Configure advanced options

On this page:

Enable User Input Security checks

In order to detect User Input security flaws, CAST provides an optional analysis feature that implements a dataflow algorithm to detect variables that flow from the user down to a critical resource without prior sanitization, allowing hackers to send them data that will harm IT applications (security vulnerabilities).

As mentioned by the Common Weakness Enumeration (CWE), SANS Institute and OWASP, **Improper Input Validation** is the top group of web programming errors that can lead to security vulnerabilities.

Improper Input Validation is defined by the CWE follows:

> *"When software fails to validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution."*

The **User Input Security** feature in the CAST Management Studio enables users to detect improper user input validation in the application's source code, which can lead to the following security vulnerabilities:

- SQL Injection (CWE-89)
- Cross-Site Scripting (CWE-79)
- LDAP Injection (CWE-90)
- OS Command Injection (CWE-78)
- XPath Injection (CWE-91)
- Path Manipulation (CWE-99)
- Avoid Log forging vulnerabilities (CWE-117)

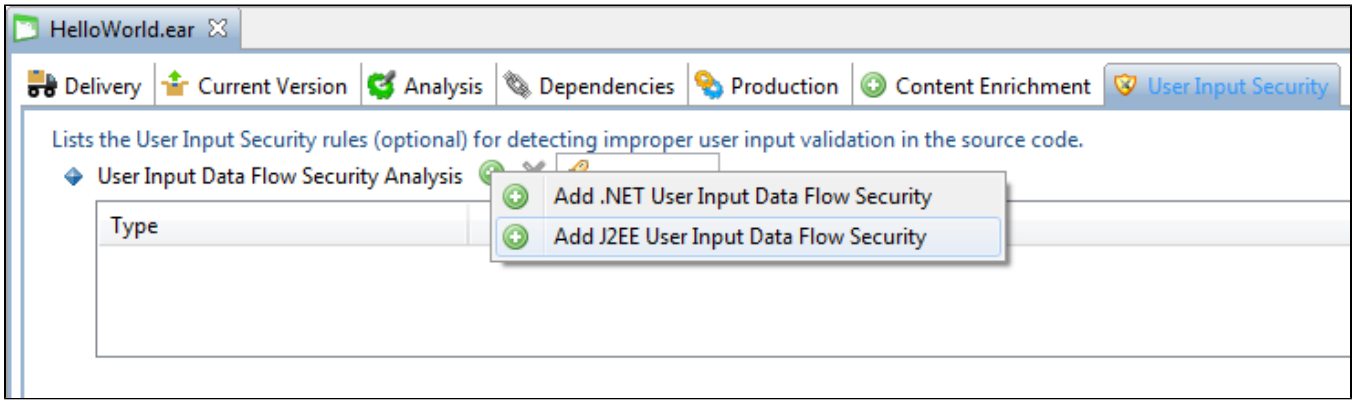
Once the source code analysis is complete, the results (i.e. the security vulnerabilities) can be viewed in the **CAST Engineering Dashboard** as standard Quality Rules. As dataflow-based Quality Rules, the flow of the user input can be tracked in the CAST Engineering Dashboard with bookmarked source code. Please see the CAST Engineering Dashboard - CED for more information about the Violation Viewer that is used to display results of User Input Security analyses.

How to implement it

CAST currently supports User Input Security for the following technologies:

- **J2EE** (using the J2EE Analyzer)
- **.NET** (using the .NET new Analyzer)

To enable User Input Security checks in your application analysis using the **CAST Management Studio**, it is necessary to activate the feature by adding either a .NET or J2EE configuration within the **User Input Security** tab in the **Application** editor:



When this is done, you can then configure the fully qualified name of the sanitization method(s) used in the application. If the data coming from the user is sent to the target methods without prior sanitization, then the analyzer will detect a security vulnerability.

The User Input Security feature will still work even if you do not input any Sanitization Methods into the GUI (however in this case, if a sanitization method is used in the application's source code, vulnerabilities retrieved by the User Input Security may not all be real security issues and will need further checking). This method is usually used, when you do not know if a sanitization method is in fact used by the application: an initial analysis without adding the sanitization method allows you to check the path of the vulnerabilities in the CAST Engineering Dashboard (Input Validation technical criteria) to find out the name and full qualified name of the sanitization method(s). Once you have these, you can then input them in to the GUI in the CAST Management Studio before re-analyzing the application.

If you cannot determine the fully qualified name of the sanitization method(s) or even ascertain its use with the help of the App Team SME you may consider a brute force approach and run the analysis without specifying the sanitization method. This will allow you to check the path of the vulnerabilities in the CAST Engineering Dashboard (Input Validation technical criteria) to find out the name and fully qualified name of the sanitization method(s). In this case, after completing the configuration of the User Input Security feature you must rerun the analysis and regenerate the snapshot.

For an in depth discussion of this feature, see also:

- User Input Security - Adding Application Specific Input or Target Methods
- User Input Security - analysis requirements and troubleshooting
- User Input Security - Internals
- User Input Security - Using FlawExplorer to speed up result check
- User Input Security - Predefined Methods
- User Input Security - configuring blackbox methods

Enable XXL table Quality Rules

XXL table Quality Rules are performance related Quality Rules that help detect incorrect or poorly performing SQL queries running very large tables, containing a large amount of data. The threshold that determines when a table is considered an XXL table can be configured by the user and it is set by default at 100,000 rows.

In order to accurately assess XXL table performance, it is necessary to provide the analyzer with table row size information from production systems - development / integration systems may not feature really large tables and would not help detect real threat on application performance levels. If the information is not physically accessible (e.g.: first release of an application with no production environment), it is worth simulating the information by identifying tables that are expected to be large and by inputting the expected row size for of the pre-identified XXL table.

For detailed instructions on how to identify database tables and configure production/expected row size for selected XXL table see XXL tables Quality Rules enablement.

Define a Module strategy (User Defined Modules)

Modules represent executable software components or tightly coupled sets of executable software components, developed and deployed together, that deliver some of the functionality provided by an Application.

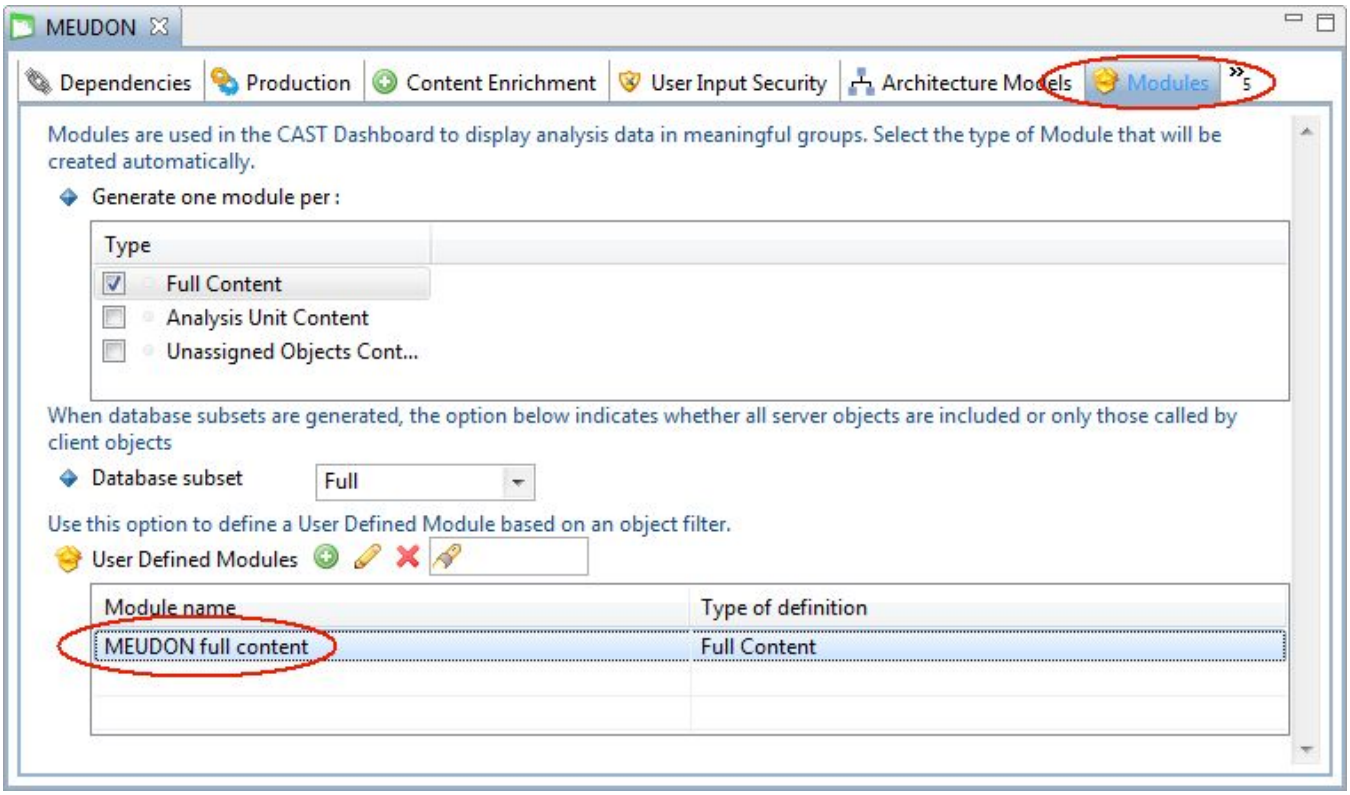
By default the "Full Module Content" module will always be created unless you change the settings - i.e. source code from all Analysis Units will be placed in one module. This default module belongs to the "My System" system. You can choose to create your own user defined modules in addition to or instead of the default, or you can use the other automatic Module strategy "one module for each Analysis Unit".

The definition of Modules within Applications impacts the accessibility and usability of Application assessment results. Specifically, in the CAST quality and quantity model, the Module is the smallest assessment entity. The definition of a Module can improve the relevance of the analysis results by linking modules to development teams and application functional layers etc.

For further consideration about Modules creation pros and cons see also: Focus on Module content

Objects produced during the analysis of an application which are not included in a User Defined Module will not appear in the Architecture Checker, Transaction Configuration Center or TRI. All objects generated during the analysis of your application must be included in a User Defined Module even if the objects are produced by a Tool in Content Enrichment.

CAST recommends creating **a Module for each functional component** of an application. Modules can be built in the CAST management Studio automatically or manually for maximum configuration flexibility:



CAST offers three automatic modules for display of source code in the CAST Engineering Dashboard:

Full Content	This Module is selected by default. It corresponds to all the source code for all Analysis Units configured in your Application. It will be called "***<Application name>* full content**". If this content is sufficient for your needs, you do not need to select any other automatic or User Defined Modules.
Analysis Unit Content	If you select this option, the CAST Management Studio will create one Module per Analysis Unit in your Application. Each Module will be called "***<Analysis Unit name>* content**".
Unassigned Objects Content	This option should be activated when you are exclusively using User Defined Modules (i.e. the Full Content and Analysis Unit Content are not ticked). When ticked, an automatic Module will be created containing any objects that were not assigned to one of your User Defined Modules, enabling you to see them in the CAST Engineering Dashboard.

Assessment Model fine tuning

Before generating the snapshot, the CAST AI Admin may also want to consider customization of:

- Aggregation rules for Modules
- Selection of Critical Violation rules (support for an Automated Action Plan generation)
- Disable selected quality rules
- Rules with parameters and Naming Convention
- Configure Mutually exclusive rules

Please see Assessment Model for more information about this.

Architecture Model configuration

The goal of the CAST Archictecture Checker is to enable Enterprise Design and Architecture Checks. Enterprise Architects can now capture and store Application Models and check overtime actual applications against these models. CAST enables real enterprise architecture checks as it offers checks of applications built using multiple languages and technologies. It fully leverages CAST's cross-language, cross-technology capabilities and CAST end-to-end application analysis.

Until recently, it was difficult to do architecture conformance checks just because architecture rules had to be very generic. Now with the Architecture Checker, Enterprise Architects can define architectures models that fit their specific application designs. Checking overtime actual applications against these architecture models is very important to avoid any **Security and Robustness issues**.

The goal of the CAST Archictecture Checker is also to help define quality rules at the application and design level without programming. Most engines that can help check design and architecture rules require users to master a programming language and a specific API. CAST Architecture Checker enables the definition of quality & architecture rules just by drag and drop. And it enables also interactive model checking and tuning directly into CAST Architecture Checker.

Summary

The CAST Architecture Checker GUI (**CAST-ArchiChecker.exe**) provides a means for you to:

- Create an Architecture Model that reflects a business application or portfolio of applications via the use of Layers and Dependencies between those Layers
- Define the contents of Layers in terms of objects resulting from a CAST analysis

Once the Architecture Model is defined, the resulting model is stored in a CASTArchitect file that can then be used in the CAST Management Studio. When your application or portfolio of applications is then analyzed in the CAST Management Studio and a snapshot is subsequently generated, the Architecture Model will be checked to control that the selected application comply with the model.

Results and information can then be consulted in the CAST Application Analytics Dashboard/CAST Engineering Platform.

Getting started with the CAST Architecture Checker

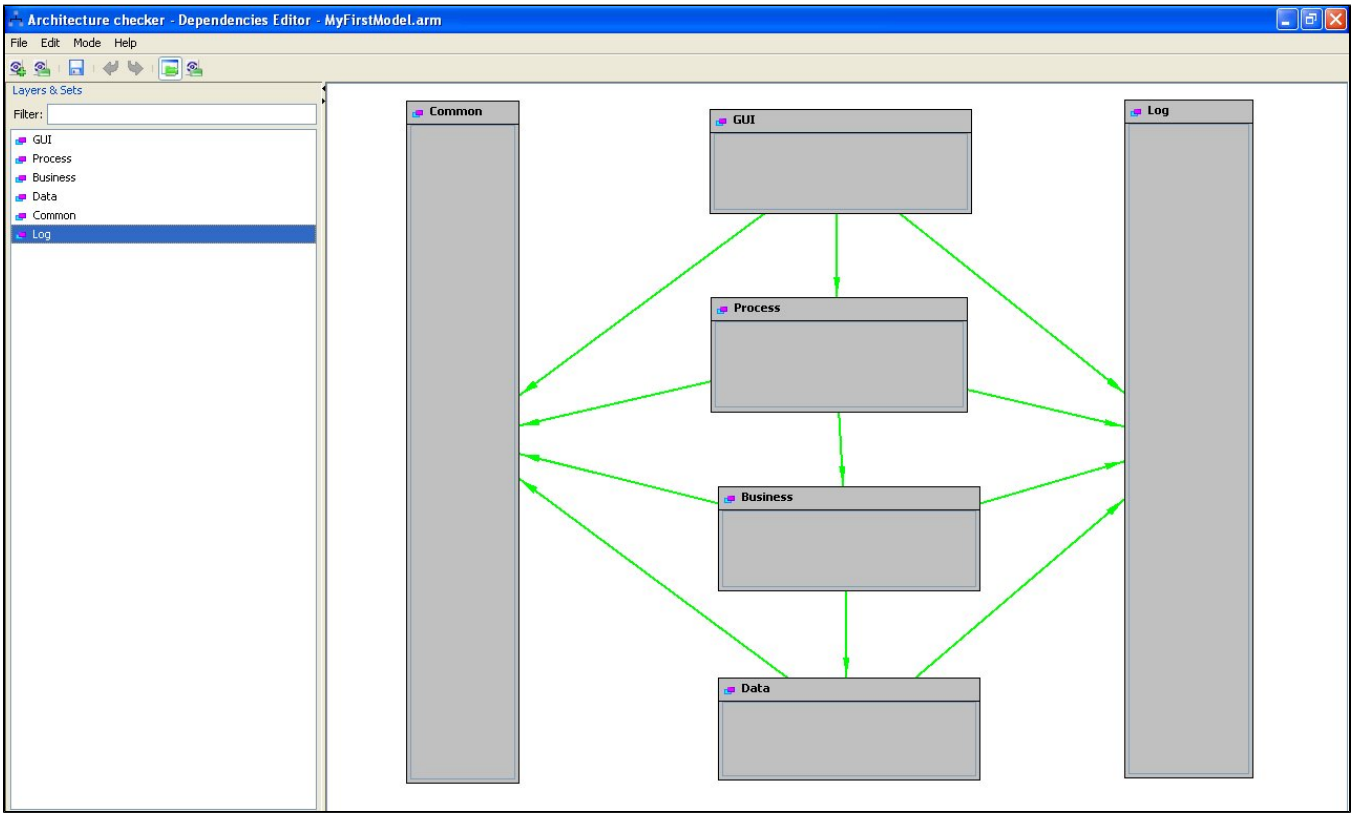
Creating a Model with the CAST Architecture Checker UI

First you need to create the architecture model to be checked according to your target application's design or according to the facts you want to check. The creation of an architecture model is done with the dedicated graphical user interface **(CAST-ArchiChecker.exe**).

An architecture model describes the layers of the application and defines the way the layers are filled with objects analyzed by CAST. Depending on your requirement, you can build two types of model:

- a model defining authorized dependencies between layers
- a model defining forbidden dependencies.

While defining the model, the user can check interactively the content of the layers and the compliance of an application by connecting to a CAST Analysis Service. Once saved, the model can be used within the CAST Management Studio to activate the automatic check for violation of the architecture and design. The CAST Architecture Checker helps you generate, without programming, the files that will be used in the CAST Management Studio to check the architecture of your application.



Interactively check your model and the Application's compliance

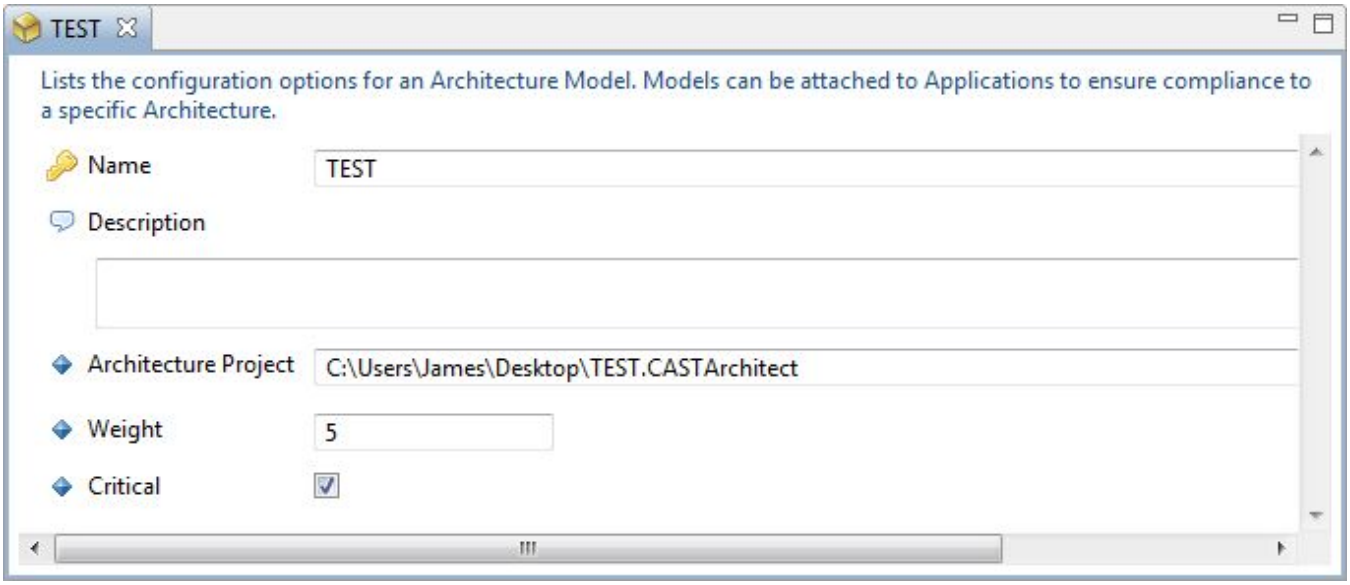
The CAST Architecture Checker enables the user to validate, tune their models interactively during their creation and before using the model in production, thanks to an on-line mode that connects to the CAST Analysis Service.

Activate the Automatic Architecture Check of your Application with CAST Management Studio

> Note that you can include more than one Architecture Model in one Application - all Architecture Models that are included will be checked when the snapshot is generated.

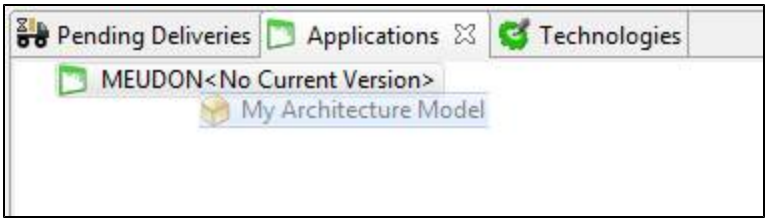
To force the CAST Architecture Checker to automatically check the defined model, you need to use the CAST Management Studio:

In the Architecture Model view in the CAST Management Studio, add a model with an appropriate name and then select the **.CASTarchitect** file created with the CAST Architecture Checker as shown below:

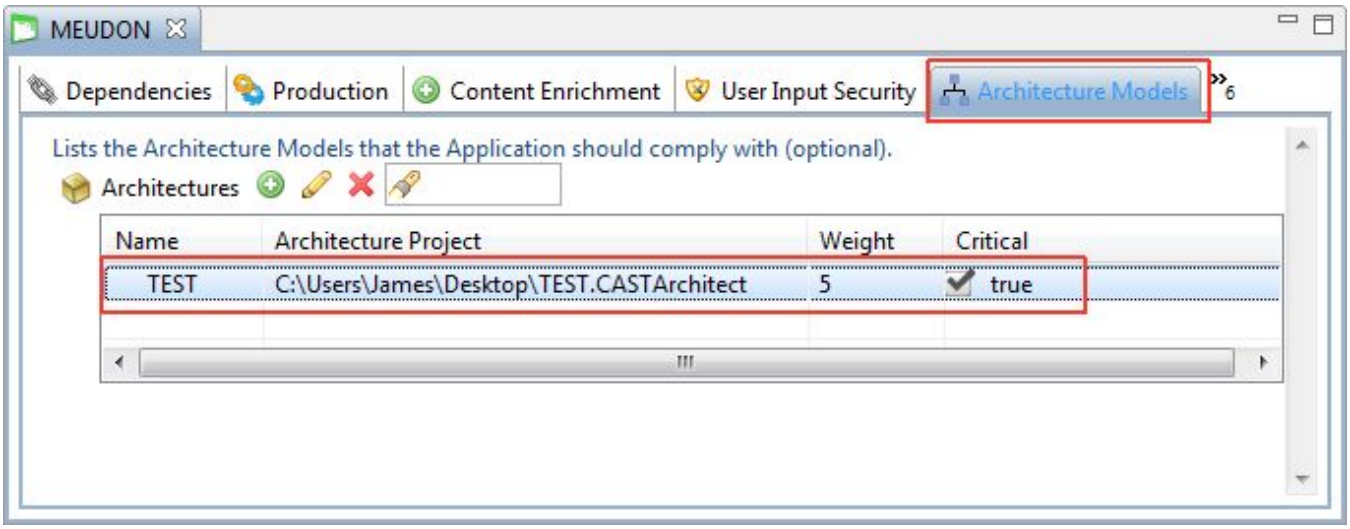


To include the Architecture Model in the Application (i.e. to activate an architecture check during the Snapshot Generation process), you can either:

- drag and drop the model from the Architecture Model view on to the Application in the Application View

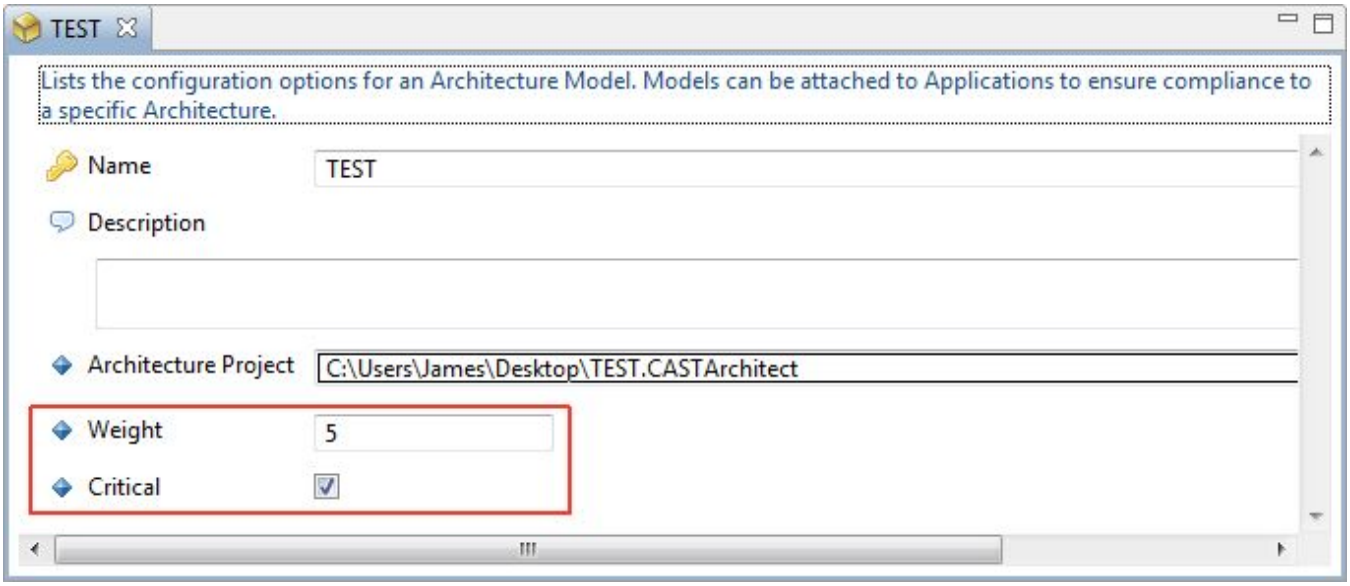


- or use the Architecture Model tab of the application in question to activate the check of your model:



Define the Weight and Criticality of the Architecture Model

Each Architecture Model that you assign to an Application will cause the creation of one corresponding Quality Rule (it will be be displayed with the name of the Architecture Model). You can define the Weight and Criticality of this Quality Rule (within the parent Technical Criterion) using the Architecture Model editor:

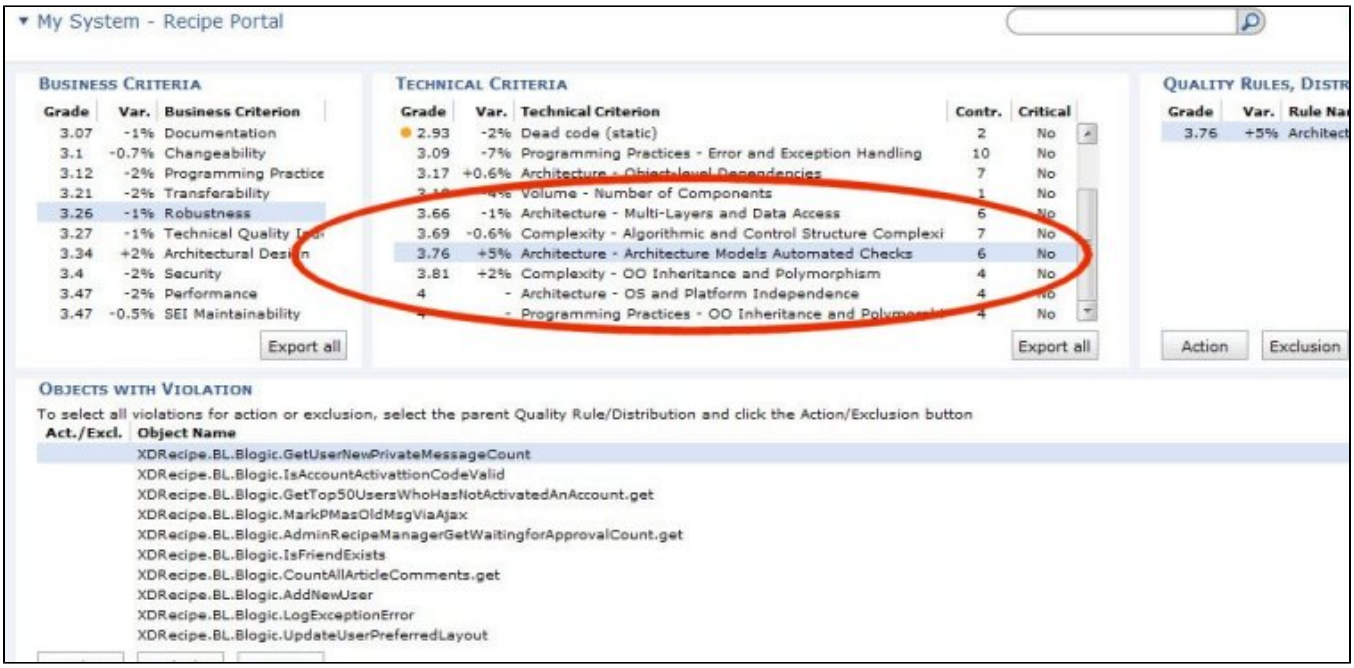


Generate the snapshot to view results in the CAST Engineering Dashboard

Once the snapshot has been generated, you can visualize the results directly under the Technical Criterion called **Architecture - Architecture Models Automated Checks** in the CAST Engineering Dashboard which contributes to the following Business Criterion:

- Robustness
- Security
- Changeability

One Architecture Model generates one Quality Rule.



Back to: 2.2.2. Validate and fine tune the Analysis