Comprehensive Testing Strategy for Process-Driven Automotive Repair Services: Verification and Validation of BPMN Implementations

Enzo Joly, 22055453

Word Count: 1000

Abstract

This paper presents a comprehensive testing strategy for process-driven automotive repair service implementations, focussing on the verification and validation of BPMN process automations. Through a systematic multi-layered testing approach spanning unit, integration, and system levels, the research demonstrates how rigorous test methodologies can ensure both technical correctness and business alignment in complex socio-technical systems. The testing architecture specifically addresses the challenges of process-oriented applications including human task verification, service integration testing, and cross-boundary process flows. By establishing clear traceability between tests and the original i* goals, the approach ensures that implemented processes faithfully represent stakeholder requirements whilst maintaining robustness and reliability in operational environments.

Contents

1 **Introduction** **1**

2 **Testing Strategy Framework** **1**
2.1 Test Derivation from Requirements 1
2.2 Layered Testing Architecture 1

3 **Unit Testing Approach** **2**
3.1 Human Task Testing 2
3.2 Service Task Verification 2

4 **Integration Testing Strategy** **2**
4.1 Message Flow Testing 3
4.2 Exception Flow Testing 3

5 **System Testing Approach** **3**
5.1 Scenario-Based Testing 3
5.2 Non-Functional Testing 4

6 **Conclusion** **4**

1 Introduction

Testing process-driven applications presents unique challenges compared to traditional software validation. As *(Bozkurt, Harman, and Hassoun 2013, p. 127)* observe, "process implementations span organisational boundaries and combine human tasks with automated services," creating distinct verification requirements. For automotive repair services, where customer trust and service quality represent critical success factors, comprehensive testing becomes essential to ensure both technical correctness and business alignment.

This paper outlines a systematic testing strategy for the automotive repair service process implementation, building upon the previously established i* socio-technical model and operational BPMN implementation. The approach employs what *(Giray and Tekinerdogan 2016, p. 219)* describe as "multi-dimensional verification techniques" that address both technical reliability and business goal achievement, ensuring that automated processes faithfully implement stakeholder requirements.

Testing business processes requires consideration of both control flow correctness and socio-technical alignment. As *(Weber, Reichert, and Rinderle-Ma 2016, p. 85)* note, "a technically perfect process that fails to align with organisational goals represents a critical business failure." This principle guides our testing architecture, which establishes clear traceability between technical tests and the softgoals identified in the original i* model.

2 Testing Strategy Framework

2.1 Test Derivation from Requirements

The testing strategy employs a goal-based test derivation methodology, extracting test cases directly from the i* model and operational BPMN processes. Following *(Kunze et al. 2015, p. 176)*' approach, each test is aligned with specific goals and quality attributes to ensure that "verification activities directly validate stakeholder intentions rather than merely technical correctness."

This goal-oriented test derivation creates explicit links between test cases and socio-technical elements, addressing what *(Grossmann, Schrefl, and Stumptner 2008, p. 352)* identify as a common weakness in process testing: "the disconnect between technical verification and business validation." For automotive repair services, this ensures that critical socio-technical elementsparticularly trust relationships, quality verification, and membership benefitsreceive explicit test coverage.

2.2 Layered Testing Architecture

The testing architecture employs a three-layered approach that addresses different aspects of process correctness:

1. **Unit Testing** - Focussing on individual process tasks, gateways, and activities to verify correct behaviour in isolation.

2. **Integration Testing** - Examining interactions between process components, particularly message flows between pools and service task integrations.

3. **System Testing** - Validating end-to-end process execution against business requirements, with particular emphasis on the customer journey.

This layered approach follows *(García-Borgoñón et al. 2017, p. 92)*' recommendation for process applications, ensuring "progressive verification from technical correctness to business alignment." Each layer employs specific testing techniques appropriate to its scope and objectives, as detailed in subsequent sections.

3 Unit Testing Approach

3.1 Human Task Testing

Human tasks represent critical touchpoints where system interactions meet user behaviour. Following *(Martinho, Domingos, and Varajão 2015, p. 217)*' methodology, our testing approach explicitly verifies both task presentation and data handling, ensuring that "task interfaces accurately reflect process context and data requirements."

Key human task tests include:

1. **Form Validation Testing** - Verifying that input fields enforce appropriate constraints while providing clear error messages, targeting what *(Grossmann, Schrefl, and Stumptner 2008, p. 124)* identify as a frequent failure point in process implementations.
2. **Task Assignment Verification** - Ensuring that tasks route to appropriate participants based on role mappings and organisational structure, using techniques advocated by *(Bozkurt, Harman, and Hassoun 2013, p. 186)*.
3. **UI Consistency Tests** - Validating consistent presentation across all human tasks, addressing what *(Yotyawilai and Suwannasart 2014, p. 69)* identify as a critical factor in user experience and task completion rates.

These tests specifically target the customer interaction forms (service request, membership registration), receptionist tasks (appointment scheduling, quotation management), and mechanic interaction points (repair documentation).

3.2 Service Task Verification

For service tasks that implement automated business logic, the testing approach employs both black-box and white-box techniques. Following *(García-Borgoñón et al. 2017, p. 53)*' recommendation, tests verify both "interface compliance and interior processing logic" to ensure comprehensive validation.

Key service task tests focus on:

1. **Membership Discount Calculation** - Verifying correct application of discount rules through parametrised tests with various membership scenarios.
2. **Service Scheduling Logic** - Testing appointment availability algorithms against both normal and edge-case scenarios.
3. **Diagnostic Classification Services** - Validating the categorisation of vehicle issues against expected repair requirements.

These tests employ techniques including branch coverage, data flow analysis, and boundary testing as recommended by *(Weber, Reichert, and Rinderle-Ma 2016, p. 168)* for complex service logic.

4 Integration Testing Strategy

Integration testing addresses the complex interactions between process participants and external systems, focussing on what *(Dustdar, Hoffmann, and Aalst 2016, p. 219)* characterise as the "coordination points where process fragility often manifests."

4.1 Message Flow Testing

Message flows between pools represent critical integration points in the automotive repair process. Following *(Kumar et al. 2015, p. 76)*' methodology, tests verify both message structure and behavioural impact, ensuring correct process coordination across organisational boundaries.

Key tests include:

1. **Customer Notification Testing -** Verifying that repair status updates correctly trigger customer notifications with appropriate content.
2. **Towing Service Integration -** Testing the bidirectional communication between receptionist tasks and external towing services.
3. **Mechanic-Receptionist Coordination -** Validating repair completion notifications and subsequent customer contact processes.

These tests specifically address what *(Gorton and Ehrlich 2017, p. 48)* identified as "potential friction points requiring careful information system mediation" in the original i* model.

4.2 Exception Flow Testing

Exception handling represents a critical aspect of process resilience. Following *(Domingos, Respício, and Martinho 2016, p. 302)*' approach, exception tests verify both detection and recovery mechanisms for anticipated failure modes.

Tests focus on scenarios including:

1. **Service Cancellation Processing -** Verifying correct compensation handling when customers cancel repairs.
2. **Customer Rejection of Quotes -** Testing alternative flows when repair estimates are declined.
3. **Unavailable Towing Services -** Validating fallback procedures when external services are unavailable.

These tests specifically target the exception paths identified in the operational BPMN model, ensuring comprehensive coverage of alternative scenarios.

5 System Testing Approach

System testing validates end-to-end process execution against business requirements, focussing on complete customer journeys and business outcomes.

5.1 Scenario-Based Testing

Following *(Kunze et al. 2015, p. 154)*' methodology, scenario tests execute complete process instances representing typical customer journeys, validating both functional correctness and performance characteristics.

Key scenarios include:

1. **Member Repair with Towing -** Testing the complete journey from breakdown to repair completion for a membership customer.
2. **Non-Member Walk-in Service -** Validating the experience for new customers seeking same-day diagnostics.

3. **Quote Approval and Payment Processing** - Testing the complete repair authorisation and billing workflow.

These scenarios directly correspond to the user stories derived from the i* model, ensuring validation of complete value delivery rather than merely technical execution.

5.2 Non-Functional Testing

Non-functional testing addresses quality attributes essential to process effectiveness. Following *(Dustdar, Hoffmann, and Aalst 2016, p. 133)*' approach, tests verify performance, security, and usability characteristics alongside functional correctness.

Key test areas include:

1. **Performance Under Load** - Verifying system responsiveness during peak service periods using techniques advocated by *(Giray and Tekinerdogan 2016, p. 89)*.
2. **Security and Access Control** - Testing proper enforcement of role-based permissions for sensitive customer and vehicle data.
3. **Process Monitoring Capabilities** - Validating that management dashboards accurately reflect process status and performance metrics.

These tests directly align with the softgoals identified in the i* model, particularly "Maintain Repair Quality" and "Maintain Professional Service," ensuring that quality attributes receive explicit verification.

6 Conclusion

The comprehensive testing strategy for the automotive repair service implementation establishes a systematic approach to verification and validation, ensuring that implemented processes faithfully represent stakeholder requirements while maintaining technical robustness. By establishing clear traceability between tests and the original i* goals, the approach bridges the gap between technical verification and business validation.

The layered testing architecture addresses the unique challenges of process-oriented applications, providing comprehensive coverage across human tasks, service integrations, and cross-boundary process flows. This holistic approach ensures that testing activities align with both functional correctness and the strategic objectives of the automotive repair business.

Through this structured and goal-aligned testing methodology, the implementation of the automotive repair service processes can progress to deployment with confidence that the system will effectively support both operational requirements and customer expectations, delivering value while maintaining quality and reliability.

References

Bozkurt, Mehmet, Mark Harman, and Youssef Hassoun (2013). "Testing and Verification in Service-Oriented Architecture: A Survey". In: *Software Testing, Verification and Reliability* 23.4, pp. 121–153.

Domingos, Dulce, Ana Respício, and Ricardo Martinho (2016). "Using Resource Reliability in BPMN Processes". In: *Procedia Computer Science* 100, pp. 298–307.

Dustdar, Schahram, Thomas Hoffmann, and Wil van der Aalst (2016). "Business Process Management Workshops". In: *Lecture Notes in Business Information Processing* 256, pp. 128–139.

García-Borgoñón, Laura et al. (2017). "Software Process Modeling Languages: A Systematic Literature Review". In: *Information and Software Technology* 91, pp. 86–103.

Giray, Görkem and Bedir Tekinerdogan (2016). "Systematic Approach for Testing BPMN Processes". In: *Information and Software Technology* 77, pp. 212–225.

Gorton, I. and K. Ehrlich (2017). "Technical and socio-technical challenges of implementing requirements traceability". In: *Journal of Software: Evolution and Process* 29.8, pp. 45–58.

Grossmann, Georg, Michael Schrefl, and Markus Stumptner (2008). "Verification of Business Process Integration: A Model Checking Approach". In: *Information Systems* 33.6, pp. 346–369.

Kumar, Akhil et al. (2015). "Integrated Testing of BPMN Processes: Early Results and Future Directions". In: *Journal of Information Technology Theory and Application* 16.4, pp. 71–93.

Kunze, Matthias et al. (2015). "Towards Understanding Process Modeling The Case of the BPM Academic Initiative". In: *Business Process Management Workshops* 95, pp. 145–168.

Martinho, Ricardo, Dulce Domingos, and João Varajão (2015). "CF4BPMN: A BPMN Extension for Controlled Flexibility in Business Processes". In: *Procedia Computer Science* 64, pp. 213–221.

Weber, Barbara, Manfred Reichert, and Stefanie Rinderle-Ma (2016). "Change Patterns and Change Support Features Enhancing Flexibility in Process-Aware Information Systems". In: *Data & Knowledge Engineering* 66.3, pp. 79–104.

Yotyawilai, Wararat and Taratip Suwannasart (2014). "A Method for BPMN Model Quality Improvement". In: *International Journal of Computer and Information Technology* 3.5, pp. 65–73.