Oakridge
Infotech

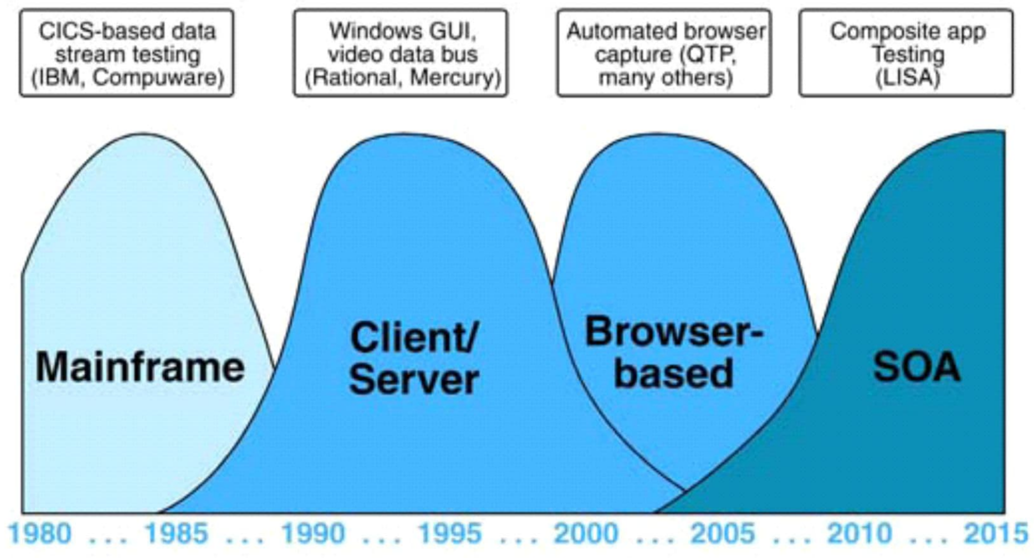
WHY SOA TESTING

WHY SOA TESTING

Service-Oriented Architecture, or SOA, is a strategy for building and integrating enterprise applications that has allowed companies to meet the needs of business with much greater agility and responsiveness. These next-generation apps are no longer built through long-term "big-bang" implementation projects. Instead, SOA applications are built from new and reused technologies, which are leveraged as smaller units of functionality called "Services," integrated as workflows in smaller iterations, and consumed at runtime.

IT BUDGETS ARE TRENDING TOWARD SOA APPLICATION METHODS

The majority of development and integration budgets are moving toward SOA initiatives, and SOA Testing capabilities must evolve with them. Services-based applications are now in the mainstream for large-scale enterprise development. Gartner estimates more than 80% of development and integration budgets will be dedicated to applications delivered in a services-based form (whether the business wants to admit it's SOA or not!). There are many reasons to be excited about the value SOA can offer, in terms of business agility and potential cost savings which we'll discuss in this paper.



However, with these potential improvements comes an increase in the potential points of failure. SOA applications introduce increased complexity and a higher rate of change. The Y-axis on this graph represents the growing complexity/interdependencies in the environment and the X-axis represents the increasing rate of change. SOA puts our focus entirely on mitigating the inherent business risks of complexity and change in the IT environment.

SOA IS SOMETHING YOU MUST DO AS AN ORGANIZATION

Achieving success in SOA requires commitment of your key people, before playing around with technology. It is no surprise that a 2008 Burton Group survey notes that at least 50% of SOA projects are either delayed or stalled due to organizational challenges.
Quality will become a primary governor on the enterprise's success in successfully achieving the agility and cost benefits expected of SOA. With so many interconnected parts making up applications that can be delivered virtually anywhere, testing and validation are no longer a mere matter of finding bugs within the developer's code, or problems that occur on the screen a given user interface. Software quality processes must evolve with the architecture, to genuinely test a business process and maintain context across the entire workflow.

ENTERPRISES CAN ATTAIN SOA QUALITY BY ACHIEVING THE THREE C'S OF SOA TESTING:

1. Comprehensive testing of business workflows across every heterogeneous technology layer of the SOA, at both a system and component level.

2. Collaborative involvement of the whole team in quality throughout the lifecycle. Enable both developers and non-programmers to define and share verifiable test cases that prove how SOA meets business requirements.

3. Continuous validation of a changing SOA deployment as it is consumed at runtime, ensuring that business requirements will continue to be met as the system dynamically changes.

The Oakridge Infotech SOA Testing, Validation & Virtualization suite was built from the ground up to offer comprehensive, collaborative and continuous testing for the next generation of SOA applications. This paper will discuss how your team, can achieve enterprise-strength quality, and mitigate the risk of frequent changes and complexity in today's SOA world.

DEFINITION IS SOA

OA is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically re-configurable end –to-end business processes realization using interface-based service descriptions.

SOA has become more than a preferred approach to application delivery – it has become a competitive mandate for any enterprise that relies upon software for core business processes (almost every large business).

GOALS DRIVING THE ADOPTION OF SOA

Increased business agility and competitive functionality

- The ability to rapidly leverage existing components, or services offered by business partners, to be responsive to customers and provide new business functionality in a short timeframe, is a leading motivator for adopting SOA.

- Faster time-to-market is realized, as businesses gain better control over their workflows and can quickly adjust them based on customer needs.

Reduced integration cost

- "Loosely-coupled" forms of integration allow a software component to be created and used independently and stitched together with other components to create a workflow at lower cost than a large implementation process.

- Industry specifications and standards for integration, such as WSDL for web services definitions, provide some common elements for integration between development teams; however every company has several elements that are far from standardized.

Increased asset reuse

- A service component within SOA can be leveraged by multiple applications or workflows. The service may also consume or rely on data from other services.

- Legacy databases and application components can be exposed as services within SOA, meaning the business will be able to maintain key existing business logic through the service where it resides, rather than rebuilding it from scratch.

Reduced business risk

- Application failures will continue to cost businesses billions of dollars in lost profitability. With a modularly constructed SOA, companies can simply repair or replace smaller units of functionality, mitigating some costly failures by containing them to smaller problem areas that are repaired quicker.

- Executives are increasingly held accountable for failure to provide accurate financial reports and forecasts. SOA Governance is being driven in part by compliance efforts such as HIPAA, and risk reduction can be better enabled through increased business transparency if the enterprise applications are well constructed.

According to Forbes study, a $10 billion company with a $500 million IT budget can save $50 million per year from a broad SOA adoption after a five-year horizon of implementing SOA in at least 75 percent of its applications. But before you can count these savings, you have to ensure that the SOA implementations will successfully meet business requirements. Quality is a mandate for success in SOA, even more so than in past architectures.

When embarking on a SOA strategy the first time, you probably had some architect or manager draw an idealized "SOA architecture diagram" on the whiteboard that represented the end goal, something like this: "We will take all our applications and expose them as services, and then we'll hook them up to our supplier services, then our customers will consume our services as well." Seems easy enough and you wondered why you weren't doing SOA all along.

Why is SOA testing such a different beast than previous forms of browser, client/server and mainframe testing? Many of the benefits of SOA correlate to the primary challenges to testing an SOA application.

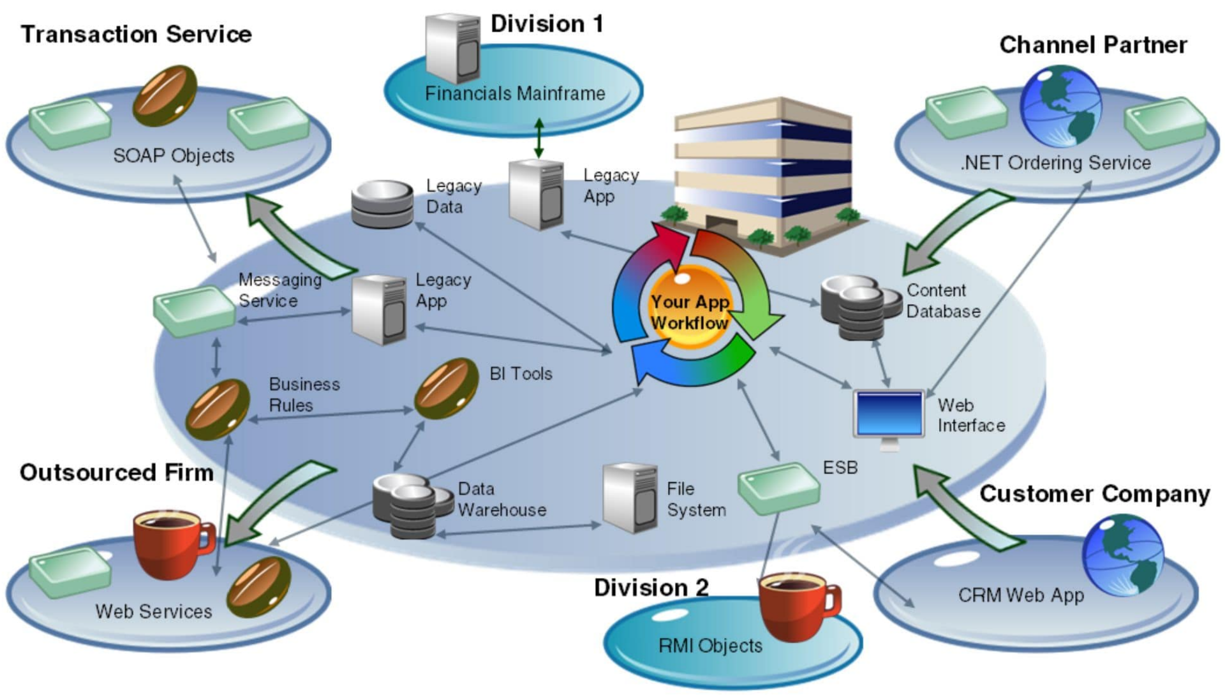
According to Gartner "Organizations that are pushing beyond basic Web services will quickly find the need for improved testing tools. Services are created and put in use, and unlike a single application that may or may not evolve a lot over the years (thus keeping defects hidden), a service will be used in ways that may never have been intended, the applications consuming it will be changing frequently, and the consumers of the services may be internal, external, technical or end users. Thus, undetected defects delivered into production could create serious downstream issues."

1. SOA is comprised of heterogeneous technology

SOA systems are based on heterogeneous technologies. No longer can we expect to test an application that was developed in a standard language, by a unified group, as a single project, sitting on a single application server and delivered through a standardized browser interface. The ability to string together multiple types of components to form a business workflow allows for unconstrained thinking from an architect's perspective, and paranoia from a tester's perspective. In SOA, application logic is generally not housed at the user interface and database level as in client/server, nor is it exclusively stored in a back-end as in venerable mainframe systems. The business rules and behaviors of an SOA live in the middle tier, operating within any number of technologies, residing outside the department, or even outside the company.

New issues and errors arise when you make multiple components talk to each other to support a business requirement. In SOA, more unique technology types, multiplied by more points of connection = an exponential increase in possible failure points.

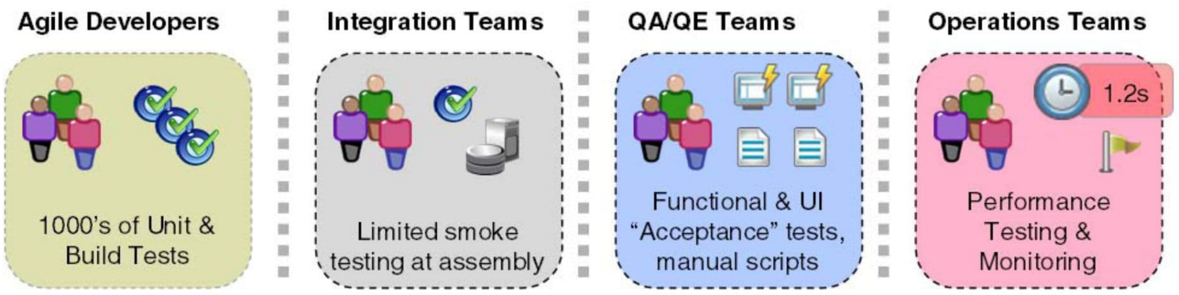
So what happens to the idealized "whiteboard SOA" shown above? It would likely look a lot more like this in reality...



The post-SOA world, as shown here, offers a vast array of options for how you assemble or consume business workflows across multiple technologies, both inside and outside your business infrastructure.

Web Services Testing is Not SOA Testing
We here at Oakridge Infotech educate customers all the time that there's a big difference between web services testing, and full SOA testing. Many people equate web services with SOA, and so from a testing point of view they equate WSDL/SOAP testing with the testing of SOA. While it is true that a number of initiatives for doing SOA are very web services centric, Aberdeen's last research on this points out that only about 50% of the SOA initiatives at large companies are web services based. There are a variety of technologies being used to create that commoditized middleware for SOA. While we love web services for this, other technologies such as ESB messaging layers, BPM and other back-end mainframes and data-driven SOA are also very relevant for SOA testing, especially in enterprises that are moving from nascent web service enablement projects to real SOA.

2. SOA teams are distributed, making collaboration difficult
Ensuring quality within a single development team, working on a single complete implementation, was difficult enough. Now, like it or not, you have a much larger, highly distributed team to match the heterogeneous nature of the SOA. With an SOA, application "stress points" can be anywhere, and will change as multiple parties inside and outside of your team can add or modify the individual services your application relies on or contributes to.


Agile Developers
1000's of Unit & Build Tests
Integration Teams
Limited smoke testing at assembly
QA/QE Teams
Functional & UI "Acceptance" tests, manual scripts
Operations Teams
1.2s
Performance Testing & Monitoring

Silo Testing Teams in SOA You have a lack of test continuity – different tools used for each component and phase of testing – therefore, your efforts don't survive long enough for the entire team to collaborate to share root causes, or cure the problem. There are often quality silos between Developers conducting Unit tests, Integration/Operations teams, and QA teams conducting Acceptance Testing. Finding the root causes of problems across the middle tiers of SOA applications is difficult. Testing at the front end user interface becomes irrelevant when it provides no insight into what is actually happening on the back end. And expecting developers to find missed requirements by conducting more unit testing at the code level doesn't get the team there either – it may find some bugs in the component-level code, but it won't demonstrate why a business requirement isn't being met.

Who is accountable for SOA quality?
Many of the quality issues that used to arise in conventional application development were easier to find simply because there were fewer parties responsible for the application. In SOA development however, you are leveraging elements that are not created by a single team – from legacy components designed by long-gone programmers, databases managed by other departments, or decision logic and commerce interfaces operated by business partners or outsourced teams.

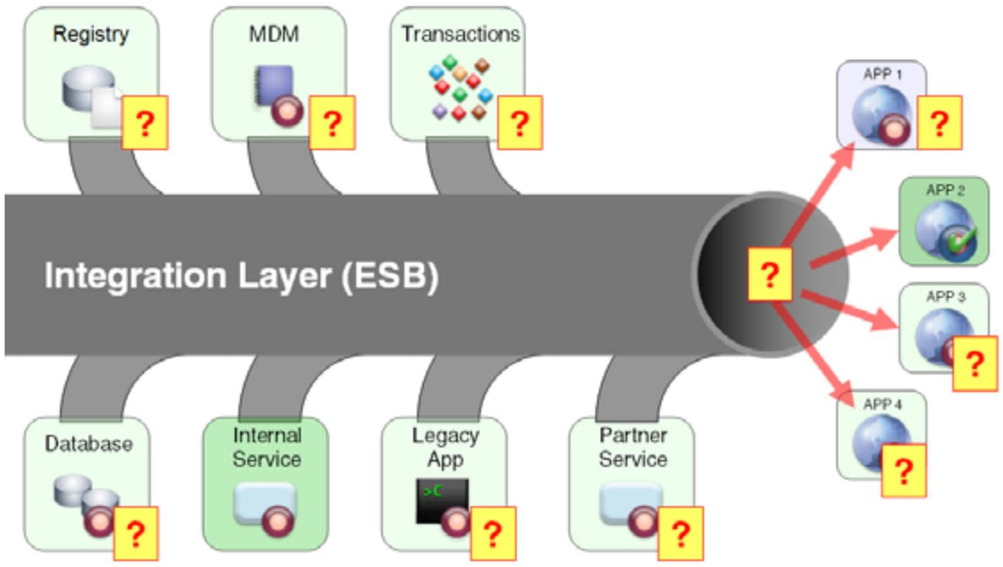
The entire extended organization of partners, stakeholders and constituents must collaborate on testing, if we are to expect to achieve the agility benefit of having business functionality that meets our needs, delivered faster.

SOA architectures are interdependent and continuously changing
The services architectures of today are bound to evolve with the enterprise and with technology in general. New features are driven into software by the process of business competition, which forces the rapid evolution of the business systems that must meet customer needs. According to the Aberdeen report, "It's no surprise that the top factor for implementing SOA, which 50% of survey respondents cited, is development of new capabilities." Take SOA evolution and complexity as a given, whether or not your systems have been validated under the new business conditions.

Testing SOA is less about structure, more about flow
Selecting an SOA approach is not like designing and building a house for your applications that you walk away from when it is done. It is an ongoing process for moving forward into a world of options. No company can afford to take a 1-3 year break from business while their systems are rebuilt and tested. Your apps must constantly integrate with new and existing tools, partners and legacy apps, while gradually moving toward the ideal strategy. And unlike the old client-server days, the test phase does not end after deployment – it is really just heating up then.

Consider a typical SOA implementation found within a larger corporation. The company will have several internal and end-user applications it must build and support. To deploy these apps more rapidly at lower cost, the applications can share and consume data and business logic housed within legacy apps and other partner services through an integration layer (often an ESB) Enterprise Service Bus). The apps, services, and enabling business rules can all be shared through one integration layer. But what is happening inside the pipe as services change, and interactions between them changes as well?

In an SOA, testing as a phase that occurs during component development and integration is useful, but it won't help you trap for unintended consequences. Since SOA involves many moving and evolving parts that are constantly leveraging each other, your testing capabilities must evolve from "What went wrong?" to "What if?," so you can continuously, proactively validate that changes in one layer of the application, do not affect other components or dependent workflows.



Unintended Consequences occur when you try to correct one aspect of an SOA application without visibility into the interdependent services it interacts with and the other applications that consume those services. Making a change to one service may fix the root cause of your application error, while potentially disrupting any other element of the SOA. It is impossible to consistently test, when you are trying to hit a moving target with fragile manual tests. Change is constant in SOA, as each service is on its own lifecycle, so you must be able to continuously validate SOA to prevent unintended consequences.

Solution: Comprehensive, Collaborative, Continuous SOA Testing