­­

Operation Orchestration Authored Code Review Utility

Author:

*Suvendu Chatterjee*

*[suvendu.chatterjee@microfocus.com]*

Contributors:

*Ankit Kumar Pandey*

*[ankit.pandey@microfocus.com]*



Contents

[Abstract 3](#_Toc62720926)

[Problem Statement 3](#_Toc62720927)

[Prior Solutions 3](#_Toc62720928)

[Our Solution 3](#_Toc62720929)

[Inference 5](#_Toc62720930)

[Evidence That Solution Works 5](#_Toc62720931)

[Benefits 6](#_Toc62720932)

[KPI 7](#_Toc62720933)

[References / External Disclosures 7](#_Toc62720934)

# Abstract

Organizations using Operation Orchestration to author code for their organization spend lot of time in reviewing code. A dedicated code reviewer or peer review is required to perform review. Cost of quality which compromise of costs related to prevention and detection of [defects](https://softwaretestingfundamentals.com/defect/) and the costs due to occurrences of defects is high.

To overcome this dependency on reviewer, reduce cost of quality and faster code delivery, automated OO code review utility is built. OO code review utility checks source code for compliance with a predefined set of rules and best practices. This is a web based utility that provides assistance to developer for reviewing their own code. Developer can upload code using Web UI provided by this utility, has option to select rules and to create exceptions in review. It reports list of warnings and error, can be downloaded and shared across.

# Problem Statement

Peer code review, a manual inspection of source code by developers/reviewers other than the author, is recognized as a valuable tool for improving the quality of code. This manual process is cumbersome and time-consuming. For small change, sometime it is required to review overall functioning of code.

Code authored using Operation Orchestration contains large number of steps compromising of several operations. Reviewing code manually against prescribed coding standard takes more than anticipated time due to ‘to and fro’ communication between the reviewer and developer. This leads to delay in overall delivery timelines. Even after doing multiple manual reviews, rule or coding standard checks might get missed which leading to bug/defect.

# Prior Solutions

After thorough research within Micro Focus or elsewhere, no precedence was found where any automated utility had been used for Operation Orchestration authored code review. It was limited to manual review either by dedicated reviewer or peer review.

# Our Solution

Our solution checks OO code against pre-defined violation matrix (consists of set of rules defined in configuration file). Utility is built by using python and utilizes flask framework for web hosting. OO content pack is uploaded as input to utility which contains code and their dependencies. Utility will parse the given content pack by identifying and extracting the Flows and Configuration items as objects. These objects are validated against defined rules and coding standards. After validation, a detailed report is generated, which contains dashboard and details of error and warning. Report can be downloaded as PDF or xls format.

Below is the high level process diagram of the solution.

**Figure 1: High Level Process Diagram**

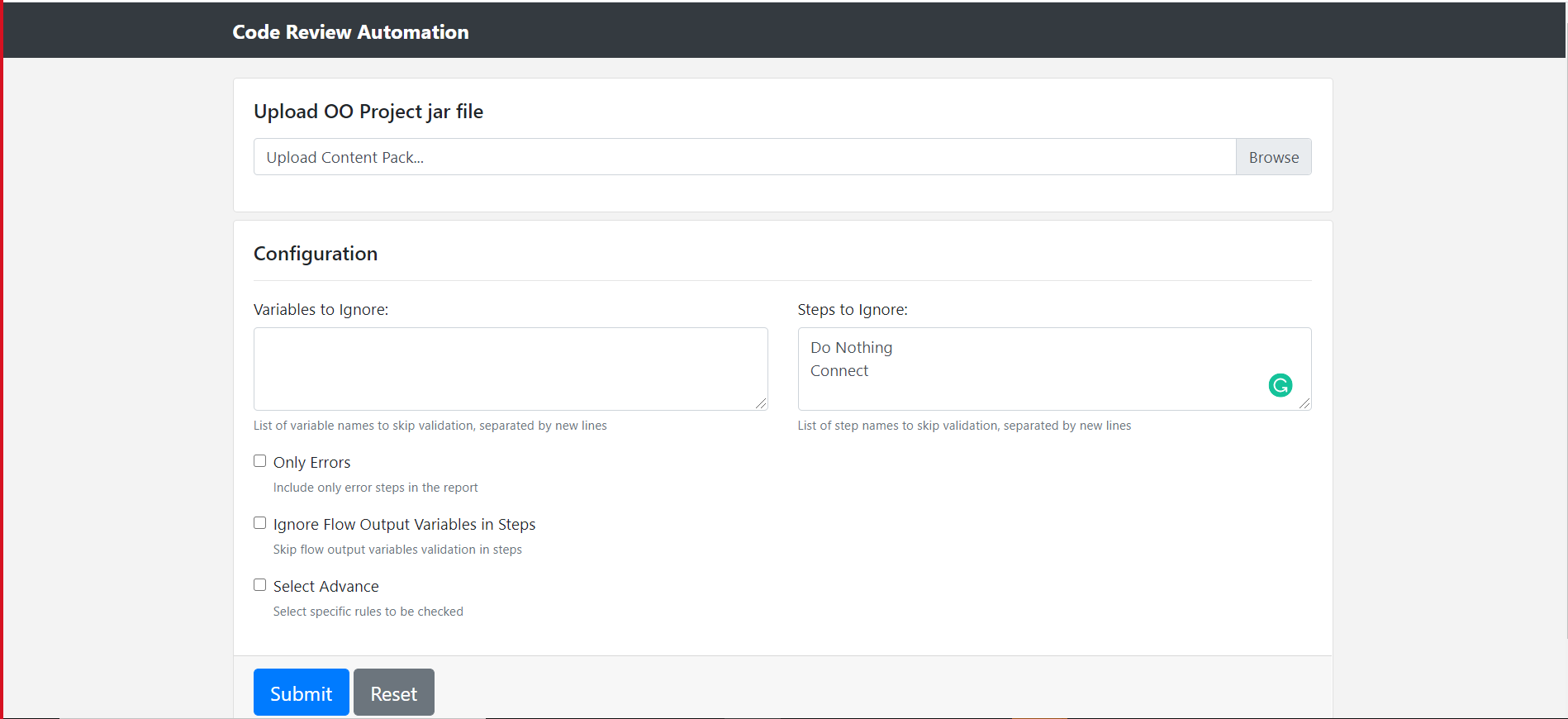
**Utility Interface:**

Interface consists of three section as below: (Refer to Figure 2)

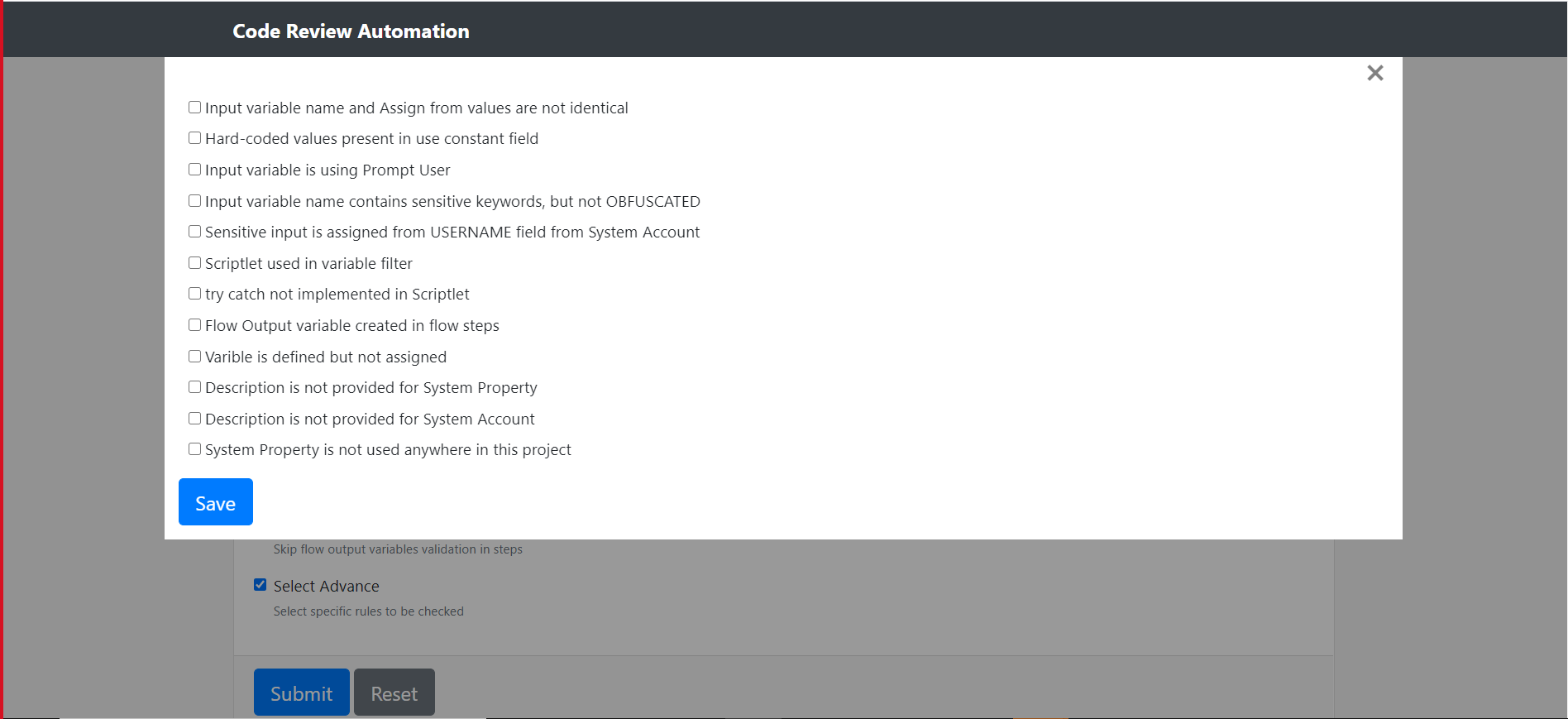
* **BROWSE:** Section to select and upload content pack
* **Configuration Section**: Different options are as below:

1. **Variables to Ignore:** List of variable to ignore
2. **Steps to Ignore:** Select list of steps to ignore
3. **Only Errors**: Reports only errors
4. **Ignore Flow Output Variables in Steps:** Skips output variable validation as selected under ‘**Steps to Ignore’**
5. **Select Advance:** Provides option to users to validate code against selected violation matrix only. Refer to figure 3.

* **Submit and Reset:** Buttons to submit for processing or resetting the form

****

**Figure 2: Utility Interface**



**Figure 3: Rule Exception List in Select Advance**

**List of Violation Matrix implemented:**

* Input variable name and Assign from values are not identical
* Hard-coded values present in use constant field
* Input variable is using Prompt User
* Input variable name contains sensitive keywords, but not OBFUSCATED
* Sensitive input is assigned from USERNAME field from System Account
* Scriptlet used in variable filter
* Try catch not implemented in Scriptlet
* Flow Output variable created in flow steps
* Variable is defined but not assigned
* Description is not provided for System Property
* Description is not provided for System Account
* System Property is not used anywhere in this project

**Technology Stack used in Solution:**

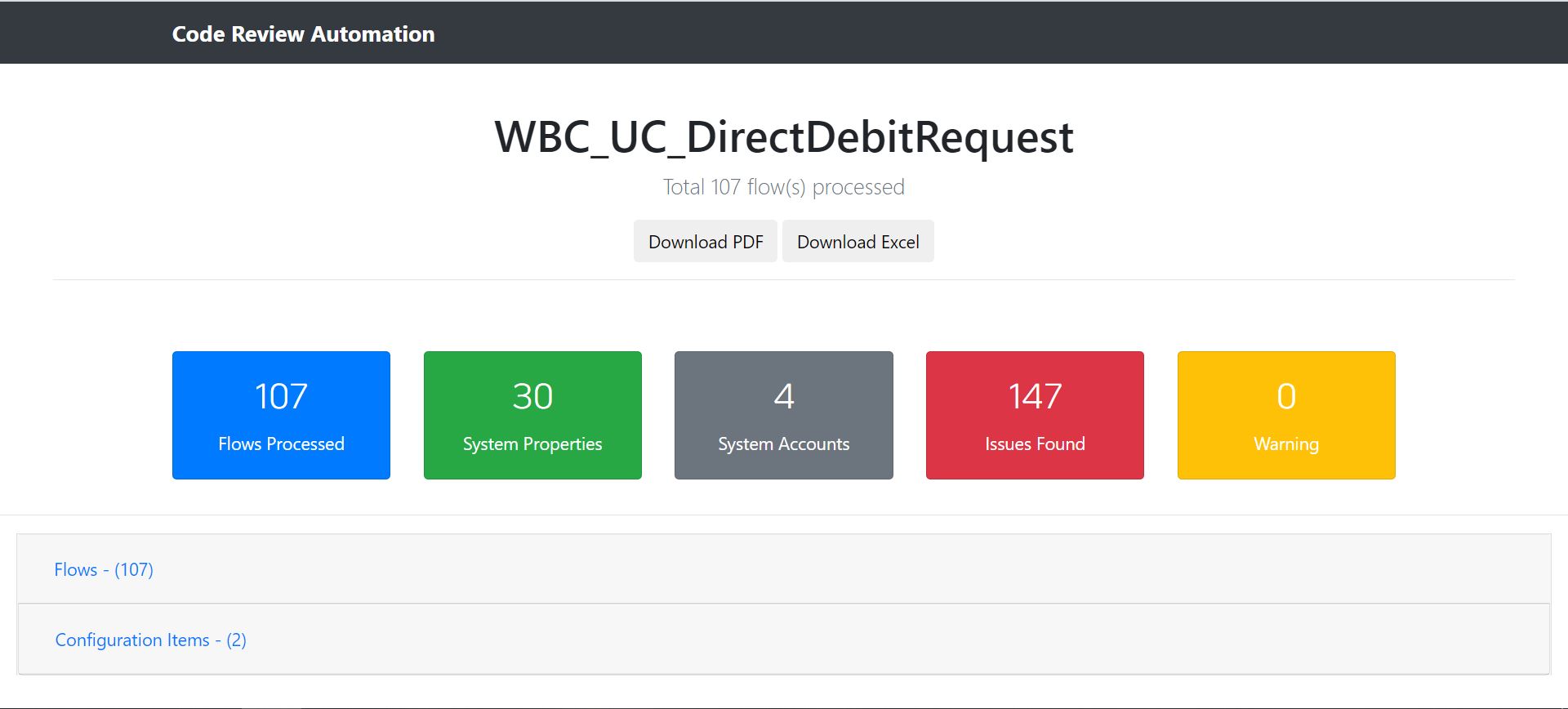
* **Python Flask**: Lightweight web application server.
* **Jinja2:** HTML Templating engine to generate interactive HTML pages to interact with web application and to generate reports.
* **Python ElementTree Parser:** XML Parser.
* **Jsbeautifier :** To format the JavaScript’s used flows
* **Bootstrap4:** CSS framework for rendering the web pages.
* **Highlight.js:** JavaScript framework to highlight JS code.
* **ZipFile:** Python module for processing jar file.

# Inference

Code review automation utility will results in much quicker identification of errors/defects, lapses in coding standards by developer itself thus reducing Cost of Quality, Defect Density, Overall delivery timeline and Compliance enforcement against coding standards which today’s Organization strive for.

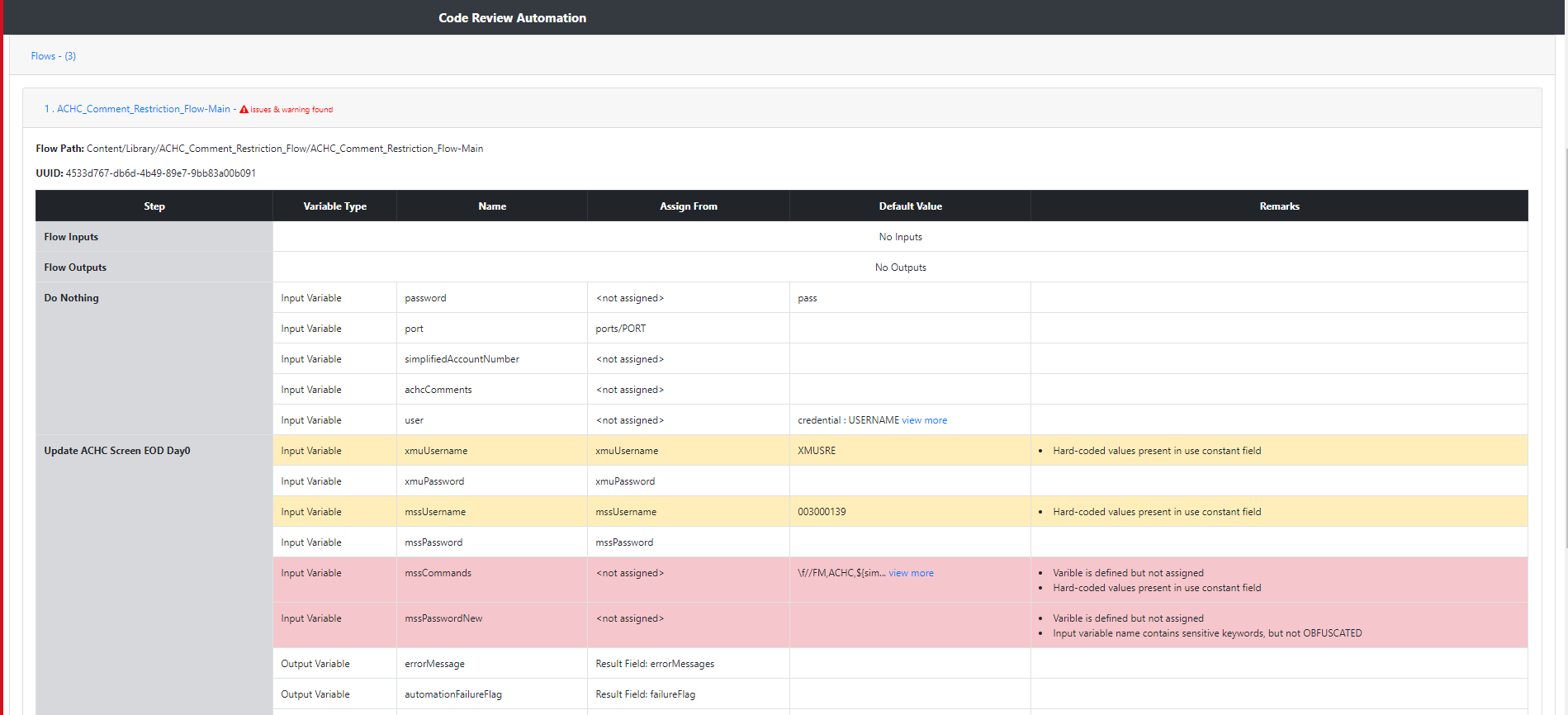
# Evidence That Solution Works

The utility has been implemented for one of Banking Corporation. Time taken to review code is drastically reduced by using this utility. Developers are extensively using this utility to enforce defined rules and coding standards. Refer figure 4 for dashboard and figure 5 for generated report.



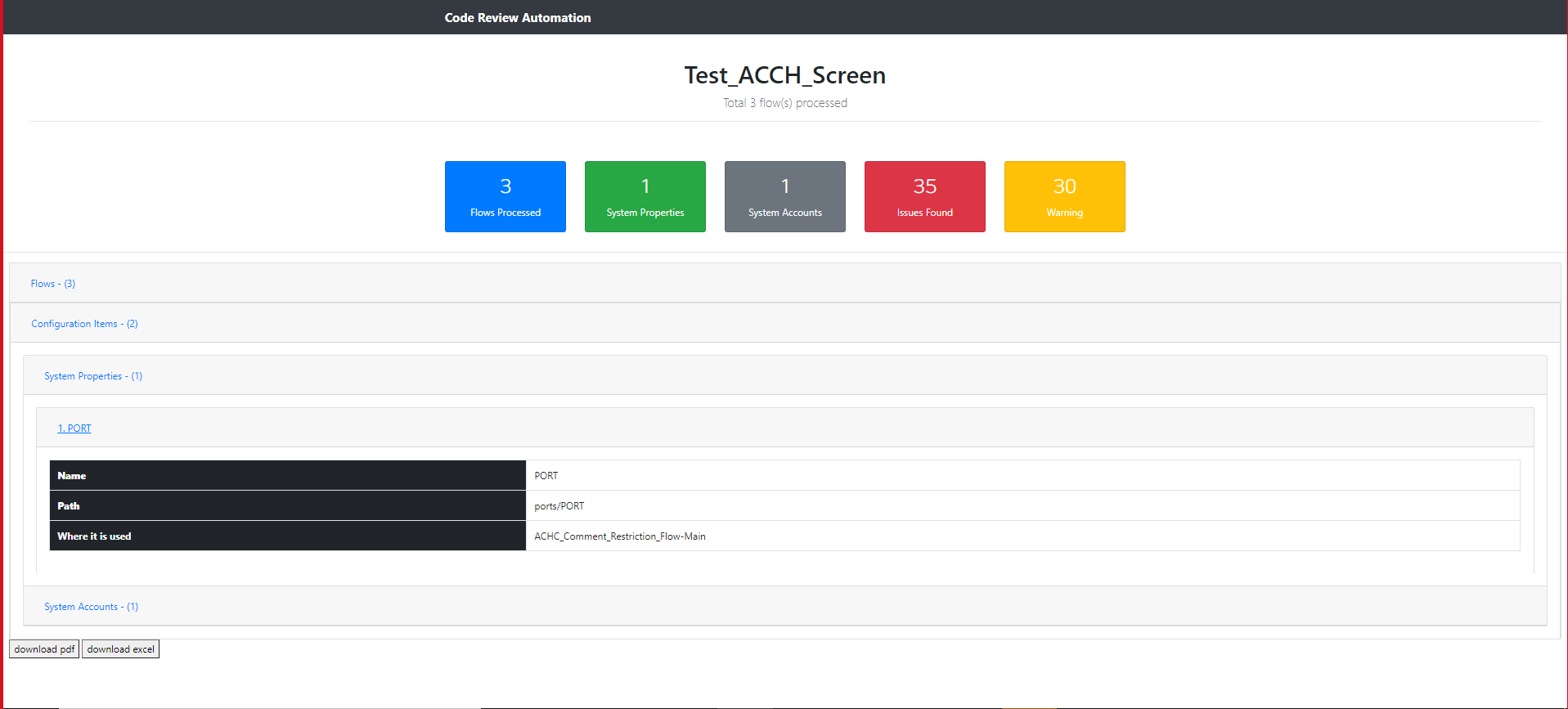
**Figure 4: Result Dashboard**

**Dashboard:** Dashboard shows count of flow processed , configuration item , error and warnings.



**Figure 5: Generated Report**

**Generated Report:** This report contains details of each step along with violation under remarks column. Color coding is used to differentiate between error and warnings.



**Figure 6: Configuration Item details**

**Configuration Items:** List of configuration item defined in code along with path, list of flows where it is used and any associated violation.

Note:- This is the link to access the utility through web browser [OO Code Review Automation](http://btp-hvm01749.swinfra.net/) .

# Benefits

Following are the benefits by using this Utility:

* Reduced cost of Quality
* Efficiency
* Code compliance
* Code quality standardization
* Self-evaluation
* Better Accuracy and Productivity
* Team management and reporting
* Removing potential human error

# KPI

Below are the key performance indicator from one of our existing project where utility is being used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Number of Modules | No. of Defects | Defect Density | Invested Hours |
| Before Code Review Automation | 100 | 54 | 0.54 | 324 |
| After Code Review Automation | 100 | 12 | 0.12 | 72 |

**Hours Saved**: 252

**% Hours Saved**: 78%

\*Single module contains on an average 20 OO operations

\*Estimated hours required to fix and test a defect is 6 Hrs.

# References / External Disclosures

[Need to think and include]