Systematic Selection of Testing Methodology for Low-Code Development

Shreyasi Warunkar, Baris Güldali
Low-Code Development & Testing
  - What is Low-Code?
  - Testing Low-Code applications

Complexity of Low-Code applications
  - Case studies
  - Complexity characteristics

Test Methodology for Low-Code
Low-Code Development & Testing

What is Low-Code?

Testing Low-Code applications

- Data Modeling
- User Interface Design
- Specification of business logic rules and workflows
- Integration of external services via third-party APIs
- Application Deployment
Complexity of Low-Code applications

Case studies selected = 100 (Outsystems = 50, Mendix = 50)

Cases from variety of application domains –
Healthcare, Finance, Logistics, Insurance, Government, NGO, IT etc.

Complexity characteristics:
- Application type
- Time to develop
- Need of training
- Integrations with other systems
- Scalable
- Safety critical
- Customized
- Testing/QA mentioned
- Agile methodology incorporated

Source: https://www.featuredcustomers.com/
https://www.outsystems.com/case-studies/
https://www.mendix.com/customer-stories/
Development Time

- Measured in weeks
- Threshold = 12 weeks
- Less than 12 weeks for 55/100
- More than 12 weeks for 28/100

Assumption:
- Low-Code enables faster application delivery (Forrester survey)
- Assumption confirmed!

Inference:
- Less development time → Experience-based test techniques
- More development time → Elaborate testing

Scalability

- Yes for 76/100
- Not specified for 24/100

Assumption:
- Low-code applications are developed mostly simple, non-scalable.
- Assumption proven wrong!

Inference:
Need of professional tester for performing non-functional tests.
Integration with External Systems

- Yes for 68/100
- No for 7/100

Assumption:
- Integrations with external systems may be difficult with Low-Code
- Assumption confirmed!

Inference:
There may be a need of all test levels – unit, integration, system, acceptance.
Need of Training & Agile Methodology

Assumption for need of training:
- Low-Code platforms are designed specifically for ‘citizen developer’
- Thus, most will need training.
- Assumption proven wrong!

Assumption for agile methodology:
- Exploratory tests can be useful to test only changes in each sprint.
- Assumption incorrect!

Information about test artifacts – User stories for agile methodology
Other Complexity Characteristics

- **Application type**
  - Web: Outsystems (24), Mendix (20)
  - Mobile: Outsystems (5), Mendix (3)
  - Both: Outsystems (15), Mendix (6)
  - Not specified: Outsystems (6), Mendix (21)

- **Customization**
  - Yes: Outsystems (39), Mendix (41)
  - No: Outsystems (0), Mendix (1)
  - Not specified: Outsystems (11), Mendix (8)

- **Safety critical**
  - Yes: Outsystems (12), Mendix (8)
  - No: Outsystems (38), Mendix (42)

- **Testing mentioned**
  - Yes: Outsystems (13), Mendix (9)
  - No: Outsystems (37), Mendix (41)
# Rule Set for Test Methodology

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Questions</th>
<th>Test Aspects</th>
<th>Option 1</th>
<th>Option 2</th>
<th>Your Answer</th>
<th>Recommended Test Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Do the requirements exist?</td>
<td>Test Techniques</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Test Techniques: Black-box testing</td>
</tr>
<tr>
<td>2</td>
<td>Does the code exist?</td>
<td>Test Techniques</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Test Techniques: Experience-based testing</td>
</tr>
<tr>
<td>3</td>
<td>How much is the estimated development time?</td>
<td>Test Strategy</td>
<td>Less than / equal to 12 weeks</td>
<td>More than 12 weeks</td>
<td>1</td>
<td>Test Strategy: Analytical, Exploratory, Checklist-based on the requirements document</td>
</tr>
<tr>
<td>4</td>
<td>Is there integration with other systems?</td>
<td>Test Levels</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Test Levels: Unit, System test</td>
</tr>
<tr>
<td>5</td>
<td>Is the application scalable?</td>
<td>Test Types</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Test Type: Functional tests</td>
</tr>
<tr>
<td>6</td>
<td>Is a professional developer involved?</td>
<td>Test Roles</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Developed by: Professional developer</td>
</tr>
<tr>
<td>7</td>
<td>Is professional testing necessary?</td>
<td>Test Roles</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Tested by: Citizen tester</td>
</tr>
<tr>
<td>8</td>
<td>Is it an agile project?</td>
<td>Test Artifacts</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Test Artifacts: Requirements, design document</td>
</tr>
<tr>
<td>9</td>
<td>Is test documentation necessary for any compliance/audit formalities?</td>
<td>Test Artifacts</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Test cases: Test cases / test scripts need not be documented</td>
</tr>
</tbody>
</table>
Evaluation and next steps

Evaluation by Testing of a Low-Code Application

- Digital notification board for technology center
- Test methodology determined & executed (see slide 11)
- Evaluation
  - Tasks/Responsibilities according to training: Development by professional developer (could be citizen developer), testing by – citizen tester (2-man principle)
  - Coverage of requirements: Out of 16, 4 were not implemented & 2 are not working
  - Defects found: 8 Defects, 2 Usability issues, 1 Security issue found
  - Testing aspects covered: test levels, test types, testing techniques, test strategy, test roles and test artifacts.
  - Time for testing ($T_T$) in proportion to time for development ($T_D$): $T_T = 10\% T_D$

Low-Code testing vs. Model-based testing

- Similar scenarios identified based on need for redundancy
- See master thesis for detailed scenario descriptions
Thank you
Any further questions?
Contact me:
Shreyasi.Warunkar@sn-cqm.de