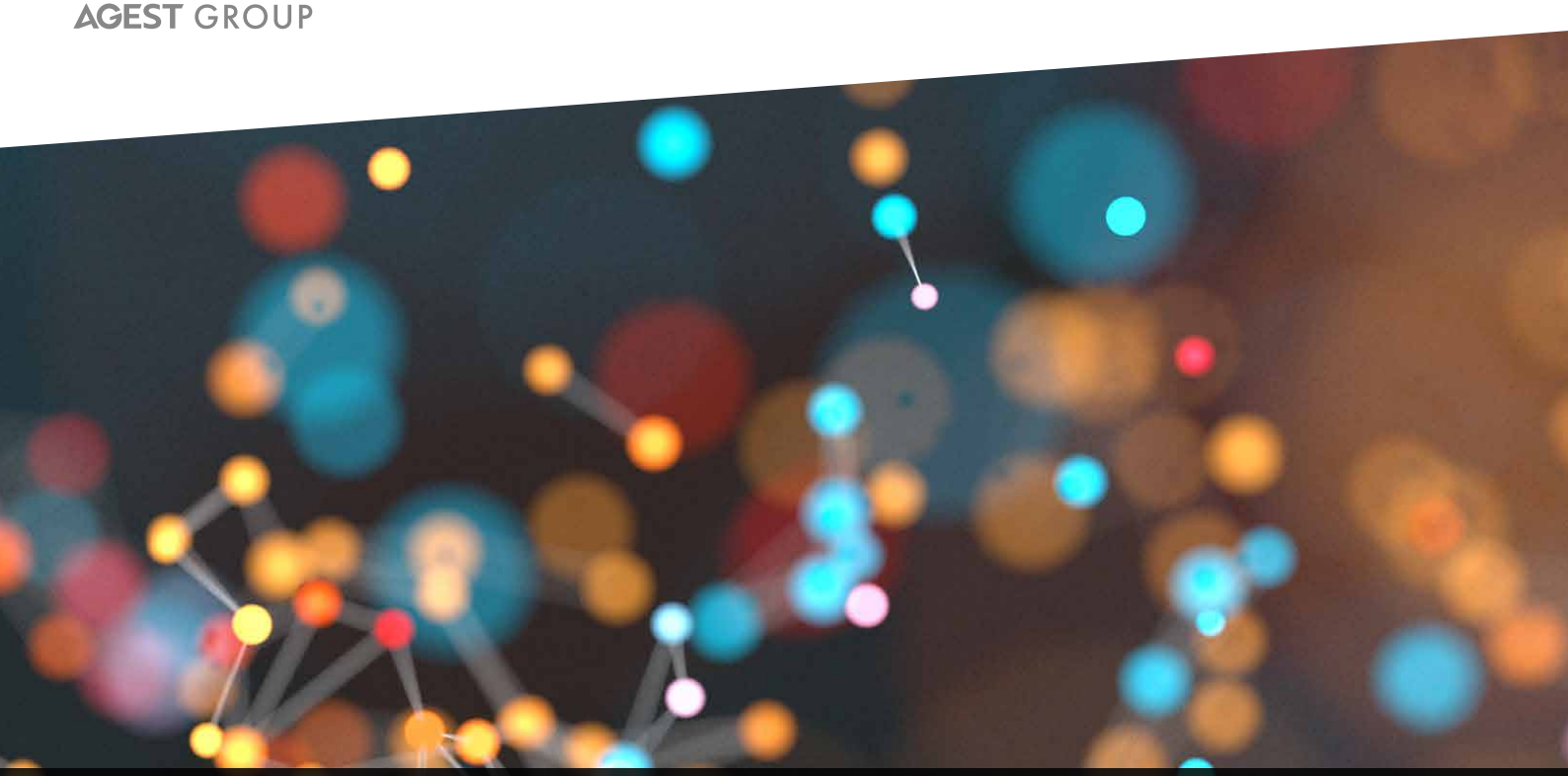
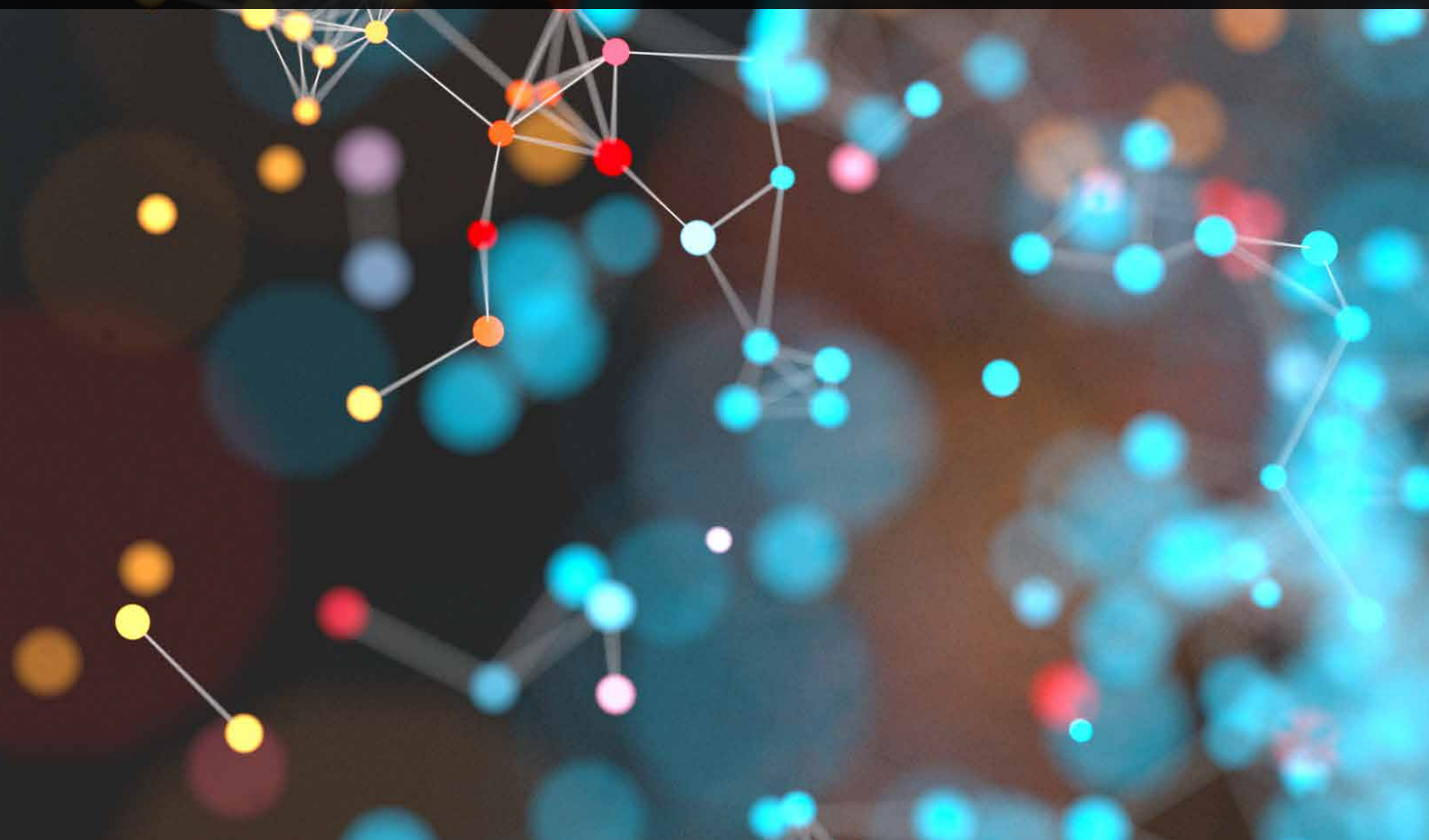
dws
AGEST GROUP



TESTING INNOVATION

A Best Practice Guide For JD Edwards EnterpriseOne



CONTENTS

WHY DO WE TEST?

When it comes to enterprise software projects, testing can account for up to 65 % of the overall time and effort. It might seem like a strange question, but why do we test?

At its heart, change involves risk and testing is all about managing and mitigating risk.

From an enterprise applications software perspective, we carry out functional testing to eliminate unwanted events that negatively impact on either system performance or user experience.

What constitutes an unwanted event? It's easy to say anything that doesn't consistently deliver the desired outcome, but we prioritise those that result in an unacceptable impact, or those that occur with an unacceptable frequency.

Physical processes have a simple input and output relationship – you put a combination of elements in one end and something predictable comes out the other. If what comes out the other end is not what was expected, there are typically few variables to be considered.

Software doesn't work that way. When it fails it tends to do so in unexpected and "mysterious" ways. The more complex the business process being managed by the software, the more difficult it can be to discover the reason for the error.

The failure of customer-facing systems can be embarrassing, we've all seen the headlines detailing website crashes during periods of unexpectedly high demand, or customer service applications stretched to breaking point following extreme weather conditions.

Today the nature of many complex and sophisticated organizations and industries means that even the smallest failure inone system can have a huge impact.

The impact of a failure might be felt across departments in a simply company, or up and down the supply chain. When systems support critical processes, such as infrastructure management or pharmaceutical production, there are potentially disastrous consequences should the software fail.

Remember, testing is not an activity, it's a process. It takes place throughout the software development lifecycle and is particularly important when enterprise applications are being upgraded, extended, enhanced or when businesses and the processes that define their business are evolving.

For some organizations, software testing is like an insurance policy. Forming a key component of quality assurance and risk management.

The amount of effort invested in testing should be commensurate with the potential risk to the business and the complexity and frequency of any changes made to the software or operating environment.

As JD Edwards EnterpriseOne customers work to continuously innovate, thought needs to be given to how they run projects. Customers need to implement a code-current strategy that will allow them to run smaller, more frequent value-adding projects. Each of these projects will require testing to ensure no errors surface downstream in the production environment.

In this best practice guide we explore how innovative testing solutions are transforming the way we run projects; making them smaller, faster and smarter.

WHAT DO WE TEST?

It's reasonable to say "test everything" when running a big project to upgrade from one major release to the next. This approach to testing is commensurate with the risk. However, as companies move to stay code-current once on release 9.2 projects will become smaller and less risky. It is on these smaller code-current change event projects that there is real value in doing thorough analysis, having informed conversations with the business, and planning to test only what needs to be tested.

How do you know what needs testing?

The biggest dilemma facing your Quality Assurance team is how to identify precisely what needs testing during a change event project. To do this you need a detailed understanding of what has changed (e.g. objects introduced or modified by an Electronic Software Update (ESU), or modifications introduced by your developers). You need to know which objects are affected and you need to identify all the dependencies within your EnterpriseOne system to truly understand the impact of those changes.

Equally important, you need to be clear regarding what does not need testing, so you don't waste time, effort and money testing things that have not been impacted by the change event.

How thoroughly do you need to test?

Once you have identified the scope of a change event project, you need to assess by how much each and every object has been affected so you can determine how thoroughly they need to be tested.

For example, is it just a yes / no dialog box that has been repositioned or are there significant changes throughout a critical workflow?

Allocating and managing test resources

Every project is different, so when you're allocating functional and management resource to a testing project, you need to consider both the functional aspects and business processes that will be affected.

Project managers will need to assign sufficient testing resource to ensure all changes and dependencies have been rigorously tested and that the changes have not caused any unexpected downstream impact.

Understanding the impact of a change event project (i.e. the scope) helps to ensure that you come up with a good plan. Then, over the course of the project, you need to ensure that the resources are being employed efficiently and making the progress expected.

Project teams that execute to a well-considered plan and that thoroughly and accurately test only what needs testing put themselves in a better position to handover to the business for end-user acceptance testing. They can clearly articulate the change that users will experience after go-live and can demonstrate the testing that has been undertaken prior to asking end-users to test and then sign-off.

HOW DO WE TEST?

Given the significant contribution testing makes to any successful change event project, it is surprising how many organizations continue to rely on inefficient, manual processes.

Adopting a more agile approach

The cost and effort required to identify and resolve issues typically escalates throughout the lifecycle of the project. However, there are several best practices that can be adopted to facilitate a more agile testing methodology.

This begins with a robust testing plan; one that is derived from a detailed assessment of the change event in question and is sequenced to fit with other requirements within the business.

The plan should be structured to reflect "bundles" of work with full integrity. The bundles should be self-sufficient; allowing for them to be tested without having dependencies at a later stage in the project.

Pros and cons of different testing approaches

Manual Testing

Research amongst EnterpriseOne customers indicates that the majority still rely on manual testing. Understandably, a degree of manual testing will be required when testing the more obscure or complex workflows – including things a user might do out of curiosity or by mistake.

However, reliance upon manual testing has several drawbacks:

- Manual testing, by its nature, is time-consuming
- Repeat testing requires the same amount of effort every time
- It is prone to human error, particularly when checking the results

With each iteration of testing, when "bugs" have to be sent back to someone on the team to fix and then returned for re-testing, the testers tend to become less diligent. As a result, bugs will creep into production, where they are more costly and expensive to fix.

Record and Playback

This approach uses tools that record the actions of a tester in a manual test, and then allows tests to be run unattended, greatly increasing test productivity and eliminating the tedious repetition of manual testing.

However, even small changes to the software under test require that tests be recorded manually again. The record and playback approach to test automation has been used for several decades. It has proven over time to be neither efficient nor scalable and Return on Investment is rarely realized.

Scripting

Scripting is a form of programming in computer languages used in software test automation. It alleviates many issues found with the Record and Playback method. Often this is done by capturing functions or use-cases and then storing them as reusable test cases to be executed by a calling script.

However, the developers of these scripts must be highly technical and specialized programmers who work in isolation from those testers performing the tests day-to-day.

Scripting is best suited for GUI testing and doesn't lend itself easily to embedded batch or other forms of systems.

As changes to the software under test require complex changes to the associated automation scripts, maintenance of ever-larger libraries of automation scripts becomes an overwhelming challenge.

Scriptless

A scriptless approach to testing does away with the need for technical specialists to write or code scripts. Instead, a tool is used to enable non-technical end-users to generate test scenarios without the need for programming knowledge.

This makes it easier for end-users to become more involved in the testing of their business processes and far less time consuming to rerun tests whenever necessary. With both the scripting and scriptless methods, test scenarios can be run as often as needed, with minimal human intervention, and accuracy can be maintained throughout.

A scriptless testing solution provides the most economic testing approach and the one that provides the best Return on Investment, as it is the easiest way to set up test scenarios without the need for expensive specialist resources.

A good scriptless testing solution should also make it quick and easy for end-users to modify and edit test scenarios, without the need for any programming skills or knowledge.

PERFORMANCE TESTING

When dealing with ERP and other mission-critical software applications, a lot of time and effort is spent on functional testing to ensure that systems perform as expected; that a given input results in a given output.

However, functional testing alone is not enough to test the resilience and predictability of your system. Running a sequence of simulated functional tests is not an accurate reflection of the real-world demands that are likely to be placed on your systems.

Function versus frequency

During exceptionally busy periods, systems do not always perform as expected. That's why stress testing or load testing your JDE EnterpriseOne system is so important.

Systems should never be designed with the lowest levels of utility in mind, they should always be built to perform during periods of sustained, heavy use. After all, it is during periods of highest demand that any systems failure has the greatest impact.

Capacity-related systems failures are less common than functional errors, but you shouldn't fall into the trap of thinking less likely means lower risk. Once a system is in production, any downtime could have a significant impact on everything from customer satisfaction to production timescales and revenue generation.

The larger your organization, the more connected systems or the more varied the workflows, the more important load testing becomes. In the real world, your systems may need to cope with hundreds, thousand, or even millions of concurrent workflows (in the case of large-scale consumer-facing websites or e-commerce platforms).

Load testing in a JD Edwards EnterpriseOne environment typically takes place towards the end of any significant project. Resilience can only be established once functionality has been defined and can be proven to work within the system.

Load testing allows you to measure more than just the continuity of a desired outcome, it allows you to assess any latency or hang-time that may result from higher volumes of use. Again, this goes back to the user experience. Availability is one thing, but so is performance.

Load testing also needs to consider the fact that not all users are the same. The broader the spectrum of users, the greater the chance of a variation in "input behaviour". Straightforward functional testing often assumes all interactions are the same. Load testing a real-world scenario requires a degree of individuality that more accurately reflects the behaviour of a community of users.

Sharing the load

If conducted manually, load testing can be difficult to manage, can be difficult to monitor, and can be a burden on both time and financial resources. That's why it lends itself so well to automation. However, in order to get most value out of automated load testing, analysts need to be able to simulate and repeat a wide range of interactions.

When selecting a test automation product, it makes sense to choose one that is designed with your specific software in mind. A generic solution is likely to require either a lot of technical setup, programming, and configuration or a compromising of quality.

As with any testing solution, it's important to be able to monitor performance in real-time and to extract and analyze test results down to a forensic level of detail.

WHAT'S STOPPING YOU?

Research suggests that the biggest single barrier to upgrading or adopting a code-current strategy is the perceived burden of testing.

We have already mentioned the contribution testing makes to the overall project effort. We've also seen that many organizations continue to rely upon time-consuming manual processes. The question is, why?

Why aren't more businesses automating their testing?

When we analyzed the use of testing software among the JDE install base, we found that market penetration for even the most well-known solutions was relatively low. Even the most "popular" of solutions was only being used by 14% of respondents.

There appears to be a shortage of recognised testing solutions for JD Edwards. Outside of Micro Focus (formerly HP UFT) and Oracle's own Oracle Application Testing Suite (OATS), there was little or no mention of suitable alternative test planning, automation and load testing solutions.

As a market leader in testing solutions, you might expect the adoption of HP to be greater, but the price point and complexity of the solution places it outside the comfort zone for many smaller to medium-sized organizations.

Software test automation is not without its own challenges. The attitudes towards test automation are remarkably similar, whether the organization has previous experience of the solution in question or not. When we explored why organizations had negative experiences, we found that the need for complex code changes and the complexity of the testing process itself were among the principal issues.

Research within the Oracle JDE install base reveals that 76% of respondents believe "Testing is too difficult and time consuming". This acts as a barrier to adoption of Oracle's preferred code-current strategy. (Fig1)

Among those respondents that had previously used software test automation solutions, we were interested to know a little more about their individual experiences. In particular, we wanted to know about the challenges they had faced using automation tools in the past. (Fig2)

Some of the negative experiences relating to testing solutions were surprising, given that test automation is designed to save time and make things simpler. So, we asked the same users what they would look for in the ideal test automation solution. (Fig3)

Fig 1.

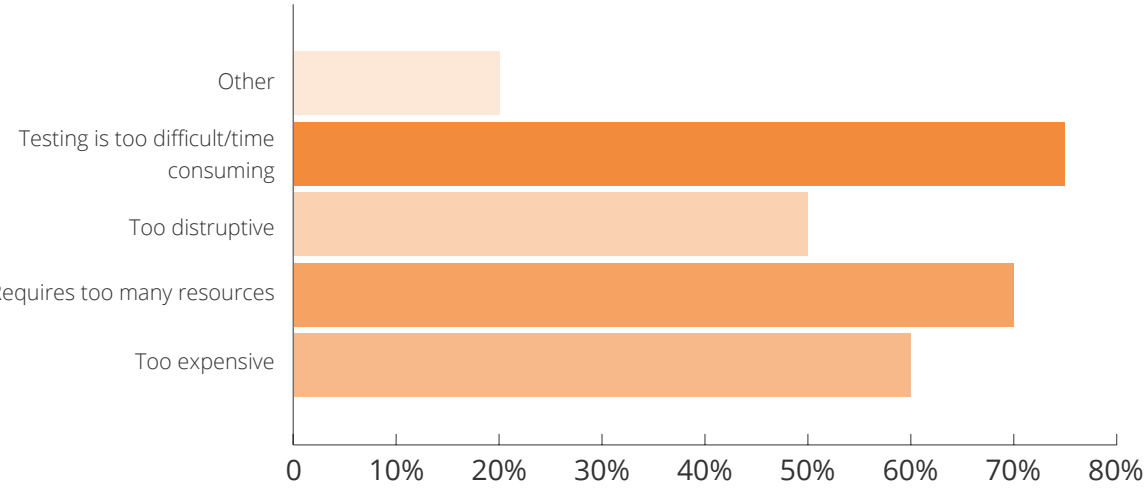


Fig 2.

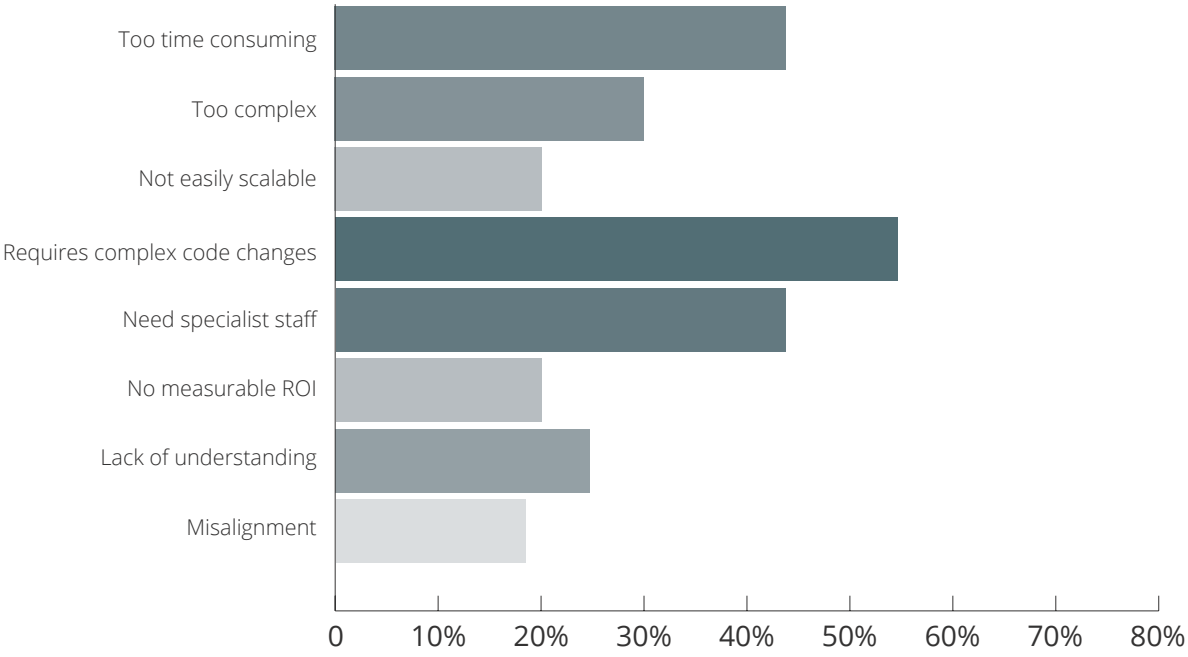
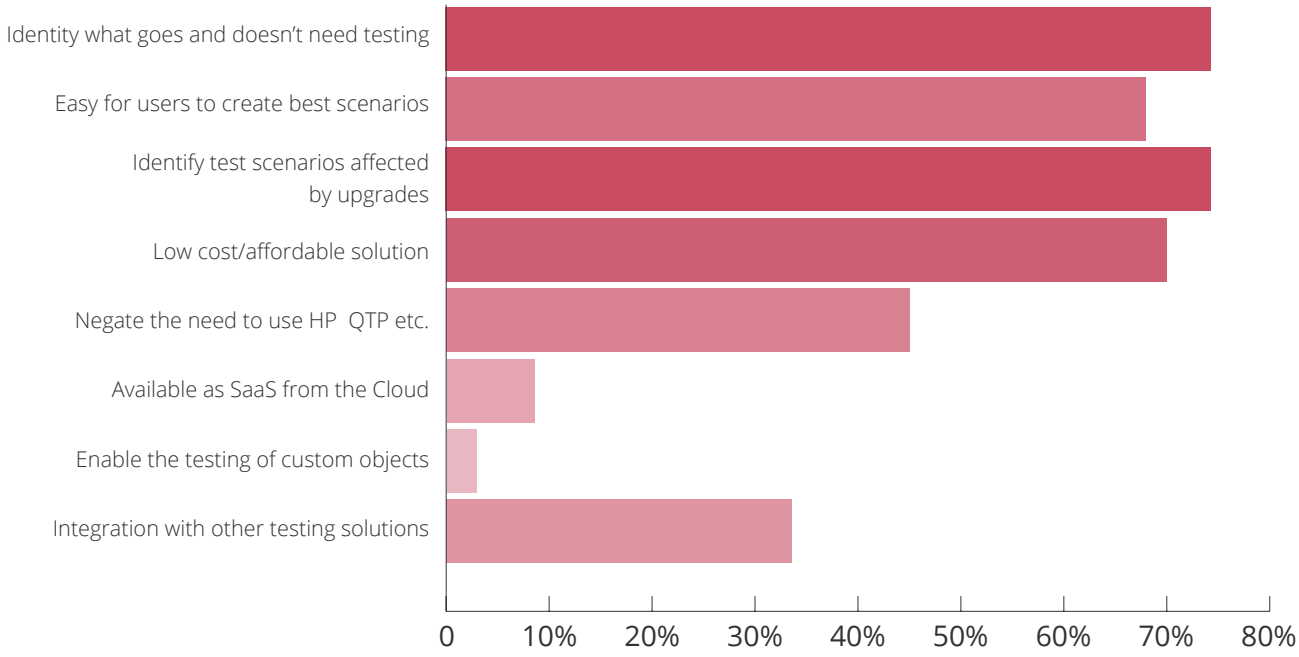


Fig 3.



WHAT TOOLS ARE AVAILABLE?

It's clear from this research that not all ERP testing solutions are created equal and that exploring the wide range of available tools and understanding their limitations is key to finding the right fit for your testing requirements.

Test Planning and Management

Traditionally, there weren't the tools available to help customers understand the impact Electronic Software Updates (ESUs) or software modifications would have on their EnterpriseOne system.

Of course, Oracle provides documentation to indicate which objects they have modified in ESUs and major version releases, and customers can set the modified flag when they make changes to the standard code, but this flag is often set incorrectly or, worse still, not set at all.

Knowing which objects have been modified is only the starting point when planning your testing. You need to understand all the interconnections between these modified objects and other objects in the system if you are to focus your testing effort in the right areas and avoid unnecessary testing of code that is not affected by the changes.

There is only one product commercially available at the current time that addresses the issues of test planning:

Dimension Focus™

The Dimension Focus application allows users to:

- Identify what needs to be tested
- See how thoroughly objects need testing
- Understand what does not need to be tested

It does this by investigating the programs that you use and how they would be affected by the ESUs; analysing the lower-level objects that are used by these programs, such as tables, business views and business functions.

The interdependencies between these programs, tables, views and functions can be incredibly complex.

Dimension Focus also understands the interdependencies (at the Event level) between these objects and can build up a multi-level hierarchy of how they are all interlinked. This cross-reference tree extends both up multiple levels above the ESU affected objects and down to lower-level objects that may be called by several 'parent' objects and events.

Using this sophisticated analysis, Dimension Focus identifies those objects that would be directly impacted as a result of installing the ESU, or your own modified code, into the EnterpriseOne system.

The number of objects affected by the change event could be as little as 5% of the total number in your system, leaving 95% unchanged, so the potential for saving test effort is significant.

Having identified all the interdependencies in your system, Dimension Focus can also identify the top-level programs that are affected by changes in the lower level objects. The result is a reduction in the number of programs in your EnterpriseOne system that need to be re-tested.

Regardless of which test management and execution method you use, the information provided by Dimension Focus will help minimize the testing effort required and maximise your Return on Investment.

Advantages

- No need to install the ESU to prepare your test plan
- Analyze the impact of ESUs and modifications down to the event or function level
- Drill up and down through your EnterpriseOne system to identify all direct and indirect impacts from any ESUs or modifications
- Identify what needs testing and what does not
- Utilize a heat map and an impact score so you know where to focus your testing effort
- Significantly reduce the amount of testing effort required
- Inform the conversation you will have with the business about the project and the acceptance testing they will need to perform

Test Automation

Oracle Application Testing Suite

Oracle Application Testing Suite (OATS) is a comprehensive testing solution deigned to ensures the quality, scalability and availability of web applications, web services, packaged Oracle applications and Oracle databases.

This integrated, full-lifecycle solution enables users to define and manage their application testing process, validate application functionality and ensure that applications will perform under load. Oracle Application Testing Suite aims to enable organizations to deploy web applications and services in less time, whilst maximizing the efficiency of their testing teams.

OATS includes the following tightly integrated products:

- Oracle Load Testing for scalability, performance and load testing
- Oracle Functional Testing for automated functional and regression testing
- Oracle Test Manager for test process management, including test requirements management, test management, test execution and defect tracking

Oracle Application Testing Suite also provides a series of integrated testing accelerators for testing Oracle packaged applications and SOA (Service-Oriented Architecture) applications. These accelerators enable enhanced scripting capabilities for more efficient and optimized testing.

Advantages

- Reduced development time for script creation
- Improved security of data during testing with built-in encryption
- Integrated scripting platform for automated functional testing and load testing
- Powerful Java-based code view built on Eclipse IDE to extend scripts
- Enables automatic generation of load test scripts from Real User Experience Insight
- The testing accelerators for JD Edwards EnterpriseOne provide support for functional test automation of EnterpriseOne and the JD Edwards Data Grid component

Disadvantages

- The tool supports only web-based applications

Micro Focus Unified Functional Testing

Micro Focus UFT (formerly HP UFT) is designed to be an omnichannel test automation tool. Centralizing the functional testing of a broad range of software applications; including SAP, Oracle, Salesforce and more.

Combined with Micro Focus ALM it can be used to execute tests synchronously, or mimic user engagement as the basis for performance testing.

UFT can be deployed in provisioned Citrix, AWS and Azure virtual environments, or to run web and mobile tests from Docker containers.

Advantages

- Easy to use, even for non-programmers to understand and start adding test cases
- The tool has an excellent Object Identification process / mechanism
- Supports different add-ins like Java, Oracle, SAP, .NET, Web Forms, Siebel, PeopleSoft, Web services, mainframe (Terminal Emulator) etc
- Well integrated with test management tools like QC
- An extensive set of default functionalities
- Supports both Windows and web-based applications

Disadvantages

- High licensing and add-on costs
- Relatively slow regression test execution
- Higher than expected levels of false positives
- Burdensome test maintenance

Dimension SwifTest™

Dimension SwifTest is the only testing solution designed specifically for, and integrated with, JD Edwards EnterpriseOne. It has been designed with the non-technical business analyst and super-user in mind and makes the creation, editing, scheduling and execution of testing as simple and efficient as possible.

Recent product innovations include a repeater function that streamlines functional testing even further. SwifTest is also available with a catalogue of sample scripts to help customers kick-start the creation of their own base catalog.

Advantages

- Native integration with JD Edwards EnterpriseOne to simplify test scenario building and editing
- Easy for non-technical end-users to build and edit test scenarios
- No specialist or technical knowledge required
- No programming or scriptwriting required
- Uses unique scanning methodology to automate the generation of test scripts
- Can be used on custom code as well as ESUs
- No need for laborious record and playback test set-up
- As test requirements or the application changes, editing the test scenarios is quick and easy
- Multiple test scripts can be submitted into a queue for automated testing
- Pinpoints all test failures with user-friendly information regarding the reasons for failure
- Pay-as-you-save subscription model delivers a rapid and measurable ROI
- Integrates with the DWS Dimension Focus planning tool for maximum ROI

Disadvantages

- Only works with JD Edwards EnterpriseOne

Performance Testing

Dimension LoadTest™

Dimension LoadTest is a powerful yet easy to use load testing tool for JD Edwards EnterpriseOne applications. It has been designed such that any experienced System Administrator/CNC should be able to create, execute, monitor and analyze load tests to effectively stress test JD Edwards EnterpriseOne applications.

Agents can be spun up and monitored with ease via an intuitive user interface. Status update messages are reported to the controller as the load is being executed and displayed on the form for the user to interpret and action.

LoadTest allows you to create test queues that can run on separate agents, testing different functional areas concurrently. This allows for a much more realistic and accurate load test.

Advantages

- Specifically designed for JD Edwards EnterpriseOne
- Very small footprint, it can be installed in minutes
- Simulate large numbers of virtual users
- Flexible SaaS based pricing, virtual user day model
- Option for cloud-based workloads anywhere in the world
- Sophisticated script building functionality is included

Disadvantages

- Only works with JD Edwards EnterpriseOne

USE CASE 1

TYNDALE

Tyndale have been using JD Edwards EnterpriseOne since February 2016, following a two-year development project.

JDE E1 is fully integrated into Tyndale, with 250 employees (up to 70 concurrent users) utilizing AP, AR, Inventory Management, Procurement, GL, Warehousing, Transportation and Expense Management and Sales Order Management modules.

Third-party integrations include CRM, CDI (website) and RF Smart for pick and pack, plus integration with freight forwarding and logistics.

Tyndale has made a significant number of customizations to its platform to support several business processes, including product returns.

The Challenge

With a small internal team of two dedicated to JDE support, testing could be time consuming. Rather than depending upon ordinary users to support quality assurance (QA), Tyndale was looking to automate as much of the process as possible.

In anticipation of further systems development, and a potential upgrade to JD Edwards EnterpriseOne 9.2 (JDE E1 9.2), Tyndale needed to find a way to reduce the burden of testing during any future change event project.

Delivery

Having assessed the alternatives, Tyndale implemented Dimension Focus™ to support test planning and Dimension SwifTest™ for test automation.

Result

Use of Dimension Focus has resulted in better test planning and eliminated wasted effort on objects that don't need testing.

The adoption of SwifTest has had a significant impact on both continuity and productivity; saving Tyndale up to 75% on the time taken to execute complex functional testing.

The experience of working with DWS has been very good; especially in terms of support. They have been very responsive... If we identify any bugs, they are fixed quickly and DWS are adding new features and functionality all the time.

Karen Riner, Systems Administrator, Tyndale

USE CASE 2

Acal have been running JD Edwards EnterpriseOne as one of their core systems since 2001.

It forms a key component of their business operations; supporting the design, manufacture and distribution of customer-specific electronic components to over 25,000 companies around the world.

JDE E1 is deployed throughout the Custom Distribution Division and operates the Sales, Purchasing, Inventory, Warehousing, Manufacturing and Finance modules – supporting approximately 270 users.

The Challenge

With just a small in-house team, Acal was looking to significantly reduce the time and resource committed to testing JDE change event projects.

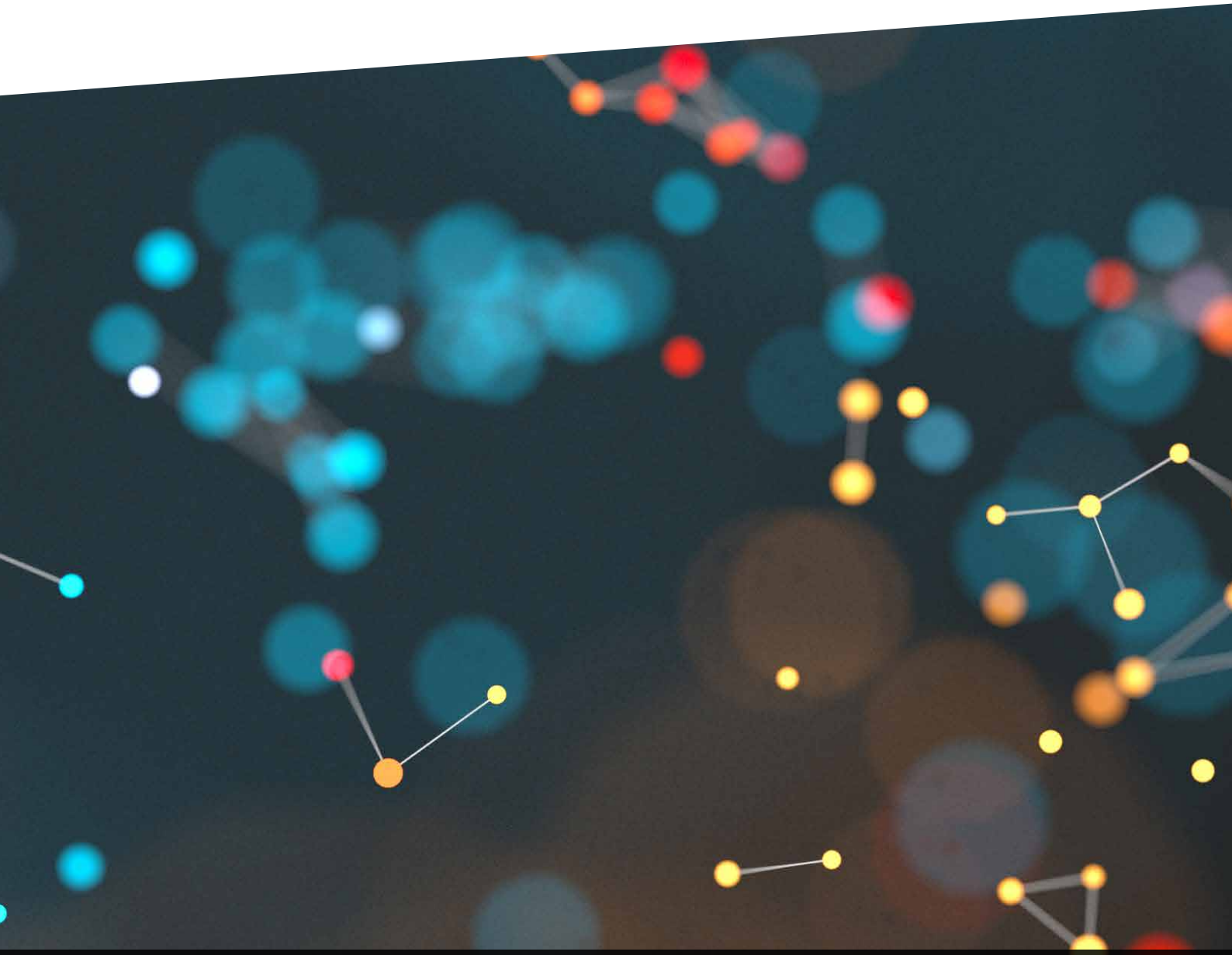
Delivery

Acal engaged with DWS to deploy Dimension SwifTest™, the test automation product designed specifically for JD Edwards EnterpriseOne users.

Result

SwifTest is used to automate the testing of all change event projects for Acal's E1 installation. Test execution times have been cut by 45%, providing a strong foundation for a code-current future.

> **"Using Dimension SwifTest™ allowed us to significantly reduce the volume of back-breaking testing we would have had to do.**
>
> ERP & CRM Applications Manager, Acal plc



About DWS

DWS is a leading provider of Oracle JD Edwards EnterpriseOne software services and products.

Since 1998, we have been providing development and technical services to organizations looking to customize, integrate, extend, upgrade or support implementations of EnterpriseOne. We also sell EnterpriseOne testing products that leverage our deep domain expertise and help customers run smaller, faster and smarter projects.

DWS serves a global client base using proven methodologies and proprietary DWS Dimension™ tools. Our best-practice approach and eye for detail help us deliver products and services that save time and money and continually drive down your TCO for JD Edwards.

For further information please visit our website, or contact us:

UK: +44 (0) 1494 896 600 US: +1 888 769 3248 ANZ: +64 (0) 9427 9956
sales@dws-global.com **www.dws-global.com**

ORACLE | Partner