b4A White Paper

Test Automation –

Improvements in stability with automated testing

best-blu consulting with energy GmbH

Your contact:
Andreas Büsching

E-mail a.buesching@best-blu.de
Phone: +49 421 491 811 80
Date: May 27, 2025

Content

Motivation

In today's IT landscapes, requirements are increasing overall, making standardization in technology and processes urgently necessary. This also includes the area of automation. In many environments, this area has differed from other parts of the IT landscape in terms of its technologies and processes. Today, automation is becoming increasingly important in companies, and so is the need to align it with the rest of the IT landscape. In addition to processes and technologies from the field of deployment, other topics are also relevant. The area covered in this document is test automation. The content is based on a contribution by Sebastian Weha, who has already managed projects in this area for several customers.

Introduction

In the field of software development, the term *software development lifecycle* (SDLC) is widely used. It describes the stages from requirements analysis to the finished software and its evaluation. There are many methods for implementing the SDLC – and many discussions about which one is the best. Among the best known are agile software development, the waterfall model, and DevOps.

All these methods have one thing in common. They place great emphasis on testing the software. This can also be applied to development in the Automic Automation Engine – at least in theory. In practice, however, testing tends to be neglected, as has become apparent in many projects. Based on this experience, a concept was developed to automate testing and provide a solution for it in best4Automic (b4A).

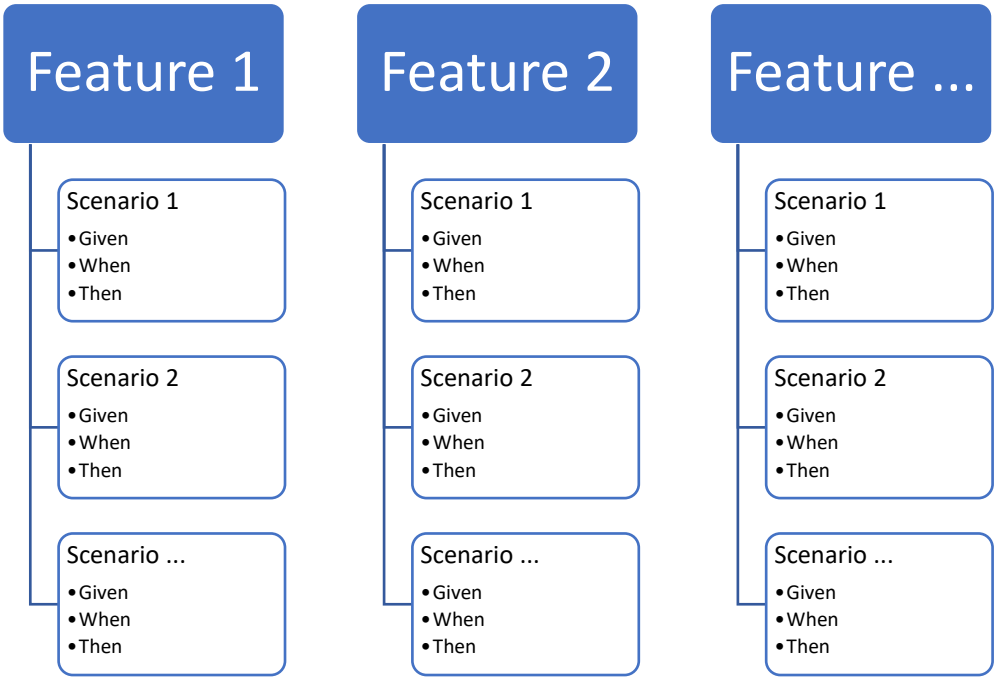
The solution is based on b4A packages, which are modeled on the structure of Automic Action Packs. The standard configuration takes the structure of Action Packs into account, so customers who already use Action Packs can get started right away. In preparation for test automation, it is necessary to define how the tests are to be performed and evaluated. Test definitions are used for this purpose. Since solutions already exist for such issues in software development, these are used.

Specification of test definitions

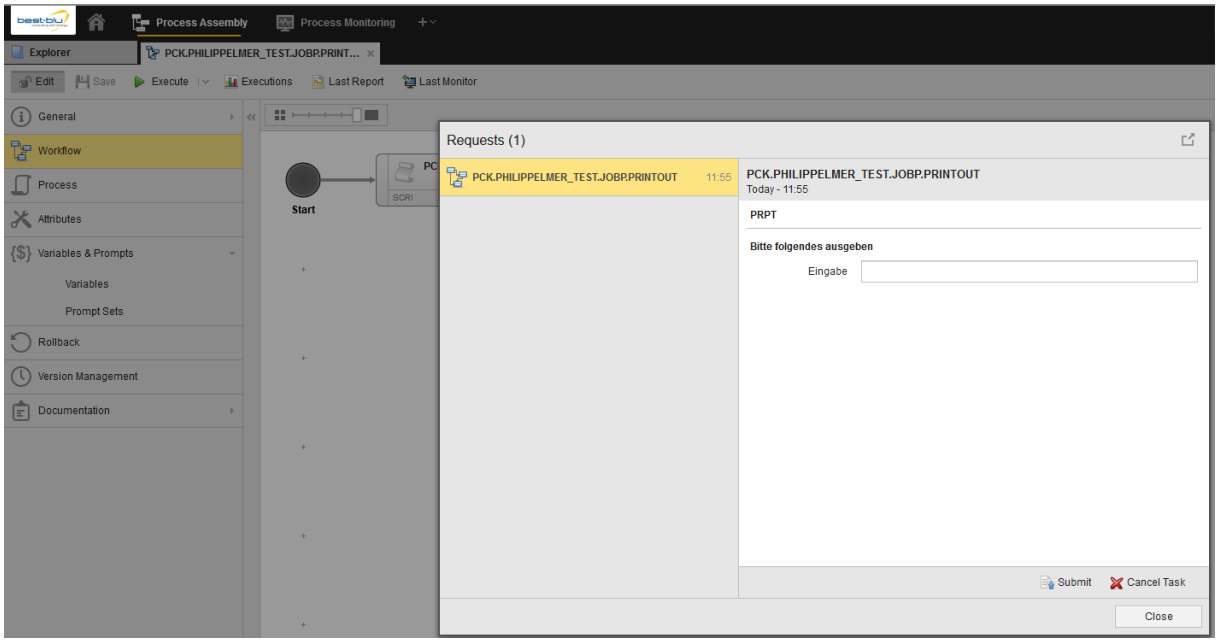
The test definitions are written in the *Gherkin* description language. This syntax is widely used and is used by many so-called *behavior-driven development tools*. The idea is to write the tests in a "natural" written language. These test definitions can be processed automatically using specific keywords.

The tests are generally structured as follows:

- **Feature**: Describes a set of test cases (scenarios) that are thematically related
- **Scenario**: Describes a single test case
- **Given**: Describes the steps that are checked before the actual test execution (preconditions).
- **When**: Describes the actual test execution
- **Then**: Describes the steps that are checked after the test has been executed (post-conditions).

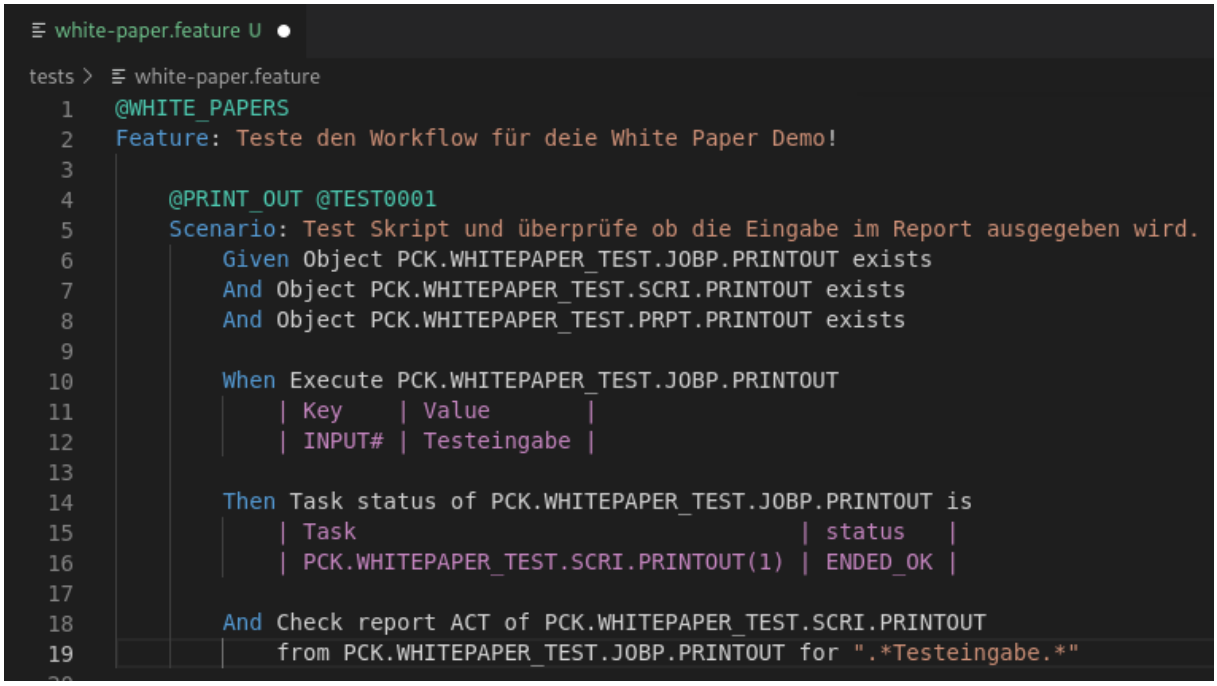


To explain this in a more practical way, a simple workflow is created below. This includes a PromptSet for text input and a script that outputs the input again.



A simple workflow with PromptSet

The function of this workflow is checked with the following test specification.



The specification contains a **feature** with a very simple **scenario**.

First, the GIVEN phase checks whether all three objects are present in the client. If one is missing, the test aborts with an error.

The WHEN phase describes the execution of the workflow and passes the PromptSet parameters. The table shows the PromptSet parameters that are passed (key: INPUT#, value: test input).

The THEN phase checks whether the task in the workflow has the status ENDED_OK and whether the report contains the text entered.

Execution of tests

The Gherkin language executes tests via the Cucumber framework and retrieves the results from there. This step is not entirely trivial, as an interpreter is required to translate Automic into the expected results and into a language that Cucumber can understand.

This functionality is part of the best4Automic Solution (*ta.Execute* module). The module can be used to execute the test cases specified in *Gherkin*. You can either transfer the test specification in a text file or store the test specification in a DOCU object. In the example, a DOCU object was used. The test execution report is also stored in this object by the module (object for report). This contains all details of the execution in an XML structure.

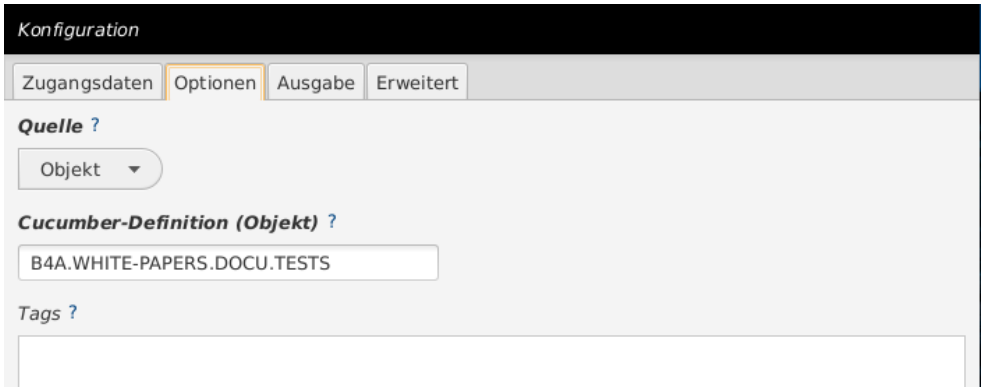
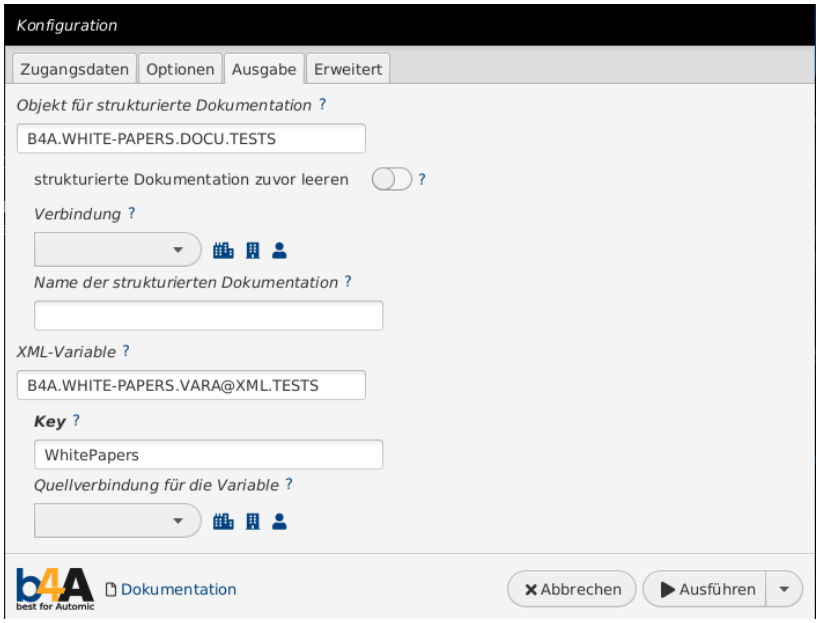


Figure1 —Specifying the source for the test specifications



2 - Output of the report to an XML variable and/or structured documentation

After the tests have been executed, a report with the results is displayed.

The test was executed successfully and the text "Test input" was found in the SCRI activation report.

The test report shown here is also available as an XML document in the object specified by the option. This allows the test results to be analyzed and further processed using Automic scripting tools. For example, the test results can be forwarded as an email or transferred to test management software.

best4Automic Test Automation now supports around 40 different test steps in the three phases, many of which can be used in different ways. In addition, the test specifications can be parameterized so that they can also be used flexibly.

Conclusion

With best4Automic Test Automation, it is possible to write test specifications in a widely used language and execute and evaluate them automatically. This means that automatic tests only require initial effort to create the specifications and can then be performed automatically, significantly improving the stability of the automation.