Architecture Final Report

Context

Introduction

Application security is a living process that must constantly be addressed throughout the application lifecycle. This requires continuous security assessments at every phase of the Software Development Lifecycle (SDLC). The idea is that, whenever new code is merged into current repository, the security test will be conducted, either locally or globally, to ensure the security of the software application.

Since a completely automated security evaluation service is not offered on commercial or open source platforms, we wish to create a Continuous Authorization (CA) service that automate the whole security test process. CAS system is intended to be integrated into the DevOps pipeline. During development stage, the primary tool will be Jenkins plugin. In the future, CAS wishes to support more continuous integration (CI) tools, such as TeamCity, Travis CI, Go CD, Bamboo etc.

WorkFlow

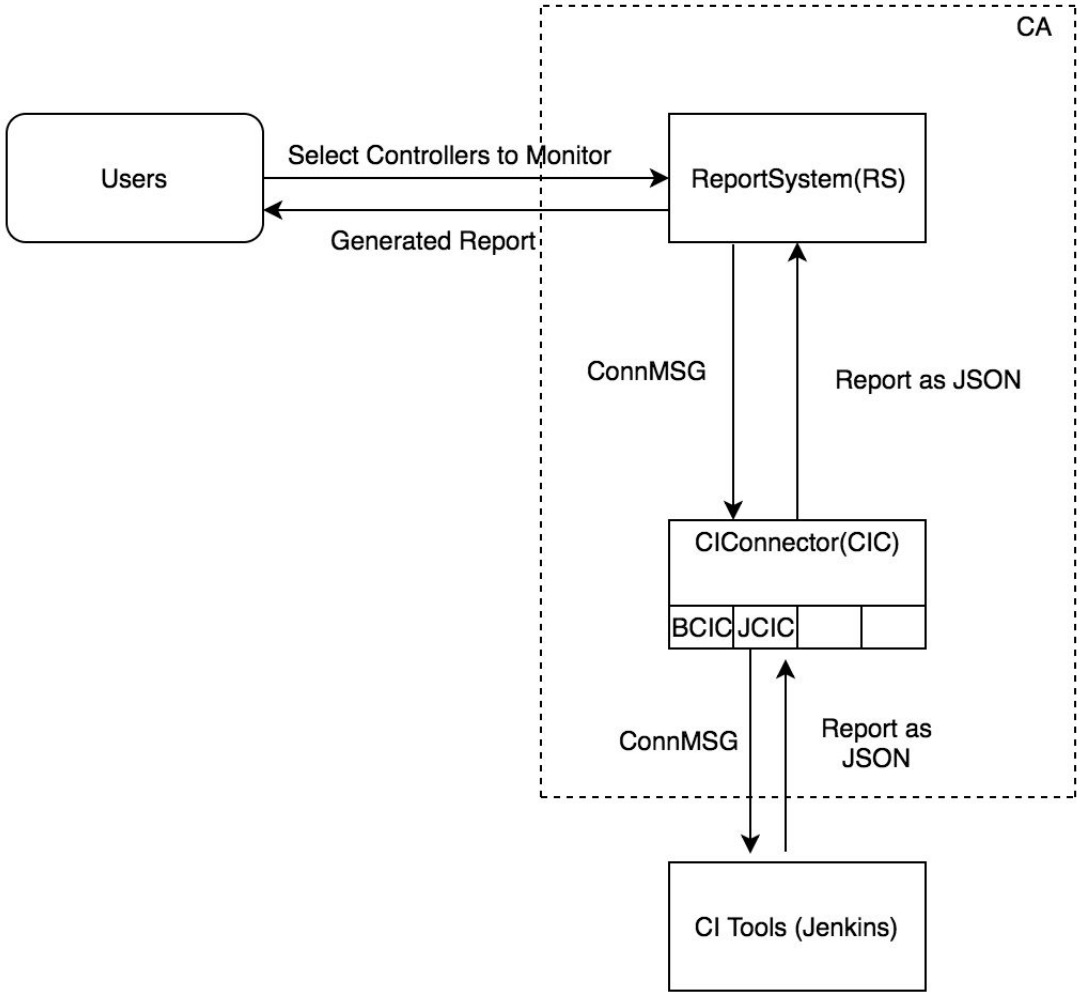

Figure-1 Context Diagram

Figure-1 shows the context diagram of CAS. Most security aspects can be checked by security controllers[1] described by Risk Management Framework (RMF). By specifying the types of controllers, users can specify which security aspects they want to CAS to track from CI.

At first, users will have to set up CAS service as a plugin tool in Jenkins. That means Jenkins will establish connections with CAS through CIConnector (CIC). CIC has different APIs as it

[1] Controllers E.g. Federal Risk and Authorization Management Program (FedRAMP) Profile Examples, https://github.com/usnistgov/OSCAL/tree/master/content/fedramp.gov

interacts with different CI Tools. Whenever the Jenkins build the project, the CIC will retrieve a report from CI that contains the test results of that build. CIC will send the report to ReportSystem (RS) for further processing.

RS system will parse the report w.r.t user-specified controllers to collect test results. RS will have to maintain a table to record controllers with their corresponding test in Jenkins. Once RS is done parsing the report, it will automatically update the dashboard, so that the users will be able to see the latest status of the security assessment.

Stakeholders

SEI CERT as Client

- An open-source platform: Continuous Authorization service
- Maintaining their reputation

Software Development Community

- An efficient product development process
- Guarantee security continuously during SDLC

User as a developing team

- Learning objectives
    - Practicing software development managing techniques
    - Practicing communication skills
    - DevOps pipeline
    - Automated testing
- Joining the open source community

Following Developers

- Maintainability
- Sustainability

Business Goals

1. This system will be integrated into DevOps pipeline via continuous integration tools.

Architecture Drivers

Functional Requirements

- The system shall be able to manage users authentication and credentials properly.
    - The system shall allow a new user to create an account through the website interface, setting his/her username and password.
- The system shall be able to let users manage project repositories properly.
    - The system shall allow a user account to create at most 12 project repositories at the same time.
    - The system shall allow a project repo to have at most 5 users.
- The system shall allow users to manage and monitor the continuous authorization process of a project.
    - The system shall be able to link each controller to different test cases provided by users.
    - The system shall allow the users to choose a different type of controllers for a certain project.
    - The system shall allow the users to import controllers settings for a project from previous projects.
    - The system shall allow the users to see the results (report) of the test of the controllers by using the dashboard.
- The system shall be able to work as a Jenkins plugin.
    - The system shall be able to be installed on Jenkins as a plugin.

Use case name:	Register project on CAS
Unique case ID:	RegisterProjectOnCAS
Primary Actor(s) :	User
Secondary Actor(s):	None
Brief description:	Users can register a new project on CAS.
Preconditions:	1. Users have access to CAS service. 2. Users have registered for a CAS account.
Flow of events:	1. Users go to the CAS website, log in to their account. 2. On the main page, users click on “Register a new Project” to go to the registering page. 3. On the registering page, users input the basic information of the project. a. Users type the “Project Name”, “Project Description”,

	“Contributors” and “Project Owner” into different input box respectively. b. Users select “Project Type” from a drop-down list. c. Users click on the register button to finish the registration. The page will automatically go to the project information page. 4. The project information page would show all the information including “Project Name”, “Project Description”, “Contributors”, “Project Owner”, “Project Type”. The page will also show the automatically generated random “Project ID” and “Project Password” used for connection to the CI platform (E.g. Jenkins).
Postconditions:	1. A new project is registered. 2. The project information is stored. 3. Project ID and password are automatically generated.
Priority	High
Alternative flow or exception:	1. In Step 3c, if the user left any input boxes or drop-down list blank, stay on the same page and show “please fill in the blank” notices on the blank boxes. 2. In Step 3c, if the input “Project Name” has already existed in the system, stay on the same page and show “Project name has already been used” notice on the “Project Name”.
Non-behavioral requirements	1. The “project types” should include all the common project types. For uncommon projects, users can select “others”. 2. QA-P05
Assumptions:	1. The project information includes “Project Name”, “Project Description”, “Contributors”, “Project Owner”, “Project Type”. 2. Users need “Project ID” and “Project Password” to build an encrypted connection to the CI platform.
Source:	

Use case name:	Configure the controller list for project
Unique case ID:	ConfigureProjectControllers
Primary Actor(s) :	User
Secondary Actor(s):	None
Brief description:	Users configure the controllers that they are concerned with for a certain

	project.
Preconditions:	1. Users have access to CAS service. 2. Users have registered for a CAS account. 3. The project has been registered on CAS.
Flow of events:	1. Users go to the CAS website, log in to their account. 2. Go to the page of “Project management”. 3. Click the name of the project that users want to configure to go to the settings page of this project. 4. Click “Manage controllers” to go to the page which shows all provided controllers in a tree structure. 5. Select some classes of controllers to get all the controllers of these classes selected. 6. Click a controller class to make the controllers of this class expand below it, then select some of them. 7. Input the title of a certain controller in the search box then find the controller the users want, then select it. 8. After selecting all the controllers to be tested, click the “Save” button.
Postconditions:	1. The controllers to be tested are configured for this project. 2. If some new tests are completed, the dashboard will show the result of these selected controllers.
Priority	High
Alternative flow or exception:	1. If users want to select the controllers which are always needed to be tested in a type of project, they can select the type of this project to have the related controllers selected. 2. If the users want to test the most of the controllers that were configured for their previous project, they can click “Import from the previous project”, then the controllers that have been saved for the previous project will be selected.
Non-behavioral requirements	1. QA-U02 2. QA-P02 3. QA-P03 4. QA-P04
Assumptions:	1. Users are familiar with the controllers.
Source:	

Use case name:	Install CAS plugin on Jenkins

Unique case ID:	JenkinsInstallCASPlugin
Primary Actor(s) :	User
Secondary Actor(s):	None
Brief description:	Users install the CAS plugin on their Jenkins service.
Preconditions:	1. Users have set up a Jenkins server.
Flow of events:	1. Users log on the Jenkins GUI 2. Select “Manage Jenkins” 3. Select “Manage Plugins” 4. Click the “Available” tab 5. Type “CAS” in the Filter field 6. Check “CAS” in the search results 7. Select “Install without restart” 8. Wait until the installation progress page displays “Success”
Postconditions:	1. CAS plugin has been installed on Jenkins
Priority	High
Alternative flow or exception:	In event flow step 7, users can also do: 1. Select “Download now and install after restart” 2. Wait until the download progress page displays “Downloaded Successfully. Will be activated during the next boot” 3. Restart Jenkins Then the postconditions can be reached with this alternative flow.
Non-behavioral requirements	1. QA-P06
Assumptions:	Users are familiar with Jenkins configuration processes.
Source:	Own experience using Jenkins and Jenkins plugins.

Use case name:	Configure CAS plugin on Jenkins
Unique case ID:	JenkinsConfigCASPlugin
Primary Actor(s) :	User
Secondary Actor(s):	None

Brief description:	Users have installed CAS plugin on Jenkins
Preconditions:	1. Users have set up a Jenkins server. 2. Users have installed CAS plugin on Jenkins. 3. Users have logged on to Jenkins GUI. 4. Users have set up a CAS server. 5. Users have registered for a CAS account.
Flow of events:	1. Go to Jenkins system configuration 2. Under “CAS servers”, click “Add CAS” and provide: a. CAS username b. CAS password c. CAS server authentication token 3. Create a Jenkins job 4. Go to the job configuration 5. Under “Build”, click “Add build step”, and select “Execute CAS check” 6. Provide the following information to Jenkins: a. Project ID generated by CAS b. Project authentication token 7. Save build configuration.
Postconditions:	1. Jenkins automatically invoke CAS check after every build of a project.
Priority	High
Alternative flow or exception:	
Non-behavioral requirements	1. QA-Se02
Assumptions:	Users are familiar with Jenkins build and configuration processes.
Source:	Own experience using Jenkins and Jenkins plugins.

Use case name:	View CAS result on the dashboard
Unique case ID:	ViewCASDashboard
Primary Actor(s) :	User
Secondary Actor(s):	None
Brief description:	Users can view the testing results of their existing project by visiting the

	dashboard.
Preconditions:	1. Users have a CAS account. 2. Users have already set up a project on their CAS. 3. Users have internet access.
Flow of events:	1. Users go to the CAS website, log in to their account. 2. On the main page, users click on the “Existing” button to go to the existing project page. 3. On the existing project page, users click on “dashboard” button view the dashboard.
Postconditions:	1. Users can see the test results of an existing project by visiting the dashboard page for that project.
Priority	High
Alternative flow or exception:	1. On the dashboard page, users can click on each testing case to view the detail information of that case. Possible information includes: a. Case name. b. Controllers associated with this case. c. Start time for that case. d. The end time for that case. e. Output for that test case. f. Error message for that test case if it fails.
Non-behavioral requirements	1. The dashboard should show a list of test cases associated with their names, start time, end time and test results. Users should be able to click on each test case to view detail information of that test case. 2. QA-P01 3. QA-A01 4. QA-St01
Assumptions:	1. The connection between CAS and Jenkins works fine.
Source:	

Use case name:	View CA history of a project
Unique case ID:	ViewCAHistory
Primary Actor(s) :	User
Secondary Actor(s):	None

Brief description:	Users can view the history of authentication testing of a project by using CAS.
Preconditions:	1. Users have a CAS account. 2. Users have already set up a project on their CAS. 3. Users have internet access.
Flow of events:	1. Users go to the CAS website, log in to their account. 2. On the main page, users click on the “Existing” button to go to the existing project page. 3. On the existing project page, users click on “dashboard” button view the dashboard. 4. On the dashboard page, click the “history” button to view the history of the authentication test.
Postconditions:	1. Users can see the test history of the authentication test for an existing project.
Priority	High
Alternative flow or exception:	None
Non-behavioral requirements	1. CAS service will start to generate the history diagram for a project when it is created. 2. The history diagram contains information about the authentication test, including: a. The number of errors in a project at each time. b. The number of successful test in a project at each time. c. Each test case status along the timeline.
Assumptions:	1. The connection between CAS and Jenkins works fine.
Source:	

Quality attributes

ID	QA-P01
Category	Performance
Source	Jenkins
Stimulus	A new testing report is generated in Jenkins.
Artifact	CAS service dashboard

Environment	Running time
Response	The dashboard should show the latest results of the testing.
Response Measure	The time CAS takes to update the dashboard in less 90 seconds.

ID	QA-P02
Category	Performance
Source	User
Stimulus	Requires to view the list of controller classes.
Artifact	CAS web service
Environment	Running time
Response	The browser shows the list of controller classes on the web page.
Response Measure	Show the information within 3 seconds after the user’s request.

ID	QA-P03
Category	Performance
Source	User
Stimulus	Requires to view the expanded list of controllers under a specific controller class.
Artifact	CAS web service
Environment	Running time
Response	The browser shows the expanded list of controllers under that controller class.
Response Measure	Show the information within 5 seconds after the user’s request.

ID	QA-P04
Category	Performance
Source	User
Stimulus	Requires to view descriptions of controllers.
Artifact	CAS web service
Environment	Running time
Response	The browser shows required controllers on the webpage.
Response Measure	Show the information within 3 seconds.

ID	QA-P05
Category	Performance
Source	User
Stimulus	Requires to registrate a new CAS account on the web console.
Artifact	CAS web service
Environment	Running time
Response	The registration process finishes.
Response Measure	After the user submits all required information, the registration should be finished within 5 seconds.

ID	QA-P06
Category	Performance
Source	User
Stimulus	Requires to install CAS plugin in Jenkins.
Artifact	Jenkins

Environment	Running time
Response	The installation process finishes.
Response Measure	The installation should be finished within 1 minute.

ID	QA-M01
Category	Modifiability
Source	Developers in open source society.
Stimulus	A directive to add new plugins for letting CAS compatible with other CI services (e.g. Travis).
Artifact	Components
Environment	Design time
Response	The new plugin component should be completed and tested.
Response Measure	1. Only one new plugin component is to be added to the system. 2. Other system components should remain unaffected.

ID	QA-U01
Category	Usability
Source	QA team (the end user)
Stimulus	QA team try to configure controllers for CAS projects on the UI.
Artifact	CAS UI
Environment	Runtime
Response	QA team make/modify the project configuration efficiently.
Response Measure	1. Task time shall be within 5 minutes on average. 2. User satisfaction shall be greater than 90%.

ID	QA-U02
Category	Usability
Source	QA team (the end user)
Stimulus	QA team try to view and select controllers on the CAS UI.
Artifact	CAS UI
Environment	Runtime
Response	The list of provided controllers to be selected should be shown in a clear view so that the users can browse and find them easily.
Response Measure	1. The controllers should be grouped into no more than 20 classes to be shown to the user. 2. The controller classes should be shown in the alphabetical order. 3. The controllers under a class should be shown in the alphabetical order.

ID	QA-A01
Category	Availability
Source	Internet error/User server error/user mistake
Stimulus	The CAS service is down
Artifact	CAS
Environment	Running time
Response	The CAS service should restart itself.
Response Measure	The time CAS takes to update the dashboard in less 90 seconds.

ID	QA-St01
Category	Stability

Source	The user or internal/ external error.
Stimulus	CA service is down or is closed manually.
Artifact	Jenkins plugin
Environment	Run time
Response	The testing reports should not be dropped. The reports should be saved by Jenkins plugin and be sent to CA service after it is back to work.
Response Measure	1. All the reports during the CA service shutdown time should be saved. 2. The reports should be sent to CA in 1 minute after CA recovering.

ID	QA-Se01
Category	Security
Source	Malicious users
Stimulus	An unauthorized attempt is made to view or change the data.
Artifact	Testing results and other data stored in the system
Environment	Normal operations
Response	The system will detect the unauthorized request to query the data.
Response Measure	All unauthorized requests should be avoided.

ID	QA-Se02
Category	Security
Source	Malicious users
Stimulus	An unauthorized attempt is to access the CAS credentials through Jenkins.
Artifact	Jenkins
Environment	Normal operations
Response	The CAS credentials should be protected from malicious users.

Response Measure	After the user input the CAS password on the Jenkins configuration page, there will be no Copy-to-Clipboard or Unmask icons that would allow a user to have access to the password.

Constraints

Technical Constraints

ID	Technical Constraint
TC01	The system shall be compatible with Continuous Integration platforms, such as Jenkins.
TC02	The CAS service shall be able to be deployed using Docker.
TC03	The CAS service shall consume less than 8G memory.
TC04	The system should be tested using Selenium and SonarQube.
TC05	The CAS service should be written in Python.
TC06	The CAS service should use the Django framework.
TC07	The Jenkins plugin should be implemented in Java.
TC08	The development process should be agile and use the Microcosm DevOps Pipeline.
TC09	The version control of the system should be managed by GitHub.

Business Constraints

ID	Business Constraint
BC01	The project has to be done within 8 months.
BC02	The budget of the project is less than $1,000.
BC03	The project will be taken over by the following developers since Aug 2019.
BC04	The project will not regular maintenance force.

Description of the design in terms of different views

CC views & sequence diagram

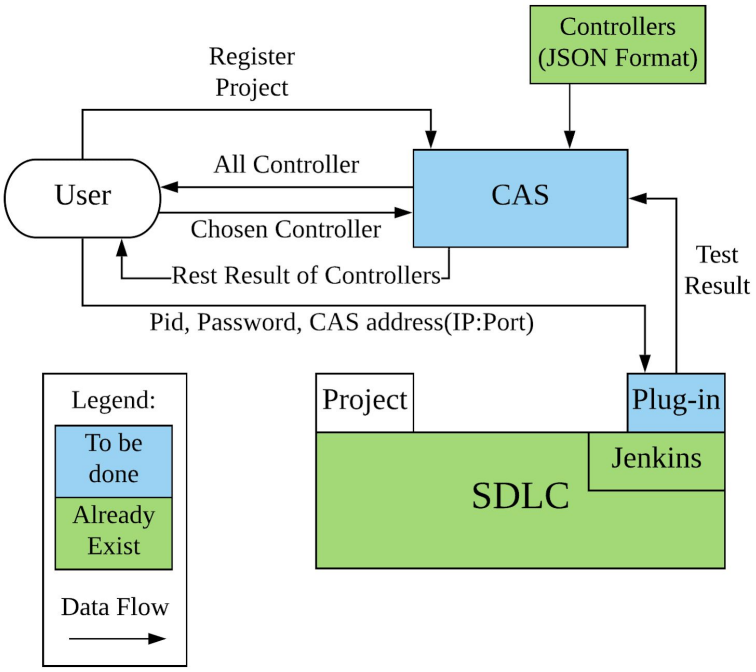

Figure 1 - Context Diagram
The components to develop and the data flow between components

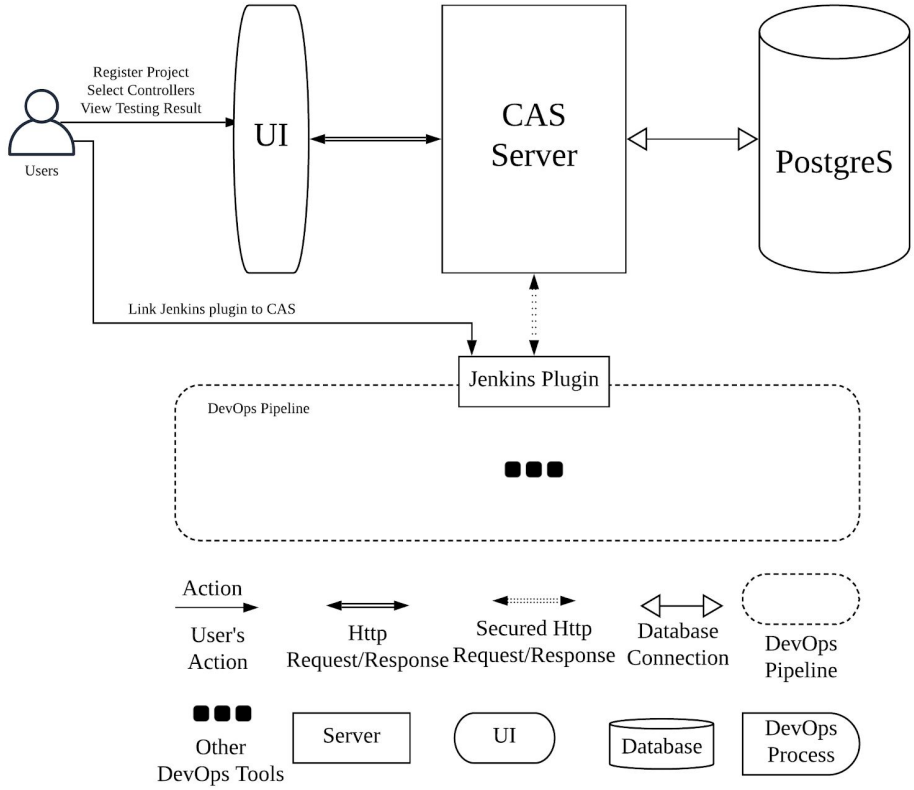

Figure 2 - C&C Diagram

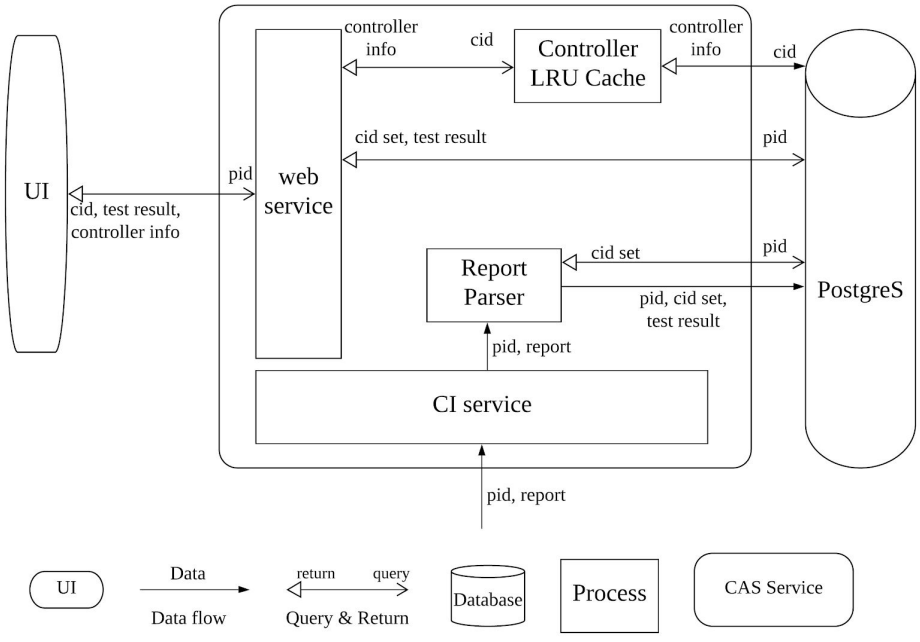

Figure 3 - Dataflow of CAS server - View Testing Result

Analysis of the architecture

Support of Key Architecture Drivers

- We build the CAS service as a Jenkins plugin.
- We have a plug-in service to support different Continuous Integration platforms (E.g. Bamboo).
- Mostly, there are thousands of controllers in the database and most of them have very long descriptions. These controllers are queried frequently and the DB connection would increase the round-trip time (RTT) of the queries. In order to reduce the RTT, we add a controllers LRU cache to CAS service. It helps to decrease the DB connection.

Key Tradeoffs

- We decided to put the static analysis testing report parser on the CAS server instead of the Jenkins plugin. A parser on the Jenkin plugin can help to decrease the data need to be transferred from Jenkins to CAS server. However, it would increase the complexity of the Jenkins plugin. Future developers need to spend more time on developing plugins for other CI tools. It is against our quality attribute of modifiability. So we decided to put parse on the CAS server.

Justification of Design