id="studyIdFetcher" class="gov.nih.nci.cabig.caaers.accesscontrol.query.impl.CaaersStudyIdFetcherImpl"
          parent="abstractIdFetcher"/>

Update above entries with new implementations . (Used class names from STEP 1 sample code )

    <!-- ORGANIZATION , STUDY  FETCHERS IMPLEMENTATIONS -->
   <bean id="organizationIdFetcher" class="edu.duke.calgb.caaers.fetchers.impl.CalgbOrganizationIdFetcherImpl"
          parent="abstractIdFetcher"/>
   <bean id="studyIdFetcher" class="edu.duke.calgb.caaers.fetchers.impl.CalgbStudyIdFetcherImpl"
          parent="abstractIdFetcher"/>
```

STEP 3 : Configure other fetchers in *gov/nih/nci/cabig/caaers/applicationContext-core-indexing.xml*

```
File to update : gov/nih/nci/cabig/caaers/applicationContext-core-indexing.xml

Remove properties (roles) from Id Fetcher bean definitions .
Existing Entry
    <bean id="participantIdFetcher" class="gov.nih.nci.cabig.caaers.accesscontrol.query.impl.CaaersParticipantIdFetcherImpl"
          parent="abstractIdFetcher">
          <property name="applicableSiteScopedRoles">
              <list>
                <value>subject_manager</value>
                <value>registration_qa_manager</value>
            </list>
          </property>
          <property name="applicableStudyScopedRoles">
              <list>
                <value>registrar</value>
                <value>ae_reporter</value>
                <value>ae_expedited_report_reviewer</value>
                <value>ae_study_data_reviewer</value>
                <value>data_reader</value>
                <value>data_analyst</value>
            </list>
          </property>
    </bean>

Remove properties from above bean definition .
    <bean id="participantIdFetcher" class="gov.nih.nci.cabig.caaers.accesscontrol.query.impl.CaaersParticipantIdFetcherImpl"
          parent="abstractIdFetcher"/>


Remove properties for adverseEventIdFetcher , expeditedAdverseEventReportIdFetcher , reportingPeriodIdFetcher , researchStaffIdFetcher,
investigatorIdFetcher
```

** Please contact caAERS support for advanced configuration .

7. References

- caAERS - User Roles and Rights