End-to-End Test Automation Framework for Web Portals in Digital P&C Insurance Platforms

Kiran Babu Boddapati

Acharya Nagarjuna University, Andhra Pradesh, India

shivakiran02@gmail.com

Abstract

In digital property and casualty insurance platforms, end-to-end test automation has assumed a strategic role as customer journeys, broker interactions, underwriting workflows, policy servicing, billing touchpoints, and document delivery increasingly depend on distributed web-based services. Portal defects in such environments can propagate far beyond the user interface, affecting rating accuracy, transaction integrity, compliance artifacts, and customer experience. This review examines academic literature relevant to the design of an end-to-end test automation framework for digital P&C insurance web portals. Key topics are web application testing, model-based testing, regression optimization, service virtualization, continuous integration and delivery, process-conscious quality assurance, test architecture maintainability, and the implications of insurance digitalization to test architecture. The literature suggests that durable automation depends on layered design rather than script accumulation; stable abstractions, domain-aware scenario modeling, risk-based prioritization, robust data management, and tight feedback from test execution are all essential. There are still gaps in insurance-specific empirical data, cross-channel lifecycle coverage, explainable risk-based orchestration, and frameworks to evaluate and relate automation metrics to regulated business results. The field is important because portal quality increasingly shapes operational trust, release velocity, and competitive responsiveness within digital insurance ecosystems.

Keywords*:* digital insurance; end-to-end testing; property and casualty insurance; test automation; web portals.

1. Introduction

Web portals have become central interaction channels in digital property and casualty insurance. Customer self-service, agent and broker distribution, underwriting referral, policy servicing, billing support, first notice of loss, document access, and cross-sell journeys are increasingly mediated through browser-based interfaces and interconnected with policy administration, rating, billing, claims, identity, and analytics services. This growth has changed the role of quality assurance. Portal testing can no longer be treated as a narrow user-interface activity because of the visible actions, which in most instances cause the state change involving multiple core systems. A quote request can invoke product set up, qualification logic, third-party enrichment, pricing services, fraud signs, document creation, and notification processes. The web application testing research has been active in arguing that the web systems are distinct in their complexity due to the dynamic content, structure of navigation and asynchronous interaction and heterogeneous conditions of deployment [1]. In the modern insurance environments that is magnified by the productization and sphere-specific regulations.

Automation has received increasing emphasis as release frequency has risen and digital channels have become central to business performance. End-to-end verification is too slow and too brittle to be practiced manually by many organizations due to continuous delivery practices, microservice decomposition, API-driven integration, and cloud deployment. Design problems are however brought about by automation itself. Empirical studies of automated testing and software quality suggest that the value of automation depends on maintainability, architectural design, and integration with delivery workflows, rather than on the sheer number of scripts [2]. Some investigations focusing on related studies about regression testing have revealed that partial reruns of large suites can easily be unfeasible and selection and prioritization are necessary due to time constraint [3]. Browser-facing insurance portals are particularly vulnerable to this tension as user-facing releases occur quite often, whereas the consequences of bugs may be spread across controlled documentation, premium computations, commissions, and even communications with customers. End-to-end framework should thus be helpful in terms of speed and control.

This topic sits at the intersection of software engineering, information systems, business process management, and insurance digitalization. From a software-engineering perspective, the topic involves test design, automation architecture, state management, mitigation of flaky tests, service virtualization, and environment control. From a process perspective, portal journeys extend beyond individual pages and span quotation, issuance, endorsement, payment, renewal, and service events. Insurance digitalization research shows that carriers are modernizing platforms that connect products, operations, and customer interfaces while enabling more data-intensive services [4]. The change poses new assurance needs as portal behaviour should not exceed what legacy core systems have to provide, what can be configured to meet customer requirements, what jurisdictional demands, and what omnichannel demands. A browser automation may be concealing more underlying orchestration risks and business-critical interactions of services may be ignored by an interface-based automation program.

A number of issues that are unsolved stand out. One major challenge concerns the appropriate level of abstraction. In most automation programs, the delicacy of UI scripts continues to be a significant problem because these scripts fail to work once the underlying business behaviour changes, despite no changes in the presentation. The second problem is related to scenario realism. Automation often verifies happy paths while overlooking endorsements, renewal exceptions, billing delinquency, referral states, accessibility issues, and role-based agent workflows. The third issue is connected with the management of the test data and environment. Insurance transactions are stateful, date sensitive, and product specific and hence controlled data generation and isolation of services are not easily attainable without deterministic end-to-end testing of the service. A fourth problem is related to observability. A failed browser step could be due to the portal layer, a service on the back-end or a dependency on a third-party, or out-of-date environmental data. Research on service-oriented testing and continuous delivery suggests that scalable assurance depends on multilayer feedback and component-aware isolation rather than black-box execution alone [5]. A final problem is the limited availability of insurance-specific empirical evidence. Studies of web testing and automated frameworks are not nearly as scarce, but studies on the details of automating digital P&C portal are limited in the literature.

This review aims at reviewing how the academic materials could be utilized in designing an end-to-end test automation framework of web portals in digital P&C insurance platforms. The entire review is a compilation of findings on the web application testing, regression optimization, model-based testing, service virtualization, continuous integration, process-conscious quality assurance, and insurance digitalization. Literature, conceptual and methodological strategies, report on the results of the studies, research gaps and future research directions will be discussed in the following sections to develop automation frameworks that are resilient, domain-driven and meaningful in digital P&C settings.

2. Literature Review

The literature relevant to end-to-end portal automation in digital P&C insurance spans multiple intersecting streams rather than a single established field. The former stream is related to testing a web application. Early work showed that web systems challenge traditional software-testing assumptions because of navigation logic, client-side session states, dynamic content, browser heterogeneity, and asynchronous behaviour [1], [6]. Later research on automated web testing supported this opinion by demonstrating that the tasks of navigation modelling, state abstraction and event-sequence generation are required to achieve sufficient behavioural coverage [7]. Research on rich internet applications and systems with heavy use of JavaScript further complicated the matter, showing that client-side logic, asynchronous rendering can cause instability to automated scripts, and be the source of error [8]. This stream directly applies to insurance portals since new policy and broker interfaces are based more and more on single-page designs, API-based information exchange, and conditional user experiences.

A second stream is on model-based, search-based and combinatorial approaches to automation. It has been demonstrated in the literature of model-based testing that systematic derivation of test paths can be enhanced by formal or semi-formal behavioral models in particular in systems with interaction-intensive behaviour that have many states and transitions [9]. Similar studies in combinatorial testing showed that commonly failures are caused by the interaction of a comparatively few factors and interaction coverage is an efficient technique to reveal defects that are hard to detect through other means [10]. This line of reasoning was further extended by search-based methods which modeled test generation, selection or prioritization as a cost and coverage constrained optimization problem [11]. These techniques are very applicable to digital P&C portal as user journeys frequently have a branch based on line of business, state or province, coverage choice, channel role, payment plan, previous policy status, and underwriting reaction. Manually designed test artifacts seldom scale effectively in such environments.

Third stream is regression, maintainability and automation economics. The common problem with reruns of full-suite tests that has been known since survey research on regression testing is that it gets more and more costly to perform as systems change [3]. Experimental research of test case prioritization demonstrated that intelligent ordering is able to enhance early faulting when the execution budgets are limited [12]. Optimization of regression suites in industrial research continued to show that regression

suites that live long are prone to acquire redundancy, volatile assets and increasing maintenance cost unless actively maintained [13]. Individual research on automating tests in industry highlighted that maintainability, the modularity of design and compatibility with the architectural limits will be the determinants of long-term automation programs [2], [14]. This direction of work is particularly significant to insurance portals since digital carriers frequently have numerous branded experiences, product variations, and channel-specific flows atop of common back-end facilities. The automation of portals may be forgotten without a structured design of disciplined frameworks that will not degenerate into a fragmented set of fragile scripts, bound to the temporary interface specifics.

The fourth stream revolves around the issue of service-based and perpetual quality delivery. Studies of service-oriented system testing proposed that end-to-end confidence requires verification of service contracts, as well as orchestration behaviour particularly in business transactions involving loosely coupled components [15]. Virtualization of services studies indicated that, simulation of unavailable or unstable dependencies can enhance repeatability, lessen the environment bottlenecks and provide the opportunity to test partially integrated system earlier [16]. Ongoing integration and delivery research contributed that the best way to use automated testing is to arrange it into a layer cake with the quick feedback at the lower levels and ever-expanding verification at the top levels [17]. This observation is especially important in portal-heavy insurance applications: not all scenarios should be validated at the browser layer. Assured calculation of ratings, document preparation, rules in commissioning, identity services and integrations of payment might need differentiated assurance mechanisms. An effective end-to-end structure must thus be layered, selective and be closely coupled with delivery processes.

A fifth stream deals with process awareness, observability and digital transformation of insurance. The studies on process-mining and conformance-checking showed that event traces have the ability to show how actual business processes are not following the desired process models [18]. The predictive monitoring literature demonstrated that past traces are applicable in predicting the outcome of paths or exception of the digital processes that may occur in the process [19]. The studies of insurance digitalization, in turn, reported the ways in which carrier operating models are transforming by modernizing platforms, connecting with ecosystems, and digitally mediating value creation [4], [20]. Such contributions are significant as the quality of portals cannot be evaluated on a page or test-script-level only. Transactional processes that have portal behaviour include quote-bind-issue, policy change, cancellation, reinstatement, payment and renewal. Process-sensitive knowledge can thus be used to spot the operationally important portal journeys, the most common variants of path, and the sequences of transactions that are more interesting to deepen the regression.

Table 1 will provide the representative studies of reference [6] onwards. The table indicates that the knowledge base is spread among the web testing, automation architecture, service based validation, process based analysis and digital insurance scholarship.

Table 1. Summary of key findings

Ref	Focus	Key Findings
[6]	Taxonomy of web application testing	Identified navigation, state, session, and dynamic content as distinctive sources of web-testing complexity requiring specialized strategies.
[7]	Automated functional web testing techniques	Showed that event-sequence generation and navigation modelling improve behavioral coverage beyond manually scripted page checks.
[8]	Testing of rich internet and client-heavy web systems	Reported that asynchronous client logic, dynamic DOM updates, and script-intensive interfaces increase automation fragility and oracle difficulty.
[9]	Model-based testing approaches	Clarified that explicit state and transition modelling improves systematic path derivation in interaction-rich applications.
[10]	Combinatorial testing	Demonstrated that interaction-focused design exposes defects driven by combinations of configuration and input factors.
[11]	Search-based software test data generation	Established that optimization methods can improve test generation and coverage under constrained resources.
[12]	Test case prioritization studies	Found that informed ordering improves early fault revelation compared with untreated regression execution.
[13]	Industrial regression-suite optimization	Reported that historical execution and cost-aware filtering can improve suite efficiency in large evolving systems.
[14]	Industrial experiences with automated acceptance and GUI testing	Highlighted maintainability, abstraction, and toolchain integration as major determinants of sustainable automation value.
[15]	Service-oriented architecture testing	Showed that contract, interaction, and orchestration testing are all necessary for reliable composite-business transactions.
[16]	Service virtualization for test environments	Found that virtualized dependencies improve repeatability, availability, and early-stage test execution in integration-heavy systems.
[17]	Continuous delivery quality practices	Argued that layered automated pipelines with rapid feedback outperform monolithic late-stage validation.
[18]	Process conformance checking	Demonstrated that event-log replay can identify deviations between intended and actual transactional behaviour.
[19]	Predictive monitoring of business processes	Reported that trace-based prediction can identify high-risk process paths, though performance depends on data quality and feature design.
[20]	Digitalization in insurance operations	Described how digital transformation expands platform interdependence and raises pressure for dependable digital-channel quality.

There are a number of common restraints that are presently evident in these studies. To begin with, a large part of the literature on web-testing is technology-focused and lacks domain specific advice towards regulated insurance transactions. Second, the maintainability and the payback of automation is frequently discussed in the industrial automation literature without providing the policy or broker workflow domain models. Third, the literature that is process-aware is robust in the path visibility but tends to short-cut to prescribing browser-automation architecture. Fourth, the studies on insurance digitalization talk about transformation and change in operating models and pay relatively minor attention to specific quality-engineering frameworks. The literature thus promotes an integrative design as opposed to direct implementation of any one of the methods families.

The other interesting trend is with respect to portal coverage and business coverage. A portal may exercise page elements successfully and still fail in business-critical downstream effects such as policy issuance, pricing accuracy, document generation, commission handling, or billing synchronization. Studies of service-based and process-based assurance propose that end-to-end reliability is a characteristic of the chain of transaction instead of the user interface per se [15], [18]. This is most applicable to digital P&C insurance, with the apparent portal being just a subset of a more comprehensive transactional system. A framework of automation will thus need to tie the interactions of browsers to business-state validation among services, documents and system records.

3. Methodology

A layered assurance perspective offers a useful conceptual framework for end-to-end portal automation in digital P&C insurance. The highest level must denote business journeys that consist of quote, bind, issue, endorse, cancel, reinstate, pay, renew and service request. Beneath that layer of journey are portal elements, API, service interrelations, data conditions and external relations. The literature indicates that a high degree of automation is possible when these layers are coincident and are not overlapping to produce one script abstraction. Web-testing studies concentrate on state and navigation modelling [6], [7], the use of models in testing concentrates on deriving scenarios formally [9], regression studies concentrate on prioritization in the presence of cost constraints [3], and [12], and the service-oriented testing literature concentrates on the contract- and coordination-aware verification [15]. The studies of insurance digitalization present the field argumentation, showing that at this point the interdependence of the interactions through the digital channels is highly embedded in the carrier operating models [4], [20].

The literature has a number of approaches that are methodological. The first methodological family is script-centric browser automation, in which user actions are executed directly through recorded or coded interactions with page objects. It is still a popular way to go as it is intuitively simple and can give substantial preliminary coverage. However, the literature has repeatedly warned that script-based automation is fragile in environments characterized by interface variability, asynchronous behaviour, and business semantics not encoded at the script level [8], [14]. Script-centric design is especially prone in insurance portals, where the necessity to rebrand and adapt product pages, and react to a shift in responsive layout may necessitate extensive maintenance even without a change in policy behaviour in the underlying policy.

The second methodological family is domain-layer and model-based automation. It is here that the user journeys are defined in terms of states, transitions, decision rules or domain actions and browser steps are formed or organized beneath the abstractions [9]. Digital insurance can benefit in a number of ways with this approach. Quotation variants, role-based routes, product choice and policy-service transactions can be represented using domain flows, as opposed to low-level locator operations. Combinatorial design can then be applied selectively to high-risk interaction zones [10]. Primary vulnerability is maintenance of models. Where the product rules or page flows are changing rapidly, then the abstractions must be highly regulated so as not to separate the framework to real behaviour.

The third family is concerned with service-conscious end-to-end automation. Research conducted on service-oriented systems suggests that the full trust of the business transactions requires transparency of the service contracts, message exchange and orchestration logic instead of the interface actions [15]. Service virtualization brings a pragmatic separation of the volatile or costly dependencies and controlled variation of behaviour [16]. This perspective is reinforced by continuous-delivery literature, which argues that automated assurance should be distributed across a layered quality pipeline rather than concentrated solely at the browser stage [17]. This would mean that portal testing as realism with customers (in the form of browser journeys), API tests to guarantee deterministic validation, and virtual dependencies to guarantee controllability, and environment diagnostics to guarantee fast fault localization should be a part of an end-to-end framework in the instance of portal testing in insurance.

The fourth family is around process-conscious, feedback-based automation. The process-mining research reveals that the footprints of operations can reveal both normal and abnormal paths and anomalies of the planned and actual business behaviour [18]. Predictive monitoring is based on that and estimates the variants of the paths that are riskiest or most likely to introduce delays [19]. When applied to automation of portals, these techniques can be used to tell which scenarios are worth more emphasis in regression, and which variants of paths are sufficiently common to warrant a special set of automation resources. This can be particularly useful when it comes to digital P&C insurance since release teams have a huge amount of potential paths to take but very little execution time. Process feedback can help distinguish common customer or agent journeys from edge cases while still supporting explicit, high-severity exception paths.

Figure 1 has a conceptual framework which is grounded on the literature. The framework aligns business journeys, domain abstractions, execution layers, dependency control and feedback learning in a single framework. This is one of the key lessons of the studied literature: effective end-to-end automation of portals is not an issue of the choice of tools, but rather of design that still maintains business intent and deals with technical changeability.

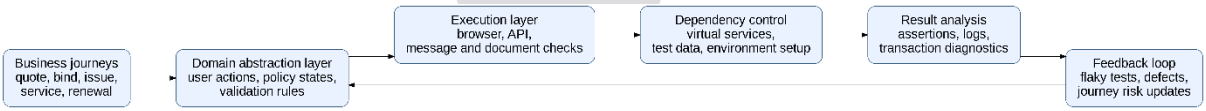

Figure 1. Conceptual framework for end-to-end portal automation in digital P&C insurance

Domain abstraction lies in the middle, as shown in the figure, between business journeys and the execution mechanics. This position is significant as it distances long-term business purpose and short-term presentation information, which is repeatedly hinted at by maintainability research in automated testing [2], [14]. The feedback loop also emphasizes that the quality of the frameworks must evolve with time in response to history of execution, defects and analysis of flaky tests and not be fixed.

The other methodological implication is that of the test oracle problem. Automation of browsers will often be subject to issues of meaningful validation particularly when portal success involves side effects at the downstream; policy generation, invoice posting, form generation or notification delivery. The literature suggests that robust frameworks combine UI assertions with API-, document-, or event-level checks

whenever governance and environmental constraints allow [15], [17]. Such multilayer oracles prove particularly valuable in insurance situations as an apparent success in a confirmation page may conceal the failure to continue transactions or controlled unresponsive output. In general, literature suggests the domain-sensitive, layered, service-conscious and feedback-oriented portal automation. It is against this background that the discussion of the results will be discussed below.

4. Discussion

The reviewed studies suggest that end-to-end automation should be implemented as an evolving framework rather than as a simple library of UI scripts. A body of research of web-testing has demonstrated that the complexity of navigation, asynchronous rendering and dynamic state changes decrease the resilience of naive browser based solutions [6], [8]. The research on automation in industries confirms this trend and discovered the maintainability, modularity and the quality of abstraction to be the determinants of the long term value [14]. This is particularly so when it comes to digital P&C insurance. The portal journeys are identity, eligibility, pre fill information, product inquiries, rating calls, payment interactions, document retrieval and downstream core-system writes. A thin page locator-bound script layer is not likely to be kept by any branding, product additions or responsive-layout additions. The literature therefore supports a transition from element-centric design toward journey- and domain-centric design.

The second outcome that is reported is the significance of overlaying and discriminating end-to-end scope. The studies on continuous-delivery and service-oriented testing demonstrate that not all the business checks can be delivered via a browser, although the aim is to provide end-to-end confidence [15], [17]. Speedy component tests, API tests, contract tests, and dependency tests on virtualized dependencies may decrease the load on browser suites and enhance fault localization. This observation is important to insurance portals since most of the failures that are caused by the UI are in fact due to back-end data failures, orchestration failures or dependency failures by third parties. An architected framework will thus utilize the browser layer to authenticate user journeys that are important and cross-system interactions and project deterministic business checks down to more stable layers wherever feasible. This remains end-to-end in terms of business coverage, even when not every verification step is executed through the browser.

The third consequence is that of the worth of model-based and combinatorial structuring. Research on model-based testing has suggested that explicit state and transition representation enhances systematically exploring applications with heavy interaction [9]. Combination testing studies also add to this fact that errors tend to be caused by interactions in a small number of dimensions and therefore coverage of interactions in a small number of dimensions can be successfully realized [10]. In the case of digital P&C portals, this is very pertinent since the line of products, region, coverage bundle, distribution channel, account state, payment option, referral status and previous policy determine variability in pathways. These pressure combs are very likely to break manual enumeration of scenarios. According to research reported,

the scalability of frameworks can be increased as journey models characterize the major flow structure and selective use of combinatorial logic on decision points of high risk instead of the path space in general.

The fourth outcome is related to maintainability economics. The literature on industrial regression and automation suggests that automated suites may become worthless with time due to redundancy, unstable tests, out-of-date assumptions and sluggish refactoring [13], [14]. This degradation is especially visible in portal-heavy systems where front-end changes occur frequently. A framework which fails to define shared components, locator abstraction, synchronization strategy, test data lifecycle and environment diagnostics is likely to accrue flakiness. Flaky tests are not some small inconvenience; the empirical and industrial literature depict flakiness as a grave challenge to a belief in automation since uncertain failures retard releases and promote the rejection of results. Despite the different terminology used in the literature, the same thing comes out clearly, i.e., automation structures demand engineering governance, rather than the performance capacity.

The strategic role of service virtualization and controlled dependencies is one of the fifth outcomes. Research on service virtualization shows that simulated dependencies can improve repeatability and reduce environmental bottlenecks, especially in integration-heavy settings [16]. The instability or cost of shared test environments can be influenced by external motor-vehicle records, property-data vendors, payment gateways, identity providers, and fraud services, document engines, and communication services. Virtualization does not remove the actuality of the real integration testing, yet the papers indicate that it may aid a more rational test pyramid by putting the completely incorporated browser situations to the journeys where the real dependency behaviour is of greatest essence. This is in favour of speed and determinism.

Large methodological approaches that are reported in the literature are compared in Table 2. The comparison shows a recurring pattern: strengths are distributed across method families, but weaknesses become pronounced when any one family is treated as a stand-alone solution.

Table 2. Method comparison

Ref	Method	Strengths	Limitations
[6]	Web-testing strategy based on navigation and state analysis	Improves understanding of web-specific behaviour and supports broader behavioral coverage	Can remain too generic for complex regulated business journeys without domain extension
[8]	Automation for rich internet and asynchronous interfaces	Addresses client-side dynamics, DOM volatility, and script-heavy interaction patterns	Increased oracle difficulty and synchronization complexity can still cause brittle execution
[9]	Model-based testing	Supports systematic derivation of user journeys from states and transitions	Model upkeep can become expensive when product and portal logic change rapidly
[10]	Combinatorial testing	Exposes faults driven by interactions among input and configuration factors	Requires disciplined constraint handling to avoid unrealistic or low-value combinations

[11]	Search-based test generation and optimization	Helps allocate limited execution resources toward high-value paths or data configurations	Optimization objectives may be weak if business risk signals are incomplete
[12]	Regression test prioritization	Improves early fault detection under tight execution windows	Prioritization does not by itself solve flaky-test or data-management problems
[13]	Industrial suite optimization	Reduces redundant execution and highlights low-value tests in evolving environments	Historical data may underrepresent rare but severe insurance scenarios
[14]	Industrial automated GUI and acceptance testing practice	Emphasizes maintainability, modularity, and framework governance for long-term utility	Practice-focused evidence can be context-bound and less transferable without domain tailoring
[15]	Service-oriented testing	Validates business transactions across service boundaries and contracts	Full orchestration coverage may require substantial tooling and environment support
[16]	Service virtualization	Improves repeatability, isolation, and early-stage testing when real dependencies are unstable	Virtual behaviour can drift from production if maintenance discipline is weak

Another trend in the literature that is reported is that of observability and diagnosis. End-to-end browser failures are costly when the only evidence available is a screenshot or a failed interaction step. Recent-delivery research and service-based testing suggests that properly designed frameworks present identifiers of transactions, service logs, event traces, and artifact status so as to facilitate localization of failures quickly [15], [17]. This plays a crucial role in insurance portals where a quote-bind trip can fail due to a portal validation error, a premium-service failure, a policy-admin rule error, a document-engine error or a third-party call failure. Cases where the delivery teams triage slowly and vaguely occur due to only capturing the UI symptom.

The table 3 gives a compilation of typical results that have been reported in the literature. The table shows that positive results are reported across multiple dimensions, including coverage, early fault detection, maintainability, repeatability, and diagnostic precision. This heterogeneity is important because an automation framework within a portal must be considered based on its usefulness to the operations and not just on the amount of execution.

Table 3. Results comparison

Ref	System	Metric	Outcome
[7]	Automated functional web application environment	Behavioral coverage quality	Event-driven automation improved traversal of interactive web behaviour beyond simple scripted checks
[9]	Model-based testing setting	Scenario derivation efficiency	Structured models improved systematic path generation for interaction-rich applications

[10]	Interaction-heavy software environment	Fault exposure through combination coverage	Targeted interaction coverage revealed defects not detected by simpler input selection
[12]	Regression testing context	Early fault detection	Prioritized execution improved defect discovery timing compared with untreated order
[13]	Industrial regression suite	Execution cost efficiency	History-aware optimization reduced redundant execution in large evolving suites
[14]	Industrial GUI and acceptance automation program	Maintainability and trust	Higher abstraction and better framework discipline improved durability of automation assets
[15]	Service-oriented business system	Cross-service validation quality	Contract and orchestration testing improved confidence in composite transactions
[16]	Integration-heavy enterprise test environment	Repeatability and dependency availability	Virtualized services improved environment stability and earlier testability
[18]	Process-aware information system	Conformance visibility	Event-log replay exposed path deviations relevant to transactional validation
[19]	Predictive process monitoring environment	Risk-focused process insight	Trace-based analytics highlighted high-risk journey variants for focused quality attention

The coded trend graph presented in figure 2 provides the areas of focus on evaluation that are most important according to the studies under analysis. The graph suggests that maintainability and fault detection receive much greater emphasis than insurance-specific journey realism or regulated-document validation. This unbalance can be used to demonstrate why organizations can have huge, automated suites and yet have no confidence in the portal releases that can impact policy servicing and compliance output.

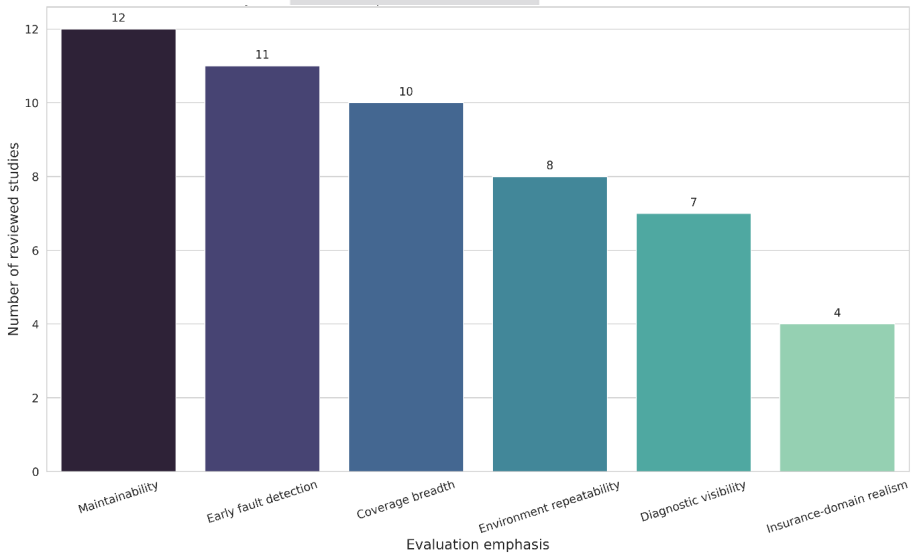

Figure 2. Primary evaluation emphases in the reviewed literature

Figure 2 indicates that the research base is comparatively mature on technical automation issues, but much thinner on domain-specific assurance outcomes. To digital P&C insurance, this is important since a portal

release may be physically acceptable and yet generate the inaccurate premium, missing forms or broken transactions between the policy and services.

The associations between the complexity of business journeys, business-journey framework abstraction, dependency control, observability, and release confidence are mapped in figure 3. The diagram indicates that release confidence does not come as a result of browser coverage. The interaction between scenario design, depth of diagnostic and controllability of dependencies shapes it.

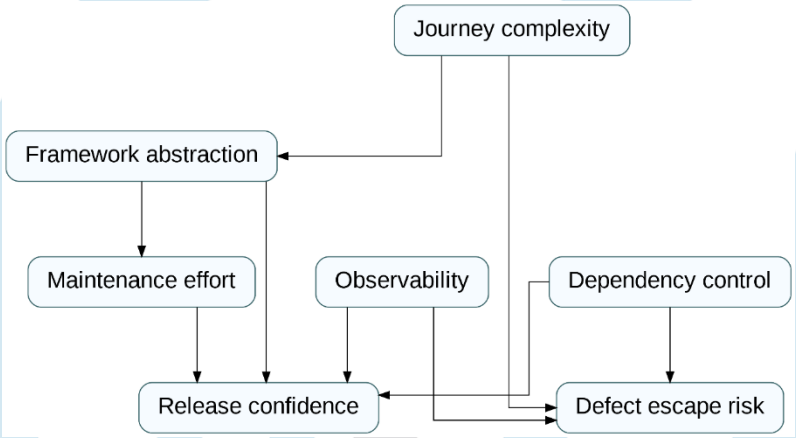

Figure 3. Relationship diagram among major variables in portal automation frameworks

The diagram points out important practical lesson of the literature. Abstraction can be used to increase confidence in release, but this should be in accordance with the complexity of the journey, and the abstraction should be well supported with good observability. With a seemingly comprehensive set of browser suites, there could still be a high risk of defect escape due to low dependency control or low diagnostics.

Figure 4 shows a model of integrated framework of automation of digital P&C portal. Journey design, domain abstractions, layered execution, dependency simulation, result analytics, and feedback-based curation come together in the model. This framework indicates the most powerful message that is repeated throughout the literature: sustainable end-to-end automation is a quality system designed, not just a set of tools or scripts.

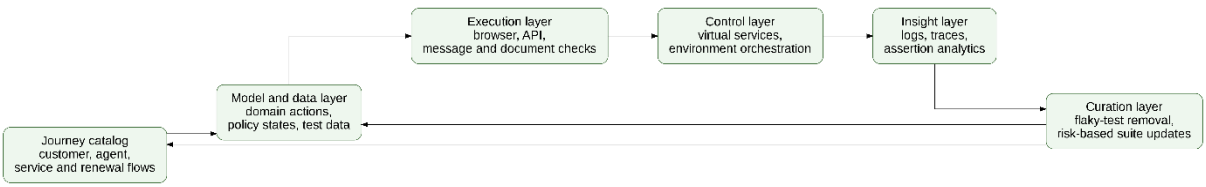

Figure 4. Integrated model for end-to-end test automation of web portals in digital P&C insurance platforms

The integrated model treats curation as an ongoing engineering process rather than as an occasional cleanup activity. This coincides with the industrial findings indicating that automation falls apart unless there is a deliberate attempt to sustain, assign more weight and get rid of the assets that are of low value or the one that is volatile [13], [14]. The other important argument that the model puts forward is the role of a journey

catalogue since the business flows must decide on the coverage of portals, on the basis of operationally and commercially important business flows.

Taken together, the reviewed literature suggests that strong portal automation frameworks depend on six related capabilities, namely domain-aware journey modelling, abstraction of the underlying page mechanics, layered validation outside of the browser, controlled dependency behaviour, rich diagnostic observability and continuous suite curation. The weaknesses of the semantics of insurance domains, portal testing as merely a front-end test, and automation in terms of number of scripts as compared to business assurance remain the primary sources of weakness. This is the main frontier to the work in the future.

5. Future Directions

One important future direction is the development of insurance-specific domain models for portal automation, especially reusable journey models for core policy-lifecycle transactions. The existing literature offers effective strategies to web testing, model-based design, regression prioritization and service virtualization, yet it does not directly discuss the journeys of digital P&C portal. The research should in the future develop a reusable quote-bind-issue, endorsement, billing-service interaction, document retrieval, cancellation, reinstatement and renewal of personal and commercial lines journey ontology. These domain models have the potential to enhance portability of structure design yet maintain business realism.

Another important direction is responsive and risk-conscious test orchestration. The practice of scheduling suites of browsers is usually in a fixed manner, although the release risk of a given product change is uneven, instability in dependencies, journeys that are traffic sensitive, and regulatory sensitivity. Future research should examine orchestration models that combine code change, service impact, production-journey frequency, and recent defect history to prioritize end-to-end execution more effectively. This would bring portal assurance in line with real business exposure as opposed to uniformity of all situations.

The third direction is based on observability-based automation. This renders browser scripts costly because of the sluggish failure diagnosis. Future frameworks would be beneficial to be more closely linked with distributed tracing, event-telemetry, document-validation analytics, and transaction-level identifiers, which would tie portal actions with the results of downstream systems. Research that integrates test automation with operational observability could substantially improve debugging efficiency and the reliability of business-state verification.

Fourth direction pertains to governance, accessibility and cross channel assurance. The usage of digital P&C portals is growing towards increasingly customer, agent, broker and service roles and mobile and API channels. Future research should examine how end-to-end frameworks can incorporate accessibility verification, cross-channel journey consistency, and controlled-output validation without becoming operationally unmanageable. Empirical longitudinal data on carrier settings would come in especially handy in demonstrating the effect of design of a framework on speed to release, defect escape and economics of maintenance with time.

6. Conclusion

End-to-end test automation for digital P&C insurance portals should be understood as a framework-design problem rather than merely as a scripting exercise. The reviewed literature shows that the complexity of web-testing, regression economics, service interactions, process visibility and insurance digitalization all converge in this field. Durable automation cannot be achieved through browser coverage alone. It is reliant on architecture that maintains business paths, stabilizes technical abstractions and facilitates trustworthy validation through UI, API, service, and artifact levels.

A number of insights can be identified. The model-based and combinatorial methods can be used to manage journey variation. The regression prioritization and suite optimization make the implementation more feasible when faced with the pressure of delivery. The repeatability and fault isolation are enhanced with the help of service virtualization and layered pipelines. Process-behaviour feedback can help identify high-risk path variants and support more intelligent coverage decisions. One more of the major aspects of long-term value is maintainability, especially in the environments that are heavy in portals, where the interface and product evolution rates are high.

Important gaps still remain. Insurance-specific empirical research remains limited, evaluation of regulated-output accuracy is still underdeveloped, and cross-channel portal assurance requires further study. The literature on longitudinal studies on the relationship between framework architecture and business outcomes in digital insurance business such as release stability, service continuity, and defect escape is also insufficient.

The limitations in the research base lean towards a coherent design agenda. The next generations will involve domain aware catalogues of journeys, multi-layered execution, simulated dependency control, rich observability and risk-based and execution-driven continuous curation. In digital P&C insurance, portal automation is evolving into a rigorous quality-engineering discipline for assuring transaction trust in modern carrier environments.

References

[1] Balsam, S., & Mishra, D. (2025). Web application testing—Challenges and opportunities. *Journal of Systems and Software, 219*, Article 112186.

[2] Wiklund, K., Eldh, S., Sundmark, D., & Lundqvist, K. (2017). Impediments for software test automation: A systematic literature review. *Software Testing, Verification and Reliability, 27*(8), Article e1639.

[3] Khatibsyarbini, M., Isa, M. A., Jawawi, D. N. A., & Tumeng, R. (2018). Test case prioritization approaches in regression testing: A systematic literature review. *Information and Software Technology, 93*, 74–93.

[4] Eling, M., & Lehmann, M. (2018). The impact of digitalization on the insurance value chain and the insurability of risks. *The Geneva Papers on Risk and Insurance - Issues and Practice, 43*(3), 359–396.

[5] Mårtensson, T., Ancher, G., & Ståhl, D. (2023). Test environments for large-scale software systems—An industrial study of intrinsic and extrinsic success factors. *Software Testing, Verification and Reliability, 33*(3), Article e1839.

[6] Kertusha, I., Assres, G., Duman, O., & Arcuri, A. (2026). A survey on web testing: On the rise of AI and applications in industry. *Science of Computer Programming, 253*, Article 103473.

[7] Sunman, N., Soydan, Y., & Sözer, H. (2022). Automated Web application testing driven by pre-recorded test cases. *Journal of Systems and Software, 193*, Article 111441.

[8] Parry, O., Kapfhammer, G. M., Hilton, M., & McMinn, P. (2022). A survey of flaky tests. *ACM Transactions on Software Engineering and Methodology, 31*(1), Article 17.

[9] Ahmad, T., Iqbal, J., Ashraf, A., Truscan, D., & Porres, I. (2019). Model-based testing using UML activity diagrams: A systematic mapping study. *Computer Science Review, 33*, 98–112.

[10] Wu, H., Nie, C., Petke, J., Jia, Y., & Harman, M. (2020). An empirical comparison of combinatorial testing, random testing and adaptive random testing. *IEEE Transactions on Software Engineering, 46*(3), 302–320.

[11] Balera, J. M., & de Santiago Júnior, V. A. (2019). A systematic mapping addressing hyper-heuristics within search-based software testing. *Information and Software Technology, 114*, 176–189.

[12] Hao, D., Zhang, L., Zang, L., Wang, Y., Wu, X., & Xie, T. (2016). To be optimal or not in test-case prioritization. *IEEE Transactions on Software Engineering, 42*(5), 490–505.

[13] Dobslaw, F., Wan, R., & Hao, Y. (2023). Generic and industrial scale many-criteria regression test selection. *Journal of Systems and Software, 205*, Article 111802.

[14] Di Meglio, S., Starace, L. L. L., Pontillo, V., Opdebeeck, R., De Roover, C., & Di Martino, S. (2026). Investigating the adoption and maintenance of web GUI testing: Insights from GitHub repositories. *Information and Software Technology, 189*, Article 107928.

[15] Endo, A. T., & Simão, A. da S. (2019). Event tree algorithms to generate test sequences for composite Web services. *Software Testing, Verification and Reliability, 29*(3), Article e1637.

[16] Benkhelifa, E., Bani-Hani, A., Welsh, T., Mthunzi, S., & Ghedira Guegan, C. (2019). Virtual environments testing as a cloud service: A methodology for protecting and securing virtual infrastructures. *IEEE Access, 7*, 108660–108676.

[17] Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access, 5*, 3909–3943.

[18] Zaman, R., Hassani, M., & van Dongen, B. F. (2023). Conformance checking of process event streams with constraints on data retention. *Information Systems, 117*, Article 102228.

[19] Teinemaa, I., Dumas, M., La Rosa, M., & Maggi, F. M. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data, 13*(2), Article 17.

[20] Braun, A., & Jia, R. (2025). InsurTech: Digital technologies in insurance. *The Geneva Papers on Risk and Insurance - Issues and Practice, 50*, 1–7.