TESTNAVIGATOR

Case Study: Developing a Test Coverage Measurement Tool for a Leading Hungarian Bank

Background

A leading Hungarian bank wanted to improve the reliability of its **User Acceptance Testing (UAT)** process. Although the bank had mature testing practices, the management team lacked **quantitative visibility** into how much of the business logic and code was actually covered during UAT. This made it difficult to assess readiness for production and often led to undetected issues slipping into live systems.

Challenge

The bank's goals were to:

- **Measure and visualize test coverage** across business processes, system components, and code.
- **Detect untested changes** before releases to reduce post-go-live incidents.
- **Introduce measurable quality gates** into the UAT phase.
- **Automate coverage tracking** without disrupting existing test management and CI/CD processes.

However, no off-the-shelf product fit the bank's complex technology landscape or its hybrid Agile-Waterfall SDLC.

Our Approach

Our company conducted a **comprehensive assessment of the bank's SDLC and testing processes**, including interviews, workflow analysis, and tooling review.
Based on the findings, we designed and developed a **tailor-made software solution** that could automatically measure UAT test coverage and provide actionable insights to testers, business users, and release managers.

The result was a **coverage-driven testing platform**, conceptually aligned with the capabilities later known from testnavigator.ai.

The Solution: A Custom Test Coverage Measurement Tool

The developed application combined code-level instrumentation, change detection, and analytics to create a **real-time picture of testing completeness**.
Its key functionalities included:

1. Coverage Measurement

- Measured executed **methods and branches** during UAT sessions with minimal performance impact.
- Provided visibility into which parts of the code and business workflows were actually tested.

2. Change Detection

- Automatically identified **code changes** between releases and highlighted newly modified modules requiring additional UAT focus.
- Enabled **change-based testing**, ensuring no untested modifications went live.

3. Test Selection and Prioritization

- Used an intelligent **test selection algorithm** to recommend which test cases should be executed first, based on recent changes, code complexity, and historical defect density.

4. Traceability and Dashboards

- Linked business requirements, user stories, and UAT test cases to corresponding code coverage data.
- Offered real-time **dashboards** and **coverage heat maps**, showing test progress and risk areas at a glance.

5. Integration with Existing Tools

- Integrated seamlessly with the bank's existing test management and CI/CD ecosystem through an **open API**.
- Enabled automated quality gates in release pipelines based on coverage thresholds.

6. Coverage-Based Quality Gates

- Supported the definition of **exit criteria** for UAT: a release could only proceed if coverage and risk metrics met predefined thresholds.

Results

After deploying the new tool and embedding it into the bank's UAT process:

- **UAT coverage visibility increased by over 40%.**
- **Post-release defect rates dropped significantly**, as untested changes were identified early.
- **Testing efficiency improved**, since business testers focused on areas with the highest risk and lowest coverage.
- Management gained **quantitative, real-time insight** into UAT readiness, enabling data-driven go/no-go decisions.

Conclusion

Through deep process understanding and targeted software development, our company delivered a **bespoke coverage-measurement** platform that transformed how the Hungarian bank approached UAT.
The solution introduced **data-driven quality assurance**, turning subjective testing decisions into measurable, repeatable processes and significantly reducing production failures as a result.