Extra Difficulties of Automated Testing in Low-Code Development

Fahmida Israt, Lutz Prechelt
03. bis 04. Juli 2025
TAV Workshop, Hamburg

1. Agenda

- Motivation: The Low-Code (LC) Automated Testing Challenge
- TDD/BDD in Low-Code: Core Struggles
- Case Study: LC ATF Experience
- Key Challenges & Opportunities
- Conclusion & Future Work

2. Motivation: The Low-Code Testing Challenge

Low-code platforms widely adopted in IT/business

Promise: Rapid dev, citizen developers, high productivity with minimal coding

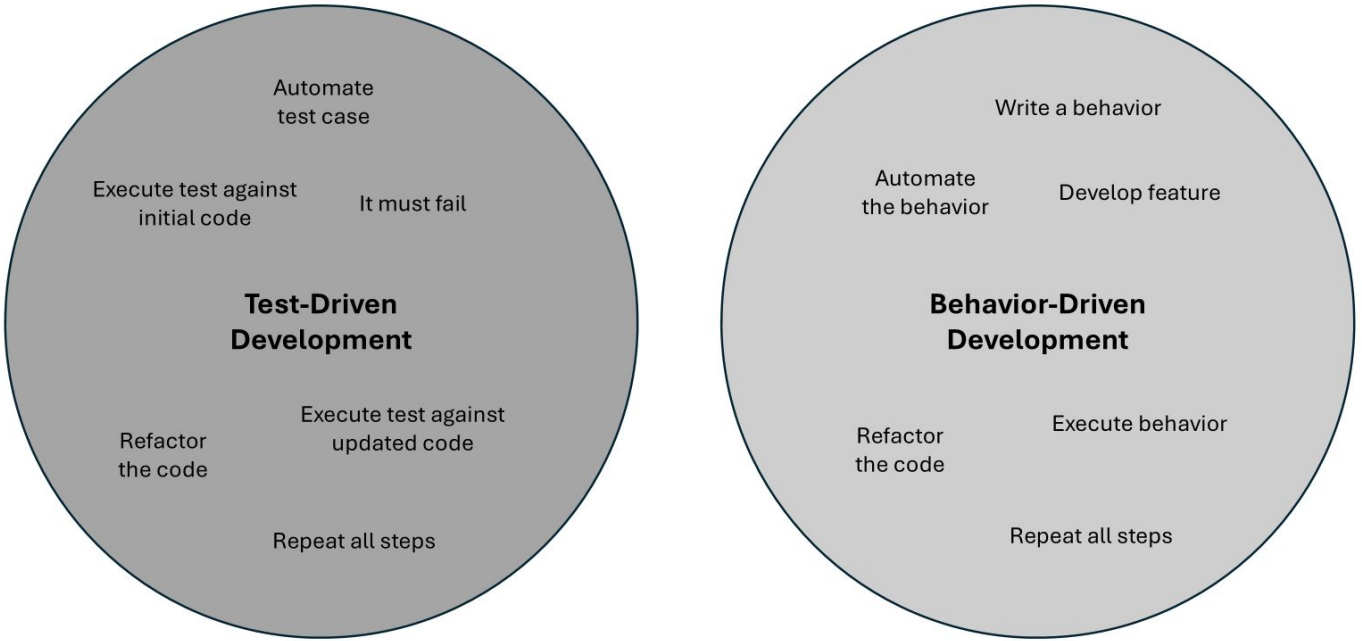
Reality:

- Quality assurance often weak
- Automated Test Framework (ATF) exists, but TDD/BDD application is complex due to User Interface (UI) centric design

Core Question: How to ensure quality when code is abstracted?

Goal: Explore limitations & testing improvements

3. TDD & BDD at a glance



4. Why TDD & BDD Matter for Software Quality

Test-Driven Development (TDD):

- Write tests first, promotes incremental design
- **Benefits:** Improved code quality, maintainability, reduced defect rates, faster feedback loops for developers

Behavior-Driven Development (BDD):

- Business-facing, behavior-focused testing; often uses natural language (e.g., Gherkin: Given-When-Then scenarios)
- **Benefits:** Enhanced collaboration between technical and non-technical stakeholders, ensures development aligns with business requirements

5. Question & Focus

Key question: Can these succeed in LC platforms?

Our Focus: Integration challenges in native LC ATFs

6. Brief Look at Related Work

TDD & BDD in Traditional Development:
Extensively studied, well-established benefits (e.g., Beck, Cui)

LC Testing Landscape:

- Often relies on third-party tools (e.g., Selenium, TestNG)
  not inherently designed for TDD/BDD workflows [Khorram et al. 2020]
- Focus on UI/end-to-end testing,
  limited support for granular unit-level testing [Khorram et al. 2020, Pulido 2019]
- *Example:* ServiceNow, Power Apps primarily focus on UI,
  lacking unit-level TDD support

Key Research Gap:
Limited studies exploring the intersection of TDD/BDD principles
within the specific constraints of LC platforms' native ATFs

Our Contribution: Industrial experience report bridging this gap

7. Manual Test Steps

Short Description: Create a work note in an Incident

Test Steps:

Go to Incident and click on ticket number to open a ticket

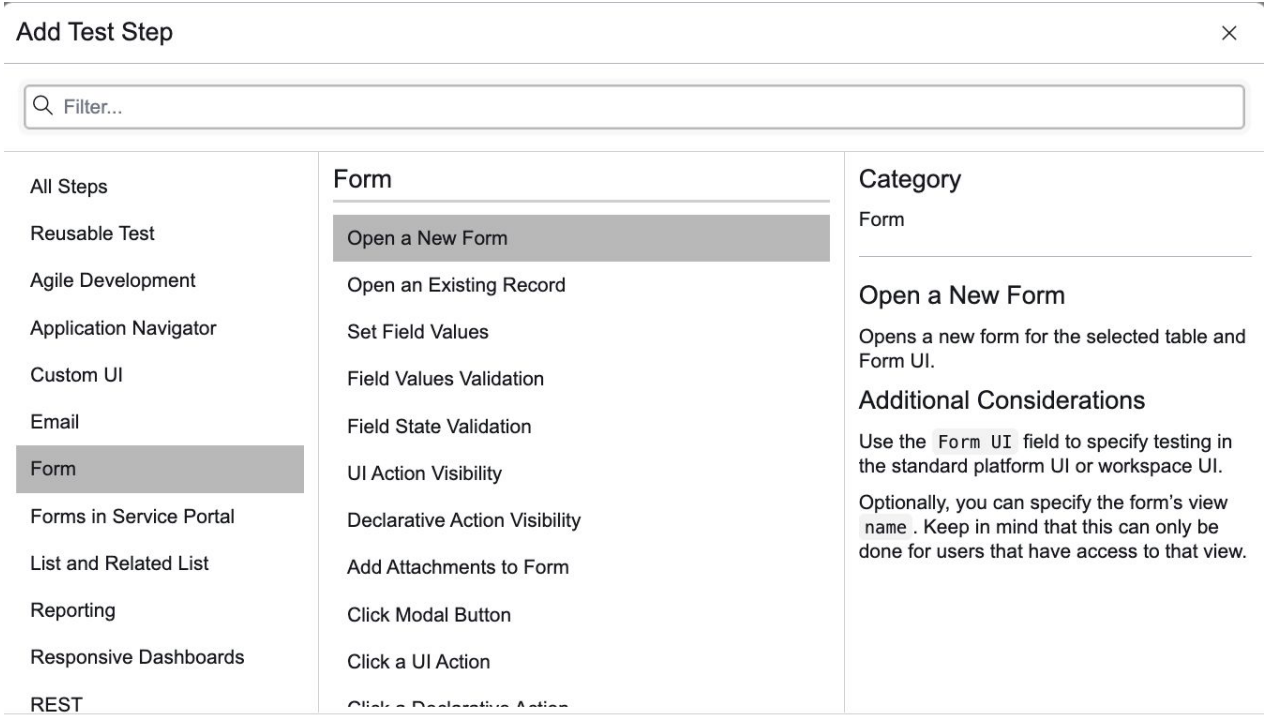
Scroll down in the ticket to the activities stream

Work note is visible

Type in a work note and post it to save it in the ticket activities

Check if work notes are now visible for all

8. Automated Test Framework UI

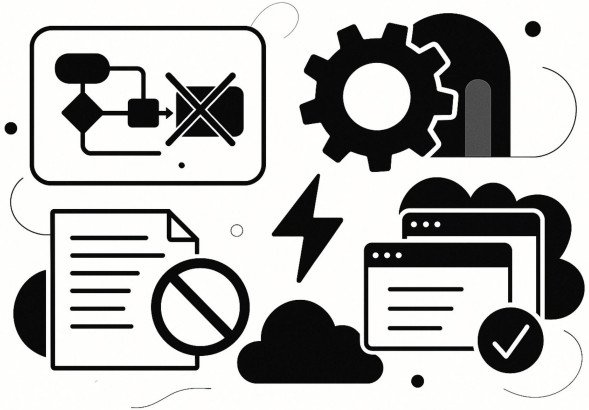


9. Final Results UI Based Test Steps

Display name	Description
Create User	• Create and impersonate user with the following values: • First name = test • and Last name = test
Open a new form	• Open a new Incident form
Set Field Values	• Set the following values on the form: • Requested for = (empty) • and Opened by = (empty) • and State = Open

10. Why They Struggle in LC?

- **Visual Dev:** Limits access to underlying logic
- **Built-in ATFs:** UI-focused, not unit/behavior
- Limited scripting/integration with standard tools
- **Result:** Mismatch with traditional testing workflows



11. Experience: Case Study

Context: Qualitative case study on ServiceNow's Automated Test Framework (ATF)

Data:

- Review of ServiceNow documentation & existing case studies
- Analysis of industry implementations of LC platform test automation
- Gathering expert feedback from developers & testers using ATF

Project Snapshot:

- Team: 2 developers, 1 product manager, 1 experienced external member
- Environment: Dedicated ServiceNow development instance; separate staging for User Acceptance Testing (UAT)
- Goal: Automate key business processes, ensure data integrity, validate ~30 static test cases

12. Development & Testing Approach (1/2)

1) **Development Process:**
   Structured 4-month dev sprint, focus on long-term maintenance

2) **Hybrid Testing Approach:**

- Combined **manual testing** (in parallel with ATF executions) and **automated testing** via ServiceNow ATF
- **Rationale:** Maximize coverage, minimize manual effort, early detection

13. Development & Testing Approach (2/2)

3) **Economic & Risk Factors:**
- Significant efficiency gains from automation: Manual 5 cases (1-3 hrs) vs. ATF (15 mins)
- Staged testing crucial for managing production data corruption risk; cross-functional admins change configuration in production directly which is not recommended

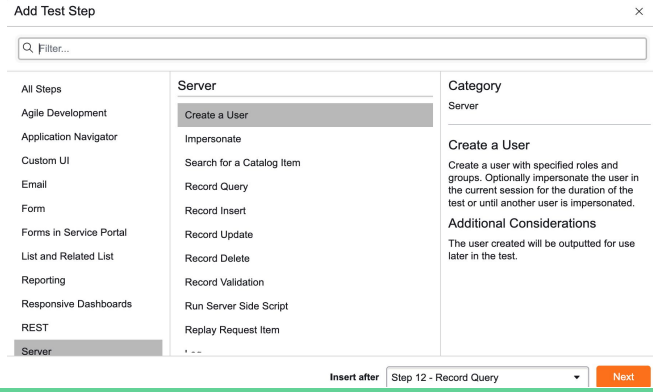
4) **User Feedback:**
Collected during UAT, vital for refining scenarios and integrating user needs into automated tests

14. ServiceNow ATF: An Overview

ServiceNow Automated Test Framework (ATF):

- Built-in tool for test automation within the platform
- Graphical interface reduces need for scripting
- Reusable test cases for UI and functional testing
- Automated updates with platform releases, simplifying maintenance

Adaptable, but not full TDD/BDD tool (challenges arise)



15. TDD in Low-Code ATFs (1/2)

1) **UI-Focused Design Hinders "Test-First":**
- ATF relies on *predefined, UI-driven test steps*
- Makes writing tests *before* functionality difficult – requires designing tests for non-existent UI
- Increased cognitive load, speculation, and misinterpretation

2) **Lack of Isolated Unit Testing:**
- Limited built-in support for testing script includes and business rules in isolation
- Essential for granular TDD, difficult to achieve

16. TDD in Low-Code ATFs (2/2)

3) **Slow Feedback Loop:**
- ATF execution, especially with UI interactions, is slow
- Incompatible with TDD's requirement for frequent, rapid test execution

4) **Limited Integration:**
- Challenging to integrate with popular Unit test frameworks (JUnit, NUnit) (**integration test)**
- Hampers importing/exporting test assets, increasing manual effort

17. BDD in Low-Code ATFs (1/2)

1) **Gherkin Syntax vs. Visual Steps:**

- BDD's Gherkin focuses on high-level behavior & business logic
- ATF uses predefined, heavily interconnected visual steps
- **Problem:** Converting Gherkin scenarios into visual steps often *distorts the intended business logic*

2) **Data-Centric Complexity & High Maintenance:**
   Effort for scenarios, frequent updates lead to decay

18. BDD in Low-Code ATFs (2/2)

3) **Fragile Visual Test Suites:**
- Minor UI changes frequently *break* existing test scenarios
- Refactoring large, complex suites is time-consuming, shifting burden to dev team

4) **Production Testing Limitations:**
- Most LC ATFs (including ServiceNow) **do not recommend running even smoke tests in production**
- Limits usefulness primarily to UAT, challenging post-deployment stability

5) **Lack of Seamless BDD Tool Integration:**
Difficulty importing/exporting scenarios or synchronizing results with external BDD tools (Cucumber, SpecFlow)

19. Opportunities for Enhancing TDD in Low-Code ATFs (1/2)

1) **Integrate External TDD Frameworks:**
   Develop middleware/plugins to allow granular, isolated unit testing (e.g., JUnit, NUnit)

2) **Leverage Generative AI:**

- **Automate Test Case Generation:** AI observing manual steps and mapping to ATF
- **Test Suite Optimization:** AI identifying and removing obsolete test cases based on requirement changes for future projects

3) **Improve Debugging Capabilities:**

- Enhanced drill-down into root causes of failures beyond basic logs
- Visualization of test execution flow for UI-heavy ATFs

20. Opportunities for Enhancing TDD in Low-Code ATFs (2/2)

4) **Enable Parallel Test Execution:**
   Significantly reduce test execution time, providing quicker feedback

5) **Quantify Gains from Test Automation:**
   Measure reductions in manual time, error rates, maintenance overhead, and improvements in reuse/scalability

21. Opportunities for Enhancing BDD in Low-Code ATFs (1/2)

1) **Enhanced Gherkin Integration:**
- Develop adapters/APIs for popular BDD frameworks (Cucumber, SpecFlow)
- Allow importing & executing Gherkin scenarios without distorting business logic

2) **Natural Language Processing (NLP) for Test Creation:**
- Replace semi-technical syntax, allowing business stakeholders to create basic test cases using natural language
- Leverage AI to achieve this goal

22. Opportunities for Enhancing BDD in Low-Code ATFs (1/2)

3) **Built-in Lightweight BDD Framework:**
   Vendor provided framework to simplify creation and execution of behavior-driven test cases directly within ATF

4) **Improved Reporting & Analytics for BDD Suites:**
   Vendor provided framework for better data-driven decision making and insights into test coverage and failures

23. Conclusion & Future Research Directions

Key Takeaways:

- LC platforms boost productivity, but applying TDD/BDD within their ATFs presents significant, unique challenges.
- Challenges stem from UI-centric design, limited code access, and integration gaps.
- Industrial experience highlights the need for specific adaptations and innovations.

Call for Future Research:

- Standardized, robust LC testing frameworks.
- Deeper AI/ML integration for test generation/maintenance.
- New approaches for granular unit testing in abstracted envs.
- Seamless bridge: Gherkin syntax ↔ visual test steps.
- Overall Goal: More effective TDD/BDD in low-code, improved quality.