jQAssistant
powered by buschmais GbR

CASE-STUDY

Standardization and Automation at E-Post Development

E-POST

Company

E-POST

Deutsche Post E-POST Development GmbH
Berlin, Deutschland

Open-Source-Framework

jQAssistant

jQAssistant powered by buschmais GbR

www.jqassistant.org
www.buschmais.de

The Berlin-based E-Post Development GmbH develops and operates the E-Post platform that is based on the microservice approach to digitally dispatch letters for business and private customers. The company and each of the development teams are constantly facing the challenge of ensuring continuous implementation of common standards. The jQAssistant open-source framework provides help in achieving this goal.

As a wholly owned subsidiary of Deutsche Post AG, E-Post Development GmbH is responsible for digitization of the corporation‘s mail business. E-Post Development therefore plays a key role in developing digital services around the mail services market for both business and private customers of Deutsche Post AG.

As part of the digitization strategy of Deutsche Post AG, E-Post Development develops software systems to optimize existing processes around postal communications. To support this strategy with technology, E-Post Development has continuously focused on providing a flexible and sustainable platform, ensuring that business processes can be quickly adapted and expanded. This places high demands on the development teams entrusted with implementation of the business processes, especially since each team is responsible for multiple software systems at the same time. In facing this challenge, E-Post Development opted for Java-based technologies throughout. Architecturally, the approach is to distribute the functionality required to operate the overall platform across separate functionally-tailored services. This approach, also known as microservice architecture, is becoming increasingly popular for the development and operation of larger platforms.

But the ensuing advantages, including improved maintainability and better scalability, present new challenges. The increased number of software systems once again highlights the importance of compliance with standards in enabling rapid initial development and sustained further development.

Therefore, the team responsible for master data management decided to look for a tool that would help them develop their services by automatically analyzing and verifying various individual aspects of a software project from configuration through project structure to dependency management. They chose jQAssistant, as it is not just limited to source code analysis, but also allows rules to be flexibly defined across various artifacts.

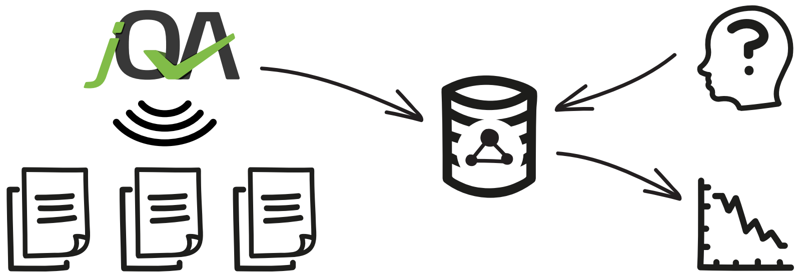

Figure 1: Software projects scanned with jQAssistant can be evaluated both manually and automatically.

Use of jQAssistant at E-Post Development

The team decided to meet the aforementioned challenges by a number of means, including a standardized build process and architecture guidelines for all software systems for which they are responsible. The jQAssistant approach of reading any artifacts including Java classes, libraries, configuration files, and directory structures, storing them in a graph database and enabling the querying of the collected data with a query language that is uniform for all captured artifacts, has proven to be particularly flexible and easily expandable (Figure 1).

The key argument in favor of choosing jQAssistant was the resulting ability to verify different aspects with a single tool, rather than use multiple tools or user-written scripts for performing the same task.

Standardizing the build process with jQAssistant

The team initially approached the task of standardizing the build process. Until then, build processes of different software systems varied in terms of procedure and configuration options. Considering the number of software systems under the control of the development team, this meant that enhancement and maintenance of existing configurations in the continuous integration (CI) system required a lot of effort. Also, over time, a large part of the knowledge about the build processes shifted from the build tool (Apache Maven) to the CI system. Accidentally deleted or incorrectly modified build processes were no longer recoverable without a significant overhead.

To improve the situation, the development team decided to standardize both the build processes—in terms of procedure and options for parameter setting—and the build procedure for all software systems.

jQAssistant on the Web

Consulting
www.jqassistant.de

Blog
www.jqassistant.org

GitHub
https://github.com/buschmais/jqassistant

Twitter
www.twitter.com/jqassistant

So they designed an Apache Maven-based build structure for all software systems. After that, verifiable rules were set with jQAssistant, to ensure compliance with the Maven build structure. Compliance with these rules is immediately verified after each check-in to the source control management (SCM) system. Violations of the rules that protect the build structure instantly lead to a build failure and allow the development team to fix the detected violations immediately.

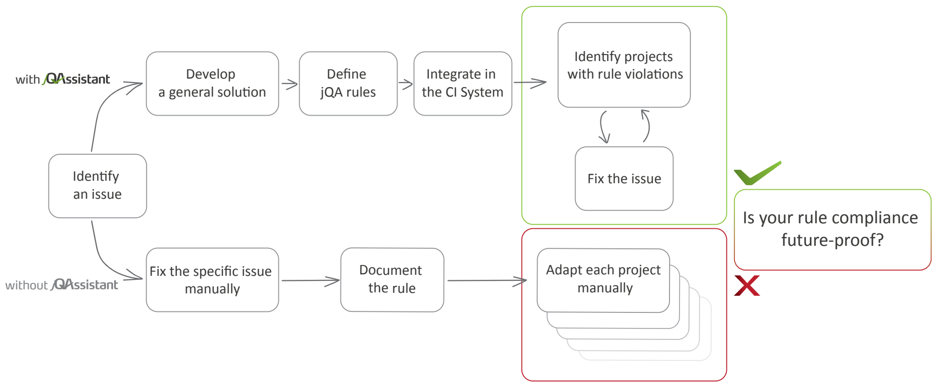
with jQAssistant
Identify an issue
Develop a general solution
Define jQA rules
Integrate in the CI System
Identify projects with rule violations
Fix the issue
without jQAssistant
Fix the specific issue manually
Document the rule
Adapt each project manually
Is your rule compliance future-proof?

Figure 2: jQAssistant now effectively prevents known issues from reoccurring.

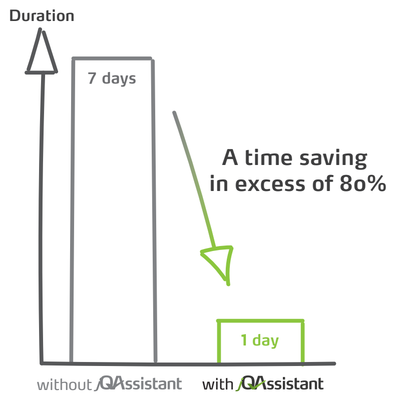

Figure 3: Recent improvements have enabled a much faster provisioning of new services.

Standardizing the software systems with jQAssistant

Further development of existing software systems and initial development of new ones depends on ensuring compliance with current architecture guidelines established by the team itself. jQAssistant was also used to create verifiable rules for these requirements. Compliance with these rules is also ensured at each check-in to the SCM system by triggering a build process in the CI environment. The range of architecture rules that are automatically verified after the implementation of jQAssistant spans different areas including logging, error handling, provision of metrics, and use of libraries. New rules are added to the existing set of rules if necessary, or if new errors are identified. This ensures that these errors are not re-introduced into the team's software systems (Figure 2).

Higher software quality and better rule compliance with jQAssistant

In master data management, jQAssistant enabled higher software quality and better compliance with applicable architecture rules after six months of use. In addition, the achieved harmonization of the setup process for services under control of the team resulted in a more rapid deployment of new services. In the past, setting up a new service took 7 working days. Today, the same task can be accomplished within 1 day (Figure 3). The team also used the new uniformity of the services to develop tools for greater automation of its work.

Your contact person

For more information or consulting offers please contact:
Mr. Dirk Mahler (buschmais GbR)
Phone +49 351 3209230
beratung@jqassistant.de

Another benefit is that jQAssistant simplifies the task of identifying, fixing, and permanently preventing multiple occurrences of error patterns across different services.

The opportunities arising from the use of jQAssistant have enabled the team to implement new business processes and develop new services more quickly and more effectively.