Verification Plan in Requirements Management Tool: Simple Traceability and Automated Interface to Regression Manager

Jan Kreisinger
Allegro Microsystems, Prague, Czechia
jkreisinger@allegromicro.com

Sanjay Chatterjee
Allegro Microsystems, Manchester, NH, USA
schatterjee@allegromicro.com

***Abstract*- Maintaining continual consistency between the design requirements, verification plan, and regression is crucial in achieving timely verification closure and high confidence in the verification results. To accomplish this, requirements traceability should be incorporated into the verification workflow and the interface between different tools across the verification process must be well defined. This paper introduces a requirements tracing workflow between a requirements management tool and a regression manager. Developing the verification plan in the same tool as the design requirements facilitates their mapping onto the verification plan elements. Benefiting from automation, this workflow provides a straightforward way of maintaining consistency between the design and verification plan throughout multiple iterations of the verification process. Moreover, different groups outside the digital team can review the requirements, verification plan, and verification progress status from a single tool.**

***Keywords*- verification, requirements traceability, Jama REST API, workflow automation**

I. INTRODUCTION

Well-defined design requirements tracing workflow throughout the verification process greatly impacts the verification task completeness and on-time project execution. Requirements shall be traceable through the verification plan to regression results, which shall be demonstrable and visible to all members of the development team. Requirements traceability is especially urgent due to the increasing complexity of the designed systems and increasing demands on quality management. Incomplete or inconsistent design requirements mapping onto the verification plan leads to a project schedule slip, indemonstrable verification results, and missed bugs. Verification plan development, maintenance, and requirements tracing throughout the verification process can take a substantial part of the time allocated to the whole verification task, which is another reason for making the effort to develop an efficient and reliable workflow.

The choice of tools, which rarely depends exclusively on the verification team, has a major impact on the implementation of an efficient workflow. A different tool may be used for design requirements control (e.g., a spreadsheet or a specialized Requirements Management (RM) tool) and for the verification plan development. The design requirements are then mapped onto the verification plan using a hypertext link, unique name, or code. This approach, however, brings considerable overhead when the design requirements are changed, added, deleted, or their identification is altered. Nonetheless, keeping the design requirements and verification plan within one tool can facilitate the mapping process, and help to maintain a good overview of the changes and their impact. It is crucial to preserve the consistency of the verification plan with the design requirements from the initial planning until presenting the final verification results and metrics. Consistency is achieved by automating the transition between different steps of the verification process using scripts based on Application Programming Interface (API) of used tools.

Mapping dozens of design requirements onto dozens of verification plan elements is often time-consuming, tedious, and repetitive work, which is prone to mistakes caused by inattention. Standardization and automation of this workflow are key approaches to confront this problem. Additionally, standardization and automation are effective means to ensure the comprehensibility and convenience of the verification process for the whole verification team. Another demand on the verification workflow is to record and present the regression progress throughout the system development.

In the context of Allegro Microsystems, Jama Connect by Jama Software [1] is used as an RM tool and Cadence Verisium Manager [2] verification platform is used to run and analyze regression. The need to reliably trace design requirements throughout the verification process is acknowledged by the commercial tool providers Cadence [3] and OpsHub [4]. Both solutions require a Verisium Manager full client license to create and edit the verification plan. For this reason, their scalability is limited as these solutions would require a large pool of Verisium Manager full client licenses for a large verification team working on many projects at the same time. Verification workflow solutions with requirements tracing were presented in [5] and [6]. Although their ideas are generally adoptable, the solution cannot be reused due to a different set of tools.

In this paper, we present a verification workflow with design requirements tracing built around the Jama Connect RM tool, and Verisium Manager. This workflow enables traceability analysis of the verification process in Jama Connect by connecting design requirements with the verification results through the verification plan. We use four key ideas to optimize the reliability and ease of use of the workflow. First, the verification plan is managed in the RM tool, where it is directly mapped onto the design requirements, which leads to immediate visibility of the changes in the design requirements and reduces the usage of Verisium Manager full client licenses. Second, two Python scripts based on Jama REST (Representational State Transfer) API [7] create the interface between the RM and regression manager tool, and minimize the manual work. Third, usage of Verisium Manager batch mode allows even further automation of the regression report generation. Fourth, uploading the regression results back to the RM tool allows a transparent demonstration of the verification progress. The presented workflow is applicable as is to the same set of tools, however, the scripts themselves were developed for a custom version of the relationship diagram and the RM tool items.

The main contributions of the presented workflow are:

- Maintenance effort reduction thanks to keeping design definition and verification plan in a single database,
- High level of automation thanks to Jama REST API and Verisium Manager batch mode,
- Benefiting from all Jama Connect features like suspect links, review center, or filters,
- Enhanced collaboration within the development team,
- Limiting the usage of Verisium Manager full client licenses.

II. Related work and existing solutions

The traceability of design requirements has been studied for many years [8], but more recent work [5] and [6] shows it is still considered an insufficiently resolved challenge for complex designs. Many papers ([5], [6], [8], and [9]) emphasize the importance of using requirements tracing to drive the verification effort.

A verification workflow with requirements tracing is suggested in [5] and [6]. Both presented workflows use automation based on API and scripting in the loop between a RM tool, verification development, and regression. The authors of [5] and [6] identified similar problems with non-automized workflows such as long execution cycles, proneness to human errors, and complications with detecting and implementing changes in the design requirements. Reference [5] includes requirements traceability in the code as well. Both references [5] and [6] use verification code generators in the loop and the workflows are based on the RM tool Polarion by Siemens [10], where the verification plan is maintained. The workflow presented in this paper, albeit developed independently, applies similar ideas and approaches to the problem while allowing traceability of higher system level requirements. However, specific demands are implied by the tools used within the workflow discussed in this paper. Employing Verisium Manager batch commands further limits the number of manual steps in the workflow, which leads to easier collaboration between multiple verification engineers and lower usage of the Verisium Manager full client licenses.

As previously mentioned, the Cadence verification platform, Verisium Manager, provides the feature *PDF Spec Annotation* to import design requirements and map them onto the verification plan elements [3]. However, this flow would not solve the issue of unavailable full client licenses when many verification engineers work on a verification plan at the same time. Furthermore, it is beneficial to develop the verification plan in the RM tool, where it can be reviewed by all members of the development team and any change in the design requirements is immediately visible as a suspect relationship to the verification plan. Another feature not covered by this flow is that, besides digital specification, system requirements should be traceable to the regression results to fulfill part of the functional safety standards such as ISO 26262. For mixed signal sensor chips with system level complexities, it becomes increasingly necessary to ensure all system level requirements are verified and traced back to the verification results.

A commercial solution for Jama Connect and Verisium Manager integration is offered by OpsHub [4]. This tool creates a direct connection to Jama Connect by importing the design requirements from Jama Connect into Verisium Manager and mapping them onto the regression results. This tool requires a Verisium Manager full client license both for the verification plan development and design requirements mapping. Moreover, the license for OpsHub solution would have to be acquired.

III. Requirements tracing workflow using Jama REST API

The presented workflow uses two scripts based on Jama REST API: *vpgen_japi* and *vmgr_jama_updater*. They create the interface between Jama Connect, where the verification plan is maintained and regression results are stored, and Verisium Manager used to run and analyze the regression. The execution of the whole workflow, depicted in Fig. 1, is described in the following subsections.

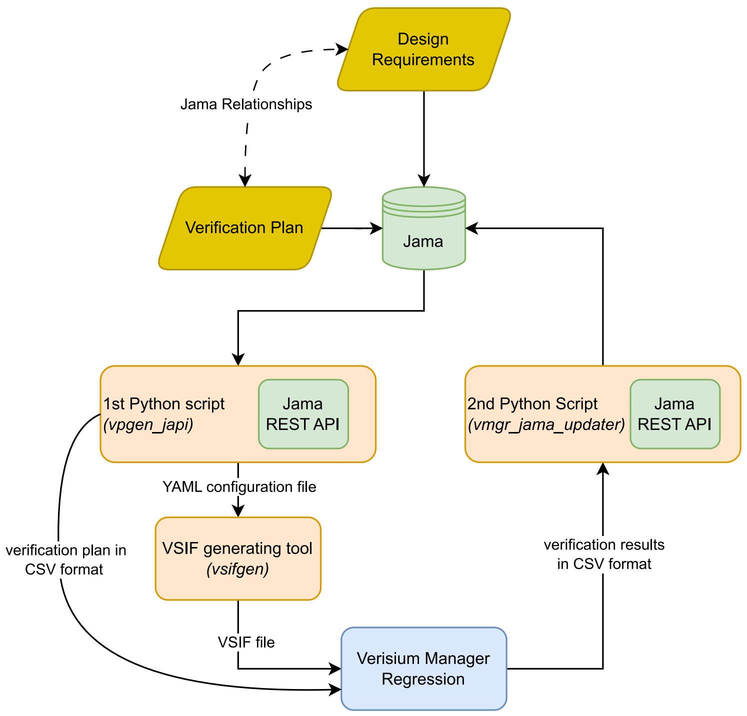

Design Requirements
Jama Relationships
Verification Plan
Jama
1st Python script (vpgen_japi)
Jama REST API
2nd Python Script (vmgr_jama_updater)
Jama REST API
YAML configuration file
VSIF generating tool (vsifgen)
verification plan in CSV format
verification results in CSV format
VSIF file
Verisium Manager Regression

Figure 1. Verification workflow using Jama API

A. Jama Connect

In Allegro Microsystems, Jama Connect is used to collect and maintain design requirements and other project-related information. Jama Connect allows a high level of customization of the project items, their purpose, and fields. Different meanings of the relationships between the items can be defined. To leverage these features, a custom relationship rules diagram was developed to accurately describe relationships between different types of items. The portion of the relationship rules diagram that is relevant to this workflow is shown in Fig. 2.

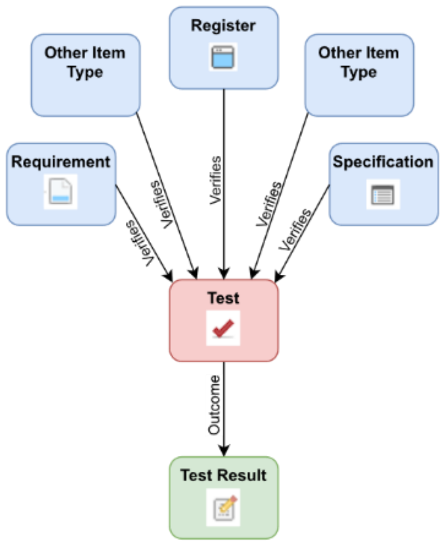
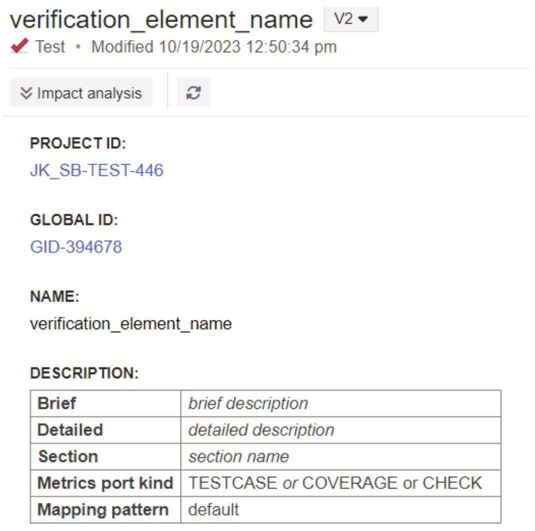


Figure 2. Test and Test Result items relationships

As mentioned before, the verification plan is developed in the RM tool to leverage its features and keep an overview of the changes in the design requirements. It consists of *Test* type items, which are universally used in all domains of verification, not only digital. Items in Jama Connect are organized under a specific item type called *Set*, which is a folder that can contain items of only one type. The *Test* item can have multiple upstream relationships of type *Verify* to different types of design requirements. On the other hand, the number of downstream relationships is limited to a single *Test Result* item type. The verification plan can be divided into multiple sections and each of its elements have a different purpose in the functional verification (a testcase, coverage, or check). Each element of the verification plan shall have a detailed description of how it verifies the upstream items. For its simplicity, the information about each of the verification plan elements is stored in a table in the format described in Fig. 3. The *Mapping pattern* row defines how the verification plan elements are mapped onto the regression results. To facilitate the initial development, the table is stored as a template and can be directly loaded to the *Description* field.


verification_element_name
V2
Test • Modified 10/19/2023 12:50:34 pm
Impact analysis

PROJECT ID:
JK_SB-TEST-446

GLOBAL ID:
GID-394678

NAME:
verification_element_name

DESCRIPTION:

Brief	*brief description*
Detailed	*detailed description*
Section	*section name*
Metrics port kind	TESTCASE *or* COVERAGE or CHECK
Mapping pattern	default

Figure 3. Verification plan element format

In case any of the upstream items have been changed, all downstream relationships become suspect. The suspect attribute of the relationship is reported in the side toolbar or trace view. This serves as a notification to the verification engineer that the related verification plan elements are out of date and the impact of the changes must be analyzed. The engineer then must manually clear the suspect relationships. The same applies to the relationships to *Test Result* items, in which case the suspect attribute means that the regression may be out of date. Having a verification plan clear of suspect relationships is a necessary condition for verification closure.

B. Jama REST API and the scripts organization

Jama Connect provides an API to access the server database and manipulate the stored data including projects, items, or relationships. The API follows the constraints defined in [11] to comply with the REST architectural style. It communicates with the database using following HTTP requests:

- GET to retrieve a record from the server,
- POST to create a new record on the server,
- PUT to update an existing record on the server,
- PATCH to update specified fields of a record,
- DELETE to delete a record from the server.

The API calls are authenticated using OAuth [7] and the data are transferred in the JSON format (see Fig. 4). An open-source Python client is available on GitHub [12] for convenient access to the Jama REST API. This client implements methods to facilitate the use of HTTP requests for different types of records in the Jama Connect database, and to simplify authentication of the API user. Other RM tools provide a same-style API (e.g., Siemens Polarion [10], or Valispace [13]), while others may provide different way to extends the tools functionality (IBM DOORS [14] uses Open Services for Lifecycle Collaboration).

Fig. 4 shows an example of an item received using Jama REST API with some fields omitted for better intelligibility. The item can be identified by a unique *documentKey* or *id* code. *DocumentKey* also provides the information about the name of the project. Other important parts of the data structure are: *itemType, fields, parent,* and *type*. *ItemType* is a unique code for each type of an item (e.g., *Test* or *Test Result*). *Fields* entry contains *name* and *description* subfields, which hold important information about the item. *Parent* field is the *id* number of the preceding item in the hierarchy, which can be a folder, set, or another item. *Type* specifies the type of the record, which can be an item, relationship, or project. The matching fields can be identified by comparing the Fig. 3 and Fig. 4. From the perspective of the presented workflow, it is mainly important to be able to use GET request to download the items and relationships, POST request to create new *vsif_cfg* item (see subsection *C.*) and *Test Result* items (see subsection *D.*) PATCH or PUT requests are used to update the items which were already created in the previous runs of the workflow. DELETE request is important to keep consistency of the automatically created items over multiple runs, however it is not crucial for the workflow execution.

```
{
        'id': 123456,
        'documentKey': 'JK_SB-TEST-446',
        'itemType': 1,
        'fields': {
                'documentKey': 'JK_SB-TEST-446',
                'globalId': 'GID-394678',
                'name': 'verification_element_name',
                'description': '<table>…</table>\n',},
        'resources': {
                'self': {
                        'allowed': ['GET', 'PUT', 'PATCH', 'DELETE']}},
        'location': {
                'parent': {
                        'item': 123455}},
        'type': 'items'
}
```

Figure 4. Test item received using REST API (includes only relevant fields)

Both scripts supporting this workflow use a common Python controller class, which includes the Python client and extends its functionality to support the custom Jama relationship rules diagram. This class implements custom methods required by this workflow (e.g., consistency check, creating or updating specific items and relationships), while the *vpgen_japi* and *vmgr_jama_updater* scripts manage the input arguments, and save and organize the output files. Both scripts are executable from the command line and require several input arguments which will be described in the following sections.

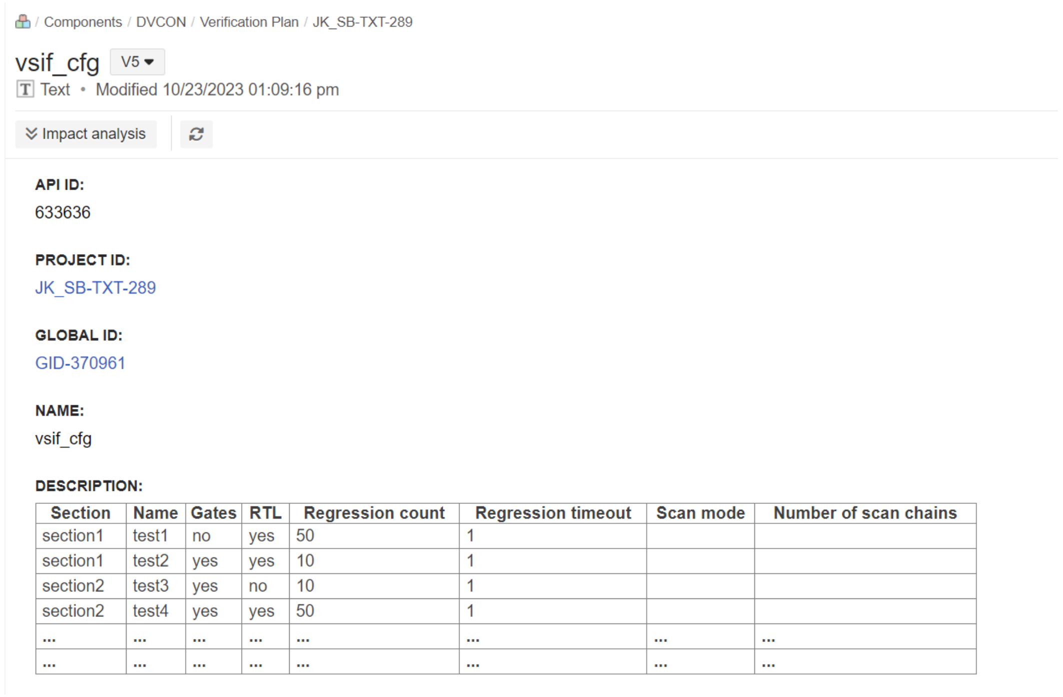
C. From Jama Connect to Verisium Manager

When the verification plan is defined using the *Test* items, the VSIF (Verification Session Input Format) file to run Verisium Manager regression and verification plan in CSV format to map it onto the finished regression must be generated. First, the *vpgen_japi* script downloads the information from the verification plan stored in the RM tool. Second, a Python tool called *vsifgen* is used for the fine-grained configuration of the VSIF files. This internally developed tool has been part of the verification process in Allegro Microsystems for many years.

The left side of the workflow diagram in Fig. 1 is managed by *vpgen_japi* script. It saves the verification plan in a CSV file, which can be later loaded to Verisium Manager and mapped onto a regression. Moreover, it creates YAML configuration files for different types of simulation (RTL, post-synthesis, and post-PNR), which are later used as the initial setup of the *vsifgen* tool. To generate the YAML files, there should be an additional item within the verification plan in the RM tool. In the case of this workflow, the item has a specific name (*vsif_cfg*) and type (*Text*), so it can be easily distinguished from the verification plan elements. Again, this item has a table in the *Description* field, where the coarse-grained configuration of the regression is stored, as shown in Fig. 5. The script has the option to create this item in case it is not found in the scope of the verification plan and fill in the table with default values. In the next iterations of the workflow, the content of the table is automatically updated by adding rows for testcases newly added to the verification plan or deleting rows with testcases that are no longer used.

The simplified execution flow of the *vpgen_japi* script is described in Fig. 6. The input argument is the *documentKey* of the *Set of Tests* which is the top item of the verification plan. Access to the Jama Connect server can be acquired either by providing the username and password or using Jama API Credentials [7]. Once access is gained, the project specified by a script input argument is found and all its items are downloaded. The folder (i.e., *Set of Tests*) with the verification plan elements is found among them using the unique *documentKey*. Next, all verification plan elements are collected in a Python list.

At this moment, the script can take advantage of using the tables in the Description field of the verification elements. They are downloaded in HTML format so Pandas Python library [13] can be used to extract the information to a list of DataFrame objects, which is a two-dimensional primary Pandas library data structure. The information is reorganized to the formats required by Verisium Manager and *vsifgen*, and saved in CSV and YAML files. Afterward, VSIF files are generated by the *vsifgen* tool using the YAML files produced in the preceding step as the initial configuration. Using these files, a regression session can be launched in Verisium Manager.



Components / DVCON / Verification Plan / JK_SB-TXT-289

vsif_cfg V5

Text • Modified 10/23/2023 01:09:16 pm

API ID:
633636

PROJECT ID:
JK_SB-TXT-289

GLOBAL ID:
GID-370961

NAME:
vsif_cfg

DESCRIPTION:

Section	Name	Gates	RTL	Regression count	Regression timeout	Scan mode	Number of scan chains
section1	test1	no	yes	50	1		
section1	test2	yes	yes	10	1		
section2	test3	yes	no	10	1		
section2	test4	yes	yes	50	1		
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

Figure 5. Special item for coarse regression session settings

D. From Verisium Manager back to Jama Connect

To complete the requirements traceability task, a regression report is generated and uploaded back to Jama Connect in the next step. This step is managed by the second Python script, *vmgr_jama_updater*, which uses the same Jama REST API client controller class as *vpgen_japi*. The usage of Verisium Manager batch commands to generate the regression report increases the level of automation and facilitates the collaboration between multiple verification engineers by merging multiple regression sessions to generate a single report.

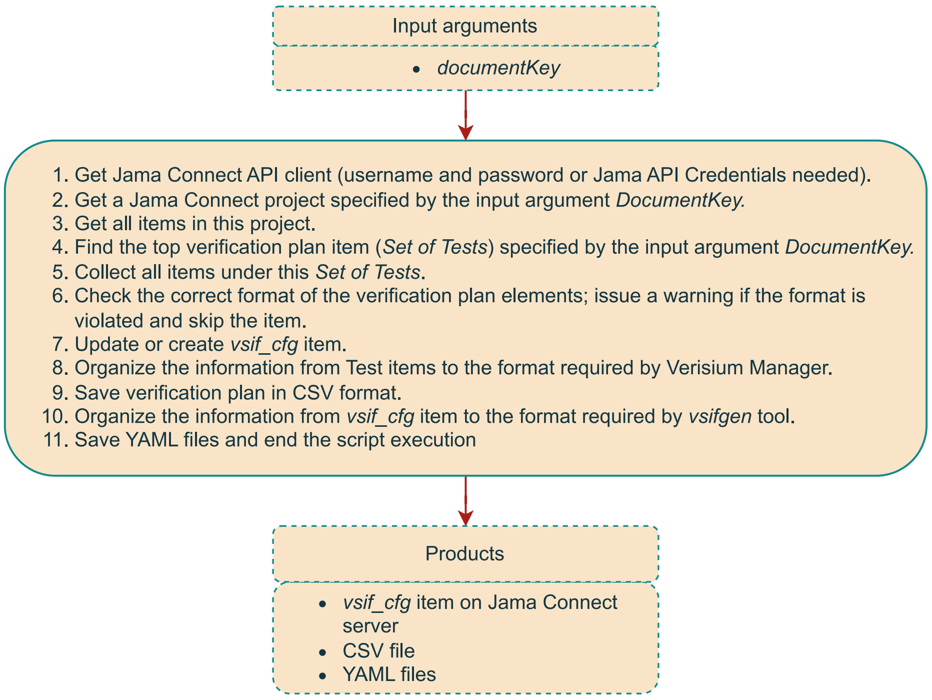
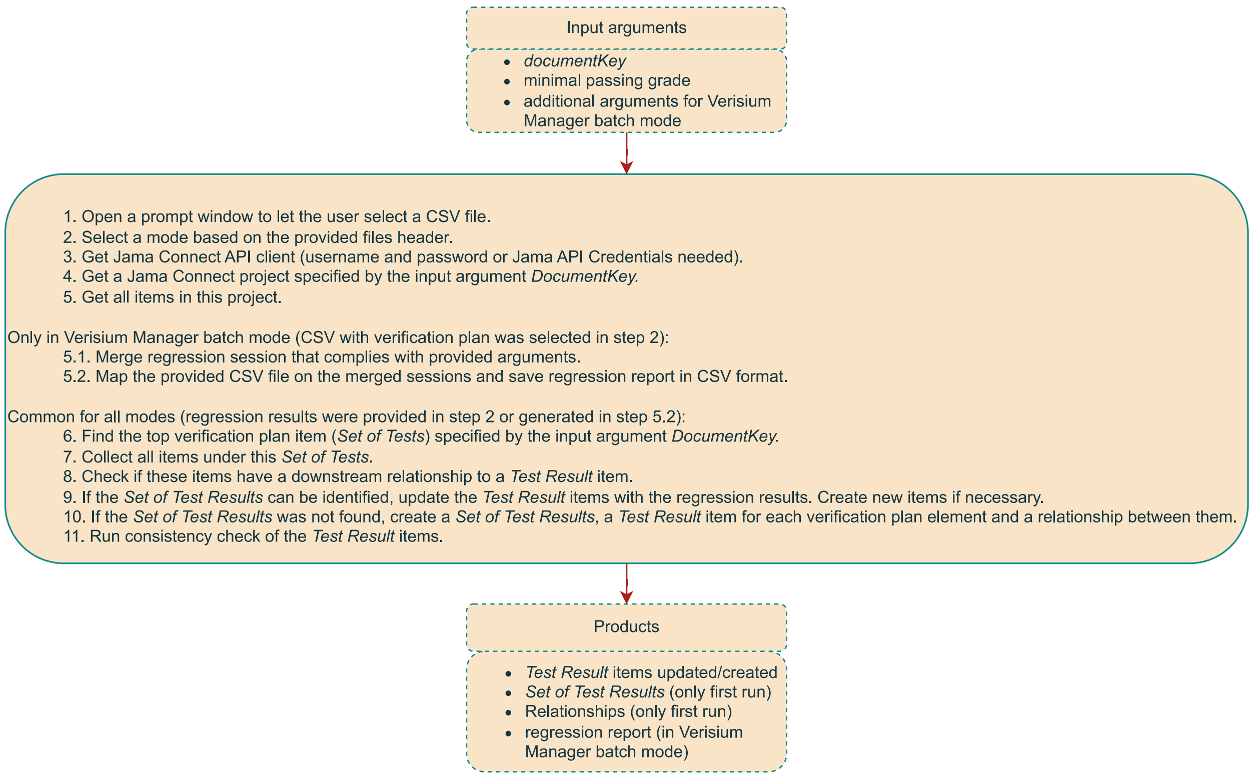

Input arguments
• documentKey
1. Get Jama Connect API client (username and password or Jama API Credentials needed).
2. Get a Jama Connect project specified by the input argument DocumentKey.
3. Get all items in this project.
4. Find the top verification plan item (Set of Tests) specified by the input argument DocumentKey.
5. Collect all items under this Set of Tests.
6. Check the correct format of the verification plan elements; issue a warning if the format is violated and skip the item.
7. Update or create vsif_cfg item.
8. Organize the information from Test items to the format required by Verisium Manager.
9. Save verification plan in CSV format.
10. Organize the information from vsif_cfg item to the format required by vsifgen tool.
11. Save YAML files and end the script execution
Products
• vsif_cfg item on Jama Connect server
• CSV file
• YAML files

Figure 6. vpgen_japi script execution flow

The execution of the script is described by Fig. 7. The script processes the regression results and uploads them to the Jama Connect server. In the end, each element of the verification plan should have a single *Test Result* item connected with an *Outcome* relationship as in Fig. 2. This item type has a *Test Result Status* field with drop-down list values *Pass*, *Fail*, and *Not run*. The last value is reserved for the regression results of the verification plan elements which were not implemented in the testbench yet. Furthermore, the type of verification element and the coverage grade it reached during the regression are reported in the *Description* field. Another demand on the script is that the workflow must be repeatable and the history from the previous workflow runs must be preserved. To fulfill this, the script first searches for the *Test Result* items that already have relationships to the verification plan elements. New items are created in case the *Test Result* items cannot be identified. Provided that this is not the first run of the workflow, the corresponding *Test Result* items are updated with the new values, and new items are created for *Test* items, which were not part of the previous runs of the workflow. A verification element passed the regression in case it reached 100% coverage grade. This default passing threshold can be changed by an optional argument while running the *vmgr_jama_updater* script.

The script can support multiple modes of operation and even more can be added if needed. The mode is selected based on the CSV file header, provided by a prompt window which opens when the script is executed. First, the script supports uploading manually generated regression report as a legacy mode. Second, it uses Verisium Manager batch commands to merge multiple regression runs from multiple verification engineers, map the verification plan onto them, generate the regression report, and upload it to the Jama Connect server. In this case, the verification plan in CSV format generated by the *vpgen_japi* script should be provided. Additionally, multiple input arguments are needed in this mode to specify which regression sessions should be used (e. g. the usernames, regression server, session name, or the time span between the first and last regression run to be merged). With the Verisium Manager batch mode, the usage of the full client license for verification plan editing is limited to only several tens of seconds. Furthermore, the verification engineer can avoid regularly checking the availability of the full client license as the script blocks the execution until it is available. The last mode is used to upload verification results from other tools (e.g., Simulink by MathWorks [16]) in CSV format with a distinguishable header.

Once all the *Test Result* items are updated or created, the script runs a consistency check to ensure that each *Test Result* item has exactly one upstream relationship to a *Test* item. In case this assumption is violated, a warning is issued, and the *Test Result* items breaching this condition are deleted.

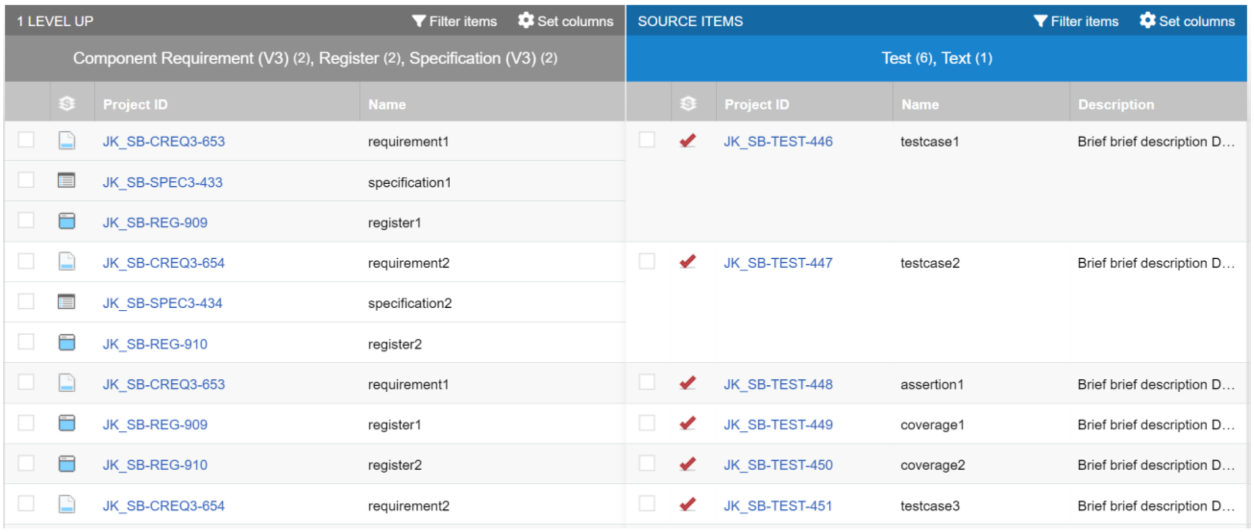


Input arguments
• *documentKey*
• minimal passing grade
• additional arguments for Verisium Manager batch mode
1. Open a prompt window to let the user select a CSV file.
2. Select a mode based on the provided files header.
3. Get Jama Connect API client (username and password or Jama API Credentials needed).
4. Get a Jama Connect project specified by the input argument *DocumentKey.*
5. Get all items in this project.
Only in Verisium Manager batch mode (CSV with verification plan was selected in step 2):
5.1. Merge regression session that complies with provided arguments.
5.2. Map the provided CSV file on the merged sessions and save regression report in CSV format.
Common for all modes (regression results were provided in step 2 or generated in step 5.2):
6. Find the top verification plan item (*Set of Tests*) specified by the input argument *DocumentKey.*
7. Collect all items under this *Set of Tests*.
8. Check if these items have a downstream relationship to a *Test Result* item.
9. If the *Set of Test Results* can be identified, update the *Test Result* items with the regression results. Create new items if necessary.
10. If the *Set of Test Results* was not found, create a *Set of Test Results*, a *Test Result* item for each verification plan element and a relationship between them.
11. Run consistency check of the *Test Result* items.
Products
• *Test Result* items updated/created
• *Set of Test Results* (only first run)
• Relationships (only first run)
• regression report (in Verisium Manager batch mode)

Figure 7. vmgr_jama_updater script execution flow

E. Verification plan traceability in Jama Connect

Finally, the traceability of the verification process can be reviewed in Jama Connect. Fig. 8 shows an example of the verification plan Trace view. Each verification plan element on the right side of the figure covers one or more design requirements (on the left side of the figure) connected by a *Verify* relationship.



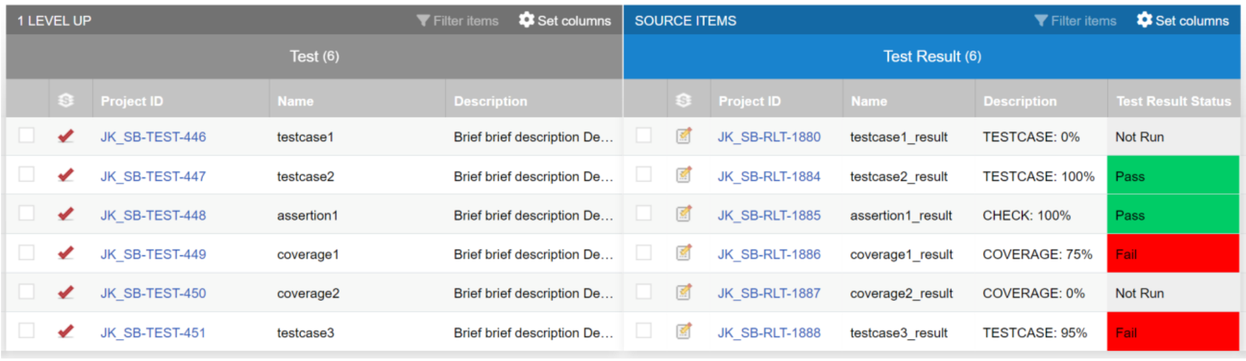
1 LEVEL UP — Filter items — Set columns | SOURCE ITEMS — Filter items — Set columns

Component Requirement (V3) (2), Register (2), Specification (V3) (2) | Test (6), Text (1)

Project ID	Name	Project ID	Name	Description
JK_SB-CREQ3-653	requirement1	JK_SB-TEST-446	testcase1	Brief brief description D…
JK_SB-SPEC3-433	specification1			
JK_SB-REG-909	register1			
JK_SB-CREQ3-654	requirement2	JK_SB-TEST-447	testcase2	Brief brief description D…
JK_SB-SPEC3-434	specification2			
JK_SB-REG-910	register2			
JK_SB-CREQ3-653	requirement1	JK_SB-TEST-448	assertion1	Brief brief description D…
JK_SB-REG-909	register1	JK_SB-TEST-449	coverage1	Brief brief description D…
JK_SB-REG-910	register2	JK_SB-TEST-450	coverage2	Brief brief description D…
JK_SB-CREQ3-654	requirement2	JK_SB-TEST-451	testcase3	Brief brief description D…

Figure 8. Trace view of the verification plan

Moving one level down in the Trace view, an example of a mid-project regression result can be seen in Fig. 9. In the left part of the figure is a list of *Test* items. Each *Test* has a corresponding *Test Result* item, which is listed in the same row on the right side of the figure. The items created by *vmgr_jama_updater* keep the same name as the corresponding *Test* items with appended "_result" string. The *Description* field shows the type of verification plan element and the grade it reached during the regression. The *Test Result Status* field is set to *Pass* if the verification element reached a higher regression grade than the minimal passing grade which was set via an input argument of the *vmgr_jama_updater* script. This field is set to *Not Run* in case the verification plan element was not implemented in the testbench. With the advantage of having a single tool for the design requirements, verification plan and storing the regression results, Jama Connect filters can be used to customize the project's dashboard for better overview of the project status. Fig. 10 presents two custom charts created using these filters. The chart on the left side of the figure shows the portion of implemented and passing *Test Result* items and the one on the right side shows how many design requirements are still not covered by an item in the verification plan.

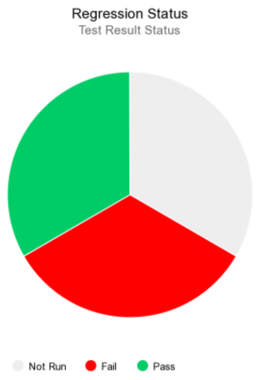


1 LEVEL UP — Filter items — Set columns | SOURCE ITEMS — Filter items — Set columns

Test (6) | Test Result (6)

Project ID	Name	Description	Project ID	Name	Description	Test Result Status
JK_SB-TEST-446	testcase1	Brief brief description De…	JK_SB-RLT-1880	testcase1_result	TESTCASE: 0%	Not Run
JK_SB-TEST-447	testcase2	Brief brief description De…	JK_SB-RLT-1884	testcase2_result	TESTCASE: 100%	Pass
JK_SB-TEST-448	assertion1	Brief brief description De…	JK_SB-RLT-1885	assertion1_result	CHECK: 100%	Pass
JK_SB-TEST-449	coverage1	Brief brief description De…	JK_SB-RLT-1886	coverage1_result	COVERAGE: 75%	Fail
JK_SB-TEST-450	coverage2	Brief brief description De…	JK_SB-RLT-1887	coverage2_result	COVERAGE: 0%	Not Run
JK_SB-TEST-451	testcase3	Brief brief description De…	JK_SB-RLT-1888	testcase3_result	TESTCASE: 95%	Fail

Figure 9. Trace view of Set of Test Results



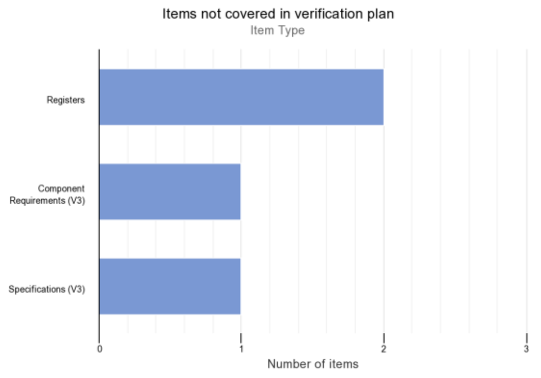


Figure 10. Example of project's dashboard charts

IV. DISCUSSION

At the time of writing, the workflow is used on a couple of projects in the development stage before tape-out. The presented scripts and workflow can be reused without any modification on each project that uses the same relationships and items definitions (i.e., Relationship Rules Diagram). It is only required to change the input arguments of the script according to the particular project and verification plan.

The presented workflow deals with the above-stated challenges and makes requirements tracing throughout the verification process efficient and easy to use. It allows maximal usage of Jama Connect features like version control of the verification plan, its reviews, and suspect links between the design requirements and the verification plan items in case of design changes. Keeping the design requirements and the verification plan in a single tool brings the benefit of immediately visible changes and simple mapping of design requirements onto the verification plan. Jama REST API-based scripts minimize manual effort and prevent human errors. As a result, the workflow is easily adoptable for the whole verification team. The usage of Verisium Manager batch commands facilitates cooperation between multiple verification engineers. Additionally, semiconductor developers can be less dependent on the licenses for developing the verification plan or on external solutions.

The presented ideas and approach are applicable throughout the whole verification community, not exclusively to digital verification teams. This workflow was specifically developed to leverage the capabilities of Jama Connect and Verisium Manager. However, the same can be used for different RM and regression management tools, on the condition an API like the one described above is provided to extend their functionality. Except for Jama Connect, these conditions are met by, for instance, IBM DOORS [14], Siemens Polarion [10], or Valispace [13]. This list is by no means exhaustive, but it shows that an API is quite a common feature of RM tools.

Even though the scripts themselves are not reusable by other semiconductor producers due to the custom Jama Connect items and relationship diagram, it is fairly simple to develop them using the presented workflow and the Jama REST API documentation as references. The scripts were developed using information from open sources [12], [15], and Cadence Verisium Manager manual, which is not publicly available. For this reason, the scripts cannot be released under an open-source license.

V. CONCLUSION AND FUTURE WORK

Design requirements tracing workflow is the base of successful design verification because it establishes a way of measuring verification progress, and confidence in the verification results. This paper presented a verification workflow between Jama Connect RM tool and Cadence Verisium Manager verification platform. Automation based on Python scripts using Jama REST API client and Verisium Manager batch commands, together with minimizing the number of steps in the workflow were the main approaches to achieve a consistent way of tracing the design requirements to the verification results. The usage of this workflow leads to:

- verification results **traceability** in Jama Connect directly for Systems architect and upper management,
- **short execution cycles** as we believe this approach will save several days or even weeks of work,
- **minimization of human errors**,
- **cost reduction** of full client license for verification plan development in Verisium Manager on each future project,
- high quality verification results with **minimal bug slippage** due to a clear mapping of all requirements onto a definite verification plan along with its associated results,
- **better requirements documentation** and clarity on what is expected to be designed and verified.

The presented workflow is scalable across multiple tool vendors with little effort in the adaptation of the scripts. When moving to a different tool, the format of the files produced and accepted by them need to be considered. Furthermore, using a different API implies the need to change the library methods used within the scripts. The presented workflow was successfully used to upload Model Based Verification regression results into Jama Connect from Simulink Test Manager [16], a regression tool by MathWorks.

The scripts are still being improved based on the most recent user experience. The two-step configuration of VSIF files could be further optimized. In the future, the configuration could be done only in a Jama Connect item, which would be directly used to generate the VSIF files. With this improvement, it would be possible to merge the scripts, *vpgen_japi* and *vmgr_jama_updater* into one script and use a single thread which would download the verification plan, launch a regression session, and upload the results back to Jama Connect. With this improvement, the workflow could be modified for automatic execution within a CI flow e.g., using Jenkins [17] or GitLab CI/CD [18].

Acknowledgment

The authors would like to thank to their colleagues Jan Kubiš, Harshil Patel, Aswini Tata and Kamil Vyštejn for evaluating the workflow and for their invaluable feedback.

References

[1] *Jama Connect*. (2023). Jama Software. https://www.jamasoftware.com/platform/jama-connect/ (accessed Oct. 26, 2023)
[2] *Verisium Manager*. (2023). Cadence. https://www.cadence.com/en_US/home/tools/system-design-and-verification/ai-driven-verification/verisium-manager.html (accessed Oct. 26, 2023)
[3] Cadence. "Requirements Traceability." Cadence.com. https://www.cadence.com/en_US/home/solutions/automotive-solution/requirements-traceability.html (accessed Oct. 26, 2023)
[4] OpsHub Integration Manager. (2023). OpsHub. Opshub.com. https://www.opshub.com/vmanager-integration/vmanager-integration-with-jama/ (accessed Oct. 26, 2023)
[5] G. Pachiana, M. Grunwald, T. Markwirth and C. Sohrmann, "Automated traceability of requirements in the design and verification process of safety-critical mixed-signal systems," in *Proc. of Design and Verification Conf. and Exhib.*, 2021. [Online]. Available: https://dvcon-proceedings.org/document/automated-traceability-of-requirements-in-the-design-and-verification-process-of-safety-critical-mixed-signal-systems/
[6] B. Craw, D. Crutchfield, M. Oberkoenig, M. Heigl and M. O'Keeffe, "Using Automation to Close the Loop Between Functional Requirements and Their Verification," in *Proc. of Design and Verification Conf. and Exhib.*, San Jose, CA, USA, 2018. [Online]. Available: https://dvcon-proceedings.org/document/using-automation-to-close-the-loop-between-functional-requirements-and-their-verification/
[7] Jama Software, Portland, OR, USA. *Getting started Jama REST API.* (2023). Accessed: Oct. 26, 2023. [Online]. Available: https://dev.jamasoftware.com/api/
[8] O. C. Z. Gotel and C. W. Finkelstein, "An analysis of the requirements traceability problem," *Proceedings of IEEE International Conference on Requirements Engineering*, Colorado Springs, CO, USA, 1994, pp. 94-101, doi: 10.1109/ICRE.1994.292398.
[9] B. Ehlers and P. Carzola, "Best Practices in Verification Planning," in *Proc. of Design and Verification Conf. and Exhib.*, San Jose, CA, USA, 2013. [Online]. Available: https://dvcon-proceedings.org/document/best-practices-in-verification-planning-presentation/
[10] *Polarion - Software.* (2023). Siemens. https://polarion.plm.automation.siemens.com/ (accessed Oct. 26, 2023)
[11] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph. D. dissertation, Inf. and Comput. Sci., UC, Irvine, CA, USA, 2000. [Online]. Available: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm
[12] *py-jama-rest-client*. (2023). Jama Software. https://github.com/jamasoftware-ps/py-jama-rest-client (accessed Nov. 1, 2023)
[13] *Valispace*. (2023). Valispace. https://www.valispace.com/ (accessed Nov. 1, 2023)
[14] *IBM Engineering Requirements Management DOORS*. (2023). IBM. https://www.ibm.com/products/requirements-management (accessed Nov. 1, 2023)
[15] NumFOCUS, Austin, TX, USA. *Pandas documentation*. (2023). Accessed: Oct. 26, 2023. [Online]. Available: https://pandas.pydata.org/docs/index.html
[16] *Simulink*. (2023). MathWorks. https://www.mathworks.com/products/simulink.html (accessed Dec. 18, 2023)
[17] *Jenkins*. (2023). https://www.jenkins.io/ (accessed Dec. 18, 2023)
[18] *GitLab CI/CD*. (2023). https://docs.gitlab.com/ee/ci/ (accessed Dec. 18, 2023)