Use service virtualization to remove testing bottlenecks

Discover integration faults early by pushing integration testing left in the software lifecycle

Contents

Complex applications, heterogeneous systems and the costs of setting up test environments challenge an organization's ability to conduct integration tests. The new pressures of faster delivery, the proliferation of mobile applications and dependence on third party systems further complicates integration testing in software activities.

A more effective means of enhancing software delivery is by using service virtualization that supports earlier and more frequent testing, even when dependent subsystems aren't available. With service virtualization, organizations can more effectively address the various testing challenges in agile development.

Complex, interconnected applications challenge agile development

One of the key principles of agile software development is that working software is the primary measure of progress. Applying principles of agile development such as test-driven development and creating working code at every milestone can significantly improve the quality of software. However, these principles are most commonly used at the level of components or small systems of components. When testing and ensuring the quality of complex systems comprised of many components, higher levels of reliability can be achieved in the components, and you can discover the bulk of issues during integration testing. Often this testing is left to stabilization iterations because the complete system is expensive to setup in hardware, time and availability of human expertise. Testing is often staged beyond the milestone to account for these challenges.

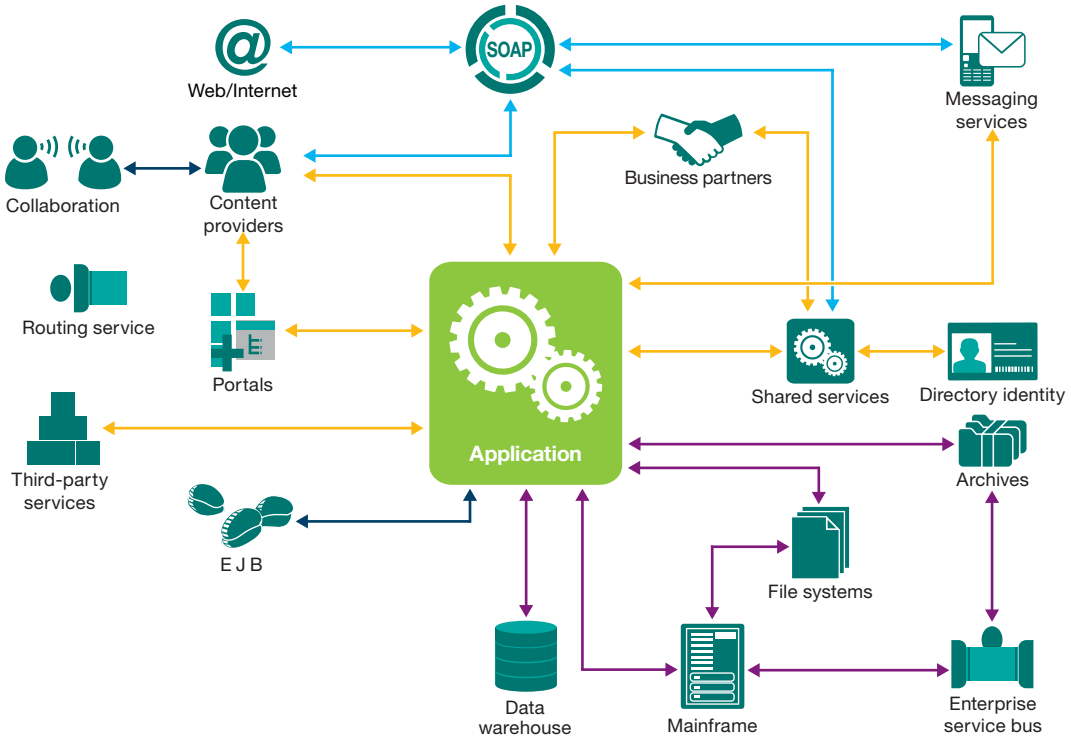

Figure 1. A complex test environment comprised of heterogeneous applications and various connected systems

Test environment complexity

Complex projects include existing systems that must be represented in various development, quality assurance (QA), acceptance and regression test environments. This representation requires a number of resources:

- Middleware and packages. Many existing systems and applications might consist of middleware and packages that require software licenses. Deploying complete test environments can potentially use a large number of expensive licenses.
- System experts. Setting up and configuring these existing systems require time potentially from a range of different specialist system experts. Often this resource is scarce, expensive and test system setup and maintenance is their last priority.
- Hardware and OS. Complex projects might include back-end systems that require multiple different types of hardware and OS, which makes setup exponentially more difficult.

Setting up and maintaining these complex and expensive test environments during waterfall development is challenging. With the pressures of short sprint cycles and milestone deliverables, the immediate availability of the test environment is critical to the success of agile development.

Regression testing coverage

Another layer of complexity in assuring quality and stability is the need for full regression tests of the application as it is being developed. However, full regression test packs with the existing systems included can be potentially weak because of their dependencies: failures in the existing hardware or various network connections can break the regression tests. And, the more the dependencies, the higher the possibility of test failures for reasons other than application fault. These are not "good breakages" that highlight code problems in the implementation. Not only do they serve no purpose, they actually reduce confidence in the regression tests.

In addition, the regression tests can be slow because of existing systems. Slowness is exacerbated by the number of systems being integrated which cause slow round-trip response time and often leads to the practice of limiting the number of tests executed in the time allowed. This practice conflicts with agile principles because the agile practice seeks to reduce the cost of change with testing feedback loops.

Common practice pitfalls

Agile development teams commonly address system complexity by hand-coding stubs. However, the time spent to create these stubs translates to time not spent writing code or testing. In addition, writing stubs is similar to editing your own paper, in that it makes finding mistakes challenging. More importantly, the stubs do not necessarily represent how the real system would behave. Therefore, this workaround both invalidates the testing and uses valuable resources inefficiently.

Architectural constraints

Constraints are imposed when integrating with existing systems. Accounting for those constraints even when applying the technique of using Sprint 0 to remove complexity and planning can be challenging. Without the availability of the complete system, the team cannot plan as effectively; the team cannot anticipate the architecture to reduce refactoring and plan the testing effort.

For example, agile techniques make a trade-off between the efforts required to be successful the first time with the effort of refactoring imperfections in later sprints or releases. When the application integrates a number of systems through multiple technology boundaries, early gaps in knowledge can cause drastic refactoring to fix problems later. The entire system being available as a sandbox can inform the original implementation and architecture to reduce this refactoring.

Continuous integration testing

Continuous integration testing can help resolve many of the challenges associated with using agile development for complex projects. Continuous integration testing relies on a fully automated and reproducible build that runs many times a day, including robust integration tests across component boundaries. Any misunderstanding of existing system constraints are surfaced earlier and addressed more quickly within each sprint.

Project planning

Teams often defer integration testing, either by staging testing into the n+1 sprint as shown in Figure 2 or by adopting mini-waterfall with a final sprint dedicated to integration testing. Both these cases include a high risk of finding defects late.

In the case of staged testing and attempts to conduct integration tests within the sprint, the planning is less effective than other types of planning because of the reliance on other system

components outside of the team's control. This reliance inevitably leads to missing components. Service virtualization specifically addresses this set of challenges.

Helping agile teams enhance quality with service virtualization

Organizations are under pressure to develop new applications and need an automated testing solution to achieve higher-quality software at reduced costs. They need solutions that reduce time and cost of building test environments, and test earlier by simulating the behavior and performance of key subsystems. In addition, automation must address testing bottlenecks by virtualizing dependent services without the need to reconfigure the original application environment. Service virtualization can benefit software developers and architects, testers, business analysts and the entire team.

Developers and architects

Build-triggered testing can be used to improve quality of complex multitier systems. Their component parts are tested in unit-focused regression tests that can be triggered when changes are delivered to that component or unit. In this case, unit tests become more robust and include integration tests across component boundaries.

Testers

Testers can author tests and code automation even before dependent components are available to enable behavior-driven testing. This ability enables the reuse of tests and services, creation of tests for services, creation of data and content for later implementation in build-triggered testing. By enabling earlier test creation and execution, the discovery of issues is "pushed left" in the software lifecycle.

Business analysts

Service virtualization can provide benefits earlier in the lifecycle. To virtualize a service, effort needs to be spent on understanding the interface and behavior of the service. This effort is the same type of analysis work that is required early on to understand constraints placed on or opportunities created for the requirements by the existing systems.

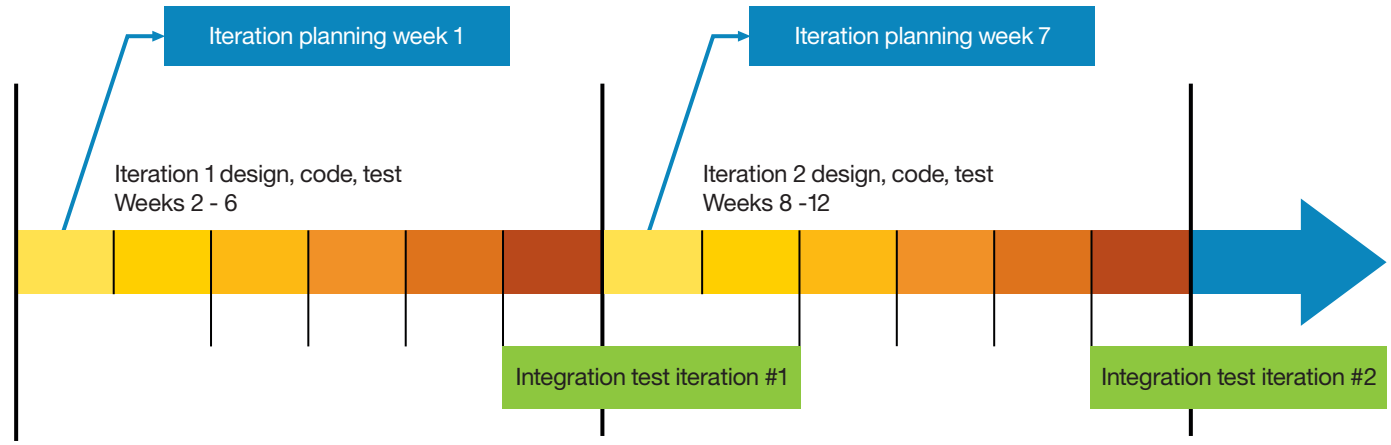

Figure 2. Integration testing lags behind the end of milestone

Analysts are able to work with developers to understand the behavior of the end-points being integrated and test their understanding by experimenting with these endpoints; checking whether their behavior meets with their understanding, and improving their understanding when it doesn't. Acceptance test criteria created by business analysts is more robust and concrete.

Testers, developers and architects are able to reuse assets created during this early exploratory work. Likewise, analysts are able to reuse existing virtualization assets created by the entire team.

Teams

Virtualized test environments can be reused by multiple users and a large number of assets can be built up around a system and shared throughout the team. These assets include:

- Tests—individual or suites
- Service virtualizations—either individual or entire subsystems
- Individual messages and complete message interactions requirements
- Assertion rules rule cache
- Data models

Delivering more reliable service architectures

Service virtualization provides benefits to teams that are developing solution components for a broad spectrum of SOA technologies, including both teams that are implementing *service consumers* and teams that are implementing *service providers*.

Example service architectures

IBM® Rational® Test Workbench and IBM Rational Test Virtualization Server can be used to test and virtualize solution components over a number of runtime platforms. Here are some examples:

- **Virtualizing a wide range of resources for a single application.** Applications built for a JEE container such as IBM WebSphere® Application Server benefit from being able to use a framework that allows them to use many different types of resources in their architecture, from services to databases and back-end systems. Development teams building these applications can get considerable use of Rational Test Virtualization Server to virtualize each of these types of resources and thereby avoid delivery planning constraints.
- **Virtualize various styles of integration.** Modern applications have a wide range of different styles of integration to choose from, each suited for their own class of integration requirements. For example, the IBM WebSphere Message Broker team often needs to deal with integrating different types of service using both web-service integration technologies and messaging-oriented middleware technologies. Rational Test Virtualization Server provides a single solution that enables you to virtualize and test integrations in either style of integration technology.
- **Reduce impact of testing new SOA appliance deployments.** The ability to both test and virtualize a service-fronted application can be useful for teams that are deploying SOA appliances such as IBM WebSphere DataPower®. These teams need a powerful service testing tool to drive load through the appliance without affecting the real business application and back-end services during their test cycles. By using Rational Test Virtualization Server, WebSphere DataPower developers can experiment and simulate scenarios with virtualized services thereby helping to avoid the complexity and effort of configuring scenarios with the real services and systems.

- **Integration testing composite services.** ESB developers implementing service components on platforms such as IBM WebSphere Enterprise Service Bus are often implementing composite services which need to integrate with other service on the bus. Much of the testing effort is spent just coordinating deployment and configuration of these dependent services, with the real complexity lying in trying to get reliable and repeatable integration behavior during tests. Rational Test Virtualization Server allows these developers to virtualize the services that their service components depend on, allowing them to achieve integration testing without the hassle and complexity of dealing with the real dependencies.
- **Integration testing automated business processes.** Developers implementing automated business processes in tools such as IBM Business Process Manager are in a similar situation to ESB developers implementing composite services. They too integrate with other services to produce composite behavior, and therefore they can use Rational Test Virtualization Server to virtualize consumed services, helping to remove the planning bottleneck and making these developers more self sufficient.

"We estimate that Rational Test Workbench reduces the time required to validate 'rate filing' by 94 percent, from 5,600 hours to 320 hours, and prior to this software, manual testing represented 67 percent of the total rate filing time."

—An insurance provider

Exposing service-enabled mainframe to mobile and portal domains

Reduce service dependencies. Modern portal applications implemented using platforms such as IBM WebSphere Portal Server are popular because of their ability to bring together flexible combinations of business data and functionality from various business applications services and back-end systems. The portal development team needs to wait for these services and back-end systems to be available before they can test, which can complicate planning. Rational Test Workbench and Rational Test Virtualization Server can help solve this problem by allowing the portal developers to test against virtualized services and systems.

Respond more quickly to mobile market needs. Modern mobile application development tools such as IBM Worklight® allow functionality to be more quickly transferred to your mobile users. Virtualized services enable development teams to implement both the mobile client applications with Worklight and back-end services in parallel. They can test these applications separately by using Rational Test Virtualization Server without being held back by dependency constraints so they fit into the tight iterations of agile development techniques.

Getting started with service virtualization

Rational Test Workbench offers functional, regression, load and integration testing to address the quality challenges of highly complex applications. And, Rational Test Workbench helps improve your team's agility and productivity.

Rational Test Virtualization Server models and simulates real system behavior to help eliminate application test dependencies and reduce infrastructure costs. The Rational solution can enable agile software development, helping to accelerate the delivery of complex test infrastructure and support integration testing earlier and more frequently in the development cycle.

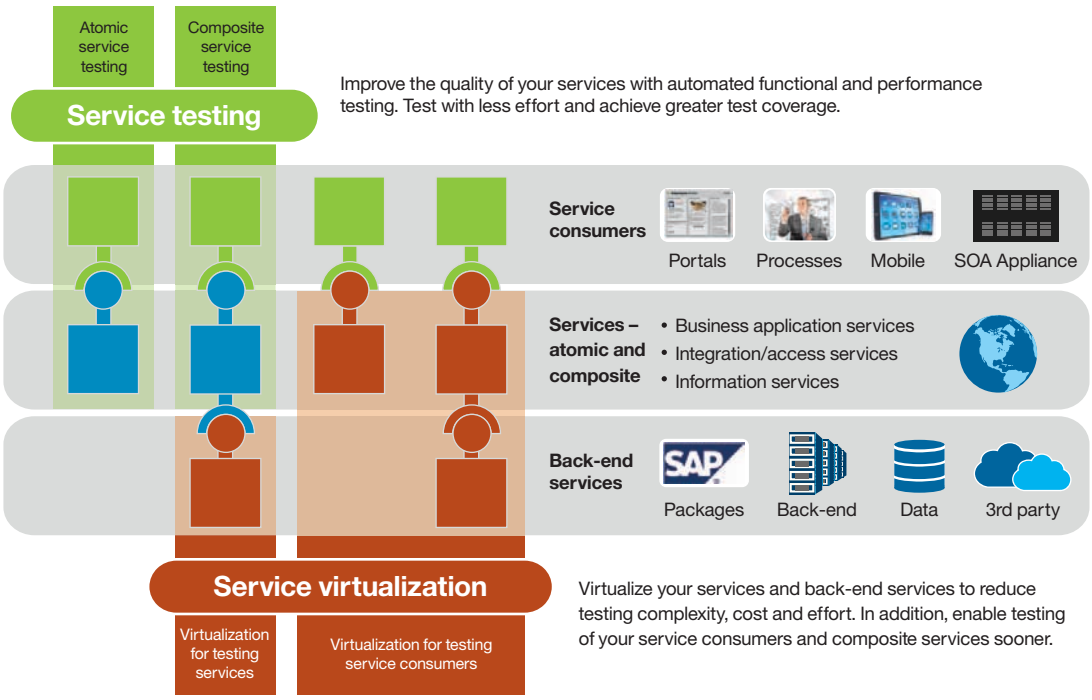

Figure 3. Service testing and virtualization of service architectures

Conclusion

Agile development focuses on delivering value consistently in short sprint cycles. Agile teams are dedicated to many practices that improve quality up front such as test-driven development, stable working code and continuous integration. But for complex heterogeneous systems, the challenges of testing the end-to-end system become a bottleneck. The impact of unexpected changes is exponentially increased when dependent components are unavailable, leading to highly weak plans. Service virtualization helps address these risks and brings system integration testing into each sprint. As a result, quality and time to market can be significantly improved. Rational test solutions provide integrated testing and help simplify the creation of virtualized test environments for optimized testing and development activities.

For more information

To learn more about the IBM solutions, please contact your IBM representative or IBM Business Partner, or visit the following websites:

- **ibm.com**/software/rational/products/rtvs/
- **ibm.com**/software/rational/products/rtw/

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

About the authors

Monica Luke

Monica Luke has over 20 years of experience in software engineering. She joined IBM Rational software ten years ago in the test organization. Since then, Monica led several test automation teams, held the role of test automation architect and earned an Outstanding Technical Achievement Award for a test automation framework that is widely used internally at IBM. In 2010, she moved into the IBM Rational Strategic Offerings team, helping to drive integration to accelerate client value throughout the Collaborative Lifecycle Management tools and starting in 2012, Monica is leading the effort to accelerate agile testing in a Collaborative Lifecycle environment with the Rational Test Workbench and Rational Test Virtualization technology.

Greg Hodgkinson

Gregory Hodgkinson is the Rational Practice Director at Prolifics. Previous to that he was a founder, director and the SOA lead at 7irene, a visionary software solutions company. He has 15 years of experience in software architecture, initially specializing in the field of component-based development (CBD), then moving into SOA. His extended area of expertise is the software development process, and he assists Prolifics and IBM customers in adopting agile development processes and SOA methods. He is still very much a practitioner and has been responsible for service architectures for a number of FTSE 100 companies. He presents on agile SOA process and methods at both Rational and WebSphere and other events. He has also co-authored a Redbook on SOA solutions, and contributes to DeveloperWorks.

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
March 2013

RAW14319-USEN-00