A Journey of Functional Test Evolution in the Public Sector

Bruno Pedraça de Souza
UFRJ and UNIRIO
Rio de Janeiro, Brazil
bpsouza@cos.ufrj.br

Rebeca Campos Motta
UFF
Niterói, Brazil
rcmotta@id.uff.br

Jessica Soares da Costa
UFRJ
Rio de Janeiro, Brazil
jessicacosta@cos.ufrj.br

Elaine Nunes
CAPGov - COPPETEC
Rio de Janeiro, Brazil
elainenunes@cos.ufrj.br

Gustavo Rocha Barreto Pinto
CAPGov - COPPETEC
Rio de Janeiro, Brazil
grbpinto@gmail.com

Rafael Maiani de Mello
UFRJ
Rio de Janeiro, Brazil
rafaelmello@ic.ufrj.br

ABSTRACT

Since 2015, a public legal institution in Brazil has relied on a large-scale information system to support its services. As the system's complexity and business relevance increased over the years, functional testing evolved from an ad-hoc, manual approach to a systematic, automated practice integrated into the development workflow. This paper reports on the evolution of the maturity stages, detailing the challenges encountered and key milestones achieved by adopting Selenium IDE automated tool testing. It also presents the main lessons learned by the testing team over this long-term process. Notably, process adjustments, regression testing strategies, and the adoption of Selenium IDE were instrumental in enhancing software quality. By sharing this trajectory, we aim to provide practical insight into other institutions facing similar constraints and demands.

KEYWORDS

Software Testing, Selenium IDE, Experience Report

1 Introduction

Software systems for the public sector play a critical role in delivering essential services to citizens. The development and maintenance of these systems often operate under unique constraints such as limited budgets, high turnover, complex regulations, and the need to serve large and diverse populations [1]. Therefore, ensuring the quality and reliability of such systems is not only a business concern but also a matter of public interest.

This paper addresses the experience of maintaining a core information system from a Brazilian legal public institution over a decade. More specifically, we report the evolution of functional software testing practices for such systems, which initially lacked formal processes and specialized roles, but gradually matured into a structured quality assurance (QA) with automated support Selenium IDE tool [2].

In the following sections, we describe the system characterization and how the testing practices evolved over time. Grounded in this experience, we compiled practical lessons, highlighting the key role of organizational support in shaping testing strategies and composing dedicated functional testing teams, including a team for test automation. We expect that our report may contribute to supporting the mapping and improvement of software quality practices in the public sector.

2 Project Characterization

2.1 Client and Development Team

This experience report is based on an ongoing long-term partnership with a public-sector organization in a state of Brazil. The client organization operates in the legal domain, specifically within the justice system. The motivation for the project came from a strategic change in several Brazilian courts of justice to replace paper-based processes with digital processes) [3].

At the beginning of the project in 2015, the client's digital infrastructure was minimal, and legal procedures were largely manual. The client recognized the need for digitally integrated but despite the discussions about establishing the solution architecture evolving quickly, proper software development strategies, including those for quality assurance and testing, were not yet defined. Today, the client operates with significantly higher digital expertise, motivated by the complexity and volume of legal services it delivers. The pandemic period accelerated a shift toward remote service delivery and digital inclusion, granting complexity to the quality assurance process.

The system was conceived and is currently maintained by a third-party team composing the [omitted] program. This program develops and maintains software solutions for the public sector, working under the coordination of a university technology foundation, offering services across project management, software development, and testing. Initially, the team had 20 software professionals and operated with limited maturity. According to early project reports, agile development was limited, focused mostly on task breakdown and iteration planning - but lacking formal design or testing specialists. Requirements activities were informal, driven by meetings, evolving document drafts, and direct client interactions, and there was no specific team for quality assurance and testing.

Currently, the system has significantly evolved, and the team has around 100 professionals, with several technical capabilities. The testing squad focuses on functional testing, a type of software testing based on the functional requirements of the system [4]. Currently, the testing squad is composed of twelve dedicated professionals, organized into manual (six people) and automated testing (six people).

2.2 System Characterization

The software system under development is composed of multiple interconnected components designed to support the evolving needs of the client institution. The system's development has followed the institutional mission of the client, which is centered around offering legal support to citizens across a state of Brazil. As a result, the software must adapt to a wide variety of workflows, user profiles, and legal domains—ranging from criminal defense and civil disputes to administrative and family matters. Over time, four distinct waves have shaped the system:

1. Process Monitoring (2015): Started focusing on enabling systems users to track and organize information about assisted individuals, including case history, appointments, and internal documents.

2. Public Assistance (2016): As demand increased, new modules were introduced to manage appointment queues, referrals, and services, aiming to reduce bottlenecks in first-contact services.

3. Monitoring of Incarcerated Individuals (2018): The third expansion responded to the need for managing cases of incarcerated clients. This involved tracking sentencing information, custody status, and rehabilitation timelines, requiring integration with legal oversight mechanisms.

4. External Integration (2020-current): It has expanded its interoperability with external parties, enabling service delivery, reflecting a broader trend toward digital transformation in legal services. Key integrations include the postal service database (Correios), inclusion of artificial intelligence (AI), the Federal Service (Receita Federal) for retrieving official data, and major judicial process systems such as PJe, EPROC, and DCP.

What began as a single-purpose platform has grown into a complex, multi-service ecosystem that serves 20 thousand internal users, 2 million processes and 2.7 million registered people.

3 Functional Testing Stages

At the beginning, functional testing activities [5] were issue-oriented. In the project context, an issue (described in user story format) may relate to the development of a new feature, the evolution of an existing one, or bug fixing, where the scope of test automation covers all.

Stage 0 – *Ad-hoc* (2015–2018). This stage began with the project kickstart. During this stage, the system was under continuous development. However, at the end of this stage, the set of features was relatively small. During Stage 0, there was no dedicated team for testing or even testing leaders. The functional tests were carried out by the developers themselves during the sprints' workflow, without the concern of writing scenarios or providing detailed test reports.

Stage 1 – Report-oriented, retrospectively documented (2018–2022). It began with the definition of a testing leader and the hiring of the first analysts devoted to manual functional testing. With the formation of the testing team, all functional tests began to be conceived and executed only by testers. Upon completing the testing cycle of each sprint, testers documented the scenarios tested for each issue in a shared spreadsheet. The description of each scenario included its goal and the steps to be followed. However, the expected results were still not documented, and testing data were often omitted or outdated.

Stage 2 – Test case-oriented, retrospectively documented (2022–2023). To promote the automation of functional testing, the project hired a software engineering researcher specialized in software testing and allocated one member from the manual testing team to focus on automation. Considering the project characteristics and the profile of the testers, they identified Selenium IDE [1] as the best option. In parallel with the creation of an appropriate environment for automation, they diagnosed three key challenges for implementing an automated testing process: **i) Lack of proper testing documentation:** As noted in Stage 1, essential elements of test cases were missing, hindering the automation process. **ii) Late documentation:** Test cases were typically written only after deployment, making it unfeasible to automate them during development. **iii) Poor programming practices for automation:** These included unfriendly navigation structures, improper element naming, and random generation of form element identifiers. To address these issues, the specialist initiated a series of targeted actions. First, the testing team received training on how to write effective and comprehensive test cases, aiming to improve documentation quality and consistency. In parallel, meetings were held with project leadership to renegotiate sprint timelines and ensure that test case writing could be integrated earlier in the development process. Additionally, the specialist identified critical navigation paths within the system and began developing automation scripts to support regression testing. To facilitate automation and maintainability, development teams were also instructed to adopt consistent naming conventions for all form elements, addressing one of the key technical barriers to automation.

Stage 3 – Partially automated, documented (2023–2024). After significant progress in addressing previous challenges, functional test automation was incorporated into the sprint workflow of the issues implemented by one of the two development teams. This team was chosen due to its smaller size and the nature of its work, mostly maintenance of less complex features. At this point, most test cases were documented during development, allowing for timely automation and retesting in the release environment. Three new testers were hired and trained to support this effort. Automated regression testing was introduced to the release testing phase, significantly reducing the number of bugs reported in production and identifying merging issues. However, timely test documentation remained a challenge. To address this, a strategy was introduced to pair manual and automation testers, aiming to integrate and accelerate documentation and automation. Despite initial success, this initiative faced scheduling conflicts and shifting priorities and was ultimately discontinued. Less than 10% of functional tests were identified as non-automatable due to test scenarios exceeding navigation boundaries (e.g., validating documents). Although this was a minor percentage, automation coverage remained between 60% and 70% per sprint due to documentation delays.

Stage 4 – Fully automated, promptly documented (2024–ongoing). Following the successful automation efforts of one team, by mid-2024, the test automation team assumed responsibility for testing across the entire system. Although the tester pairing initiative did not yield all the expected results, it effectively raised awareness about the importance of accessible, timely test documentation. Taking advantage of a postponed deployment in late 2024, technical leaders prioritized clearing the testing documentation backlog, renegotiating priorities, and refining sprint workflows. As a result, automation coverage increased significantly in the subsequent sprints.

4 Lesson Learned

During the stages followed for automating the functional tests, we learned important lessons. we discuss the major ones:

The efficiency of regression testing strongly depends on the coding standards. The automation of functional tests allowed the adoption of regression testing for the first time in the project. After regression testing, we observed a considerable and stable decline in bug reporting by system users. From regression testing, it is expected to be fast and autonomous (re)execution. However, the effort spent with regression testing may considerably increase if poor programming practices are applied. Some practices, such as the random generation of identifiers, may lead testers to manually fix several recorded steps, which may take more time than manual testing.

Selenium IDE is a practical alternative for populating testing data. In large and complex business systems such as our system, it may be challenging to continuously develop and maintain database scripts to update testing data after restoration. Alternatively, our experience showed that recording and maintaining data inclusion scripts through Selenium IDE is alternative. Besides, it minimizes the need for support from database teams for testing.

Selenium IDE is a practical tool for beginners in automation. The initial version of the automated testing team was composed of functional testing experts, but most of them were inexperienced with testing automation. Despite this limitation, they quickly embraced Selenium IDE due to its intuitive interface, ease of use as a browser plugin, and the ability to record automated test scripts. These characteristics significantly streamlined the adoption process and accelerated the training of new team members.

Selenium IDE logs may be used for self-document functional testing. The testing team is responsible for assuring that the system meets user requirements through testing activities. With the adoption of Selenium IDE, the testing team can export user-friendly logs describing each navigation step applied. Besides, Selenium IDE commands allow testers to easily assign each test case description and its steps in the logs.

Sharing knowledge to overcome tool limitations. Despite the ease of using Selenium IDE, its command library has some key limitations. Regardless, Selenium IDE allows testers to insert JavaScript (JS) functions and instructions in its recordings. Besides, it is possible to store and manipulate the values returned from running these scripts. A simple example addresses using a JS command to get the current system date to be used in native Selenium IDE commands. In this sense, sharing knowledge is essential to avoid reworking. The automated testing team maintains a single repository to support all automation activities.

5 Conclusion

This experience report presented the long-term evolution of software testing practices in a public sector information system. Over nearly a decade, testing activities matured from informal, undocumented practices to a structured, partially automated, and eventually fully automated process. Each stage reflected organizational shifts, growing technical complexity, and awareness of quality assurance's value. Key enablers of this evolution included team specialization, early documentation, and the adoption of tools like Selenium IDE. Despite setbacks, such as documentation delays and process misalignments, the initiative demonstrated that significant quality gains are achievable even under the constraints of public administration.

Based on this trajectory, we identify three promising research directions: (i) evaluating the cost-benefit and scalability of lightweight automation tools in public sector systems, (ii) exploring how automated test artifacts (logs and reports) contribute to traceability, auditability, and compliance, and (iii) studying the adoption curve of no-code and low-code tools among non-programmer testers. By sharing this journey, we hope to inspire other public institutions to invest in testing practices as a path to delivering more reliable and sustainable services.

ACKNOWLEDGMENTS

This paper is a result of a real Research&Development project. The authors also thank CNPQ and CAPES - Finance Code 001.

REFERENCES

[1] Yuri Lima, Luana Passos and Jano M. de Souza. 2025. Analysis of automation and retraining opportunities for the Brazilian federal public service. *Conference on Digital Government Research*. https://doi.org/10.59490/dgo.2025.1031

[2] Selenium IDE. Retrieved July 1st, 2025, from https://www.selenium.dev/pt-br/documentation/ide/

[3] Marcus Parreiras, et al. Inteligência artificial aplicada para o aumento da produtividade no atendimento de intimações. WCGE. SBC, 2022. p. 180-191.

[4] Jan Severo and Valéria Lelli. 2023. Testing Maze: an educational game for teaching functional testing. *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. 2023. https://doi.org/10.1145/3613372.3614191

[5] Glen J. Myers, Corey Sandler and Tom Badgett. 2011. The art of software testing. John Wiley & Sons.