A Rational approach to integration testing

Early and continuous testing to contain defects and optimize test efficiency
Contents

2 Introduction

3 Incremental integration

3 Employ test virtualization

5 Use continuous system-level testing and asset sharing

5 Plan for effective data management

6 Reduce E2E testing and isolate the GUI

6 Test earlier

7 Avoid the Big Bang

7 Summary

Executive summary
Applications are evolving at an ever-faster pace. These applications are not discrete islands; they build on a complex, interconnected set of components that includes disparate technologies, developers, deployment topologies and organizations. Developers must deliver high-quality applications while limiting their testing expenses. In this challenging environment, a combination of automated integration testing and test virtualization can enable test teams to improve quality and keep up with the rate of change.

Introduction
Applications are evolving at an ever-faster pace. These applications are not discrete islands; they build on a complex, interconnected set of services that includes disparate technologies, developers, deployment topologies and organizations. Developers must deliver high-quality applications while limiting their testing expenses. In this challenging environment, a combination of automated integration testing and test virtualization can enable test teams to improve quality and keep up with the rate of change.

A simple measure of success for any quality professional is the ratio of captured defects versus escaped defects. However, success or failure is not simply determined by the number of defects that have escaped into production. Categorization of defects to determine where the defect should have been found can dramatically reveal the efficiency or inefficiency of your testing. For example, if a functional defect is found during end-to-end system testing, the costs of remediation would far exceed the costs of fixing the defect as it was introduced in an earlier development phase. The increased costs would be due to factors such as: more regression testing, more test resources, usage of more live-like environments and greater requirement for coordination.

The issue of defect phase containment is a critical factor in integration projects and in service-oriented architecture (SOA) projects in particular; it represents a considerable challenge to a test manager who is working within financial and schedule limitations.

Indeed, SOA presents great challenges to the testing community. Complex business processes mean that services have to collaborate. This necessary collaboration can lead to a vast number of permutations in the data and correspondingly in the required
Test cases. Testers not only need to test combinations of messages, message data and service-to-service interactions, but ultimately they have to demonstrate that the solution supports their business requirements. SOA testing has usually followed the same traditional testing roadmap: requirements management, functional testing, integration testing, end-to-end (E2E) system testing, user acceptance testing (UAT), performance testing and operational acceptance testing (OAT). This roadmap places enormous importance on functional and integration testing to ensure compatibility, interoperability, functionality and performance of services in various applications, systems and the enterprise.

Test automation for SOA and integration testing is no longer an option; it is a necessity. But test tools alone are not enough. A proactive approach to testing is required: strategic approaches that will mitigate the risks of system change and prove quality in a structured and controlled fashion. Using the integration testing capabilities in the IBM Rational Test automation solutions combined with a proactive integration testing methodology can help you achieve defect phase containment, early and continuous testing, and an optimized return on testing effort.

A proactive testing approach requires an appropriate integration structure: incremental integration. But the approach also requires specific tools. IBM Rational Test Workbench, IBM Rational Performance Test Server and IBM Rational Test Virtualization Server are powerful tools that are designed to help you as you implement proactive test methods.

**Incremental integration**

Accurate and non-ambiguous requirements are particularly vital for integration testing. Business processes must be defined and broken down into their constituent parts so that corresponding testing requirements can be defined at the most granular level of service interaction. Requirements coverage is thus tracked at a functional level, a service interaction level, the business process level and finally at full system level. This approach to requirements management offers considerable potential cost benefits for SOA or integration projects.

If granular requirements are fully visible, project leads and test managers can more easily cooperate and plan an effective and controlled release schedule. A gradual and controlled introduction of functions and components into test environments greatly expedites the isolation of faults and defects.

After establishing an incremental integration plan for the project, test professionals should consider the following techniques to help achieve a more proactive test procedure.

**Employ test virtualization**

In test virtualization, a real component is replaced by a virtual component, sometimes called a “stub.” These virtual components can be used to model and simulate real system behavior. The virtual environment helps eliminate application test dependencies and reduce the setup and infrastructure costs of traditional testing environments. A proactive test plan should use test virtualization to make virtual components available for key service components, which allows various situations to be simulated and tested more easily.

Some industry professionals use the term “service virtualization” to refer to this process of emulating missing dependencies. However, these emulations can also use non-service-based approaches, so the term “test virtualization” is more appropriate.
Test virtualization is a powerful tool to use when carrying out integration testing. With test virtualization, you can create virtualized integration environments for functional or performance testing, which in turn can provide great advantages in reduced downtime, cost savings, early error detection and requirements clarification prior to development and integration testing of hardware and software.

Test virtualization capabilities are at their most powerful when used strategically. For instance, scheduling issues can be extremely disruptive to integration testing. Carefully planned release schedules are fraught with dependencies and assumptions. A proactive testing strategy that employs service virtualization can help you remove as many dependencies as possible.

IBM Rational Test Virtualization Server provides production-level simulation of required services. Developers and testers can build simple stubs that return fixed responses or a variety of different responses based on inputs, or more complicated stubs with stateful behavior. You can also use data tables to provide parameterized stub behaviors that can be extended simply by adding another row to a spreadsheet.

In a similar way, for performance simulation of required services, developers and testers can use IBM Rational Performance Test Server to generate loads at the application and integration levels. This functionality is designed to help you get a complete view of the performance and scalability of all application components.

Virtualized services can support nearly any test purpose and test execution technology. When the IBM Rational Test Workbench integration capabilities start a test, the program can automatically start the necessary virtual services to go with it, helping ensure that unresolved system dependencies can be provided with predetermined behaviors. The user is in control of which stub is used, allowing a wide variety of different situations to be modeled, depending on the testing scenario.

An assessment of critical integration points, both inside and outside the scope of change, will yield an understanding of what services should be virtualized. Environment availability and late service delivery are a considerable risk. Proactive test coverage using service simulation can provide you with the flexibility to adapt to scheduling crises as they arise.

As components are built and delivered, the virtualized components can be swapped out and testing can continue (see Figure 1).

If you plan to use virtualized services as part of your proactive testing protocol, it is usually beneficial to create a simulation template in the form of a script at the beginning of the project. You can then adapt this script very quickly as the demand for the virtualized service becomes apparent. A thorough understanding of the technological and business risks, as derived from effective requirements management, can help you prioritize simulations in the most effective manner.
Use continuous system-level testing and asset sharing

One of the great advantages of using Rational Test Workbench is how quickly and easily you can execute tests. Its simplicity means that full regression cycles can be run whenever new or virtualized components are introduced. This provides an immediate feedback loop to the development team who can run the exact same scripts, replicate and resolve issues with minimal effort (see Figure 2). This promotes innovation rather than remediation. The tools encourage developers and test teams to collaborate by sharing integration tests and virtualized services throughout the software development life cycle (SDLC).

Plan for effective data management

Representative and appropriate data is required to support the necessary test coverage and achieve the appropriate level of confidence in a delivered solution. Data considerations should begin at the requirements stage and should be factored into test creation and execution. Given limited timescales and budgets it is usually necessary to use equivalence partitioning or boundary analysis to identify the data that is absolutely essential to the project. Test data management is such an important activity that it is often assigned to dedicated individuals.

All test scripts should be data-driven and Rational Test Workbench can handle various file sources as inputs to drive this data through interfaces. Because testing is continuous, it is important to consider clean up scripts; these scripts can return the systems to their baseline states and are designed to ensure that data can be reused where appropriate.

Virtual services can also be data-driven based on the requirements of the test environment. Consistency is often required between the data set used by tests and the one that supports the virtual services.

When you establish a data plan, remember that virtualized services can make it easier to gather the appropriate data. For example, solutions such as IBM Optim™ Test Data Management can automate the extraction of a relevant subset of data from production environments, obfuscate and privatize this data as required, and transform it for multiple test purposes. One primary goal is often populating the corresponding databases in a test environment. However, alternate samples of the production data can be consumed by automated test scripts and virtualized services, maintaining consistency between the data driving the tests and the range of responses offered by the simulated services. In many cases, virtualized services can simplify data management and make it more cost-effective to rerun tests as needed.

Figure 2: Reusing test assets contributes to constant asset evolution.
Reduce E2E testing and isolate the GUI

Because integration testing is incremental, E2E testing takes on less significance. If you implement a proactive test approach, you can expect to spend far less time performing costly end-to-end testing, because by the time full end-to-end functionality has been created in the test environment, the functional, integration and business-process-level tests will have been executed many times. Incremental and continuous testing will have removed many of the risks of the integrated solution (see Figure 3).

In order to maintain a sufficient rate of test execution, you should isolate GUI components in environments where they interface with virtualized services. GUI components can then be periodically introduced, validated and withdrawn through the project life cycle; they should only be formally introduced after completion of all other integration testing. Rational Test Workbench can also help you automate these tests. The combination of automation at multiple isolated layers of your application and a more predictable execution environment for GUI tests helps mitigate the traditional challenges of a GUI-only approach to testing.

Test earlier

The above considerations are all undertaken in order to test more effectively at an earlier point in the software development life cycle. It is widely accepted that testing and defect resolution are more expensive if undertaken at the latter stages of integration (see Figure 4).

---

**Figure 3**: An early and continuous test protocol helps reduce E2E testing.

End-to-end testing should mainly focus on driving through the E2E processes via the various GUI front ends. You can use Rational Test Workbench to implement automated testing that occurs at the service layer and bypasses the GUI. In our experience, this testing method is faster to create, faster to execute and more robust over time than using GUI-driven automated scripts.

**Figure 4**: Achieve cost savings through early and continuous testing.
**Avoid the Big Bang**

A traditional “Big Bang” test method brings all integration points together for end-to-end testing. With this test method, there is a sudden threshold at which it becomes possible to run many more test cases. Because of this increase in the number of cases, there is a sudden drop in the percentage that have been tested and passed (see Figure 5).

A proactive approach to testing supports a structured and controlled incremental integration of your application, and helps you keep test costs under control.

**Summary**

- Test integrations incrementally and reduce your end-to-end testing costs. A proactive plan is essential:
  - Employ test virtualization strategically to remove critical dependences and reduce downtime.
  - Enable system-level asset sharing to accelerate defect remediation.
  - Be proactive about test data management.
  - Think about isolating the GUI testing. Could this accelerate your test execution?
  - Test earlier for maximum efficiency.
  - Avoid the Big Bang.

Applications and manufactured products are becoming increasingly complex, with unprecedented levels of connectivity and dependency between systems, processes and infrastructure. Therefore, application integration testing in general, and SOA testing in particular, will continue to present new challenges for the testing community. As new applications are developed to cooperate with existing services, it can often result in a situation where there is no user interface within the scope of change. The testing that is required in such cases sits somewhere between white box testing (testing of internal systems) and black box testing (testing of functionality).

Due to the technical nature of these projects, the components and interfaces are often tested via programs or stubs that the developers have created themselves. This can lead to a situation where the developers are setting the criteria for their own success, which can degrade overall project quality. Furthermore, the tests and stubs may not be available or may be limited to specific test purposes, applications or teams.
It is imperative that developers and testers adopt a new testing strategy so that they can successfully manage change and deliver lasting quality. This may require that the test team have a more technical skill-set than usual. However, the axioms of testing should not be dismissed entirely in favor of a purely technical focus. The general principles of testing, to identify and fix defects as early as possible, are increasingly important as projects become more complex and the scope of change expands.

**For more information**
To learn more about IBM Rational test automation solutions, contact your IBM sales representative, IBM Business Partner or visit: [ibm.com/software/rational/offerings/quality](http://ibm.com/software/rational/offerings/quality). See also:

- **IBM Rational Test Workbench**

- **IBM Rational Performance Test Server**

- **IBM Rational Test Virtualization Server**

Additionally, IBM Global Financing can help you acquire the software capabilities that your business needs in the most cost-effective and strategic way possible. We’ll partner with credit-qualified clients to customize a financing solution to suit your business and development goals, enable effective cash management, and improve your total cost of ownership. Fund your critical IT investment and propel your business forward with IBM Global Financing. For more information, visit: [ibm.com/financing](http://ibm.com/financing)