Executive Summary

Data-Driven Testing (DDT) is crucial for complex insurance software systems like Guidewire InsuranceSuite, yet traditional approaches often fall short. Kimputing's CenterTest platform introduces a revolutionary patent-pending DDT approach, transforming automated testing into a strategic asset for insurers. At its core, CenterTest innovates through the use of object-oriented (OO) test data managed via Excel spreadsheets, representing complex entities as simplified, reusable objects.

This approach significantly reduces data redundancy and improves scenario management. By separating data from test logic, CenterTest enhances maintenance efficiency, scalability, and collaboration between technical and business teams.

Key innovations of CenterTest's Data-Driven Architecture include:

- Object-oriented test data representation, improving reusability
- Excel-based management bridging technical and business teams
- Intelligent parameterization for test customization without altering core logic
- Scalable test generation from minimal base entities
- Advanced execution management with parallel processing and intelligent filtering
- Comprehensive scenario testing with minimal data input, allowing swift adaptation to market demands
- Guidewire Version Compatibility also extends to test data, ensuring tests written in v7/8/9 also run in the Cloud.

These features address critical DDT challenges, transforming QA processes, accelerating development cycles, and reducing costs for complex system testing. The results have been remarkable, with an 80%+ reduction in effort for Guidewire projects, including support for complex end-to-end tests. For Guidewire Cloud Migration customers, the results are even more impactful, allowing you to test on the first day of your cloud migration if you already use CenterTest.

The following sections of this whitepaper detail CenterTest's approach, its solutions to DDT challenges, and its impact through real-world applications, including a case study demonstrating these impressive efficiency gains. For insurers seeking to maximize their

InsuranceSuite investment and navigate an increasingly complex technological landscape, CenterTest offers a pivotal tool for modern software testing.

Critical Challenges in Data-Driven Testing

Before delving into CenterTest's solutions, it's crucial to understand the key challenges that have long hindered traditional DDT approaches. These challenges span various aspects of the testing process, from data management to execution performance, and have significant implications for the efficiency and effectiveness of testing efforts.

1. Data Management. Data management in DDT involves handling data redundancy, addressing organizational difficulties, dealing with inefficient data structures and inconsistent data formats across environments. These issues can lead to ***complications in test data preparation, maintenance, compatibility, and test reliability***.

2. Parameterization. Parameterization is the mechanism used to incorporate test data into the test itself. It involves efficiently retrieving the test data from its source, effectively incorporating this data into the test, and ensuring the tests remain readable and maintainable. ***Effective parameterization is crucial for creating tests that can adapt to various scenarios without requiring extensive rewriting of test logic.***

3. Test Maintenance. Test maintenance challenges include time-consuming updates, error-prone modifications, and difficulty adapting tests to application changes. Understanding how changes in test data impact test behavior and results adds to the challenge, ***potentially slowing down the testing process and reducing overall efficiency.***

4. Scalability. Scalability concerns include handling large volumes of test data, managing extensive test case libraries, and adapting to growing system complexity. As systems expand, maintaining ***effective and efficient testing becomes increasingly challenging, often leading to issues like data bloat and slower test execution***.

5. Data Quality. Data quality issues involve ensuring data accuracy and relevance, maintaining consistency across test suites, and identifying and addressing data issues. ***Poor data quality can lead to unreliable test results and missed defects, undermining the entire testing process.***

6. Test Coverage. Test coverage challenges include achieving comprehensive scenario coverage, creating edge cases and complex scenarios, and adapting to diverse industry contexts. ***Inadequate coverage can result in undetected software issues, potentially leading to critical problems in production environments.***

7. Environment Dependencies. Environment dependencies include managing data across different environments, ensuring consistency between environments, and handling environment-specific configurations. ***These issues can lead to inconsistent test results across different stages of development, complicating the testing process.***

8. Test Data Security. Test data security concerns protecting sensitive information, complying with data protection regulations, and balancing security with test authenticity. ***This is particularly crucial when dealing with production and production-like data in testing environments, where data breaches can have significant financial and reputational consequences.***

9. Collaboration. Collaboration challenges involve bridging technical and non-technical team members, facilitating contributions from various stakeholders, and aligning tests with business requirements. ***Effective collaboration is essential for comprehensive and relevant testing that meets both technical and business needs.***

10. Automation Integration. Automation integration includes integrating with existing infrastructure, supporting various data storage technologies, and maintaining consistency across diverse tools. ***Seamless integration is key to maximizing the benefits of automated testing and ensuring efficiency across the testing ecosystem.***

11. Execution and Performance. Execution and performance challenges involve efficiently executing large test suites, optimizing resource utilization, and balancing execution speed with test depth. ***These factors significantly impact the overall effectiveness and efficiency of the testing process, particularly in large-scale or complex testing environments.***

By addressing these challenges, CenterTest aims to revolutionize the DDT landscape, offering a comprehensive solution that enhances testing efficiency, effectiveness, and scalability across diverse enterprise environments.

Object-Oriented Test Data

CenterTest introduces a paradigm shift in data-driven testing through its innovative use of OO test data. This approach, unprecedented in the industry, incorporates the principles of abstraction to represent complex real-world entities as simplified, reusable objects with specific attributes.

The Test Data

Core to CenterTest's approach is the representation of real-world entities as OO data structures. These structures encapsulate the properties and behaviors of various elements commonly encountered in software testing scenarios. By abstracting these entities, CenterTest enables testers to create more flexible, maintainable, and reusable test data.

To illustrate this concept, let's consider some common entities that CenterTest can represent:

1. Person: This entity might include attributes such as names, birth date, and contact information. It's a versatile object that can represent customers, employees, or any individual relevant to the system under test.
2. Address: An address entity typically encompasses properties like street address, city, state, postal code, and address type. This object can be associated with persons, businesses, or any other entity that requires location information.
3. Vehicle: In scenarios involving automotive or insurance applications, a vehicle entity becomes crucial. It might include attributes like make, model, year, color, new cost, and VIN number.

These examples demonstrate how CenterTest breaks down complex real-world scenarios into manageable, reusable OO components. This approach allows for greater flexibility in test data creation and management.

While CenterTest supports OO-capable technologies like JSON, XML, and databases, its standout feature is the ability to implement OO test data using Excel spreadsheets. This approach combines the power of OO data with the familiarity and accessibility of spreadsheets, enhancing team-wide management and collaboration.

In Excel, abstracted data entities are represented as worksheets within a spreadsheet, with columns containing an entity's properties. These "reference" sheets contain keys for efficient organization and retrieval and the data needed for testing.

Let's look at an example **Account** reference sheet containing person, company, and address entities. Note that each sheet contains specific types of entities, each row is an entity, and columns contain each entity's data.

code	firstName	lastName	dateOfBirth	officialID	phone	phoneType	gender	email
JohnSenior	John	Doe	-p68y	123-45-6789	512-111-1111	Mobile	male	johnd@test.email
MaryAdult	Mary	Fitzgerald	-p33y 3m	123-45-1111	201-777-0123	Home	female	
PeterTeenager	Peter	Smith	-p 18y		512-111-1111	Mobile		pete@test.email

Person | Company | Address | +

Code	AddressLine1	City	State	StateCode	PostalCode	AddressType
Austin	43 Rainey St	Austin	Texas	TX	78701-1111	Home
Chicago	171 N Wells St	Chicago	Illinois	IL	60606-1234	Home
NewOrleans	1140 Royal St	New Orleans	Louisianna	LA	70116-2211	Home

Person | Company | Address | +

And now the **PersonalAuto** reference sheet which contains vehicle, driver, and coverage information.

code	make	model	year	vin	color	newCost
Lambo	Lamborghini	Huracan	2018	3A876H89	Orange	215,000
pickup	Dodge	Ram1500	2022	1Y887NM90	Black	40,000

Vehicle | Driver | Coverage | PolicyChange | +

code	Person	FirstName	LastName	YearLicensed	DateOfBirth	LicenseNu
Driver1	MaryAdult			2000	-p25y	555555
Driver2		Stephen	Strange	-6	-p40y	666666

Vehicle | Driver | Coverage | PolicyChange | +

Code	Coverage	CovTerm	CovtermValue
MedPay	Medical Payments	Medical Limit	15,000
Collision	Collision		
Comp	Comprehensive	Comprehensive Deductible	1,000
Towing	Towing and Labor	Towing and Labor Limit	25

Vehicle | Driver | Coverage | +

Several things to note include unique codes for each entity, explicit and relative date processing within the same field, and even blank fields that do not fail even if they do not exist.

To effectively use OO-based reference test data, we override provided data to meet specific test requirements rather than creating unique entities for each requirement. This approach reduces redundancy and avoids data bloat, potentially allowing fewer than a dozen Person and Vehicle entities to support thousands of tests—an efficiency unattainable with traditional flat data structures.

By combining OO test data with familiar data storage systems like Excel, CenterTest provides a robust, scalable solution for DDT. This approach dramatically reduces data redundancy, improves clarity, and simplifies updates and maintenance across various industries, revolutionizing test data management while leveraging familiar tools.

Next, we will see how to choose the data entities for use within a test.

The Test Container

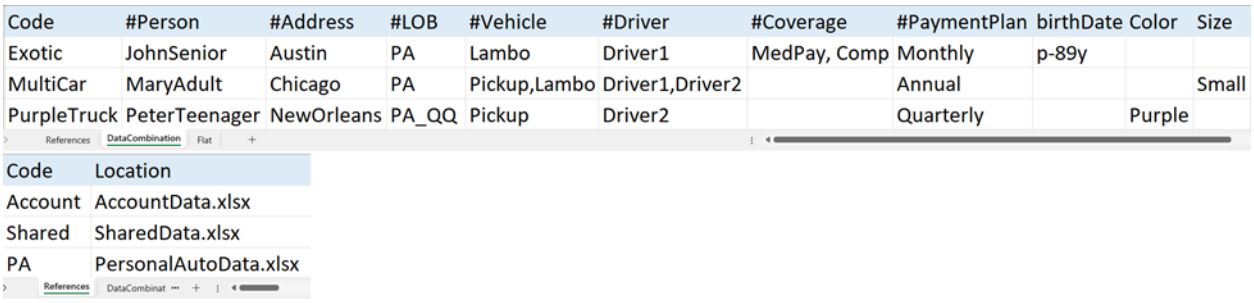
CenterTest's approach significantly reduces data redundancy and improves scenario management. Instead of creating unique entities for each requirement, which can lead to data bloat, the system allows for overriding reference data to meet specific test requirements. This efficiency can result in fewer than a dozen Person and Vehicle reference entities supporting thousands of tests—an improvement unattainable with traditional flat data structures.

The test container in CenterTest focuses on two primary considerations:

1. Identifying the reference entities needed
2. Overriding default values for specific needs

This approach of referencing small sets of reusable test data entities and then overriding specific criteria forms the cornerstone of test maintainability and data simplification.

Let's break down the PersonalAutoDataCombination test container, or **Data Combination** spreadsheet, for an insurance policy.



Code	#Person	#Address	#LOB	#Vehicle	#Driver	#Coverage	#PaymentPlan	birthDate	Color	Size
Exotic	JohnSenior	Austin	PA	Lambo	Driver1	MedPay, Comp	Monthly	p-89y		
MultiCar	MaryAdult	Chicago	PA	Pickup,Lambo	Driver1,Driver2		Annual			Small
PurpleTruck	PeterTeenager	NewOrleans	PA_QQ	Pickup	Driver2		Quarterly		Purple	

References | DataCombination | Flat | +

Code	Location
Account	AccountData.xlsx
Shared	SharedData.xlsx
PA	PersonalAutoData.xlsx

References | DataCombinat ... | +

Data combination spreadsheets are used to combine multiple reference sheets into their OO data entities for each test. Each Data Combination spreadsheet does several things, including:

- The **References** sheet (tab) identifies which reference sheets to use for the test
- The **DataCombination** sheet defines the tests, with each row being one DDT test with the Code being the test name

- Each reference column (# columns) provides the key to the associated reference entity
- Multiple reference keys can be included in a single reference column.
- Individual values can be overridden on a test-by-test basis

Let's consider **MultiCar** test from the second row to understand the process.

1. **MultiCar** references **MaryAdult** in the **#Person** column (# signifies a reference column). In the **Person** reference sheet, MaryAdult refers to **Mary Fitzgerald** who has a relative age of "**-p33y 3m**". Relative dates are java Period elements, so Mary is **33 years** and **3 months** old
2. Vehicles are retrieved in the same way so **MultiCar** refers to two vehicles, a **Pickup** and a **Lambo**. Going to the **Vehicle** reference sheet returns an **orange 2018 Lamborghini Huracan** and Pickup returns a **black Dodge Ram 1500**
3. To override data, we process each column that does not begin with a **#** excluding reserved columns such as **Code**. For example

- the **Exotic** test lets you override John's **Age** to **89** years old,
- the **MultiCar** test needs something in a **Size Small** for Mary,
- and the **PurpleTruck** test overrides Peter's pickup to be the **Color Purple.**

Before we look at the test, consider what a traditional flat DDT input file might look like, with 47 columns containing all the data from the OO entities. This is where data bloat comes from. As tests are added, data is duplicated.

A	B	C	D	Z	AA	AB	AT	AU
code	firstName	lastName	dateOfB	vehicle2_year	vehicle2_vin	vehicle_c	payment_Plan	size
Exotic	John	Doe	1937/04				Qurrterly Payments	
MultiCar	Mary	Fitzgerald	1987/02	2018	3A876H89	Orange	Monthly Payments	Small
PurpleTruck	Peter	Smith	2006/03				Payment in Full	

The following list identifies the 47 properties found in the associated referenced sheets.

Person	Address	Line of Business	Vehicle 1	Vehicle 2	Driver 1	Driver 2	Coverage	Coverage	Payment Plan	Custom
person_firstName	address_Line1	lob_lineOfBusinessTitle	vehicle1_vin	vehicle2_vin	driver1 FirstName	driver2 FirstName	coverage_Coverage	coverage_Coverage	payment_Plan	code
person_lastName	address_City	lob_effectiveDateOffset	vehicle1_modelYear	vehicle2_modelYear	driver1_LastName	driver2_LastName	coverage_CovTerm	coverage_CovTerm		custom
person_dateOfBirth	address_State	lob_quoteType	vehicle1_make	vehicle2_make	driver1_YearLicensed	driver2_YearLicensed	coverage_CovtermValue	coverage_CovtermValue		
person_officialID	address_StateCode	lob_quoteOnly	vehicle1_model	vehicle2_model	driver1_DateOfBirth	driver2_DateOfBirth				
person_phone	address_PostalCode		vehicle1_costNew	vehicle2_costNew	driver1_LicenseNumber	driver2_LicenseNumber				
person_primaryPhone	address_AddressType									
person_gender										
person_email										

Now let's look at a test.

The Test

To use the test data in your data-driven test, CenterTest tightly couples the Data Combination (provides the test data and any overrides) to the physical test.

```
1  @CenterTest
2  public final class PersonalAutoSubmissionTest implements PC {
3
4    @CenterTestCase
5    @DataDriven(datasource = "PersonalAutoDataCombination.xlsx")
6    public void run(ScenarioContext scenarioContext) {
7      var context = scenarioContext.getInvocationContext("su");
8      var data = DDTHelper.getPersonalautoPersonalAutoDataCombination(scenarioContext);
9
10     // Move the override values into the root values for the test data
11     data.getPerson().withDateofbirth(data.getDatacombination().getAge());
12     data.getVehicle().withColor(data.getDatacombination().getColor());
13
14     // Lines 11 and 12 are unecessary when using the internal override process
15     // but that process has been excluded from this white paper
16
17     PC.loginToPC(context).execute();
18     PC.newPersonAccount(context, data).execute();
19     PC.personalAutoSubmission(context, data).execute();
20   }
21 }
22
23
24 public class NewPersonAccount extends {
25
26   public void execute() {
27     createPersonAccount();
28     createAddress();
29   }
30
31   protected void createAddress() {
32     var addressContainer = createAccountPage.getGlobalAddressContainer();
33     var address = data.getAddress();
34     addressContainer.getAddressLine1().set(address.getAddressline1());
35     addressContainer.getCity().set(address.getCity());
36     addressContainer.getPostalCode().set(address.getPostalcode());
37     addressContainer.getState().set(address.getState());
38   }
39 }
```

Note the following DDT specific aspects that make the entire process work.

- **Line 4** identifies this test as a CenterTest test.
- **Line 5** defines this test as being data-driven and using the PersonalAutoDataCombination spreadsheet as described earlier.

- **Line 8** loads the OO data-driven entities from the spreadsheet and makes each entity and its data available to the test.
- **Lines 11 and 12** override the original values from the special override columns in the PersonalAutoDataCombination spreadsheet. As stated in the commented lines (lines 14 and 15), this is not the typical way, but shows one way to do overrides. Overrides are normally done directly in the spreadsheet, but that process is being excluded from this white paper as it is a proprietary process and the goal of this whitepaper is to explain OO test data and how it transforms DDT.
- **Lines 33-37** pass the data-driven data directly to the POM (page object model) fields, allowing them to be set, ignored, or handled in special ways.

Challenges Addressed

CenterTest's innovative approach to DDT primarily addresses the challenges of **Data Management** and **Parameterization** through its OO (OO) test data structure implemented in Excel spreadsheets. This method effectively tackles data redundancy and organizational difficulties by using reference sheets for different entities (e.g., Person, Address, Vehicle) and Data Combination sheets for test scenarios.

The system's design contributes to improved **Test Maintenance** and **Scalability** by allowing data overrides and reuse of entities across multiple tests, potentially supporting thousands of tests with a minimal set of base entities. **Collaboration** is enhanced through the use of familiar Excel interfaces, bridging the gap between technical and non-technical team members. The approach also touches on aspects of **Data Quality** and **Test Coverage**, offering a structured framework for data consistency and the flexibility to create various test scenarios through data overrides.

We will elaborate on these challenges and provide additional techniques for those challenges not yet addressed.

1. Data Management

CenterTest's OO approach to data management goes beyond the core concepts described earlier. It incorporates additional features to enhance data organization and accessibility:

- **Automated Analytics:** The system automatically collects analytics, such as time to set values and tracking of changes, providing valuable insights into test execution.
- **Override Tracking:** When data values are changed from their defaults in reference sheets, these overrides are specifically tracked and reported.

- **Version Control**: For test data, version control is equally crucial, as ***untracked or outdated data versions can lead to unreliable test results*** and hinder the ability to replicate and troubleshoot issues effectively.

These intelligent features and more, streamline the parameterization process by removing coding requirements, enhancing test robustness, and providing greater visibility for test data usage and modifications.

2. Parameterization

Parameterization is the mechanism used to incorporate test data into the test itself and involves several key challenges in DDT. CenterTest's capabilities extend beyond basic data incorporation:

- **Propagation of Changes**: Modifications to both reference data and structure automatically propagate across all dependent tests, significantly reducing maintenance overhead.
- **Automated Data Generation**: CenterTest can automatically generate Data Combination files from existing tests, bridging the gap between traditional hard-coded and DDT methodologies, thus speeding the adoption of DDT for new tests.
- **Intelligent Data:** CenterTest also incorporates many smart features into test data at the internal layer, features like automated set verification, simplified assert processing, internal analytics at the field level, and more.

These features streamline parameterization in CenterTest by eliminating coding requirements and enhancing test robustness. They provide greater visibility for test data usage while automating complex data scenarios. This approach not only improves test reliability and data security, but also ensures parameterization remains efficient and adaptable to changing test requirements.

3. Test Creation and Maintenance

Beyond the core data management functionality already discussed, CenterTest can automatically generate Data Combination files from existing tests, providing a bridge between traditional hard-coded and DDT methodologies. This preserves the value of your current test assets while adopting more flexible and maintainable testing practices.

Furthermore, CenterTest leverages its OO foundation to ensure that modifications to both reference data and structure propagate across all dependent tests, significantly reducing maintenance overhead. The integrated DC/DR (Data Combination/Data Reference) utility - see **Data Quality** - complements this by enabling quick identification of tests affected by

changes, thereby simplifying impact analysis and supporting efficient test suite management.

CenterTest Guidewire Version Compatibility is the **only** technology able to take tests written for Guidewire v7/8/9 and use them in the Cloud. This lets Cloud Migration projects start testing their Cloud Upgrade project immediately by using tests already written for their current implementation. **Imagine the value of Day One Testing your Cloud Upgrade without having to write new tests with true parallel testing of the old and the new!**

4. Scalability

Handling large volumes of test data and test cases efficiently has been a persistent challenge in DDT, particularly in complex, multi-system enterprise environments. As businesses grow and systems become more intricate, the ability to scale testing efforts becomes crucial.

CenterTest's OO data structure allows complex scenarios to be easily represented with combinations of simple OO entities, enabling the creation of vast numbers of test scenarios with minimal data overhead. A simple cartesian product of 6 PersonalAutoDataCombination reference entities (excluding LOB) with 5 possibilities each would produce 15,625 tests (5^6). Though this is an unreasonable number for a single test scenario, CenterTest supports it. In Test Coverage, we will show how to reduce this number to 35 tests.

To effectively run and choose tens of thousands of potential tests, CenterTest implements two key strategies:

- **Sophisticated Runtime Manager**: CenterTest includes a parallel test execution capability that utilizes as many threads as the hardware supports. It dynamically monitors active threads and distributes tests to optimize completion time.
- **Advanced Suite Selection**: CenterTest allows the filtering of tests by package, name, workflow, or even specific data in the Combination spreadsheets (e.g., state or province). This feature not only simplifies execution but also aids in defect analysis and test management.

By efficiently running and managing vast numbers of test scenarios, organizations can continue to expand coverage as their systems evolve, ensuring more robust and reliable deployments.

5. Data Quality

CenterTest's separation of test data from test code simplifies data management and automatically propagates changes across all associated tests, maintaining consistency while significantly reducing maintenance overhead.

Building on this foundation, **relative date processing** addresses time-sensitive data challenges by letting testers specify dates in relation to the current date. This feature ensures tests remain valid across different time zones and test environments with varying system clocks, eliminating the need for constant updates and enhancing test reliability across diverse settings.

To further enhance data management, CenterTest's **Data Combination/Data Reference** (DC/DR) utility analyzes the test data. This tool helps identify tests affected by specific data modifications, unused test data, and more, streamlining the maintenance process and ensuring optimal data usage.

CenterTest's OO data structure incorporates advanced capabilities to enhance test robustness and security. The system automatically handles **blank values**, preventing test failures when encountering non-existent properties. It can **generate realistic random data** when needed, increasing test variability and coverage. It will retrieve values from **secure vaults or secrets** as needed, ensuring the protection of PII/PFI data throughout the testing process.

These data quality features collectively address key challenges in DDT, improving efficiency and security while enabling teams to focus on creating comprehensive test scenarios rather than managing data complexities.

6. Test Coverage

CenterTest approaches coverage in two ways: handling sophisticated end-to-end tests covering real-world scenarios, including edge cases, and minimizing the number of tests needed for the expected growth of comprehensive coverage.

For end-to-end testing, CenterTest excels in supporting complex scenarios across multiple Guidewire applications. It enables the creation of test flows that span PolicyCenter, BillingCenter, and ClaimCenter within a single test, supporting multi-user and multi-Center interactions. Advanced features, such as time travel testing and simulation of delayed processes (e.g., document generation) are seamlessly integrated into both test flows and test data, ensuring robust validation of intricate, interconnected insurance operations.

To address the challenge of comprehensive coverage, CenterTest supports intelligent test data generation. As test scenarios grow more complex, the number of possible

combinations can quickly become unmanageable. For example, a simple PersonalAutoDataCombination with 6 reference sheets and 5 options each could potentially require 15,625 tests (5^6) for full coverage.

CenterTest tackles this exponential growth through **pairwise** testing, a combinatorial method that maximizes coverage while minimizing the number of tests. Kimputing's AnkrPt Pairwise generator, a free pairwise generator, can be used with CenterTest to further optimize test generation. For instance, a scenario that would require 15,625 tests can be effectively covered with just 35 tests, maintaining a high degree of confidence in test coverage.

This dual approach allows organizations to thoroughly test complex business processes, including edge cases, without the burden of executing an impractical number of test cases. It significantly enhances both the depth and efficiency of the testing process, ensuring comprehensive validation across the entire Guidewire suite while keeping the testing effort manageable.

7. Environment Dependencies

CenterTest effectively addresses environment dependencies through features that enhance test reliability and consistency across diverse enterprise landscapes. At its core, CenterTest provides environment-specific processing for crucial elements like login information and Single Sign-On (SSO) processing, allowing tests to adapt automatically to the requirements of different environments or geographical regions without manual intervention.

This principle extends to reference spreadsheets, which can be environment-specific if desired. This capability enables fine-grained control over test data across different environments, managing variations between development, testing, and production environments, or even between countries. Consequently, the risk of environment-related test failures is significantly reduced.

Furthermore, CenterTest's locale-based relative date processing ensures accurate test execution across different time zones and system clocks, maintaining consistency in time-sensitive scenarios. This feature is particularly valuable for preserving the integrity of tests involving date-dependent processes, a common requirement in many business applications.

These features collectively provide robust support for managing environment dependencies, improving both the efficiency and reliability of test execution across diverse enterprise landscapes.

8. Test Data Security

Protecting sensitive test data is a critical concern across industries and regulatory environments, especially when using production-like data. CenterTest addresses this challenge with a multi-layered approach that balances security with test authenticity.

CenterTest's primary strategy is the use of realistic non-sensitive data, thus avoiding PII and PFI concerns altogether, as demonstrated in its reference sheets with generic names and non-specific addresses. This approach minimizes security risks from the outset. For scenarios requiring more realistic data, CenterTest can generate synthetic information that mimics production data without compromising security, allowing for authentic testing without exposing sensitive information.

When production data use is unavoidable, CenterTest supports integration with secure vaults for protected data access. It offers controlled usage and automated removal of sensitive data post-test. For instance, in rate matching projects between legacy and new systems, CenterTest can retrieve PII from secured locations, store this information in the target environment during testing, and automatically remove the data upon test completion, regardless of the test outcome.

This comprehensive approach enables organizations to maintain compliance with various data protection regulations while conducting thorough, authentic tests. By effectively balancing security and test authenticity, CenterTest facilitates robust testing without compromising sensitive information, crucial in an environment where data breaches can have significant financial and reputational consequences.

9. Collaboration

CenterTest bridges the gap between technical and non-technical team members, addressing a long-standing challenge in DDT. By leveraging Excel, it creates an environment where business analysts, domain experts, and QA teams can collaborate effectively.

The system's clear separation of core data and test-specific data allows technical team members to focus on underlying structures and logic, while business users can define and refine test scenarios without needing to understand the framework's intricacies. This approach reduces the learning curve for advanced testing concepts and enhances communication between QA teams and business stakeholders.

Importantly, business users can directly update test data in spreadsheets and run tests, fostering a truly collaborative environment. This level of engagement results in more comprehensive and relevant test coverage, ensuring that testing efforts are not only

technically sound but also closely aligned with business requirements across various sectors.

By facilitating this collaboration, CenterTest improves the alignment between technical testing efforts and business objectives, ultimately leading to more effective and efficient testing processes across diverse enterprise environments.

10. Automation Integration

Automation integration presents a significant challenge in software testing, particularly in diverse multi-system enterprise environments. CenterTest addresses this challenge through its innovative use of abstraction layers, providing a flexible and robust solution adaptable to various industries and technological landscapes.

These abstraction layers allow for flexibility in choosing the most suitable options for their specific needs. Options such as the data format for DDT (such as Excel, JSON, XML, or databases), or the test manager or defect-tracking system to use.

CenterTest's design philosophy of flexibility and adaptability means it can potentially complement existing automation systems or incorporate their testing into itself, protecting your investment. This consistent interface for test data and scenarios further enhances CenterTest's flexibility, facilitating the creation of efficient automated testing ecosystems that can adapt to various environments.

Ultimately, CenterTest's approach to automation integration enables faster, more reliable software delivery across diverse industries and technological platforms. It transforms the challenge of integration into a strategic advantage by offering a comprehensive, adaptable testing solution that can evolve with an organization's needs.

11. Execution and Performance

CenterTest significantly enhances test execution performance through its advanced parallelism capabilities and sophisticated time-based testing features. The system's Runtime Manager leverages the full processing power of the testing environment, executing as many threads as desired and available. Sophisticated load balancing ensures tests are distributed evenly across these threads, synchronizing end times to minimize overall execution time, even for large and complex test suites across diverse environments.

At the core of CenterTest's suite-building process are intelligent filters that complement traditional suite definitions. These filters enable teams to create smaller, dynamic suites, effectively managing large numbers of tests regardless of their complexity. This approach facilitates more efficient monitoring and allows multiple suites to execute simultaneously, substantially reducing overall execution time.

Relative date testing can result in tests requiring clock changes, referred to as time travel testing. CenterTest's robust support for time travel testing includes automated clock changes, crucial for time-sensitive systems, self-managed execution to synchronize clock changes, perform best-day calculations to minimize the number of changes, and automated test restarts as needed, letting teams conduct high-volume time travel testing without risking costly errors due to mistaken changes.

By combining advanced parallelism, intelligent test suite management, and sophisticated relative date and time travel testing capabilities, CenterTest optimizes resource utilization and dramatically improves testing efficiency and effectiveness. When considering time-sensitive testing, efficiency improvements are at a new level.

Case Study: A New Guidewire Customer

Background

A mid-level insurance carrier based in the United Kingdom, using Guidewire InsuranceSuite for Personal Auto, faced significant challenges with its testing approach. The carrier relied on an in-house automation team using pure Selenium for their testing needs, focusing solely on regression testing. Their existing test suite was minimal, comprising only 26 tests for submission, 15 for policy transactions, and some additional tests for other workflows. This limited suite, developed over approximately 1.5 years, highlighted the time-consuming nature of their test development process and the constraints it placed on their quality assurance efforts.

Challenges

The carrier's testing approach presented numerous obstacles, with the time-consuming test development process being a critical one. This lengthy development cycle hindered the team's ability to keep pace with system changes and new features, creating a bottleneck in their quality assurance process.

The key challenges faced by the carrier included:

- Limited test coverage across product brands and scenarios
- Difficulty in maintaining and expanding the test suite
- Inability to test complex insurance scenarios
- Lack of comprehensive testing for all InsuranceSuite components
- Time-consuming test development, critically impacting overall testing

To address these challenges, the carrier decided to explore new testing solutions that could provide more comprehensive coverage while reducing development time. This led them to consider implementing CenterTest, a specialized testing tool designed for Guidewire insurance carriers.

Implementation

The carrier initially planned a 3-week Proof of Concept (POC) to implement CenterTest. However, the results were achieved far more rapidly than anticipated.

The entire POC, including the original requirements and additional enhancements, was completed in less than one week, with the core implementation taking only 3 days.

This accelerated timeline included one day to set up CenterTest, connect to the test endpoint, and identify and review the tests to be covered. Once completed, the core work of the POC began with the development of tests to cover their existing suite.

This work was completed primarily by one senior Kimputing CenterTest SDET who accomplished the following:

- Generated the CenterTest client framework directly from the client's Guidewire codebase.
- Modified OOTB hard-coded policy workflows to meet the client's specific Guidewire codebase to confirm test functionality. This required test code updates.
- Converted the hard-coded tests to data-driven tests using OO test data. This included extending the OOTB test data. All future tests were implemented using DDT.
- Expanded policy submission coverage by extending test data for complex scenarios (multiple drivers, multiple claims), adding post-binding verification steps, and expanding coverage to all brands.
- Added additional workflows covering remaining needed tests and more.

This rapid implementation dramatically outpaced the carrier's previous testing development efforts, showcasing the efficiency and power of the CenterTest solution.

Results

The rapid implementation of CenterTest not only dramatically outpaced the carrier's previous testing development efforts, but also led to significant improvements in their testing capabilities. These improvements far exceeded the carrier's initial expectations,

demonstrating the efficiency and power of the CenterTest solution across multiple dimensions of their quality assurance process.

Key results of the CenterTest implementation included:

Dramatic increase in test coverage:

- Submission tests increased from 26 to 135 (419% increase)
- Policy change and other transactions increased from 15 to 48 (220% increase)
- Total tests increased from 41 to 183 (346% increase)

Remarkable efficiency in development:

- The entire implementation of 183 tests was completed in approximately 3 workdays by a single developer
- This represents a significant improvement over the previous 1.5 years spent developing a smaller test suite

Streamlined test management:

- All tests are now driven by a single Excel spreadsheet, simplifying test data management and scenario creation
- This approach allows business analysts and non-technical team members to easily contribute to test scenario creation

Enhanced flexibility and maintainability:

- The DDT approach allows for more adaptable tests that can handle a variety of scenarios without code changes
- Changes to test scenarios can often be made by simply updating the Excel spreadsheet, without requiring code modifications

Comprehensive coverage:

- Tests now cover multiple brands and complex scenarios like multiple drivers and claims
- End-to-end testing now includes post-binding verification in external systems, ensuring more thorough validation

This substantial improvement in testing capabilities positioned the carrier to ensure more robust quality assurance for their insurance products and services, while significantly reducing the time and resources required for test development and maintenance.

Lessons Learned

The successful implementation of CenterTest provided valuable insights into effective testing strategies for insurance carriers. These lessons highlight the importance of choosing the right tools, leveraging data-driven approaches, and fostering collaboration between technical and business teams.

Key lessons learned include:

- **Power of DDT:** Separating test data from test logic created a more flexible and maintainable test suite, allowing for easier updates and expansions.
- **Efficiency of specialized tools:** CenterTest's features, particularly its ability to handle blank fields and generate client frameworks, greatly enhanced the efficiency of test creation and execution.
- **Importance of comprehensive coverage:** The ability to easily test multiple brands and complex scenarios revealed the value of thorough, varied test cases in ensuring product quality.
- **Collaboration benefits:** The Excel-based approach opened up test creation to non-technical team members, fostering better collaboration between development and business teams.
- **Rapid implementation potential:** With the right tools and expertise, significant improvements in testing capabilities can be achieved in a remarkably short time frame.
- **Scalability of automated testing:** The dramatic increase in test coverage demonstrated the scalability of well-implemented automated testing solutions.

Conclusion

CenterTest represents a significant leap forward in DDT for complex software systems, especially insurance carriers using Guidewire. By introducing an object-oriented approach to test data management and leveraging familiar tools like Excel spreadsheets, CenterTest addresses longstanding challenges in the field of software testing.

Key advantages of CenterTest include:

1. Dramatic reduction in data redundancy and improved test data management through its OO test data structure
2. Enhanced collaboration between technical and business teams via Excel-based data management
3. Significant improvements in test maintenance efficiency and scalability
4. Robust solutions for test data security and quality assurance
5. Advanced execution management with parallel processing and intelligent suite filtering

The case study presented in this whitepaper demonstrates the transformative potential of CenterTest, showcasing how it enabled a mid-level insurance carrier to increase their test coverage by over 300% in just a matter of days - a feat that could take years using traditional methods.

As the insurance industry continues to evolve and face increasing technological complexities, solutions like CenterTest will play a crucial role in ensuring software quality, reducing time to market, and ultimately improving insurance carriers' bottom lines.

The future of data-driven testing in the insurance sector looks promising, with tools like CenterTest leading the way. By addressing the core challenges of data management, parameterization, scalability, and collaboration, CenterTest not only improves current testing practices but also paves the way into the future.

As insurance companies continue to invest in digital transformation, adopting advanced testing solutions like CenterTest will be key to staying competitive, ensuring system reliability, and delivering high-quality services to policyholders.