Project Number: 318786

Model and Inference Driven Automated testing of Services architectures

White Paper

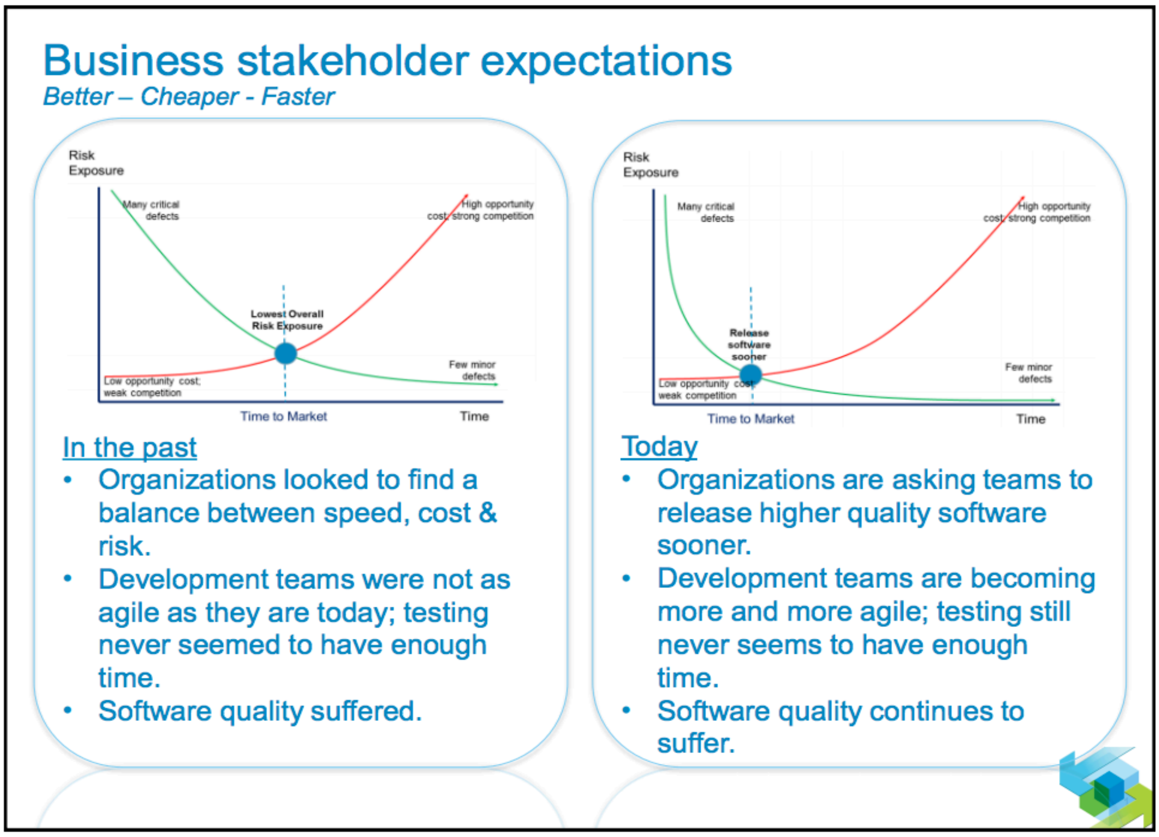
Risk and Impacts of Inadequate infrastructure for Service Oriented Architecture Testing

Revision Chart

Version	Date	Author (Partner)	Description
1	2 December 2015	Andrea Nicolai	Final Version

1 MARKET ANALYSIS AND MIDAS MARKET NICHE

Software testing represents perhaps one of the last major opportunities for businesses to reduce IT overheads and improve efficiency through automated testing tools. Many of the traditional barriers associated with software testing (ill-defined testing and requirements processes, limited understanding of actual testing costs) can be overcome by selecting an automated testing, indeed automation of software testing can reduce the cost of the testing function (by 25% or more, as well as improving time to market and overall software quality).



Business stakeholder expectations

Better – Cheaper - Faster

Risk Exposure
Many critical defects
High opportunity cost; strong competition
Lowest Overall Risk Exposure
Few minor defects
Low opportunity cost; weak competition
Time to Market
Time
Risk Exposure
Many critical defects
High opportunity cost; strong competition
Release software sooner
Few minor defects
Low opportunity cost; weak competition
Time to Market
Time

In the past
- Organizations looked to find a balance between speed, cost & risk.
- Development teams were not as agile as they are today; testing never seemed to have enough time.
- Software quality suffered.

Today
- Organizations are asking teams to release higher quality software sooner.
- Development teams are becoming more and more agile; testing still never seems to have enough time.
- Software quality continues to suffer.

Figure 6 Business expectations

This is vital because so many businesses now heavily rely on IT applications functioning faultlessly, time after time. Testing activities are closely linked to the challenges that companies currently face: flexible reaction to changes on the market and customer side, high speed of introduction of new products through optimized "time-to-market", and greater efficiency in delivering services. Software-testing and quality assurance is one of the most important IT topics for companies, and there is a high interest in collaboration with professional service providers for software-testing and quality management and a large set of companies already integrate external service providers into their testing activities, in the majority of cases regularly or on the basis of long-term with managed test services agreements.

Service Virtualisation Testing (SVT) particular sweet spot is when application development and testing teams are under pressure to develop and test faster while also dealing with complex application scenarios.

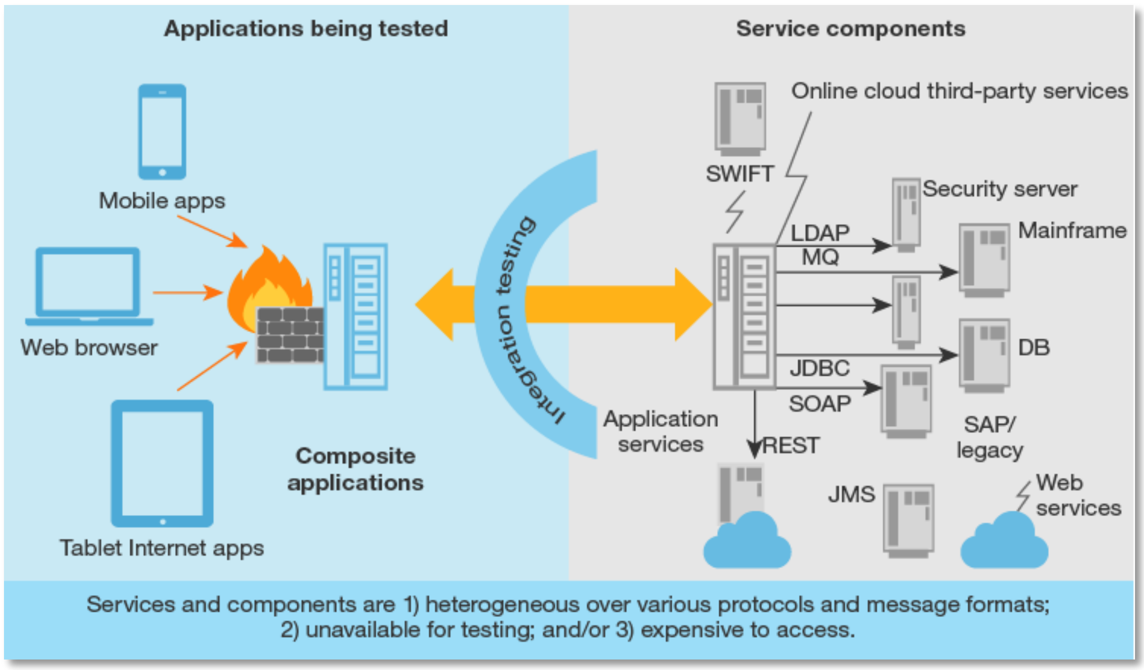

Applications being tested
Service components
Mobile apps
Web browser
Tablet Internet apps
Composite applications
Integration testing
Online cloud third-party services
SWIFT
Security server
Mainframe
LDAP
MQ
JDBC
SOAP
DB
SAP/ legacy
Application services
REST
JMS
Web services
Services and components are 1) heterogeneous over various protocols and message formats; 2) unavailable for testing; and/or 3) expensive to access.

Figure 7 Integration Testing Complex Architectures

The pressure increases as the demand for more speed increases. Firms adopt SVT to:

- Create and provision complex test environments. A hallmark of SVT tools is their ability to simulate hard-to-duplicate production environments. Applications are increasingly composed of a plethora of services that run on diverse application infrastructure; these services are not always easily accessible for testing and can be expensive to access repeatedly. For example, a mobile insurance claim app may access customer profiles from an eCommerce platform, claims data from a mainframe, and third-party services like Google Maps. It can be a challenge to create a test environment that represents or duplicates all of these interdependent services. SVT enables testing through service virtualization and test scenario simulation to recreate a test environment that's as close as possible to the production environment.

- Test earlier and more often. Multiple development teams working in parallel often face service dependencies that hold up full integration testing. For example, team A may use a service developed by team B but delay testing until team B finishes an initial version of the service. Team A can create a service stub to continue its work, but that requires additional development time and may not represent the full functionality required to perform a proper test. SVT tools allow developers to create virtual services that more closely simulate the functionality of the service. This allows developers and QA professionals to start testing much earlier, leading to a shorter SDLC and fewer bugs in production. If we look at results from surveys we can understand how today developers are approaching testing, more than half does not test at all.

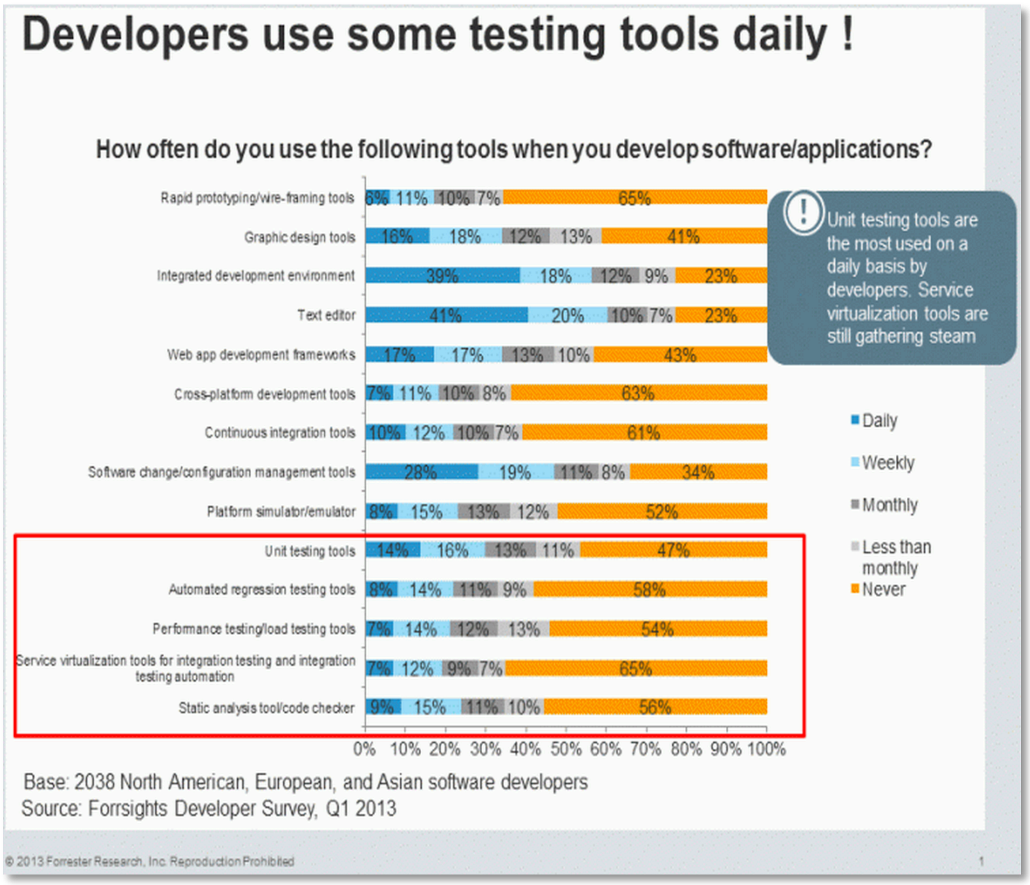


Figure 8 Developers Use of testing tools

- Automate regression testing. Many applications integrate with old legacy systems for which no one has the source code or the knowledge to automate regression tests. These legacy apps might carry large batteries of manual regression tests that consume a lot of time and energy. SVT enables the automation of manual regression tests by recording payloads exchanged with legacy apps on the wire and then virtualizing services based on the payloads so that the team can run regression tests automatically. This eliminates the need to know the code or app behavior to create virtual assets to test against, saving time and improving delivery speed even in the presence of complex legacy systems.

- Safely enable continuous integration. Development, testing, and operations are under increased pressure to deliver new application features and updates more frequently. As a result, many firms are moving to an Agile development SDLC and continuous integration of new versions in production. The danger of releasing application updates more frequently is that new bugs will appear. SVT tools combine simulated production and automated regression testing to provide continuous testing for the SDLC, resulting in safer, more successful continuous integration. The integration of SVT tools with release automation and DevOps tools enables continuous testing support for broader application life-cycle management (ALM).

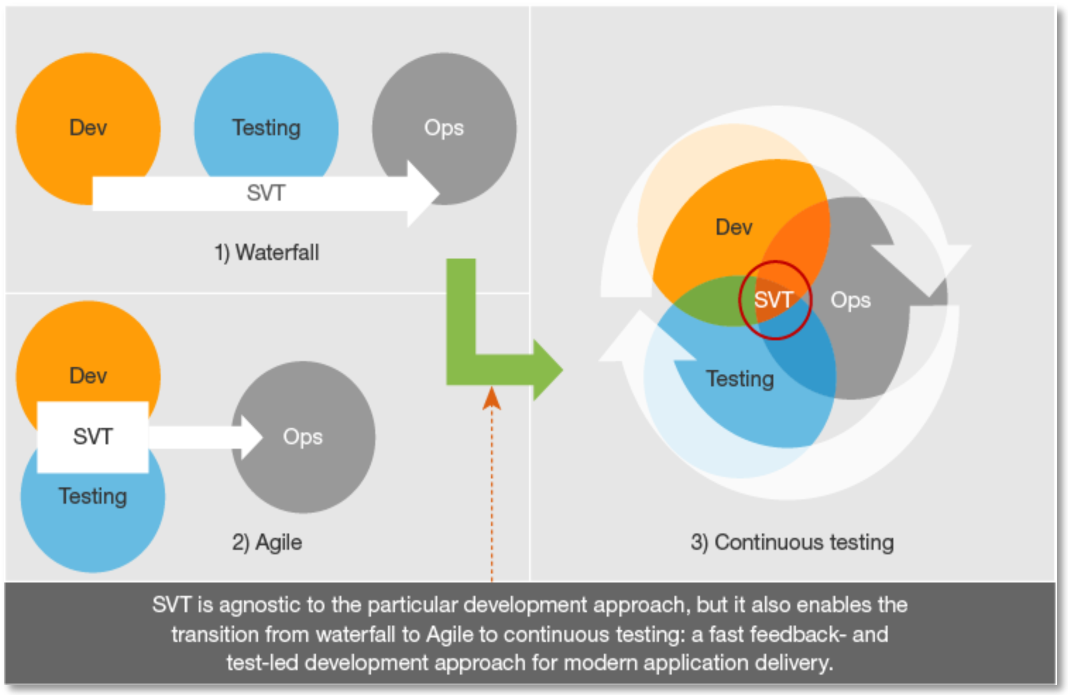

Dev
Testing
Ops
SVT
1) Waterfall
Dev
SVT
Testing
Ops
2) Agile
Dev
SVT
Ops
Testing
3) Continuous testing
SVT is agnostic to the particular development approach, but it also enables the transition from waterfall to Agile to continuous testing: a fast feedback- and test-led development approach for modern application delivery.

Figure 9 SVT and Development approach

- Lower stub development costs. If developers want to test the integration of services that aren't available, they have to create and test their own stubs of these services. While this might be convenient in some cases, it often gets quite expensive, as developers spend time building code that doesn't go toward new features or create value for the business. SVT reduces these costs by providing supporting tools to create stubs more quickly and allowing them to be shared across multiple teams efficiently and for later testing purposes. Applying SVT increases team productivity and lowers development costs.

1.1 MARKET OVERVIEW AND SIZE

Software failures are everyday more expensive. On the day of the announcement of a software failure, organization lost an average of -2.3 Billion dollars of shareholder value. This equates to about -3.75% of shareholders value, With social media and news feeds on mobile devices – news outlets are ready to pounce on issues raising. News articles about an organization's second offense increase on average of 167%.

Just as an example a series of software failures and security breaches left Sony’s gaming services down for weeks, several analysts called for the ousting of the Sony CEO. Estimated cumulative loss for Sony was around $18Billion

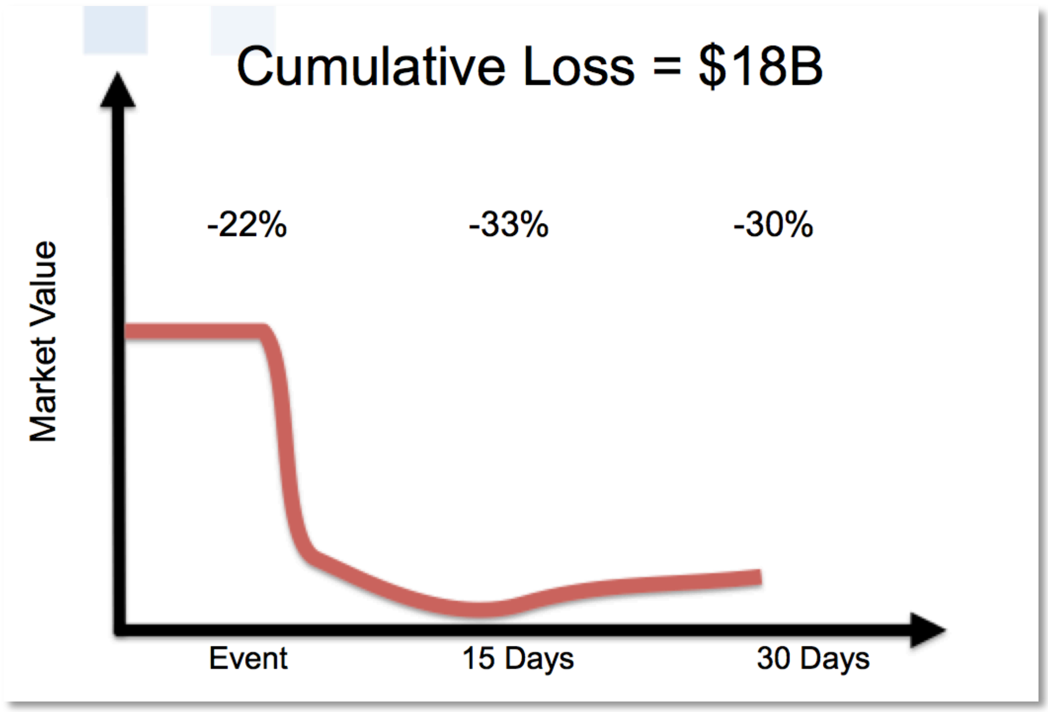


Figure 10 Sony estimated loss for software errors in gaming errors

Also, notable is that the markets don’t forget. Organizations that had a second offense were punished harder with an average of -5.68% decline in stock price.

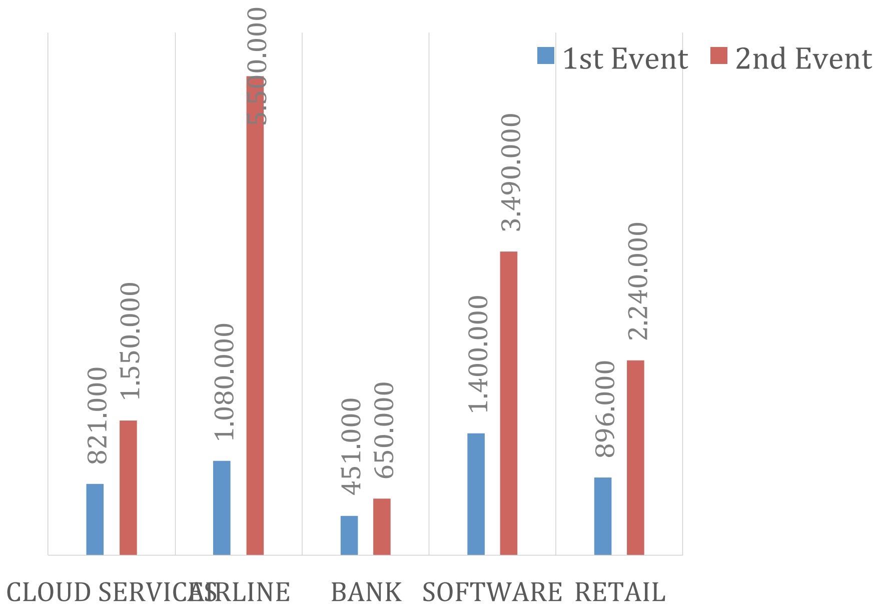


Figure 11 Loss of software errors

With social media and news feeds on mobile devices – news outlets are ready to pounce. News articles about an organization’s second offense increase on average of 167%.

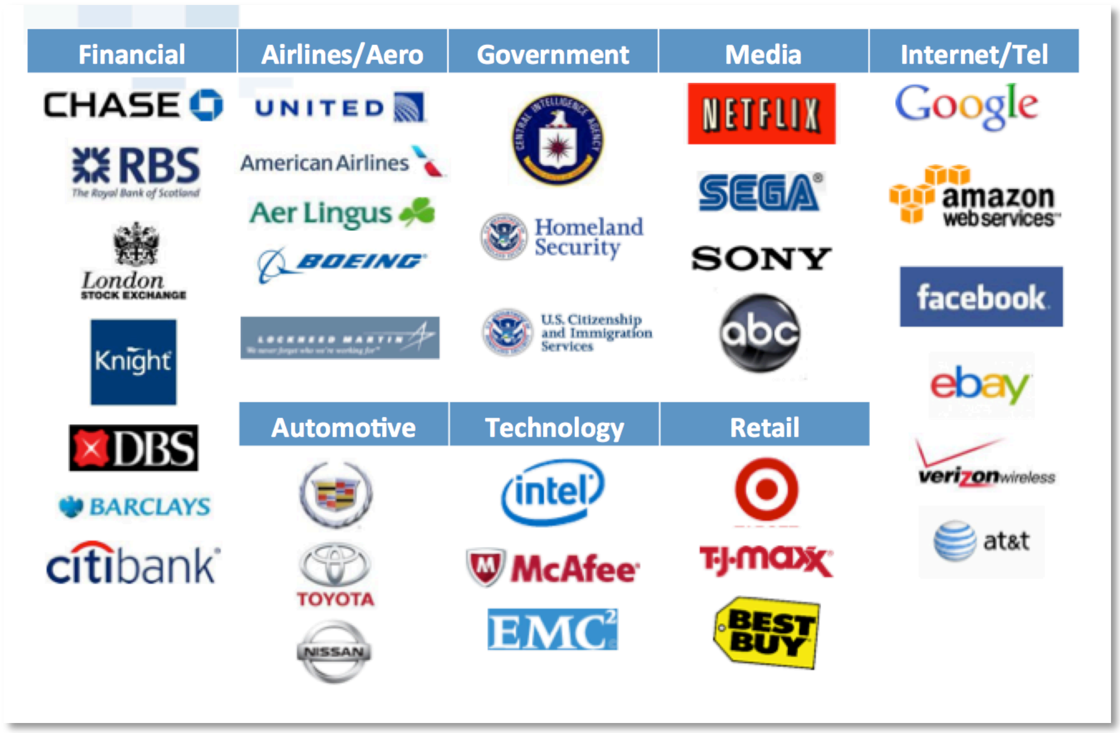


Figure 12 Companies affected by serious software errors

Assessment of automated testing market is quite a difficult tasks given the segmentation of the market and of the different testing approaches. For this document we will focus on what is defined as service virtualisation of testing (SVT).

Global Testing outsourcing market has increased three times every four years. It was evaluated at 6.1B$ in 2005 and is estimated to be about 60B$ today.

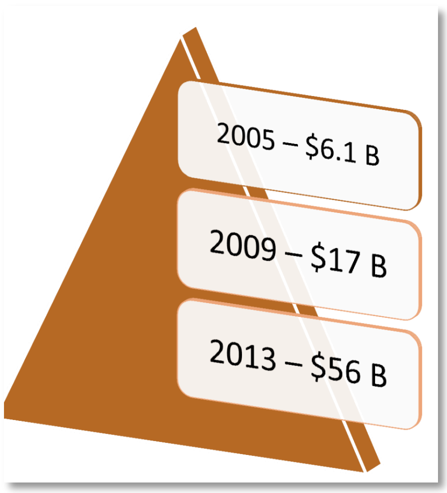


Figure 13 Testing global market size estimation

Several research organization saw then a high double-digit growth in 2011 at 33.2% on low numbers in the cloud testing and Automated Software Quality SaaS with continued reinvestment. User engagement is driving adoption and uptake in this competitive area for automated software quality. Although the market revenue size is still slight and somewhat early in its evolution, high growth numbers on a small base in 2011 (and also in 2010) are significant. Among others IDC assessed the cloud testing and ASQ SaaS market at $261.4 million for 2011, up from $196.2 million and 28.6% growth in 2010, with a higher growth of 37.8% expected for 2012. Cloud testing and ASQ SaaS combined represented around 11.8% of the overall ASQ $2.2 billion market for 2011

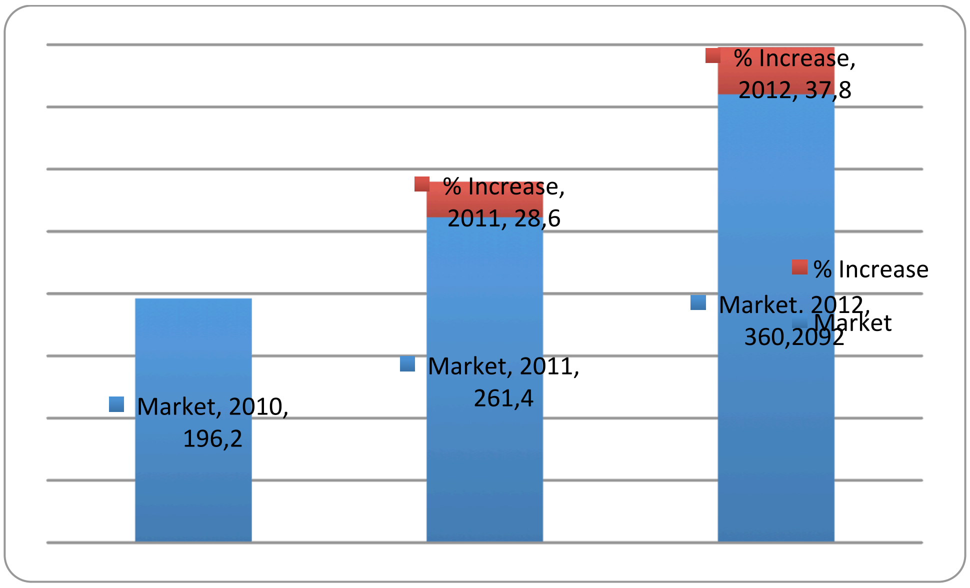


Figure 14 Cloud testing market trends

1.2 MARKET TRENDS

The testing tool landscape is changing under several pulls of user needs. Some analysts do identify the following trends:

1. Testing and development are getting more and more integrated and becoming an integral part of the SDLC from its very initial phases. Developers are adjusting to this trend;
2. Developers love open source tools and choose them as a first option. Because developers are getting more involved in testing, specifically in unit and automation (functional and nonfunctional testing), performance and integration testing, they also have a strong say in the tools used for testing. You will see in the testing tools doc research that open source tools are flourishing in all main testing categories: test management, test data management, automation, and performance. In tools, categories like test-driven development, unit testing, code quality, and bug tracking are open source and developers are an entrenched binomial.
3. Service virtualization tools. With software becoming more strategic to enterprises, software quality and testing are becoming first-class citizens aiming at primary budgets. So simulation in the form of service virtualization (a higher level of abstraction compared with the virtualized test environments and labs you are probably more familiar with) becomes an important enabler for providing a stable testing environment that simulates a real complex software integration landscape.
4. Companies like IBM, HP, and CA are all heating up their marketing engines to position themselves in this nascent market. Parasoft has a solution in this space too. We are seeing an interesting evolution of this whole area into testing optimization, where integration of service virtualization with downstream automation tools provides a complete end-to-end testing and continuous deployment solution (both IBM's acquisition of UrbanCode and CA's acquisition of Nolio give an indication of this).

The huge change of testing is undergoing has an impact on all of the testing tools categories: from test management tools to functional and nonfunctional automation testing to performance and load testing to security and test data management tools and, finally, to the newcomers' services virtualization for software testing tools. More than ever there is a need for a constant flow of information among BAs, developers, and testers. As testing shifts more in the hands of developers, vendors need to adapt tools that easily plug into the developers' integrated development environments (IDEs), while QA and other software professionals prefer tools that offer a higher level of abstraction and are easy to use. The pressure on tool vendors is to emphasize qualities like collaboration/communication, continuous automation, simplicity, and seamless integration.

1.3 COMPETITIVE ENVIRONMENT: SOA TESTING INDUSTRY

There is a number of SVT testing tools available in the software market. Although the core functions of these tools are similar, they differ in functionality, features, usability and interoperability. CA Technologies, HP, IBM, Parasoft, and SmartBear Software do concentrate on service.

CA Technologies and IBM are among the leaders with rich features and a compelling strategy. CA and IBM lead with rich, comprehensive, in-depth SVT capabilities coupled with a strong strategy and rich post-sales services and training. CA focuses on the downstream part of the life cycle by integrating the LISA (formerly ITKO) SVT solution with its release automation (acquired from Nolio) and API management (acquired from Layer 7 Technologies). IBM leads with its RTVS (formerly Green Hat) SVT, but has a more comprehensive end-to-end DevOps (via its acquisition of UrbanCode) and ALM testing integration tool strategy. CA has more accurate simulation capabilities, while IBM has stronger test lab provisioning. CA has a larger user base; IBM has stronger market momentum.

HP and Parasoft are strong performers. Parasoft’s features are on par with the leaders, but its strategy was not good enough to qualify for that tier. Parasoft is a solid choice for customers that prefer broad capabilities, ease of use, and quick startup. Parasoft supports continuous testing during the SDLC and a clear separation of roles between testers who design, develop, and consume virtual assets and those who manage environments.

HP arrived late to the SVT market; until 2011, HP offered ITKO until CA acquired the product, at which time HP decided to go with its own product, HP SV. HP is especially attractive to its huge captive market. Its product provides ease of use and an enjoyable user experience when testing scenarios need to leverage integration with its own ALM and testing tools like LoadRunner. HP’s strategic product road map is particularly interesting for implementing self-service testing factories.

SmartBear Software satisfies techies but lacks key SVT features. SmartBear focuses narrowly on the SOAP and REST web services market. SmartBear SoapUI Pro provides developers and technical testers with scripting and wizard-based features that need to create their own stubs or mocking services for testing. SoapUI Pro is for clients that take a very lightweight approach to SVT. SmartBear also provides a very successful and open source mocking tool, SoapUI, which acts as a gateway to the SoapUI Pro commercial solution. Both products have thousands of users and better market penetration and momentum than the other SVT players. In addition to these general-purpose solutions, firms that wish to benefit from SVT solutions may also consider these alternative or complementary tools.

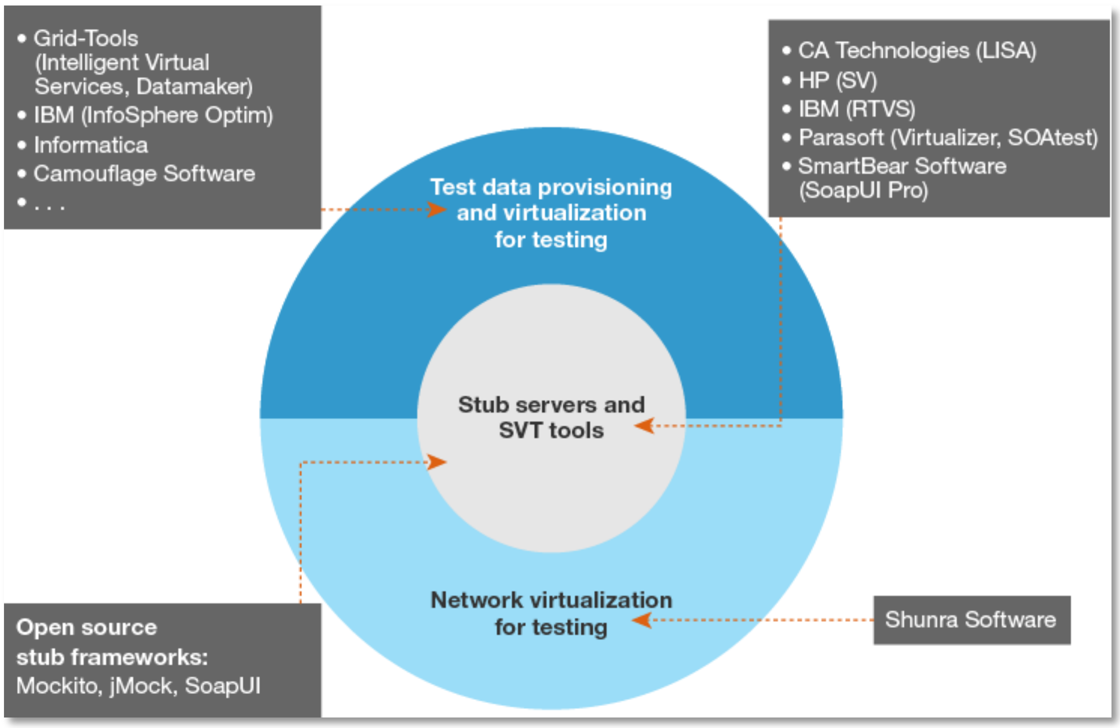

• Grid-Tools (Intelligent Virtual Services, Datamaker)
• IBM (InfoSphere Optim)
• Informatica
• Camouflage Software
• . . .
• CA Technologies (LISA)
• HP (SV)
• IBM (RTVS)
• Parasoft (Virtualizer, SOAtest)
• SmartBear Software (SoapUI Pro)
Test data provisioning and virtualization for testing
Stub servers and SVT tools
Network virtualization for testing
Shunra Software
Open source stub frameworks: Mockito, jMock, SoapUI

Figure 15 SVT Complementary tools

Stubbing or mocking servers. These tools integrate with development environments and offer features, scripting languages, or programming frameworks to speed stub development. They're the little brothers of SVT tools, originally created to test service-oriented architecture; now, firms often use them in conjunction with more comprehensive SVT suites. Clients use mocking servers as an alternative to broader SVT suites when they don't have a complex application scenario with multiple platforms, protocols, and message formats or aren't looking to scale SVT as an enterprise service for continuous testing and development. Parasoft offers a little brother to Virtualize called SOAtest, while SmartBear offers open source SoapUI in addition to SoapUI Pro. Other open source options are Mockito and jMock.

Test data provisioning and virtualization tools. These are comprehensive data management and test data management (TDM) tools that are highly synergistic with SVT. TDM tools allow firms to discover, subset, mask, refresh, and analyze test data to improve quality and testing. Integrating TDM and SVT tools enhances testing and quality by making hard-to-reach "proper" test data virtually accessible. TDM tools allow for masking and desensitizing virtualized test data and subsetting the quality of test data to be used in simulated scenarios. TDM players active in the SVT space include Grid-Tools with Intelligent Virtual Services and dynamic message masking and IBM with InfoSphere Optim Test Data Management. It does not appear that other TDM players — like Informatica or Camouflage Software — are active in SVT.

Network virtualization tools (NVT) for software testing. NVT allows firms to emulate network conditions like latency, limited bandwidth, packet loss, and jitter to test application performance and other parameters. NVT is especially effective when multiple network connections are required to support third-party services or external resources, and when unexpected loads or concurrency happen in any geography. Shunra Software is the most significant — and only — player with an NVT solution that we know of and is a key partner of most of the SVT tool providers. The integration of Shunra and SVT tools enables the testing of complex end-to-end application and service performance scenarios while taking lower-level network variations into consideration.

Microsoft takes an alternate approach, with just-in-time production testing and monitoring. Microsoft does not offer SVT tools. Instead, it takes an alternate approach within the hosted Visual Studio Online service and the Visual Studio IDE that leverages production application data analytics. Microsoft has been offering IntelliTrace for quite some time, which takes the problem away from production and gives it to developers to solve, helping them to immediately analyze the root cause. Today, Application Insights is the new collector of IntelliTrace..

The aim of the MIDAS project is model-based ***extreme automation*** of SOA/API testing tasks. The MIDAS TAAS user builds a collection of structural, functional and behavioural models of the services architecture under test that allow the MIDAS test methods to automate: (i) the production of test cases and oracles and (ii) the planning, scheduling, execution, arbitration and reporting of unit and integration test campaigns.

The ***measure of the impact*** of the MIDAS framework and technology on the software and service industry practice is an important issue. It is acknowledged in the MIDAS DOW as an objective to be attained and has been put under consideration from the beginning of the project. The problem can be formulated in terms of the following question: what is the good method that allows estimating how the SOA/API testing practices will be impacted - qualitatively and quantitatively - by the availability of the MIDAS technology? The search for an answer to this question is not easy per se and is made even more complex because of two facts:

1. Service testing is not only an activity of service providers, but also of service users. Service providers and service users are ***roles*** of developers whose software uses services in order to implement the services that it provides. Hence, service developers have to test the services that their software uses,

in addition to test the services that it provides, and this from the beginning of the SDLC (*test driven development*). This need, which is brought by the soar of the SOA/API economy, pushes for a real change with respect to the actual development and testing rules of game.

2. The MIDAS SAAS approach - and the underlying cloud technology - downsizes of one or more order of magnitudes *all the testing equipment costs*, making these equipment accessible to people that traditionally do not practice or practice very little testing (such as SMEs). Moreover, on the basis of pay-per-use policy, the MIDAS SAAS approach downsizes the licensing costs of test methods and tools too. In other terms, the MIDAS technology *disrupts* the SOA testing tool market. The objective is that people (such as SMEs) practicing little testing or no testing at all enter the "new" SOA/API testing-as-a-service market. As a consequence, in order to estimate correctly the impact, we should measure the actual cost of **no testing** too.

After almost two years of research, we should agree on some facts:

- There are very little publicly available data and studies on the testing costs in general and, a fortiori, on SOA testing costs. The only interesting source of data and arguments is the famous NIST study[1]. In particular, the European Commission hasn't issued anything comparable for the European Union. Moreover, reports of market research organisations such as Gartner are about the (SOA) *testing tool market*, not the SOA *testing practice*.
- There are no data and no studies at all about the consequences and costs of **no testing** (except in the aforementioned NIST study).
- Interviewed professionals - managers, developers and consultants – have limited knowledge and awareness of the *testing and no testing practices and costs* and are reluctant to talk frankly about this issue.

Putting aside the equipment and licensing costs, it is really difficult to obtain reliable data of the *human effort* needed to perform the testing activity and of the eventual downsizing of this effort brought by the MIDAS technology. This proposal, which is firstly directed to the MIDAS UMC, aims to collect this kind of data. The general idea developed in the remainder of this document is a sort of "reverse engineering" of the problem that also satisfies the principle of "comparing what is comparable". This document is not a work plan; it simply exposes and arguments the idea and calls for remarks, suggestions and a discussion about the issue.

[1] NIST-02-3 (2002). *The Economic Impact of Inadequate Infrastructure for Software Testing*. Planning Report 02-3. National Institute Of Standards & Technology.
URL http://www.nist.gov/director/planning/upload/report02-3.pdf

1.4 THE MAIN CONTRIBUTIONS OF MIDAS INNOVATION

MIDAS is a Service/API testing facility accessible as a service on the Internet. The MIDAS four key innovation points are:

1. Low-cost Testing As A Service on cloud.
2. Programmable testing facility through APIs.
3. Evolutionary portfolio of enhanced test methods.
4. Extreme testing automation.

These points are illustrated in the paragraphs below.

1.4.1 Low-cost Testing As A Service on cloud

The MIDAS testing facility is delivered as a service implemented on cloud. This delivery mode ensures: (i) a reduction in equipment costs of at least one order of magnitude and (ii) the shift from capital expenditure to operational expenditure.

The service is self-provisioned, and the logical and physical resource allocation is no only scalable, but, above all, highly elastic. The pay-per-use policy can be applied in a radical way because it is coupled with transparent accounting, pricing and billing based on fair metering.

The “radical” elasticity of resource allocation and radical “pay per use” pricing policy are the most important disruptive innovations in terms of costs. There is no comparable offer in the market today, even for more basic testing functionalities[2].

Fair metering means that the MIDAS utilisation price could be calculated on the basis of the underlying cloud platform price of the computational resources effectively used by the customer, plus a margin. This policy can be applied only by using cloud providers that practice radical pay per use policy and sophisticated accounting features[3]. Furthermore, in a dynamic perspective, the public cloud price will lower[4] rapidly. The choice that cloud-based service providers, such as the MIDAS facility provider, are confronted with will be between (i) following strictly the cloud provider lowering price policy and (ii) practicing extra-margins. Probably the realistic stance is in-between. In any case, the prices of the testing as a service delivery mode will decrease fast in the future, as a mechanical effect of the lowering of cloud costs. The disruptive cost cut (making the same thing with costs that are at least one order of magnitude lower), as well as decreasing costs, are essential traits of the MIDAS offer.

The usage of a public cloud provider for the underlying infrastructure could raise a security concern. The conclusions of an objective analysis are that security concern is still present as a perception but is far from reality for a MIDAS customer. In

[2] It can be compared with free products such as SoapUI, that the user runs on her/his premises.

[3] This is the case for AWS, the current MIDAS cloud provider.

[4] In the past eight years AWS has lowered its prices 42 times (once every 2/3 months).

fact, what is running on MIDAS cloud is the test system, not the systems under test. The customer executable code is never uploaded on the MIDAS cloud. The customer critical data are never uploaded on the MIDAS facility. The customer uploads on the MIDAS cloud only models for testing and testing data. We discuss in the next paragraphs privacy concerns about these models and data.

Models for testing describe the externally accessible functions, the interfaces and the external behaviour (at the interfaces) of the SAUT components and their mutual service dependencies.

In general, services/API are classified as *public* (available on the Internet), *partner* (available in infranets of business partners) and *private* (accessible on the intranet of the organisation). Services architectures can be classified in the same categories. A services architecture is: (i) *public* if all its services are public; (ii) *partner* if at least one service is partner and there are no private services; (iii) *private* if at least one service is private.

For public architectures there is no privacy concern by definition. Partner architectures are multi-owner and deployed in partner networks, whereas private architectures are mono-owner and deployed on the owner private network. For both, models (service interfaces and services architecture topology) could be confidential.

Generally speaking, service interfaces can be private[5] but are seldom confidential. The trend, in the most important business domains is towards standardisation. This is the case for the MIDAS pilot domains (Health and Logistics) that can be extremely sensitive to privacy. The data exchanged in the field within these architectures are classified (especially for Health), but the data formats are standard and public. The MIDAS pilot partners have put in place services architectures on the basis of business standards of interfaces and data formats.

Testing data are confidential only if also the interfaces are confidential, because from instantiated data and concrete interactions it is possible to reverse engineer the data formats, interfaces, exchange protocols. But testing data per se are not confidential because they are fictitious.

Testing data **must not be** customer's real data. Using real data for test cases/oracles is very bad practice and, in some business domains (such as Health), it is totally forbidden by current regulations. A customer can utilise as testing data the result of some transformations of real data (data obfuscation, anonymization). MIDAS could supply downloadable data obfuscation software components to be executed on the customer premises. Of course, MIDAS cannot give any non-disclosure guarantee about real data that are used as testing data and uploaded on the MIDAS platform.

The last security concern is about the interaction of the MIDAS platform with the customers' services architectures, whose components are installed by their owners on premises or on their public or private clouds. MIDAS interacts with the SAUT, not

[5] Interfaces, as the source code, can be copyrighted. This means that you cannot run your service using interfaces copyrighted by someone else. As copyrighted, they are disclosed. The only private interfaces that are also confidential are covered by the trade secret (and they are not copyrighted).

with the SAIF (Services Architecture In the Field). The SAUT **must not** be the SAIF. The same considerations made about real data apply. Testing directly the SAIF, concurrently with business operations, is **very bad practice** and, in many business domains (such as Health), it is forbidden by current regulations. Of course, MIDAS cannot give any guarantee about functional and operational reliability of a SAIF that is tested concurrently with the flow of real business operations. The obvious good practice is to put in place a SAUT that is totally separated from the SAIF.

The conclusions about security concerns, such as access control, privacy and integrity of data are that: (i) a very large part of the MIDAS potential audience does not have special strong security concerns about models for testing and testing data and (ii) a more general trend is that Cloud FUD (fear, uncertainty and doubt) is still present as a perception but far from reality[6]. Hence, the current security features of the MIDAS platform (access control, user sandbox, etc.) based on the underlying cloud platform security services seems to be sufficient for a large part of the potential market. Moreover, the delivery mode (TAAS) and low costs of the MIDAS testing services allow decentralised fruition by enterprise departments in BYOC (bring your own cloud) mode[7].

1.4.2 Programmable testing facility through APIs

The MIDAS functionalities are fully accessible through public APIs that are documented on the MIDAS portal. This key innovation (in the world of testing tools) allows MIDAS entering the digital service economy.

Four customer/partner populations are concerned with this key innovation feature: (i) end users (business developers); (ii) software engineering tool editors, (iii) business package and vertical services vendors; (iv) consultancy, including professional testers.

The MIDAS APIs allow the business service developer (the end user) to “integrate” the MIDAS testing services with her/his home made CAST (Computer aided software testing) environment. The integration process, supported by pay per use and radical elasticity, is smooth. The business developer can start consuming only some of the MIDAS testing functionality and end replacing her/his environment with MIDAS. In any case, users are confident because they can easily modify the type and frequency of MIDAS utilisation without paying extra-fees (if you do nothing, you pay nothing). Of course, s/he can stop the MIDAS utilisation at any moment.

The MIDAS APIs allow software engineering tool developers to add value to their offer by proposing to access MIDAS test methods and functionalities directly from their tool in an integrated manner.

The tools that are candidate for integration with MIDAS could be classified in the following categories:

[6] According to a Ponemon Institute LLC study (http://www.ponemon.org/) focused on cloud security, 66% of businesses that are focused on cloud security put sensitive information in the cloud, while only 40% of the businesses considered to be less concerned with security do the same.

[7] Such as Dropbox. A credit card (and a minimal budget) suffices to use MIDAS services.

1. Modelling tools,
2. Integrated Development Environments (IDEs),
3. Application Lifecycle Management (ALM) tools,
4. Computer-Aided Software Testing (CAST) tools,
5. SOA governance tools,
6. API management tools.

Globally speaking, actors of this category could be resellers of MIDAS services. They can be also MIDAS competitors (especially CAST tool editors). Possibly, the MIDAS services can be consumed in “OEM” mode, as if they were directly offered by the tool. Of course, a specific marketing strategy with detailed contractual policies must be designed and put in place towards this population. Partnerships with actors of this category create value chains.

The third category of customers/partners includes vendors of business packages and business services in any business domain (starting, for instance, with the Health and Logistics domains). This category is represented in the project consortium by Dedalus.

Today, in the digital service economy: (i) business packages must be able to provide and to consume services through APIs, and (ii) business services (API) are consumed in unexpected manners. The focus of MIDAS is service integration testing, hence these actors: (i) may utilise MIDAS for internal testing needs, (ii) may propose to their customers the MIDAS facility, packaged MIDAS artefacts that facilitate MIDAS use in a specific context and an accompanying support to integrate (and test the integration of) their package or service within more general services architecture. These actors can be MIDAS resellers. Of course, also for this category, a specific marketing strategy with detailed contractual policies must be designed and put in place. Partnerships with actors of this category create value chains.

Last but not least, the consultancy partners can foster the MIDAS diffusion and are key actors of the MIDAS success. Many businesses can be interested in the MIDAS services, but do not have the time and the personnel for entering the MIDAS world. Two important issues must be managed: (i) the model-based approach to test automation and (ii) the interaction through API[8]. On both points, consultants can help customers to put in place quickly effective testing with MIDAS. Consultants are facilitators: it must be made clear that first level consultancy on testing with MIDAS is not the business of the MIDAS Company[9], whose job is building e-learning tools and

[8] A probable evolution of the MIDAS facility, similar with the evolution of AWS, is that the most current utilisations of the MIDAS facility will be driven through a scripting language alongside the invocations of the MIDAS APIs by the client software.

[9] MIDAS Company is a generic and conceptual term to cover the **who** of “who will carry out the exploitation”. It could be an individual partner, sub-consortium, full consortium, joint venture, new company, third party or any other type of entity.

ease of use, and that creating an independent MIDAS experts' ecosystem around the MIDAS facility is a important factor of the MIDAS success.

The MIDAS business maintains an ambivalent relationship with professional testers. On one side, MIDAS extreme automation makes obsolete many of the activities that are today performed "by hand" by professional testers. On the other side, professional testers could enter very quickly the MIDAS consultancy market, certainly quicker than "generic" IT consultancies (that are known for having low testing culture and competence). MIDAS boosts the productivity of the professional testers, and the smartest of them can take advantage of this situation and use MIDAS to enlarge their market footprint.

Of course, also for this category, a specific marketing strategy with detailed contractual policies must be designed and put in place. Partnerships with actors of this category create value chains.

1.4.3 Evolutionary portfolio of enhanced test methods

The fruition mode of the MIDAS functionalities (TAAS on cloud, APIs accessible on the Internet) requires continuous evolution: MIDAS is a living service that shall continuously offer new, up to date, enhanced test methods and functionalities.

The third MIDAS key innovation point is that the MIDAS facility is built on an *open platform* that is loosely coupled with the test methods that it hosts. The researchers and experts on testing and the developers of testing tools (*test method developers* in the MIDAS terminology) upload, register and install their test methods, developed on their premises, on a sandbox of the MIDAS platform that allows safely checking, verifying and testing the methods. Once a rigorous certification process is achieved successfully, these test methods are made available to the MIDAS users.

MIDAS can become an attracting point for research and development about testing (for instance, for current and future European projects[10]). A specific strategy for capturing new advances in the research and practice of SOA/API testing and making them available as new enhanced test methods that foster the automation of effective test generation and test run cycles must be put in place. It could be interesting to create a community of MIDAS test method developers, to organise test method contests, to develop consensus about how to measure test method efficacy and efficiency, etc.. Apart from important technical problems of test method verification and validation, this strategy must clarify issues such as:

- What about the intellectual property of these test methods?
- How to reward the authors?
- Who is in charge of the maintenance and evolution?

[10] There are ongoing talks with the FP7 Prowess project (http://www.prowessproject.eu/), a sibling of MIDAS. The general idea is that some test methods developed within this project (that doesn't target exclusively service testing as MIDAS does) can be uploaded on the MIDAS platform and become part of the MIDAS test method portfolio. There are no intellectual property concerns about this operation, because these methods are open source.

Partnership with researchers and experts on testing and with the developers of testing tools creates value chains.

1.4.4 Extreme testing automation

Last but not least, extreme SOA testing automation is the final MIDAS objective and key innovation point. The other innovation points / objectives detailed in the preceding paragraphs are *instrumental* with respect to this one.

Services/APIs are the building blocks of the digital economy. Modern applications are composite aggregating not only components but also private, partner and public Services/APIs. We witness a progressive shift from 'specify, design and implement' components towards 'select, evaluate and integrate' *services through APIs*.

Services/APIs are the enablers of critical business transactions and, potentially, the weakest links in these transactions. The quality of the APIs that a business produce and consume is now more important than ever - if you consume it, you own it.

The business impact of any application failure is the same regardless of whether the fault lies within the components directly developed by the business or the APIs that the business consumes. Finger pointing does little to foster customer satisfaction and brand loyalty.

When there is no alternative to digital economy, i.e. when all relevant business processes become digital, and there are no more backup manual processes, all processes and the services that support them become critical.

Testing the APIs that a business provide and consume, and the services architectures the business is involved in is the only means for increasing the confidence of the stakeholders. Test driven development of services and test driven incremental integration of services architectures are no more options.

With the rising adoption of agile development, incremental integration and continuous delivery, change is a continuous process. SOA/API testing should not be a singular event anymore, but a SDLC continuous activity.

Here we are. The question is: Why is service test so little practiced with, as a consequence, enormous costs of no testing for everybody, including the general public?

After all, SOA/API **no testing** is a risky business for both service providers and consumers: the business risk of the delivery of non-dependable services is high, because when the service is your business, non dependable service is non dependable business.

The answer has been given by the well-known NIST report [1] more than ten years ago: because SOA/API testing is expensive and hard and an effective and cheap "infrastructure technology"[11] that supports testing is lacking.

[11] The term 'infrastructure' is utilised here in a general sense. In this sense, we can say that the MIDAS facility is an infrastructure technology.

SOA/API testing is expensive because:

- it is a labour *very* intensive activity - manual test case/oracle design, manual configuration of the test environment, eyeball arbitration of test outcome, handmade test reporting, human-based scheduling of test runs, human-based planning of test cycles;
- the needed equipment (hardware, licences, maintenance) is steep;
- with the currently available technologies, the duration of the standard quality testing phases goes far beyond the time to market.

SOA/API testing is hard because:

- The testing activity is both low-rewarding and knowledge intensive. The involved skills (deep functional knowledge of the services architecture to be tested, competence of effective testing methods and tools) are scarce resources.
- Human cognitive abilities are not well-adapted to an activity that, in front of complex services architecture, requires sustained attention for long periods and the ability to cope with an enormous quantity of detailed information. The immediate consequence is that this activity is error prone (useless test cases, wrong oracles, false positives, false negatives).
- Commercially available tools offer limited functionalities, i.e. the mechanisation of few clerical tasks applied only to single-service unit testing.
- Test is difficult to plan, to schedule and to manage.

The MIDAS facility offers the complete automation of all the testing tasks in the domains of functional and security/vulnerability testing[12]. It automates all the basic testing tasks of the generation cycle (test case production, test oracle production) and of the run cycle (test execution, test arbitration, test reporting). Furthermore, it automates also the middle-management tasks (the workflow of the generation cycle and the scheduling of the test run cycle). Finally, a MIDAS objective is the automation of the complete test cycle. Moreover, the components that drive the high level tasks (the scheduler and the planner) are implemented as Bayesian agents that are able to cope with the fundamental characteristics of the test as an empirical activity based on heuristics and performed in an uncertain environment.

Extreme SOA/API testing automation, coupled with the key innovation points presented in the preceding paragraphs (low-cost testing as a service, programmable testing facility and evolutionary test method portfolio) enables a service development life cycle that is model based, test driven and adapted to incremental service integration and test.

[12] Automated performance testing should complete the commercial offer. Performance testing is not in the scope of the MIDAS project.

Extreme automation means that once the end user has built the SAUT structural, functional and behavioural models, the testing activity runs automatically, asynchronously and in background. The end user can focus on her/his job which is designing great services and MIDAS takes care of the continuous test of these services.

Conclusion

Testing activities are closely linked to the challenges that companies currently face: flexible reaction to changes on the market and customer side, fast introduction of new products through optimized “time-to-market”, and greater efficiency in delivering services.

The MIDAS SaaS approach - and the underlying cloud technology - downsizes of one or more order of magnitudes all the testing equipment costs, making the MIDAS solution accessible to all the relevant stakeholders (users) in testing domain that traditionally are excluded or perform very little testing activity(such as SMEs). Moreover, on the basis of a typical pricing policy for cloud computing, the MIDAS SAAS approach downsizes the licensing costs of test methods and tools too. In other terms, the MIDAS technology disrupts the SOA testing tool market, allowing “all” to enter the “new” SOA/API testing-as-a-service market.