Compliments of IBM

IBM Limited Edition

Testing SAP Solutions FOR DUMMIES®

A Wiley Brand

Learn to:

- Reduce the cost and risk of testing on SAP projects
- Test SAP projects using IBM software
- Realize the value of standardized testing

Allan Wagner
James Hunter
Nick Portalski
Bernd Eberhardt

To learn even more about how IBM can help organizations test and deliver higher quality SAP solutions faster, visit **http://ibm.biz/IBMDevOpsforSAPprojects.**

Testing SAP Solutions

FOR DUMMIES®

A Wiley Brand

IBM Limited Edition

by Allan Wagner, James Hunter, Nick Portalski, and Bernd Eberhardt

FOR DUMMIES®

A Wiley Brand

Testing SAP Solutions For Dummies®, IBM Limited Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

ISBN: 978-1-119-04720-9 (pbk); ISBN: 978-1-119-04722-3 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Carrie A. Johnson

Editorial Manager: Rev Mengle

Business Development Representative: Sue Blessing

Production Coordinator: Melissa Cossell

Author's Acknowledgments

The IBM authors would like to thank their colleagues Kay Johnson, Glyn Rhodes, Stuart Walker, Carsten Siegler, and Lynn Giles, who provided vision, content, review, and assistance to help make this book possible.

Table of Contents

Introduction

Welcome to *Testing SAP Solutions For Dummies,* IBM Limited Edition. We hope you find this material helpful as you begin or continue your work to measure the quality of your SAP solutions implementation. With much of today's business relying on integrating SAP solutions with a mix of established custom applications and commercially available solutions, the task of testing these complex applications continues to become more and more challenging. To help organizations achieve results they desire, IBM software has the tools and technology to achieve continuous delivery of high quality software while reducing the cost and risk of managing change in the SAP landscape. IBM calls this set of tools and technologies DevOps for SAP systems (DevOps is short for development and operations).

The main themes of IBM's approach to DevOps is to accelerate software delivery while at the same time balancing speed, cost, quality, and risk *and* improving the client experience — no small task, as you may expect. Because testing makes up a significant portion of the delivery life cycle, it can be a major contributor to the success of the business. When the testing goes wrong and announced release dates are missed, you can damage your brand's reputation or (even worse) lose customers. By testing efficiently and releasing quality software in a timely manner, you can grow your market share while delighting your customers.

About This Book

In this book, we share how IBM software and solutions can help an organization manage quality and begin testing SAP solutions and the integrations they depend on earlier, continuously, and with greater test coverage.

SAP is a well-established leader in the packaged applications market, providing software with a rich set of composable application modules and configurable functional capabilities, delivered by a comprehensive enterprise application software suite.

However, organizations typically modify the packaged business processes to meet their specific needs. Testing those changes becomes even more important because many organizations adopting SAP software remain *heterogeneous* enterprises — enterprises where SAP implementations are integrated with a variety of non-SAP enterprise systems, portals, messaging infrastructures, security, systems of record, systems of engagement, and more. These complex implementations must work properly — end-to-end across a wide variety of technologies.

IBM offers a complete quality management and testing solution enabling companies to continuously test and measure quality levels throughout the software development life cycle on into production. This should be done without concern for whether their implementation is a stand-alone SAP deployment or integrated with other critical business systems. By bringing together functional, performance, and integration testing, combined with *service virtualization* — the ability to simulate dependent yet unavailable software and systems for the purpose of testing — organizations are able to deliver higher quality products and services to market rapidly with greater efficiency, higher predictability, and reduced cost.

Icons Used in This Book

This book uses the following icons to call your attention to information you may find helpful in particular ways.

The information in paragraphs marked by the Remember icon is especially important, so we'd ask you to take special note of it.

The Tip icon indicates extra-helpful information. You may discover how to implement an especially useful testing implementation or find out crafty ways to help you save time or money.

This icon marks places where technical matters are discussed. Sorry, it can't be helped; plus, the information is intended to be helpful.

Paragraphs marked with the Warning icon call attention to common pitfalls that you may encounter.

Chapter 1

Setting the Stage

In This Chapter

- Meeting testing timelines
- Testing software the DevOps way
- Going for lean and mean

You'll often hear that testing takes as long as it takes. To dig down and determine what this statement actually means — other than an attempt to avoid committing to a timeline — you'd have to closely examine the many factors contributing to how much time "as long as it takes" actually takes. Things to look at include

- The complexity of the technologies being tested
- The availability of complete and realistic test environments
- The particular approach to testing: manual, automated, or a combination of the two
- The number of resources you can devote to testing
- The testing experience of the quality professionals
- Knowledge of the application being tested

Taking Testing Seriously

This idea of "testing taking as long as it takes" could certainly apply to organizations developing and testing SAP solutions. These projects tend to be complex and also tend to take a long time. One could also imagine accepting a testing taking as long as it takes approach if the delivery team was in fact achieving maximum efficiency without any distractions or interruptions. However, "maximum efficiency" isn't the case for many teams.

Now, if your crack team is able to deliver higher quality software and meet all planned release dates with room to spare, you've got little to no risk for the business, and your stakeholders are happy. When the day comes, however, that delivery expectations are missed or the release date slips, those very same stakeholders may feel that your all-star delivery team has ended up with egg on its face.

To meet customer expectations and deliver quality software at the speed customers or end-users demand, teams need to shift to the more responsible side of the tactics spectrum — they need to become more efficient in their testing practices, they need to involve testers much earlier in the process, and they need to begin testing earlier and continuously so they can release software of higher quality much more frequently.

Testing Software Using a DevOps Approach

To reduce costs and win in the market, companies are now looking to extend agile and lean principles — increasing efficiency, reducing waste, and embracing best practices to shorten development cycles — in a DevOps approach to software development. This desire and need to improve their capability aligns perfectly with IBM's definition of a DevOps approach to software development — an enterprise capability for continuous delivery of quality software that enables organizations to seize market opportunities and reduce time to customer feedback by releasing new functionality in smaller increments more frequently.

By applying lean and agile principles across the software delivery life cycle, IBM DevOps helps organizations deliver a differentiated and engaging customer experience, helps them achieve quicker time to value, and works to increase their capacity to innovate.

For the delivery team to meet the needs of the customer and the employers at the speed demanded, the team needs to become more efficient and look for ways to reduce bottlenecks. Working in *silos* — with analysts, developers, testers, and operations all working in isolation with limited collaboration — creates inefficient handoffs between the individual groups. This approach doesn't work when time is of the essence. All

roles should focus on working together as one delivery team where all team members are contributing to the quality of software being developed.

It is really all about time and taking less time to get to the end of your project while maintaining or increasing the level of quality in the software being delivered. So, to deliver software that is better, deliver it faster, and deliver it at a lower cost, the delivery team needs to look for ways to begin testing earlier and continuously so it can shift release dates to the left on your time scale, rather than settle for always shifting release dates to the right.

Shifting left means breaking down the silos and eliminating the idea of handoffs by building quality into the delivery process earlier — during blueprinting and design, code reviews, requirement approval, and so on. But again, building quality in doesn't mean that testers are the only ones responsible for the quality of the software to be delivered.

In a DevOps approach to software development, everyone contributes to quality, and everyone should be contributing to the shared goal of delivering a positive user experience. Delivery teams need to expand the effort to do more unit testing, increase the level of integration testing done to check services, and decrease the level of effort focused on testing through the user interface (UI). See Figure 1-1.

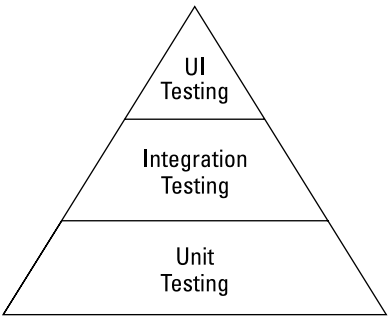

Figure 1-1: The layers of testing.

Unit testing refers to testing carried out in the development environment by the person who wrote the code or carried out the configuration. *Integration testing* (also known as API testing) requires the delivery team to test that all integration

points between the many components making up the complete application are interacting with each other properly.

Neither unit testing nor integration testing require a stabilized user interface — that means you can start with these types of testing very early on in the development process. *User interface* (UI) testing is performed by testers interacting with the application's user interface to measure whether the application functionality is working properly. This type of testing can either be done manually or by using test automation tools once the user interface is available from the development team.

Having developers do more unit testing — or get that first round of unit testing up and running if your organization hasn't yet incorporated that aspect into the delivery life cycle — makes sense since it's the developers who write the code.

Next would be increased testing at the service layer, continuously validating the integration between components to ensure the messages being exchanged are processed appropriately and the components are performing as expected. In situations where all the components aren't available for testing, service virtualization can be used to simulate what's missing.

Leveraging automation techniques to expand the coverage of testing at the unit and service levels frees up testers to do other value-added activities such as exploratory testing, asking better questions, and improving test design — activities that would definitely improve the overall level of test coverage for many organizations.

By being able to test earlier and test continuously across the entire delivery life cycle, teams are able to improve efficiencies to such an extent that they'll no longer be tempted to fall back on the tried and true ways of just throwing more money (or more people) at the problem. We're here to tell you that there is a better way to test and measure the quality of your SAP solutions and to do it at a lower cost. Read on to find out more about what's available to help organizations manage quality and automate testing of their SAP solutions while helping testers test more efficiently and effectively.

Chapter 2

The Challenges of Testing on SAP Projects

In This Chapter

- Examining typical approaches of testing SAP projects
- Working with multiple user interfaces
- Looking to the future

Teams embarking on developing and deploying high-quality SAP solution implementations are sure to encounter several distinct challenges — challenges they'll need to be overcome if they ever hope to succeed. The delivery team needs to be able to

- Consistently manage change when faced with testing both SAP and non-SAP solutions
- Deliver a high level of quality on time and within budget
- Validate that the business processes have been checked and are working as expected

It goes without saying that all this is supposed to be done with the goal of minimizing errors, reducing costs, and aligning the business goals with the development effort. So, in order to truly test SAP applications effectively, you really need to align the project blueprint — the business process hierarchy, or BPH, to use more technical language — with the quality management effort. It is the BPH that defines the scope of the development effort by documenting the processes to be implemented and drives the test design for testing changes.

When properly aligned with the BPH, any SAP project test plan is going to contain multiple test cases, each of which is going to contain multiple test scripts (wheels within wheels within

wheels, as it were). Within the test script are the specific, detailed steps a test is going to execute, either manually or through some form of automation. The test execution results are then linked to the test case, along with any associated defects created as a result of a test failing. Finally, and to ensure that a high level of quality is maintained, teams need to validate that the changes haven't impacted the existing functionality and performance of the deployment.

Sounds like a tough row to hoe, and it can be, but approaches exist that can ease the burden to a certain extent. The rest of this chapter takes a look at a few.

Examining Typical Approaches to Testing on SAP Projects

While there are many approaches to testing SAP implementations, the driving factor that needs to be accepted across the entire team is a constant focus on delivering quality while being aware of potential roadblocks along the way. The following list, while far from exhaustive, can give you a sense of the challenges your team will be facing in this regard:

- The uncertainties of testing your SAP implementation when it has been integrated with legacy applications and deployed in environments made up of many differing technologies and commercially available solutions
- Setbacks that may occur if some members of your team (or maybe all members) are hampered by a lack of specific testing skills
- The (potential) impact to a delivery schedule when performing a high volume of manual tests
- The (again, potential) creation of bottlenecks when test environments and test data aren't available when teams need them

But it's not all doom and gloom; help is available — help to assist teams in managing quality, automating their tests to reduce test execution time, and removing testing bottlenecks like waiting for test labs to become available so testing can begin. Read on to find out the details.

Manually testing

In many organizations, details of the test case and the detailed test scripts are documented and maintained in a collection of documents and spreadsheets and managed by the SAP project test manager or test coordinator. These *test cases* (detailed descriptions of how the software will be tested) and *test scripts* (the step-by-step instructions spelling out how a test will be carried out) are then assigned to the tester. The tester in turn performs the steps as documented in the test script and manually records which test steps were completed, which verification points passed — and which failed. Other information captured during the manual test execution process may include whether a defect discovered is blocking the tester's ability to complete the testing or if any other observations were made. This is also a great time to capture any discrepancies or possible errors in the test script itself. The test results are then returned to the SAP project test manager or test coordinator.

An SAP project test manager is a member of the testing team who has good knowledge of quality management as well as of testing methodologies, the tools available to run testing procedures, and of *defect management* — the process that focuses on preventing defects, catching defects as early in the process as possible, and minimizing the impact of defects. In addition, a competent SAP project test manager also displays a deep understanding of all aspects of SAP applications — from the various SAP components and implementation methods, to the integration between the various application components, to an appreciation of the role SAP applications play with the business domain itself.

Automating your testing

Automated testing has often been overhyped as some kind of a silver bullet that would solve any organization's testing problems. We're here to state two important truths:

- Not all tests can be automated.
- Not all tests *should* be automated.

However, we're saying that automated testing should play a significant part in your testing strategy. Applying an 80/20 approach to automation is a good rule of thumb. (***Note:*** 80 percent means 80 percent *automation*.) Automated testing is sure to speed up your test execution. Together, automated testing and service virtualization can produce tremendous time savings and allow teams to stand up realistic test labs quickly and begin testing the application end-to-end earlier.

Learning to Live with Multiple User Interfaces

In today's world, the same business process or application functionality may be available to the end-user through a number of different user interfaces (UI) deployed on many different platforms. Perhaps an end-user uses a rich client interface while at the office, accesses the same application via a thin client (browser) while away from her desk, or needs to do some last-minute work from a mobile device. The multiple channels still need to be tested individually, and a single test script doesn't lend itself to test everything.

To get a sense of the lay of the land, check out the following UIs, which are quite common in SAP solution implementations:

- **SAP Graphical User Interface (GUI):** SAP GUI is the graphical UI installed on the end-user's desktop to access the *back-end system* of an SAP R/3 deployment — the application server and database, in other words.
- **SAP UI5:** UI5 is an extensible library provided by the SAP development team that removed the need for organizations to roll their own custom frameworks when building mobile and web interfaces for their SAP implementations.
- **Custom UIs:** Many SAP implementations have dependencies on other systems, such as databases or services, resulting in the integration of SAP components with non-SAP components. Such custom applications will inevitably need access to SAP systems to retrieve data, making this another channel where the interface presented to the user requires testing.

In cases where the same business functionality is presented in a variety of UIs or channels, teams should consider modularizing the same test steps into a single test script and maximize reuse. You may know this as *keyword scripting*.

Navigating a Changing SAP Landscape

When working to manage quality and test your SAP solution deployments, it would serve you well to be aware of how SAP landscapes and deployment models may change over time. Change is inevitable — being prepared for that change, however, is never a sure thing. You need to be in a position to make the right decisions early on when radical changes threaten to disrupt your business model. Staying aware to what the future may bring will go a long way to determining whether such changes require a complete overhaul of your test assets or whether the changes to the test assets are minimized.

Dealing with business changes

Large organizations around the world are pursuing enterprise transformations designed to streamline their business as well as their IT landscapes by leveraging prepackaged enterprise application suites. These transformations often replace a significant portion of the existing IT landscape and wouldn't be possible to accomplish without a prepackaged solution. Understanding which legacy components will remain and which ones will be replaced plays a crucial role in effective change and quality management.

Without a solid quality management foundation offering traceability across requirements management, test management, and change management throughout the entire development life cycle, many organizations would struggle in understanding what tests need to be updated, which test scripts need to be created, and which tests need to be run to protect the interests of the business.

Dealing with technology changes

As you might expect, we are of the opinion that the effort to test SAP solution deployments is both huge and often underestimated.

SAP project rollouts are typically large discrete projects with many dependencies on other subsystems deployed in heterogeneous environments. The ability of the *composite* application (SAP applications plus the complex collection of dependent systems) to maintain a system of record while accessing multiple data sources is critical. (When that ability is lacking, you can never really ensure that your business will continue to operate without issue.) Organizations need to ensure that the exchange of data between these disparate systems is working as expected by executing integration tests.

In addition to testing the integration points between your SAP components and non-SAP components, you may choose to run a full set of regression tests on the system as a whole. In this context, *regression tests* are tests designed to ensure that the applications that worked before continue to work properly after implementation.

As always, forewarned is forearmed: Knowing the impact of change to workflows, reports, interfaces, conversions, enhancements, and forms as a result of the upgrade can go a long way toward helping test teams understand what new test assets, test cases, and test scripts need to be created.

Dealing with legislative changes

At times, changes required by new legislation may force organizations to make modifications to their SAP landscape — and those changes must of course be tested. Existing SAP implementations must be tested for functionality, performance, security, user experience, and so on. More importantly, any change mandated by legislative action must work, or the business could be at risk of some type of legal action. IBM's quality management solution can help the SAP project test manager manage and prioritize the test effort by delivering the information they need to understand which tests are needed to check which business process or process steps. Without this insight, teams could end up in a situation where they're running the wrong tests or creating/running unnecessary tests. Such missteps could potentially delay a release, which could potentially result in hefty fines.

Chapter 3

Developing a Testing Game Plan for SAP Projects

In This Chapter

- Choosing the proper business scenarios
- Coming up with a test execution and reporting strategy
- Knowing what to do when things go wrong or plans change
- Ensuring the proper team spirit
- Making sure your testing covers all the (environmental) bases

Like many things in life, such as renewing your driver's license or seeing your doctor for that annual health checkup, testing SAP solution deployments, upgrades, or new business functionality tends to take longer than you might expect. Given that fact, the more effort you put into planning your deployments — including foreseeing all the steps that may slow you down — the smoother the process will be. Defining your strategy early on and determining your reporting needs will help keep the team on task and deliver the necessary feedback to your stakeholders.

Knowing Which Business Scenarios to Test

By using the Business Blueprint, a feature of SAP Solution Manager that you can use to document business scenarios supported by an SAP system, you can come up with a common strategy of how your business processes are to be mapped into one or more SAP applications. Your Business Blueprint (if you have one) details the scope of business scenarios,

business processes, process steps, and the requirements of an SAP project implementation.

In an SAP project, a Business Blueprint brings together the following structural elements into an ordered hierarchy:

- Organizational units
- Master data
- Business scenarios
- Business processes
- Process steps

You can assign content — project documentation, for example, or business configuration sets (BC Sets) or transactions — to individual structure elements with the help of SAP Solution Manager. For more on working with SAP Solution Manger, see Chapter 7.

If you're using SAP Solution Manager, then specifying a Business Blueprint is a necessary first step for testing. It allows you to control two crucial aspects of your strategy:

- **Configuration:** When you configure your business processes with reference to the Business Blueprint project structure, you are essentially drawing a detailed map of how the business expects to operate. With that information in hand, you're in a better position to determine what should be tested and where.
- **Test organization:** Your test plans can be generated based on the Business Blueprint project structure. The transactions that you assign in Business Blueprint process steps are then put into test plans when you generate them, and you can run these transactions as transaction function tests.

If you use IBM Rational Quality Manager (RQM), you can import the Business Blueprint to an RQM project area and automatically create requirements, test plan, and test cases. The auto-generated test assets are linked back to the original Business Blueprint hierarchy, providing full traceability — achieved by creating links and relationships between project artifacts — across business requirements, test cases, test results, and all the way down to any associated defects.

Preparing Your Test Execution and Reporting Strategy

If showing up is half the battle, then showing up with a proper plan of execution means you're three-fourths of the way there. The next sections spell out how you can prepare for testing success.

Test execution

The test execution effort is highly dependent on the number of test cases that have to be executed in a complete system test. The recommended approach is to create one test case for the entire business process. The test case is assigned at the business process level of the Business Blueprint.

The advantage of this approach is that you end up with a smaller number of test cases, since test cases are now created at the level of the business process, or its variants or flavors, drastically reducing the test case development and maintenance effort.

An alternative approach is to create a test case for each process step of an end-to-end business. With this approach, each test case is assigned at the relevant step of the business process in the Business Blueprint.

The advantage of this second approach is that test results and associated defects can be managed at a lower level with communication, status reporting, and root cause analysis being much more transparent. However, this approach might result in an extremely large number of test cases, where test case creation and maintenance effort could potentially outweigh any benefits.

Reporting

With test status reporting in IBM Rational Quality Manager, you get a summary of the test results for the reporting period, including analysis of result information. The summary describes how well the test execution is progressing.

IBM Rational Quality Manager comes with a dashboard that you can use to track your test plan and its execution progress in real-time throughout the testing cycles. These dashboards

are highly customizable and can be easily adapted in their appearance. Predefined and customized queries help you filter the right data from the project so you can quickly collect the metrics required for a test status report.

The test status report identifies those activities completed in the current reporting period and those planned for the next. Some of the measurements produced by this report include the following:

- Number of defects found
- Severity levels of each defect
- Status of defects (outstanding, investigated, explained, fixed, retested, accepted, and so on)
- Rate of defect arrival (in other words, defects found per test executed and/or per tested product component, organized by test period and level)

The report also includes information on planned tasks, planned schedules, the actual status of both tasks and schedules, the forecasted outlook, issues, recommendations, and an action plan. With this information in hand, you are in a position to

- Review actual status against plan
- Identify and address constraints, concerns, and issues
- Identify items requiring containment and/or escalation to resolve issues

Appropriate consolidation and filtering of data is necessary to ensure that key information is both evaluated and assimilated at the respective planning meetings. You'd then be able to use any statistically valid conclusions to predict the quality level achieved by the tested application and compare that level to the target level documented in the test plan.

Dealing with defects and change requests

In all the tests we've ever run, we've always ended up with two main results:

- **Test logs:** These contain the test results with all your pass/fail information
- **Defects:** These contain the information and details about a problem with the application that has been found during testing — typically a deviation from the requirement's expected result or outcome of a test.

You need to record defects in a defect management system. A proper defect management tool, such as IBM Rational Team Concert, provides a workflow engine that can help project teams track defects and manage requests. Inefficiency in managing defects and change requests means you'll most likely end up spending a lot more time than anticipated in order to resolve problems. Team members are unable to work together effectively, which often results in development and testing efforts being duplicated. Lacking an effective tracking tool can also result in poor quality as important changes are lost or missed because of a lack of prioritization.

Many organizations that run on SAP applications require *heterogeneous environments* — environments that make use of both SAP components and non-SAP components — in order to deliver the end-to-end business process. Often a change request to a business process will require different teams to do work to deliver that change. As a result, you often need to parcel out the responsibilities for managing and testing a change to different parties.

This parceling out often leads to unclear ownership and a lack of visibility in the overall process. Adding to the complexity, managing and testing these application changes is frequently performed using different change control tools, resulting in quality management that's often unstructured and dispersed, with insufficient sign-off procedures. Additionally, due to a lack of sufficient automated testing, whether that be functional, integration, or performance testing, the testers are unable to keep up.

To make matters worse, reporting across all the various concurrent activities can be really difficult. All this is meant to say is that change is hard, but we want to assure you that it shouldn't represent an unbearable burden. Delivery teams with both SAP components and non-SAP components who adopt common processes and tools for quality and change management reap the benefits of

- Shared requirements definition
- Familiar workflows
- Consistent defect management
- One source of the truth reporting via real-time dashboards

The next few sections spell out in a bit more detail how such a collaborative process would work.

Collaborating with test partners and teams

It is common on SAP projects to work with multiple teams and partners. Teams from different organizations need to be able to work together effectively, following standardized test processes and leveraging a common technology platform. Geographically dispersed test resources need to be able to communicate and collaborate effectively, wherever they are in the world and whatever their working hours. Additionally, the quality/test manager needs to be able to monitor the progress of any outsourced test work in real time.

For complex SAP testing projects to work successfully, the key is collaboration. Testing teams must be able to communicate and share test data, tools, scripts, and even skills both internally and with each other — in an appropriate and governable manner. To achieve this, you must have a quality management platform that is *open* (allows other tools to plug into the platform) and can ensure consistency and traceability. This will enable all parties to have confidence in how data will be shared and metric reporting can be assured.

A secure, strong, reliable, and open technology backbone is usually the answer. An open quality management platform makes it simple for each member of the team to find the information that is most relevant to them — for example, the business owner might want instant access to overall project status reports, while a tester would prefer specific information on the tasks assigned to that individual and their immediate team.

Collaborations can be difficult to manage. To keep your headaches at a minimum, keep the following in mind:

- **Make onboarding easy:** Bringing new team members onboard when new or enhanced skills are required needs to be fast and easy. This will give organizations the flexibility they need to leverage business and technical talent wherever it is located.
- **Ensure governance:** Focus on delivering real-time insight into testing on SAP solution programs, projects, and resource utilization to help teams make better informed decisions and continuously measure progress against desired business outcomes.
- **Be realistic about standardization:** When working with outsourced partners, it can sometimes be challenging or expensive to insist they use a particular tool. The answer is to provide a quality platform that is open for them to plug in or connect to. IBM's solution is built on open Web and OSGi standards and provides an open and extensible architecture designed to provide the flexibility to assemble a quality platform that can be tailored to the unique needs of any test process leveraged by required outsourced providers, vendors, and partners.

Testing all business scenarios across both SAP systems and non-SAP systems

As non-SAP products, applications, and vendors are added to the IT landscape, companies must undertake extensive integration work and testing to ensure that all components function correctly. As the IT landscape grows in complexity, managing such cross-system integrations can be a difficult task.

IBM fosters collaboration

A successful collaborative quality management solution needs to be built with open, flexible services and on an Internet-based architecture. IBM achieves this by delivering a collaborative life cycle management solution, including quality management, that's built on an open platform. This enables diverse tools to be used together, providing users with an integrated experience.

Where dependencies exist between SAP projects and non-SAP projects, SAP delivery teams must keep in sync with other teams' release cycles, enabling them to manage projects holistically. Without close collaboration and a shared toolset for change control, ad hoc IT changes or application releases can cause business disruption.

The answer, of course, is to manage and test SAP projects and non-SAP projects in a unified way. This means you need a comprehensive and automated approach for requirements, quality, and change management. IBM Rational Quality Manager integrates seamlessly with SAP Solution Manager, allowing testers and developers to exchange SAP Business Blueprints, test results, and incident information quickly and easily. The business scenarios from the Business Blueprint may depend on SAP technology and non-SAP technology, but the implementation of the test plan, test case, and test scripts — as well as the reporting and analysis of the test progress and results — is not constrained by any particular technology or vendor.

There's just one hitch: In order to achieve this seamless testing of business process across SAP systems and non-SAP systems, you need a functioning system to test — not always a given. We suggest you consider the option of deploying virtualized services, software, and applications to fill any gaps in the end-to-end scenario during testing. The solution from IBM means you can accelerate the delivery of any complex test environments and enable integration testing to begin earlier in the development cycle, to be performed more frequently, and to be completed faster.

Making Sure You Can Test Across All Environments

The test environment is the environment used for testing — no surprise there, but keep in mind that it should be set up to resemble the production environment as closely as possible. In addition to the system under test, it also contains the software tools for test execution (manual and automated), simulators, and other supporting software and hardware to conduct a test. The test environment may also include additional items, such as the facilities, processes, and personnel required to carry out the test.

The setup of a test environment is always an area of potential high risk, particularly from a scheduling point of view. If the plan for setup does not cover all aspects of selection, acquisition, and training for new components, or is missing components, there may be significant delays in beginning the actual tests. If the setup is not comprehensive and misses key configuration items, there may be significant schedule setbacks. If risks in the setup are not identified and managed, they may derail the entire project.

Documenting the setup in the form of a test environment specification is crucial. Such a specification helps ensure that

- The results of testing activities are accurate and valid.
- Testing activities have a higher likelihood of being systematically reproduced.

A test environment specification documents the processes and tools required to set up, operate, and maintain the infrastructure, particularly any processes and tools that are unique to the test environment. It often includes documentation of any differences from the production environment, likely impacts on testing, and the risk management plan for such impacts.

Items covered in the test environment specification include

- Project/Application identification
- Contacts
- Schedule
- Availability of the test environment
- Test platform requirements

For each node (mainframe, server workstation, and so on), you'll want to specify test platform requirements for the following:

- Hardware
- Software
- Voice/data network
- Security

- Tools and Utilities
- Applications
- Internal/external application interfaces
- Data
- Facilities
- Personnel staffing to establish and maintain the test environment

Important considerations for any test environment are

- How many environments will be required, and how will the code, test data, and infrastructure components be managed, transferred between them, and maintained?
- Will there be separate environments for each level of testing? If not, how will the testing be isolated to keep one level from interfering with another?
- Configuration management is an essential process. The test environment's documentation requires automated assistance.
- Test environment and configuration automation will also help in managing backup and restore functions for testing. A key element often forgotten is the capture of the environment at pre- and post-test states. This capture is essential for efficient test results re-creation, variance investigation, and auditable testing.
- At the conclusion of testing, enough environment data should be captured to enable reconstruction of the environment to repeat any tests. This requirement may vary directly in proportion to the risk involved and any legal or statutory considerations.
- After the test environment is established, the environment build team should conduct a set of verification tests to ensure that the environment is operational and ready for the test team to conduct their tests. The use of a setup checklist is helpful.
- At the conclusion of testing, once again the use of a checklist is helpful to ensure that all test environment teardown activities have been completed.

Chapter 4

Getting Started

In This Chapter

- Setting up the proper environment
- Working out an effective test management strategy
- Ensuring your test execution is flawless
- Aiming for excellence

SAP solution implementations are very seldom done in isolation from other systems the organization may already have in place to drive their business. When you consider all of the technologies involved as well as the diversity of the protocols and data shared across such a heterogeneous environment, it's clear that a truly effective testing effort will be far from trivial. And with such a large effort comes confusion on where to start. We're here to tell you where your team should focus their attention early on so they can be sure to reap the benefits of effective testing in the later stages.

Knowing Where to Start

First things first: The trick to ensuring success in any SAP project is to make doubly sure that your testing techniques are able to assess the risk of implementing change.

In the context of an SAP project, this means being able to test all possible business scenarios, as spelled out in Chapter 3 of this book. We know a common response to such a piece of advice is to exclaim "Easier said than done." An equally common response is to ask "Where should I start?"

Here's the beginning of an answer: To work out a successful end-to-end testing solution for SAP projects, you need to focus on three parts of the implementation process — your test environment, your test management, and your test execution.

Providing a test environment

The test environment needs to be realistic and as complete — *production like* — as possible. In order to test all business scenarios, the complete test environment must include all required SAP technology and non-SAP technology. Making all the required hetergenous services and/or applications available for testing requires the virtualization of any missing parts of the test environment. By ensuring the team can virtualize all or any of the test environment, you ensure that the testing team(s) can have access to the exact test environment they need when they need it. This *service virtualization* of your SAP landscape is your necessary first step in implementing a successful end-to-end testing solution.

Preparing your test management

Quality management is much more than simply executing tests. The requirements, planning, execution, reporting, and analysis needs to be supported by a collaborative hub for all appropriate stakeholders, whether they be the business owners, technical teams, outsourced partners, and/or vendor support. The idea to is have a central space that can be used by all to plan, to share information, and to review the latest quality status of a project.

Test management includes the identification and support for a testing process. Whenever possible, it is usually most effective for test teams to follow an *iterative* testing process, where testing early and often is the key. Doing so integrates quality management and continuous testing across all stages of the project work flow, as opposed to relegating test activity to the end of the project.

Changes to an SAP landscape may come from implementing a new SAP application, an upgrade, an enhancement pack, a new or revised business process, or something else entirely. Part of preparing test management is defining how to

determine when the level of risk associated with that change is sufficiently reduced (and maybe not completely eliminated) to align with the time-to-market objectives of the business. Sometimes, changes to business processes require more retesting than other change factors, so the test management hub also needs to be able to help the test team minimize the cost of rerunning tests. It should do this by identifying the minimum amount of re-testing that needs to be done following a change to one or more business processes. This is known as *business process change analysis.*

On an SAP project, an important consideration for test management involves, first, whether or not the team will utilize SAP Solution Manager, and second, if such a utilization is desired, how precisely that will take place. The test management hub needs to be able to connect and link to requirements and dependencies from the entire business, technical, and enterprise landscapes. From a business process perspective, this means understanding how SAP Solution Manager is being used to manage business processes and how the team intends to interact with vendor support.

Finally, an important activity in preparing for test management is thinking through how to utilize any existing investment in testing skills, tools, and data that may already be in use. Some organizations opt for a "Big Bang" approach to implementing continuous testing, meaning they introduce a set of standardized test processes and tools to as many people and projects as possible at the same time. This is often to try and reduce the training costs and to keep disruptions confined. However, organizations are typically dealing with projects that are already in the process of testing and a Big Bang approach would disrupt and add risk to those projects. To address this, the organization must select a quality management tool that is built on an open platform that can allow different test execution tools to plug in for common and standardized test planning, management, reporting, and — most importantly — collaboration.

Executing your tests

Having established a test management protocol, the test team needs to prepare to start test execution. As part of that preparation, your test team must decide what type of test execution is required. Your choices are as follows:

- **Manual testing** as the lowest level approach to testing where each step of the test script is run by a human instead of a test automation tool.
- **Integration testing** to ensure all required dependencies work as expected.
- **Performance testing** to validate the performance and scalability of the business process and/or business scenario.
- **Business process validation** to ensure the business will run as expected even after complex changes.

The testing capabilities from IBM support testing across a broad set of middleware, messaging technologies, applications, and industry-specific requirements. This is the only way business objectives and test execution can be aligned and an assessment of risk made visible to all necessary stakeholders.

So, that's the 1-2-3 punch to prepping your successful end-to-end testing solution for SAP projects. In the next section, we want to offer some practical advice about the products and capabilities that can help you turn your plans for an end-to-end testing solution for SAP projects into a successful reality.

Creating Your Own SAP Testing Center of Excellence (CoE)

Having all things test-related under one roof (whether that roof is literal or virtual) makes sense because it is easier to ensure quality when activities are concentrated in one place. SAP, for example, has its own Testing Center of Excellence (CoE) in Walldorf, Germany, where it uses testing tools from IBM Rational as part of its testing strategy. Many organizations want to set up and run their own Testing CoE for their own SAP projects, so here are the key things to consider to ensure sustainability of your own Testing CoE from day one.

Preparing a flexible test environment with flexible access to testing tools

One of the most frequent comments we hear back from organizations that set up and run their own CoE for their SAP projects is that the demand for their services can suddenly increase and they then struggle to respond. Not only can it prove difficult just to provide proper access to testing environments, but it can also be a hassle to provide enough licenses for test tools. Given these facts, it is important to prepare for access and licensing issues from day one.

From an infrastructure point of view, the key to sustainability is to ensure that your teams can virtualize any missing part of the testing environment at any time. Every team must be able to get access the test environment they need when they need it.

From a test tool licensing perspective, the key to sustainability is enabling practitioners to use the right tool at the right time, as well as factoring the need for additional tools that are potentially required later in a project life cycle. IBM addresses this challenge by providing *license tokens* — a flexible method of licensing the end-to-end test tools needed by a testing CoE. Tokens have a predefined value that replaces the traditional view of a static license quantity. The CoE then maintains a pool of tokens available to users so they can deploy the right IBM product at the right time throughout the life cycle of the project. Often testing CoEs charge other departments for use of their services or offerings and so manage the token pool by role or department or project.

Preparing for different approaches to testing

If there is not a corporate approach for testing, then teams may request support for different testing processes, such as agile or waterfall. If there is a corporate standard, then the testing environment must support governance and templates,

but if it turns out that project and test managers have the power to choose their own approach, then the testing environment must be able to adapt for that particular project.

A testing CoE supporting SAP projects, needs to be ready to support SAP Solution Manager. The specifics about how a project SAP Solution Manager can vary significantly. Indeed, the way a project wants to utilize the information stored in SAP Solution Manager for the express purposes of testing may vary. IBM and SAP have worked together to addresses this challenge and have come up with a simple, flexible configuration interface that allows teams to choose the relationship (and automation) between information captured in SAP Solution Manager and how it is used in the rest of the testing process. This flexibility allows your testing CoE to support SAP projects that want to utilize SAP Solution Manager.

Playing well with others

It is common for SAP projects to outsource part of a project to one or more partners. This could mean that different testing activities are parceled out to various partners, using different tools.

In such a world, flexibility is key. To be successful, you must ensure that the testing CoE is open enough to allow other organizational (or individual) teams to plug in their own tools or testing environments as appropriate for their work. This way, the testing CoE maintains governance and control while maximizing flexibility.

Chapter 5

Getting to Know the Components of Continuous Testing for SAP Projects

In This Chapter

- Going for an "all testing, all the time" model
- Checking under the UI hood
- Getting your priorities straight
- Testing your various components

Every test you execute probably focuses on shifting the risk of software delivery further to the left on your project timeline — that "shifting left" we talk about in Chapter 1 of this book. The aim for all testing is to test as thoroughly as needed and as early as possible — well before you deploy into production. In this sense, it pays to think of software testing as the headlights of a car — it allows you to see what's coming on the road ahead in your development life cycle. With this capability, you can make better-informed decisions around your system as you're building it. The better the headlights, the more confidence you have. So where to start?

Testing on SAP Projects Earlier and Continuously

Business demands mean SAP implementations operate in a distributed and highly integrated environment where stakeholders of a business process can be both internal and

external. This forces you to think of applications as the sum of their parts, and testing is no exception to this rule.

When building a test plan for a new application or a plan to test a change to an existing application, it is important not only to think about logic, data, and exit criteria but also to understand the mechanics of how the application works from an *integrated* perspective — more specifically, how the system is architected and how that affects your ability to execute a chosen set of tests.

With complex, distributed systems, the primary risk is integration. It is true that SAP systems are designed to talk to other systems — other SAP systems, systems incorporating components delivered by other vendors, or even systems that don't reside within your enterprise. But even the best designed systems can run into problems, so you need to test these integrations as early as possible and as continuously as possible. The problem is, you're going to encounter a tapestry of differing standards, overlapping testing schedules, conflicting methodologies, and contention around test environments when trying to move forward with your testing strategy.

Testing below the Presentation Layer

The days of solely relying on testing via the user interface (UI) are long gone. The model you are presented tends to have several layers, with the UI often being the outermost. Oftentimes, the majority of the risk lies at the integration layer. The integration layer may contain the business logic, data dependencies, and various technologies.

Automated integration testing below the UI provides great benefits to projects, uncovering integration defects early in the development life cycle. In fact, testing below the UI is a fundamental tenet of any "shift left" strategy.

The principles for automated integration testing of SAP scenarios are the same as those for UI testing; you execute test steps and compare what happens to an expected result. There are however, some differences to bear in mind:

- **You'll have multiple message types and permutations of data to validate.** See Figure 5-1. It's not feasible to do all this manually. To get satisfactory coverage, you need to automate.
- **The test steps are contained within the payload of the message.** Payload can be data driven in much the same way as automated UI tests.
- **Sending a message via the integration and validating the response do not always provide full confidence in a particular scenario.** Integration testing introduces the concept of testing what can be referred to as side effects. In other words, what else happens when I send this message? Is a database written to as expected? Is the log file updated as defined? Test tools, like IBM Rational Test Workbench, allow you to validate at this level.

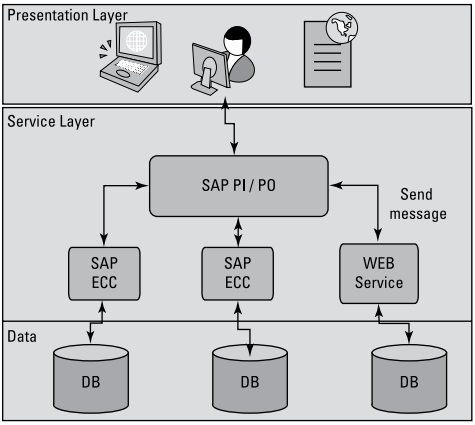

Figure 5-1: Seeing what lies below the user interface.

Service virtualization

In addition to testing below the UI, the other key development in the testing of distributed SAP systems is *service virtualization*. In a typical end-to-end process, you may encounter a number of distinct architectural components, which may be delivered on different timelines and be owned by different teams. Virtual services can stand in for these components in your test environment when they are unavailable, as shown in Figure 5-2. This introduces greater flexibility into the testing

approach. With service virtualization, you can plan to test around availability bottlenecks and also plan to test much earlier than would have traditionally been the case.

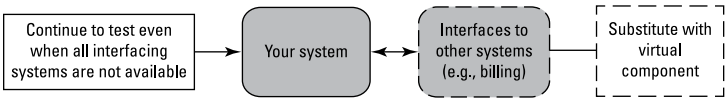

Figure 5-2: Virtualizing components.

Testers in an SAP solution environment should consider two basic scenarios where they can use service virtualization:

- The virtualization of SAP components
- The virtualization of non-SAP components

Working beyond the firewall

Service virtualization is clearly a great tool to have when working with a complex, heterogeneous internal system, but it also can be extremely helpful when you have to deal with a system that sits outside of the corporate firewall, say a third-party system or a business-to-business (B2B) interaction. In the bad old days, when you wanted to test, especially when you wanted to test early and end-to-end, you were at the mercy of the priorities and availability of another company. These rarely aligned with your own goals and timelines and caused delays — often weeks of delays — while you waited for a suitable slot around availability of those environments (systems) and the resources to provision and manage them. With the coming of service virtualization, you can sidestep at least some of these barriers and thus mitigate the risk to the project.

Shifting Left and Prioritization

The ability to test below the UI and create virtualized services has changed the acceptable risk profile for the delivery of SAP systems. These two new approaches mean that you can approach the prioritization of test cases much more aggressively. As tempting as it is to execute the easy tests first, try to avoid the low hanging fruit until later on. Prioritize on the

more technically challenging and commercially beneficial functionality first and expose the bulk of the risk early on in the software development cycle — shifting the timeline left, as it were.

It used to be the case that we'd put off testing of a particular functionality until "later on" because a component in the end-to-end test scenario was not available — the SAP Process Integration (PI) server or SAP ERP Central Component (ECC) server, for example, or a non-SAP component such as a web service. With service virtualization on the scene, being able to virtualize an unavailable component means bottlenecks can be avoided and test managers can plan to execute their tests as early as possible.

Integration testing and virtualizing the key SAP components

When tackling automated integration testing, you inevitably have to deal with two major components in a typical SAP landscape: the PI server and the ECC server. In addition, you have various adapters that are used to integrate the message protocols and middleware. These are important not only because they offer test and virtualization points but also because the tester must think about whether testing the functionality of the adapter is an important part of the test case.

In the next few sections, you take a look at a typical automated integration testing process, where you see the recommended approach for the high-level aspects of the landscape — PI and ECC — and then move onto more granular considerations around the messaging patterns. You look at the specific types of testing SAP integration testers might do, why they might want to test in this way, and possible approaches they could take. In each case, you'll be looking at the specific tools testers would have at their disposal, including discovery and recording. These concepts apply just as equally to SAP technologies as well as a range of non-SAP technologies.

Testing and Virtualizing PI

SAP Process Integration (PI) acts first and foremost as an *orchestration engine* — a tool for modeling and executing business processes — but it also provides a bridge between the proprietary technologies used by the ECC and the open standards used in a typical service-oriented architecture (SOA) enterprise. SAP PI accomplishes all this using the standard pattern of adapters, as shown in Figure 5-3. On the way into SAP PI, messages are first adapted from incoming formats and technologies to an internal XML representation known as XI to be processed and then adapted back into outgoing transports. In this way, an EDIFACT message — an electronic data interchange (EDI) conforming to a UN-sanctioned standard — can be received over the message queue (MQ), processed, and then sent as an intermediate document (IDOC) into ECC.

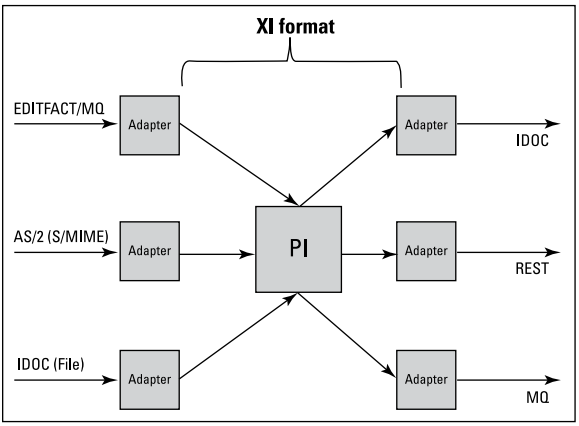

Figure 5-3: The pieces of the PI puzzle.

A PI system can contain two distinct technology stacks: SAP's own Advanced Business Application Programming (ABAP) language and Java. The ABAP stack has similar functionality to the remote function call (RFC) and IDOC features of an SAP ECC system, so we'll cover that in the ECC section later.

From a tester's point of view, testing PI means testing that the business processes (rules) captured within it are operating correctly, and any side effects are as expected. This is executed by sending messages via the adapters and verifying the results. As PI will be both retrieving information from

and sending information to other systems, it is common for a tester to virtualize those downstream systems to provide a controlled, deterministic set of test conditions. (This type of virtualization is called *service virtualization*; we address that particular topic in greater detail in a bit.)

Discovery and SAP PI

Within a typical large organization, an SAP PI system will be orchestrating between large numbers of systems over a wide range of technologies. The ability to discover and record is very important in an environment such as this in order to test quickly and efficiently without environment dependencies.

Discovery is the process by which you bring in information about the SAP environment into the test tool. That information could be environment variables, business flows, or identification of downstream dependencies. With this information, you're able to bring recording business processes as they flow across the SAP environment.

Recording allows the tester to non-destructively pull messages from the system as they flow through. These messages are stored and manipulated within the tool. From these recordings, testers can create automated integration tests and virtualized services.

Traditionally, testing tools in an SAP PI context were tasked with discovering RFCs and IDOCs in the ABAP stack, but test tools today can now discover a whole host of other information. Generally speaking, this discovered information can be divided into two categories:

- **Services that the PI system is hosting:** A tester may want to test hosted services (to check that PI is orchestrating correctly) or virtualize them in order to make testing of a dependent system easier.
- **Services that the PI system is dependent on.** These are services that PI uses to fulfill its orchestrations or that it passes information to. A tester is interested in such dependent services because they represent *validation points* (you'll want to test that PI has sent the correct message to a downstream system) and *virtualization points* (you'll want to virtualize systems that PI is communicating with to gather information in order to make the environment simpler and test cases deterministic).

An SAP PI system contains a number of interfaces that allow test tools to discover the landscape that the PI knows it is operating within. If testers are using a tool that can properly interrogate these interfaces, they can quickly discover the landscape and start recording real interactions that occur between the SAP PI and connected components. For example, this capability should allow them to discover web services and import Web Services Description Language (WSDL) files, identify queue managers for MQ/JMS communication, and identify which databases PI is interacting with.

Recording PI interactions

The fastest way to create tests, test data, and virtualized services is to record real interactions with real systems being tested. With an SAP PI system, you have two recording options, depending on which adapter architecture is present.

For all versions of SAP PI, your first choice is to record the message stream coming into or out of the PI adapters. This could be an MQ or JMS queue, SOAP, REST, or XML message over HTTP. Most automated integration testing tools have a way to record these messages and to use those recordings to build tests. The recordings can provide the template for messages to be injected into the adapters to test PI as if they had come from an external system. In addition, such recordings provide a template for the validation of message structure and content that the PI sends out to other systems.

Virtualization with SAP PI

Testers should start by thinking about what they are trying to achieve with virtualization. Do they want to get on with testing even if part of the environment isn't ready yet? Or do they want to get a service to respond in a particular way to satisfy test conditions? Or is the idea to introduce errors or delays for negative testing? After figuring out precisely what it is you want to achieve, you'll be in a better position to identify the most appropriate virtualization point.

Deciding where to virtualize calls for a book in itself — in fact, there's already one out there, appropriately entitled *Service Virtualization For Dummies* — so we can't devote as much space to the topic as we'd like, but we'd at least like to look at two scenarios:

- **Virtualizing the entire SAP PI system.** In this scenario, the tester is testing a system that *depends* on the SAP PI system. It may be using SAP PI to pull information from a variety of sources, or it may be pushing data to SAP PI to update other systems. Testers want to simplify their environment and create deterministic test cases. Virtualizing away the entire PI system lets them do that.
- **Virtualizing other systems that SAP PI is communicating with.** Generally, you'd take this approach because the tester wants to test the orchestrations within SAP PI itself. These orchestrations can be complicated, as they tend to communicate with other downstream systems. Setting up all of these systems to provide just the right results can be a time-consuming and complex process. A tester can create virtual services to represent those systems and easily create test data to force the orchestration into various test cases.

Testing SAP ECC

The ECC system will be at the heart of many organizations. Although an SAP PI server is set up to provide the SOA-based communication and orchestration capabilities you need, in many cases it's also acting as a bridge to allow non-SAP systems to work with ECC. See Figure 5-4.

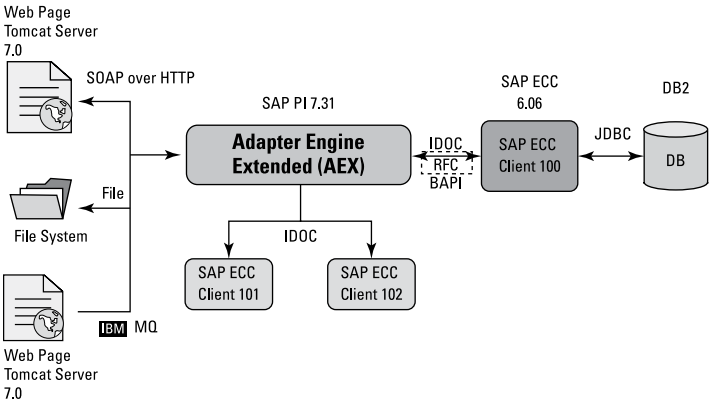

Figure 5-4: SAP ECC and PI in a sample architecture.

There are times when a tester needs to test the ECC system directly to ensure that the core is working before stepping back and testing the outer layers of the integration-onion. Testing the outer layers is easier if you can virtualize away certain aspects of the ECC system too, so you'll want to look at that option as well.

Given that you're not working at the UI level, you have three ways to interact with an ECC system:

- Call RFC in order to cause some behavior or check some data.
- Publish an IDOC into the ECC system in order to cause some behavior.

  "Publishing" here can include writing an IDOC file to disk to be imported into ECC.
- Watch the side effects of the above two actions, which include IDOCs being sent from the ECC system to other systems.

Let's start with the simplest use case, testing an RFC.

Testing and virtualizing with RFCs

An RFC call can be viewed as a simple operation that exposes some ECC functionality to the outside world. It's similar to a remote procedure call or stored procedure call in that you pass in some data as structured arguments and get some data back. There are typically two reasons for testing an RFC: a new one has been written (or an existing one changed) or for regression testing as part of an SAP system upgrade or other system change.

There are two aspects to the discovery of RFCs. An ECC system is sure to host hundreds of RFCs, each with its own signature. A testing tool must allow a tester to interrogate the ECC system to discover details of these RFCs and make it easy to construct an RFC call and to understand the results. Typically, RFCs are logically grouped together into business application programming interfaces (BAPIs), and discovery is often easier if a tester can drill down to RFCs from the BAPI level.

The other aspect to discovery comes into play when you're testing a system that depends on ECC — an application that makes RFC calls in order to retrieve or modify data from the ECC system, for example. At this stage, don't worry about testing the ECC system proper; in fact, it may be easier if you just virtualize the ECC system away to make testing the dependent system a much simpler task. Without the ECC system complicating things, you not only remove the need for a "just right" environment but also you're now able to create virtual RFCs that give simple, deterministic results in order to run the correct test cases.

In this scenario, the discovery process means making it easy for testers to find out exactly which particular RFCs the application they're testing is using — more specifically, which RFCs it's using for the test cases the tester cares about. You want to spend the minimum effort virtualizing just what you need in order to satisfy your test conditions.

Using the recording functionality to build tests

When building regression tests for an existing application, the best way to do so is to record the RFC calls that the application makes. This is a great way to start for two reasons:

- You only spend time virtualizing RFCs that the application actually uses.
- The recorded calls contain sample requests and responses that a tester can use to quickly build virtual RFCs.

It's also possible to use the recording to create tests you can then use to regression test the ECC system sometime in the future. These tests allow you to confirm that the ECC system is still working correctly in the event of an upgrade before you regression test the other applications in the environment. Again, discovery and recording accelerate the creation of integration tests that allow you to test core parts of the environment first — and in isolation — so you can find problems more quickly and cost-effectively.

Of course, it's possible that the RFCs you need aren't written yet. In this case, you of course can't record them, but the tester can still work with the ECC development team to understand the RFC signatures. From this understanding of the RFC call and response, the tester can create a virtual RFC to allow development and testing of the new application to proceed.

Testing: IDOCs

Most business processes within an SAP environment revolve around IDOCs. An IDOC is a structured collection of logically grouped information relating to a business process — details of a material movement, for example, or a change to employee details. An ECC system holds metadata that describes the structure of each type of IDOC; an IDOC itself is a set of data conforming to the structure defined by the metadata. This is very similar to the idea of an XSD schema and an XML document that conforms to this schema.

Given that the majority of business processes will be initiated by an IDOC showing up at the door of an ECC system, a very common task for an SAP tester is to test a particular process by publishing an IDOC into the ECC system. (For those familiar with other messaging technologies, this is similar to publishing an XML document into a JMS or MQ queue for a business process engine to pick up and process at a later date.)

But what are you actually testing by publishing an IDOC into the ECC system? That act in itself doesn't give an immediate result: it's a fire-and-forget situation, where the ECC system essentially just says, "Okay, I've got it." What the tester really wants to check in this scenario are two things: the final result and the expected side effects.

Looking at final results

At the end of the business process, *something* is going to happen — you'll end up with a result, and you'll want your tester to check that result. The result could be that the IDOC is augmented with additional information and passed to another system, or it could be that the ECC system creates one or more new IDOCs and sends them to a number of downstream systems to process. The final result could even be as simple as a database being updated or a file being written out.

IDOCs bring with them their own challenges. Typically, they're large documents containing a great deal of data. That means coming up with one for testing purposes can be a hassle. Let's face it; creating an IDOC by hand, from scratch, is time-consuming. It's a far better use of your time for you to track down a real IDOC and then use it as the template for your test.

This is where discovery and recording comes in. A tester needs to be able to discover the IDOCs that an ECC system is aware of — the ones it has metadata for, in other words. Typically, any EEC system is aware of quite a number of IDOCs, so testers need to be able to quickly search and filter to get to the IDOC they want to create.

However, this assumes that testers know which IDOC is being used for a particular business process. It also leaves them the not insignificant challenge of entering all of the data into the IDOC to create a valid test case. A (distinctly) better alternative is recording.

In the world of SAP solutions, recording means watching IDOCs as they're received by an ECC system and observing how they flow around the system and where they're sent. Testers want to be able to record copies of all of these IDOCs in their testing tool of choice for two reasons:

- They can now tell exactly which IDOCs are used in a particular business process and use this information to create their own test cases.
- Testers now have which can significantly accelerate test creation time.

Looking for side effects:

Watching IDOCs as they're received by an ECC system and observing how they flow around the system and where they're sent can be an effective way of testing for adverse side effects. In other words, simply publishing an IDOC into an ECC system isn't as meaningful a test as watching the IDOC as it moves through the process, checking the content at each stage, and checking that child IDOCs are created and routed correctly.

So far in this section, we've talked about scenarios where the complete environment exists. In other words, we've assumed the existence of a downstream system that the ECC can push an IDOC to as part of a business process. This isn't always the case — the system may not have been built yet, and even if it does exist, sending an IDOC to it may be undesirable (as it may have side effects the tester would rather avoid). In this case, the tester needs to be able to use the testing tool as the endpoint — effectively virtualizing the entire downstream system so that SAP ECC directs the IDOC to the testing tool

so it can be validated instead of to the downstream system. Typically, this involves changing SAP configuration parameters, which means you're going to have to dirty your hands a bit rather than being able to simply sit back and watch the flow.

Running the test

So, you've discovered and recorded the necessary IDOCs. You now have one or more IDOCs ready to use in a test. The time is ripe to construct some data-driven tests. This means identifying the IDOC fields that need to change to satisfy each test case and populating those from a spreadsheet at run time. This *parameterization* means you'll have fewer assets to maintain as test conditions change.

Finally, you'll need to consider how the IDOC will be published into ECC. Typically, this will either be via an RFC call or by writing to a file in a directory for the ECC system to consume. The test tool must look for results from this process, which could also mean waiting for a file to be written back to disk.

Looking at the big picture when testing SAP solutions

Virtualizing effectively is about understanding the environment, understanding precisely which part of the environment you're trying to test, and then understanding where virtualization should be used to make the testing task easier with the minimum amount of effort. Discovery and recording are vital tools in doing this.

With so many different types of adapters and technologies out there, the full range of virtualization possibilities is beyond the scope of this book. We can only recommend that you look long and hard at your options and opt for the one that can be achieved not only with the minimum of technical effort but also one that can be tailored to how much of the environment will be available during the test run.

Chapter 6

Making Sure Your SAP Deployments Perform as Expected

In This Chapter

- Ensuring the expected outcomes
- Understanding the importance of performance testing
- Capturing goals and user acceptance criteria
- Defining a strategy

SAP business processes rely on two fundamental elements — functionality and performance. Functionality is what the SAP application enables its users to accomplish, while performance is the system's ability to complete transactions and present the requested information rapidly and accurately. *Performance testing* of the system refers to being able to measure the system's response time and determine whether it is acceptable, whether processes are running fast enough, and whether a specific concurrent user load can be supported and maintained for an extended period of time. That is certainly useful, but performance testing is also designed to be more than a means to identify deficiencies in the infrastructure or to reveal opportunities to tweak configuration settings; the idea here is that performance testing will also help identify process bottlenecks in the code itself.

Keeping Your Focus on Performance Testing

Some teams may attempt to test and measure the performance of a system manually, but we can tell you that the manual route is not the way to go. Bringing in an army of testers equipped with stop watches is not going to capture all you need to know about your business processes, so you can rule out a manual approach right from the start. Automation of performance testing and implementation of a performance testing solution like IBM Rational Test Workbench combined with IBM Rational Performance Test Server (to generate the load) is the only viable approach to significantly reduce the time required to complete performance tests. Any other approach would be time consuming, labor-intensive, extremely inaccurate, not repeatable, and very expensive.

In addition to the manual testing crowd, there are those who simply ignore the need for performance testing at all, letting it be known that they just don't have the time or the resources to put in place a testing strategy. How many organizations have deployed into production only to realize the pain of system failures? The potential damage to the brand and loss of revenue as customers flock to a competitor should be enough to make sure this is not the norm.

Then there are those organizations who defer performance testing until later — at the last possible moment — as they wait for all the modules to have been deployed and the entire system to be available for testing. Inevitably, teams may be challenged in scheduling time to use a realistic test environment or delayed due to any number of interruptions. With respect to performance testing, procrastination often leads to expensive defects being realized far too late in the process. By that time, often the only recourse is to delay the release.

Don't let the bad habits of others lead you astray. Take a proactive approach to performance testing and begin measuring the scalability and stability of the system under test early in the development lifecycle. And don't be dismayed by any speed bumps that may pop up along the way. Software that is functionally sound, or perhaps presents some issues that can be resolved with workarounds, may be challenged in delivering the desired level of performance. Take comfort in knowing that these issues are being identified in your test labs and not

at the expense of all your customers who happen to be using the production system.

To make sure your deployment performs at an acceptable level, you'll want to set specific goals and identify what are the minimal and optimal configurations. Some goals you may consider adopting are listed below:

- Measuring system response times
- Meeting requirements and performance goals
- Comparing to benchmarks
- Complying with service level agreements (SLAs)

Teams will also want to know the maximum number of users for a system (component, transaction, or configuration) to assist in capacity and scalability planning.

Regardless of whether the effort involves SAP modules or not, deploying any software into production without testing the performance of the release is not only a bad idea, it adds unnecessary risk to the business. Any software outage due to software or hardware constraints will impact the user community and potentially one's brand and its position in the marketplace. Make sure you release software with confidence, which includes being confident that the deployment will perform as required.

Prioritizing What to Test

When it comes right down to it, performance testing is often educated guesswork, where teams measure the ability of a system under test to process requests and deliver responses at an acceptable speed while running in a test environment that is as close to the production system as an organization can afford. Rarely do organizations have an exact clone of their production environment just sitting there waiting for the testers to use.

As performance testing can be a very long process, with teams working continuously to tune the system under test, you then have to consider the level of risk your stakeholders will find acceptable. What is an acceptable response time? What processes are known to exceed the expected response times? Determining the performance testing criteria and determining what levels you need to test for by business scenario or business process is critical to defining scope.

Otherwise, performance testing can go on forever as the team constantly tweaks, tunes, and tests.

Next, one must identify the specific processes to be tested, matching that list to which processes are critical to the business and/or used most often by the end user or customer. Then one needs to look at the impact to the business if one of these priority processes was to either start performing poorly or fail to such an extent that the entire system is brought down. Reviewing and ranking the importance of the business processes to the business will help you better understand where to focus your energy as you begin creating the test scripts. A basic rule is to prioritize the performance testing of those business processes that present the highest level of risk to the business if they were to fail or perform poorly.

Considering the Interface — GUI Client versus Thin Client

When preparing to create a set of performance tests, keep in mind that there are subtle differences in how you configure your environment depending on whether you are accessing the SAP system using the SAP GUI client or via a browser. The approach, the configuration/setup, and the playback considerations vary depending on the user interface, as the next sections make clear.

Configuring SAP R/3 via the SAP GUI client

You'll need to configure the SAP R/3 server in order to carry out performance test recording and execution, and you'll also need to install the SAP GUI client on the remote computers put in place to generate the load.

To enable scripting on the SAP R/3 server:

1. **Check that there is a `Scripting` directory located in the SAP GUI installation directory.**

   If this directory does not exist, then the SAP Scripting API is not installed, and you must reinstall SAP GUI with the SAP Scripting API option.

2. **Run the SAP GUI client and log on to SAP R/3 with your user name and password.**

   Administrator privileges might be required to enable scripting on the server.

3. **In SAP, run the transaction `rz11`, type the parameter name `sapgui/user_scripting`, and then click Display.**

   If the parameter is not found, make sure you have the correct support package level.

4. **If the Current value is `FALSE`, first click the Change Value button and then set the New value to `TRUE` in uppercase characters.**

5. **Click Save to end the transaction.**

To enable scripting in the SAP GUI client, do the following:

1. **In the SAP GUI client toolbar, click the Customizing of Local Layout toolbar button, and then select the Scripting tab.**

2. **Select Enable Scripting, and then disable both the Notify When a Script Attaches to Running GUI and the Notify When a Script Opens a Connection options; click OK.**

3. **In the Help menu, select Settings, and then select the F4 Help page.**

4. **In the Display menu, select Dialog (modal) and then click the Enter button.**

Performance testing SAP applications via the SAP GUI client with IBM Rational Test Workbench

Performance testing will always involve a discrete number of steps, depending on the testing tool used. When using IBM Rational Test Workbench with your SAP GUI client, you'll usually want to follow these general steps:

1. **Test creation.**

   You create your test by recording a session with the SAP GUI client. Typically, the recorded session starts when you log on to the SAP R/3 server. You then interact with the application in order to produce a relevant performance test, and the session ends when you log out. The recorded session is split into transactions and SAP application screens. Response time measurements and verification points are automatically added to transactions and SAP application screens.

2. **Test editing.**

   After recording, you can edit the events in each transaction and SAP application screen. With the SAP Protocol Data view, you can use snapshots to edit the events. You can replace recorded test values with variable test data or add dynamic data to your tests. You can also set verification points on field values or window titles to validate that the test behaved as expected.

3. **Test validation.**

   Before deploying the test, you can run the test as a single virtual user to make sure that the test runs smoothly and produces the expected results in a nominal environment with minimal server load.

4. **Workload emulation with schedules.**

   When the test runs repeatedly, you specify an execution schedule and user groups to emulate a workload that is generated by a large number of virtual users. You can add batch input tests to the schedule to simulate a heavy load on the servers while minimizing virtual tester resources. (We'll discuss batch input tests in more detail later in this chapter.)

5. **Schedule execution.**

   Running the schedule deploys test execution over virtual users hosted on remote hosts — the load generators. Each virtual user runs an instance of the SAP GUI client. Response time results are provided by the SAP R/3 server and recorded. Verification points are checked and results are recorded.

6. **Evaluation of results.**

   You evaluate the results produced by the performance tests through the various reports that are generated during execution. You can also design custom reports to performance test execution results with your stakeholders.

Performance testing SAP applications via a thin client using IBM Rational Test Workbench

As mentioned earlier in this chapter, there are differences in performance testing using a browser over http/https. Note how these steps differ from the steps outlined in the previous section:

1. **Test creation.**

   Although it's possible to write a test from scratch, you generally create HTTP performance tests by recording representative interactions with an application. These actions are saved, and a test is generated from these recorded actions.

2. **Test editing.**

   After recording a test, it is a best practice to run it individually and inspect the results to make sure that the HTTP tests are doing what you expect. You might want to edit your tests. Typical changes that you might make are

   - Replacing a value in a recorded test with values in a data pool in order to produce a more realistic test.
   - Adding dynamic data to a test (referred to as *data correlation*).
   - Enabling verification points, so that you can determine whether an expected behavior occurred.

3. **Emulating workload with schedules.**

   After editing HTTP tests, you create a schedule. You add user groups to the schedule and add appropriate tests to each group to emulate a task. A typical schedule requires, at a minimum, user groups and tests to run. Optional items that may be included in a schedule are loops, delays, and think time behavior settings, to further emulate a load.

4. **Evaluation of results.**

   Reports are displayed during the schedule run for teams who are using the IBM performance testing solution. You also have the ability to regenerate reports after the run, customize reports, and export reports in HTML format, so that others who don't have the performance testing software installed can see them.

Batch Loading the Back-End Servers

As the resources required to execute performance tests via the SAP GUI are high, an alternative approach is to capture the performance test results after having loaded the back-end servers with a batch process that would stress the system while applying a smaller load via the SAP GUI client. This approach decreases the system resources required to deliver sufficient load to accurately measure the capability of the system under test.

To play back batch input tests, you must first install the SAP Java Connector (JCo) libraries on local and remote computers running the load generators or agent controllers.

Things to Watch Out for: The Common Pitfalls

We're not saying that something bad will inevitably happen — we're not pessimists at heart — but we are saying it pays to be prepared for what are charitable referred to as "surprises along

the way." To minimize the possibility of encountering such a surprise, keep this list of "Don'ts" in mind:

- **Don't drive too high a user load too early.** Before deploying and executing the test using a large number of virtual users to drive the load, we recommend that you initially run the test as a single virtual user to make sure that the test runs smoothly and produces the expected results in a nominal environment with minimal server load. As the test script is actually execution of code, you may experience multiple cycles of test editing and validation cycles before your test is robust.
- **Don't undersize the load generating agent machines for your SAP GUI performance tests.** As the SAP GUI client services are launched for each virtual user being created, you can quickly consume your system resources. It is important to monitor the load generating computers and make sure they do not become constrained as this may negatively impact the test and compromise the results. It is also important to remember to disable antivirus software, screen savers, and automatic updaters that may be present on the load generating computers. And more importantly, avoid using the load generating computers for other purposes while a test is running.

  

  The SAP GUI testing API wasn't originally designed for performance testing and consumes a large compliment of the available system resources if you launch multiple instances of the SAP GUI client on a single computer. It is also important to note that the SAP GUI testing API is only available on Windows, so you can't use non-Windows computers for load generation.
- **Don't run your tests for too long.** When tests exceed 24 hours, resource issues can cause problems. The Long Run Mode option in IBM Rational Test Workbench increases the reliability of long duration tests for SAP protocols by running the tests in multiple processes. Testers may choose to enable the long run mode in the Performance Schedule Editor.

Verifying That the Functionality Is Working As Expected

If you rely on your own heterogeneous SAP and non-SAP application development to serve your needs or your customers' needs, then you know that application quality is a prerequisite for success. However, ensuring that SAP applications work in a manner that meets or exceeds project expectations requires a level of project discipline and efficiency that can be hard to achieve. Typically the domain of quality assurance departments, functional testing allows users to verify the successful implementation of requirements in categories such as functionality, usability, and appearance. Though you can perform functional testing using a purely manual approach, you'll find that automation can bring great benefits to the process.

Most organizations tell us that automated testing is the only way they can test the effects of change end-to-end across their complex application landscapes — applications that may include web technologies, mobile apps, hybrid cloud environments, and dozens of heterogeneous enterprise applications, customizations, and integrations as well as custom and legacy applications!

Planning for automation

The first consideration of automated testing for SAP projects is to think about what your organization wants to achieve with automation. Do you simply want to reduce the cost of testing? Or are you perhaps looking to speed up the time to deployment? The value of automated testing is that it is not concerned with how processing occurs, but rather only with obtaining the results of whether the processing ran as expected.

As the processing of an end-to-end SAP business process can be complex, it is important to note that not every manual test can and should be converted into an automated script. When making the decision to either stay manual or go automated, consider the bottom line: If the automated test does not lead to significant cost savings, then it may not be pragmatic to convert the manual step to an automated one.

Reusing existing manual scripts

If time and effort has been put into creating manual tests, perhaps by an outsourced provider, then these should be reused to help capture the first automated test recordings. Once recorded, the tests can be rerun when required.

Having business domain experts record tests

Take advantage of the expertise of your business domain experts. To accelerate success, make sure the chosen automated testing tool doesn't require complicated scripting knowledge in order to create tests. That way, your (highly skilled, if somewhat script-handicapped) business users can record and execute the tests to help with the testing process. Without any existing manual tests to start from, it's a good idea to utilize business domain experts to use the test system as they would normally expect it to work. Behind the scenes, you can have your automated test tool capture the process flow so the scenario can be run again whenever a change is made to the system.

This ability to leverage the knowledge of your business domain experts represents a fundamental change in the testing process. Rather than being the sole domain of a specialist test role or developer, testing has been opened up to include business users as partners in the process.

Ensuring automated scripts can be reused

Automated tests are designed to be run multiple times so it is imperative to make the tests and scenarios easily accessible and reusable. It is a useful tip, then, to ensure that teams can share the automated test assets and seamlessly feed into the overall approach of quality management for analysis, reporting, and collaboration.

This can significantly increase confidence in stakeholders within the business, with partners, and also with delivery teams. It can also reduce the cost of testing and speed up the time to deployment, and the end-to-end business process validation significantly reduces technology risk.

This way, automated testing becomes an integral part of any attempt to achieve continuous testing.

Putting the Pieces Together to Achieve End-to-End Testing

Different roles in your SAP solution implementation use different capabilities, but some capabilities are utilized by all roles. It is this matrixed relationship that is simplified by using a common tool for the test environment and for test management. It is only the specific test execution roles that require different capabilities.

While the roles are pretty straightforward — test environment, test management, and test execution — the IBM solution capabilities may need brief explanations:

- **Rational Test Virtualization Server** can be used by all testers to create the exact test environment they need, when they need it.
- **Rational Quality Manager** is the open, collaborative hub that connects with SAP Solution Manager and the test execution tools and other third party vendor tools to provide test analysis, planning, management, reporting, and collaboration between all stakeholders.
- The test execution tools (**Rational Test Workbench**, and **Certify**) utilize Rational Quality Manager to know what to test and to provide results and data for reporting and test analysis.

When combined, the tools fit together seamlessly to deliver an end-to-end testing solution for SAP projects.

Chapter 7

SAP Solution Manager in Continuous Testing

In This Chapter

- Integrating SAP Solution Manager into your architecture
- Documenting your business process requirements
- Analyzing business process change
- Linking with IBM Rational Quality Manager and IBM Rational Team Concert

Businesses that run on SAP applications need to adapt to changing needs quickly and can benefit from a unified end-to-end application life cycle management (ALM) platform based on industry standards — and the more open and extensible the platform, the better.

SAP Solution Manager often plays an important role within the end-to-end life cycle. It provides the option for centralized documentation of business processes and central availability of required information. SAP Solution Manager integrates with IBM's Collaborative Life Cycle Management (CLM) to provide a truly comprehensive approach to testing.

As SAP applications are often part of a much larger heterogeneous implementation, deploying an end-to-end life cycle management solution means your organization can take advantage of a comprehensive and automated approach for requirements, quality, and change management across your enterprise. The IBM Rational solution for CLM helps organizations manage change across both SAP environments and non-SAP environments in a manner that reduces risk and cost.

The benefits of extending SAP Solution Manager with a CLM solution from IBM Rational should be pretty clear. Your organization will be in a better positon to manage change quickly and with lower risk. This is because with the IBM Rational Connector for SAP Solution Manager, you can achieve a holistic approach to managing and testing the life cycle of heterogeneous applications in a unified way and easily share data.

SAP Solution Manager 7.1 SP 2 or higher is required if you want the unified platform for life cycle management.

In this chapter, you get a chance to explore the combined capabilities of SAP Solution Manager and the IBM solution so you can better see how this combination optimizes quality management and testing for both SAP projects and non-SAP projects.

Working with SAP Business Blueprint

Created within SAP Solution Manager, the SAP Business Blueprint documents the business process requirements of a company and shows how your business processes are mapped onto one or more SAP systems. It provides the option of adding further details on the scope of business scenarios, business processes, and process steps. The business process content may also be extended to maintain important content for systems, interfaces, software vendors, transaction codes, training materials, and process documentation.

Setting up a requirements hierarchy

With the IBM Rational Connector for SAP Solution Manager installed and configured, the Business Blueprint can easily be pushed to the IBM requirements management software (for example, IBM Rational DOORS Next Generation) to start the quality management process. The result of the blueprint push is the creation of a requirements hierarchy with test plans and test cases in the IBM Rational solution for CLM.

Connections are automatically created back into SAP Solution Manager's documents. The test plan(s) and test cases are also automatically linked to the newly created requirements to ensure full life cycle traceability, aiding in understanding the impact of change. Figure 7-1 offers a graphical representation of this process.

DOORS Next Generation serves two purposes in the requirements creation process:

- It provides the business blueprint information from SAP Solution Manager to testers, who will use the requirement as a basis for their test planning.
- It supports linkage and traceability from non-SAP requirements (generic legal or IT requirements, for example) to the business blueprint.

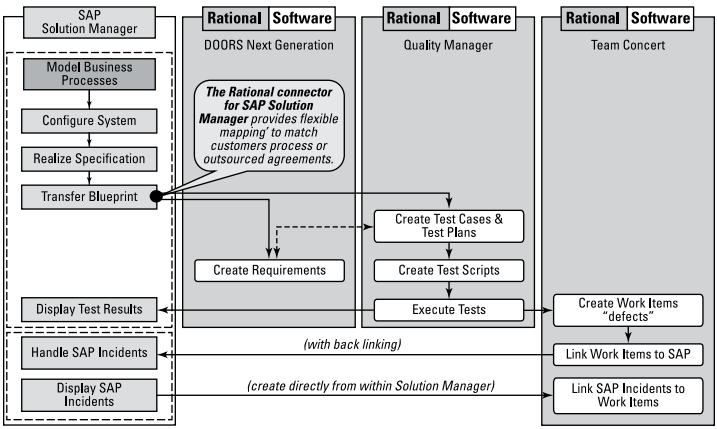

Figure 7-1: Work flow for the IBM Rational solution for CLM.

SAP Solution Manager provides many options to specify the scope of the blueprint push. Depending on the settings, the blueprint is transferred as a whole or as individual business processes. In Figure 7-2, you see the complete business process hierarchy. All business scenarios, business processes, and process steps are selected and will be transferred.

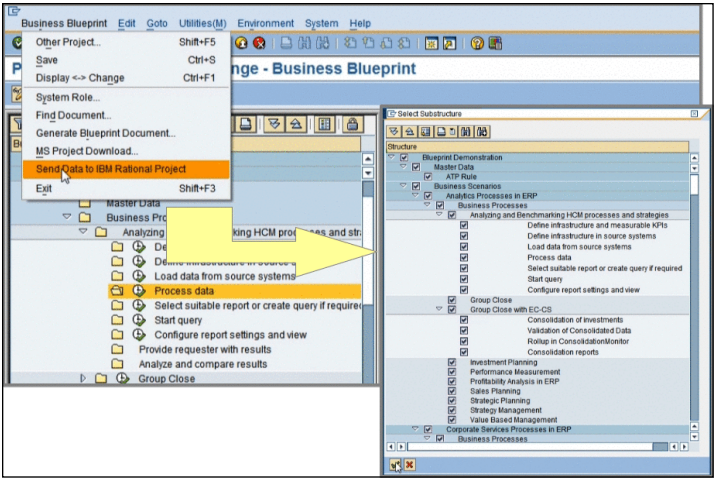

Figure 7-2: You can choose to send your data to the IBM Rational Project.

After the blueprint push has finished, the delivery team may review the transferred information by launching a thin client (browser) and navigating to the IBM Rational DOORS Next Generation and IBM Rational Quality Manager applications deployed on the IBM Rational Jazz server.

Connecting the dots

The IBM Rational Connector for SAP Solution Manager, developed jointly by SAP and IBM, is the link between the blueprint tree structure, the corresponding collections and requirements in the DOORS Next Generation project, and the test plans and test cases hosted in an IBM Rational Quality Manager (RQM) project area. See Figure 7-3.

How you'd transfer a specific document type being managed in SAP Solution Manager to the IBM Rational Solution for CLM as a specific artifact type depends on the mapping configuration you choose within the IBM Rational Connector during the setup process. You could, for example, create a business scenario for SAP Solution Manager and connect it to a test plan in RQM, while connecting a test object to a test case and a business requirement to a requirement in IBM Rational DOORS Next Generation.

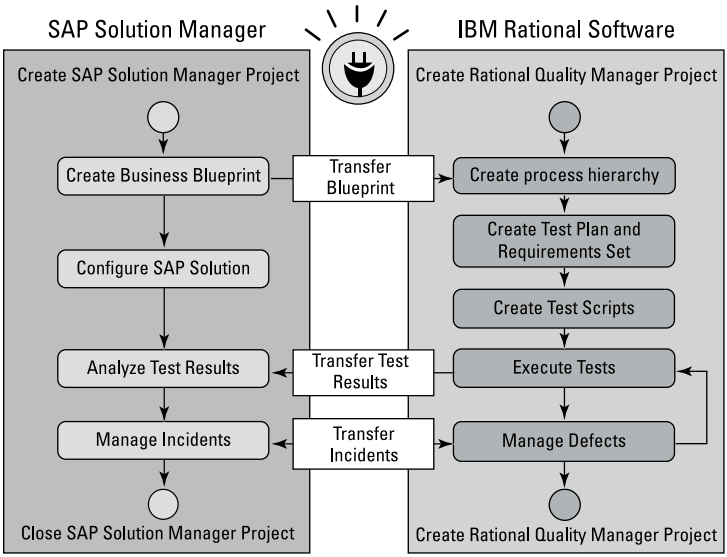

Figure 7-3: Keeping your avenues of communication open.

With the test plan(s) and test cases automatically loaded in RQM, the team can now associate any manual test scripts — authored, maintained, and executed from within RQM — to the appropriate test case. If the delivery team has chosen to leverage test automation, it can also manage and link those automated test scripts to the appropriate test case.

And as traceability has been established between the SAP documents and the CLM artifacts, delivery teams have a better overview of the testing process as a whole as well as the ability to quickly report on the results of the test execution through dashboards and detailed reports. Stakeholders now have access to the latest data on quality available at their fingertips as the test results transfer from RQM to SAP Solution Manager, either automatically or manually.

SAP Solution Manager provides many options to analyze, report, and document data and information. A simple method to display the IBM Rational Quality Manager test results in SAP Solution Manager is by using the Testing with IBM Rational Quality Manager option, which is accessible by calling the SOLAR_EVAL transaction directly.

Before a test manager initiates an analysis in SAP Solution Manager, he or she can specify search parameters and thereby limit the analysis to a substructure of the Business Blueprint, a particular team member, or other object attributes.

The result of the analysis displays the project structure all the way down to the business process step as well as any and all test results that have been transferred from RQM. The test results also contain information about the test case, the test execution record, the test state (passed or failed) and the affected test object.

Gauging Change Impact with Business Process Change Analyzer (BPCA)

For an organization working to balance quality and speed in its change management policy, knowing what has changed, what needs testing as a result of change, and how to optimize the testing effort for maximum coverage is of crucial importance.

One option in keeping abreast of business change is SAP Business Process Change Analyzer (BPCA). By launching and executing SAP's own impact analysis, BPCA is able to return details on all of the affected test cases in IBM Rational Quality Manager and other information such as the execution type, node text, referred object, node type, or logical component.

Armed with the list of impacted test cases, the team can choose to create a new test plan in RQM to perform a full regression test or they could use the list of test cases to optimize the testing scope. The idea here is to find time savings and maximize coverage while potentially reducing the effort.

SAP BPCA capability helps the individual tester and delivery team to analyze and look for opportunities to concentrate their testing in IBM Rational testing tools on what truly matters. By reducing coverage parameters to the risk tolerance level an organization is willing to accept and by prioritizing test

cases based on their importance to the business, teams are able to improve testing efficiency while minimizing risk to the business.

The SAP BCPA also provides the SAP project test manager or test coordinator with additional details regarding the optimized test scope through the business process hierarchy view. Through this view, the team can see which nodes are impacted by change, whether a test case exists in IBM Rational Quality Manager, as well as estimates of the test execution effort — manual, automated, and cumulative.

With its scope optimization exercise completed, the test team can now set up a test plan in the RQM project area. Upon review of this new test plan, only those test cases sure to deliver the most amount of coverage with the least amount of effort while minimizing risk to the business will be presented as options. That means only the test cases that truly matter have been listed for execution, which results in less time being spent checking business processes that are of low priority to the business.

Connecting with the SAP Service Desk Component

In an integrated SAP Solution Manager/Rational CLM scenario, two distinct management systems exist for defects and change requests or problems and incidents. RQM integrates with IBM Rational Team Concert (RTC) as a defect management tool. This allows testers to quickly document the defects found as a result of a failed test run. Alternatively, SAP Solution Manager provides the Service Desk component to manage problems and incidents. The IBM Rational Connector for SAP Solution Manager links the RQM and RTC change management system and the SAP Service Desk component. The result is an integrated change management system that allows a bidirectional exchange of information by creating incidents in SAP Service Desk from defects in RTC, and the other way around.

To understand how this would work in practice, below are the details of how easily you would create an incident in SAP Service Desk using the defect form in RTC:

1. **Click the Links section of the defect form in RTC.**

   Doing so opens a new tab that allows the selection of an external location of a related change request.

2. **Create a new Service Desk Incident Form by adding a Related Change Request to the RTC defect form.**

   The New Service Desk Incident form is created through the connector, and it provides live field values and data from the SAP Service Desk. The field values are all valid and up to date, because they're sent to the connector at association time.

3. **Provide the necessary defect information and click the Submit button.**

   After the new defect has been submitted, the defect is updated with a bidirectional link to the new corresponding incident in the SAP Service Desk.

It is also possible to create a new defect in RTC from the SAP Service Desk. Once connected to each other, the SAP Service Desk incident record and the RTC defect artifact will remain connected via the connector.

Chapter 8

Ten Considerations for Testing on Your SAP Project

In This Chapter

- Making your stakeholders happy
- Benefitting from the IBM and SAP Alliance
- Setting the (early) test date
- Extending the value of SAP Solution Manager

The complexity of today's software and the driving need to release higher quality software faster is challenging many organizations. Teams need to work more efficiently together and with greater effectiveness. They need to build quality in from the start, test deployments end-to-end earlier in the development life cycle, and ensure that the released software performs as expected. This chapter gives you ten ways IBM software for SAP solutions can help.

Increase Confidence across Stakeholders

Success depends on how well you manage change — change coming either from business needs or from required technology needs. IBM software integrates seamlessly with SAP software to enable traceability and analysis even when change is occurring. For example, if an employee changes a business process in a SAP Business Blueprint, other users can see which nodes of the Blueprint have been affected, determine which test cases to run. The ability to understand the impact of change across the enterprise, view the status of any ongoing validation, and be able to collaborate increases confidence across stakeholders.

Reduce Your Cost of Testing

To reduce the cost of testing on SAP projects, Sandhata Technologies Ltd. sought to improve the efficiency of its software testing processes. After implementing testing tools from IBM, Sandhata deployed and used virtualized services, software, and applications to test more efficiently. The testing efficiency gained by using IBM software cut the average project length by 40 percent.

"The biggest challenge Sandhata's customers' face with implementing SAP is the dependency on the end-to-end environment, both existing and third-party, to test the functional logic provided by the SAP core components," says Gary Thornhill, general manager at Sandhata. "That is a great challenge because if those environments aren't there, you can't test the functionality of the applications, and it causes project delays."

Benefit from the IBM and SAP Alliance

IBM spends over $7 billion annually on research and development. This investment has led to many innovations, enabling both business and technical stakeholders to have improved confidence that their business will continue to operate as expected after any SAP implementation or change.

Cooperative dedication is what makes the IBM and SAP Alliance so valuable to clients worldwide.

Test as Early as Possible

With the ability to virtualize required SAP services or non-SAP services that are unavailable at the time of testing, test teams can start testing early in the development life cycle. IBM's service virtualization capability together with test automation means testing can be continuous through all stages of the delivery process. This in turn leads to earlier acceptance testing, deployment, and feedback from operations and customers.

Extend the Value of SAP Solution Manager

IBM and SAP have worked together to integrate key areas of SAP Solution Manager with IBM's testing tools, enabling the sharing and exchange of data. Keeping the data shared between SAP Solution Manager and IBM tools in sync allows the entire team to understand the scope of change and to be sure to test what matters.

Realize the Value of DevOps

IBM enables organizations to adopt a *DevOps* approach, bringing together your *development* initiatives with your *operations* initiatives, which includes shifting testing to the left. The testing capabilities offered by IBM means SAP testers can begin integration, functional, regression, and load testing sooner than ever before and provide continuous feedback to the development team and the stakeholders.

"What's unique about the IBM Rational solutions for SAP is they allow you to accelerate the whole delivery, so you have that DevOps theme of continuous delivery throughout all your projects," said Gary Thornhill, General Manager, Sandhata.

Reduce Business Risk

Validating post-change business processes across heterogeneous application landscapes can be complex. However, the business needs to change — there's no getting around that — and the SAP landscape needs to validate that the business can run as expected. To simplify the end-to-end testing approach across both SAP applications and non-SAP applications, IBM provides end-to-end integration, functional, regression, and load testing for pure SAP landscapes as well as for heterogeneous landscapes.

Protect Your Investments

IBM Rational's quality management platform is open and integrates with a wide range of environments and third-party products. This gives teams the ability to execute tests with legacy tools or data — and still collect test results in a central location and integrate with SAP Solution Manager. Such openness in quality management means teams can utilize or phase out other legacy testing tools and reduce the risk of compatibility issues during the testing process as they transition to test execution tools from IBM.

Minimize the Impact of Testing

IBM supports test automation on many platforms and provides a collaborative hub for managing business-driven software and systems quality across SAP platforms and non-SAP platforms. These capabilities mean test teams can share information seamlessly, use automation to accelerate all types of testing, and easily report on the execution results offering continuous feedback to the many stakeholders — feedback that enables agility in steering the overall business.

Control Your Requirements

To help get your project requirements under control, IBM supports *requirements-driven* testing, where you can link your test plans to a collection of requirements that may or may not have come from SAP Solution Manager. This keeps test cases in sync with changing requirements and helps to streamline negotiations between test teams and their stakeholders.

Notes

Notes

Manage quality while minimizing the impact of testing on your SAP projects

If your organization uses SAP applications, success depends on your ability to deliver change. Change can be frequent and complex, and the demand for testing on SAP projects seems overwhelming and expensive. But there is an answer! *Testing SAP Solutions For Dummies*, IBM Limited Edition, helps you understand how you can deliver sustainable, continuous testing on all your SAP projects.

- ***Start testing early* — virtualize required SAP or non-SAP services**
- ***Deliver quality* — collaborate continuously, enable automation, and easily report and analyze results**
- ***Realize value of DevOps* — achieve continuous delivery, reducing the cost and risk of managing change across your SAP landscape**
- ***Achieve interoperability* — standardize testing across your projects**

Open the book and find:

- **The challenges of testing on SAP projects**
- **The components of continuous testing**
- **How to make sure your deployments perform as expected**
- **The top ten considerations to help you test SAP projects**

Go to Dummies.com®
for videos, step-by-step examples, how-to articles or to shop!

FOR DUMMIES®
A Wiley Brand

ISBN 978-1-119-04722-3
Part #: RAM14028-USEN-00
Not for resale

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.