MIDAS: a cloud platform for SOA testing as a service

Alberto De Francesco

Istituto di Scienza e Tecnologie dell'Informazione – CNR,
Via G. Moruzzi, 1, 56124, Pisa, Italy
and
IMT – Istituto di Studi Avanzati di Lucca,
Piazza S. Francesco, 19, 55100, Lucca, Italy
Email: alberto.defrancesco@isti.cnr.it

Claudia Di Napoli* and Maurizio Giordano

Istituto di Calcolo e Reti ad Alte Prestazioni – CNR,
Via Pietro Castellino, 111, 80131, Napoli, Italy
Email: claudia.dinapoli@cnr.it
Email: maurizio.giordano@cnr.it
*Corresponding author

Giuseppe Ottaviano, Raffaele Perego and
Nicola Tonellotto

Istituto di Scienza e Tecnologie dell'Informazione – CNR,
Via G. Moruzzi, 1, 56124, Pisa, Italy
Email: giuseppe.ottaviano@isti.cnr.it
Email: raffaele.perego@isti.cnr.it
Email: nicola.tonellotto@isti.cnr.it

Abstract: The increasing adoption of the service-oriented architecture approach makes software quality more challenging since traditional approaches for software testing are inadequate for SOA-based applications. The MIDAS platform is an integrated platform for SOA testing automation, designed and architected according to the SOA computing paradigm, and deployed on a public cloud infrastructure to tackle the variability in the computational resources necessary for testing SOA applications. It is available to end users as a testing as a service on a self-provisioning, pay-per-use, elastic basis. The cloud-based software architecture envisioned for the MIDAS platform and the strategies adopted for both its development, and its deployment on the target cloud infrastructure are here discussed, together with the solution designed and implemented in order to monitor the usage of MIDAS services and resources. Also, the strategy adopted to provide the MIDAS platform with the management of elastic resources is outlined.

Keywords: cloud computing; Amazon AWS; service-oriented architecture; SOA; software testing.

Reference to this paper should be made as follows: De Francesco, A., Di Napoli, C., Giordano, M., Ottaviano, G., Perego, R. and Tonellotto, N. (xxxx) 'MIDAS: a cloud platform for SOA testing as a service', *Int. J. High Performance Computing and Networking*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Alberto De Francesco received his Master degree in Telecommunication Engineering from the University of Pisa in 2010. He is a PhD student at IMT Lucca, and has been a research fellow at ISTI-CNR since 2012. His main research interests concern cloud computing, and web and microblogging information retrieval.

Claudia Di Napoli is a researcher at the High Performance Computing and Network Institute of CNR. She received her degree in Physics (with specialisation in cybernetics) from the University of Naples in 1992. Her research interests include service-oriented architectures, cloud computing, software agents, and automated negotiation.

Maurizio Giordano is a Research Scientist at the High Performance Computing and Network Institute of Italian CNR. He received his degree in Physics (specialisation in cybernetics). He is a Lecturer at the Computer Science Faculty of University of Naples. His research interests are high performance and cloud computing, SOA, and artificial neural networks.

Giuseppe Ottaviano has been a postdoc researcher at ISTI-CNR since 2013. He received his PhD in Computer Science from the University of Pisa in 2013. His main research interests include machine learning, cloud computing, information retrieval, and search engines.

Raffaele Perego is a senior researcher at ISTI-CNR, where he leads the HPC Laboratory. His research interests include HPC, web information retrieval, and data mining, with particular emphasis on efficiency issues of web search. He has co-authored over 120 papers on these topics published in journals and in proceedings of international conferences.

Nicola Tonellotto received his PhD in Information Engineering from the University of Pisa and in Computer Engineering from the Technical University of Dortmund in 2008. He has been a researcher at ISTI-CNR since 2006, and has been a Contract Professor at the Computer Science Department of the University of Pisa since 2009. His main research interests include cloud computing and web information retrieval.

This paper is a revised and expanded version of a paper entitled 'A SOA testing platform on the cloud: the MIDAS experience' presented at Cloud Interoperability and Federation Workshop in conjunction with the 2014 International Conference on Intelligent Networking and Collaborative Systems, Salerno, 10–12 September 2014.

1 Introduction

The service-oriented architecture (SOA) paradigm (Papazoglou and van den Heuvel 2004) supports the development of applications resulting from the integration of new and pre-existing components, *aka services*, in a business workflow. SOA services are loosely-coupled, discoverable, reusable, inter-operable and platform-agnostic components, implemented according to well-defined standards of the web service technology (Alonso et al., 2004). They can be described, published, categorised, discovered and dynamically assembled, bound or unbound at any time and as needed.

The key characteristics of SOA-based applications are posing new challenges to software testing since conventional testing methodologies are inadequate to cover SOA testing requirements (Kalamegam and Godandapani, 2012). In particular, the lack of observable behaviour of the involved systems, the lack of trust in the employed engineering methods, and the lack of direct control of the implementation life-cycles, make SOA testing a heavy, complex, challenging and expensive task. There are also additional factors that make SOA testing harder such as: the late binding of the systems, the fundamental uncertainty of test verdicts, the organisational complexity, and the elastic demand of computational resources (Canfora and Di Penta, 2009). Some solutions have been proposed to address these challenges, but still far from exploiting the full potential benefits of SOA testing (Dustdar and Haslinger, 2004).

In this context, the European FP7 Project MIDAS (http://midas-project.eu) aims at designing and building an integrated platform for SOA testing automation, itself designed and architected according to the SOA computing paradigm. It provides facilities for black-box testing of single services and grey-box testing of their interactions. This platform will be made available on demand to end-users as a software system providing advanced and off-the-shelf testing methods and capabilities as a service to be paid-per-use in the digital economy era. The MIDAS TaaS is designed as a cloud-based application, deployed on a public cloud infrastructure. From the end-user perspective it will be made accessible over the internet as a multi-tenancy software as a service (SaaS), we refer to as a SOA testing as a service (SOA TaaS).

The rest of the paper is organised as follows: Section 2 describes the main features of the MIDAS cloud platform for SOA testing, and its design; Section 3 discusses why several testing software vendors/providers are moving to the cloud, together with the main MIDAS requirements that drove the choice of the underlying cloud infrastructure; Section 4 describes both the development and the deployment strategies for MIDAS, together with the environment set up to develop the MIDAS platform components; Section 5 describes the monitoring solution implemented to account for the MIDAS resources and services usage; Section 6 reports the approach followed to design the elasticity facility planned for MIDAS. Finally, Section 8 reports some concluding remarks.

2 MIDAS: a platform for automated SOA testing

The MIDAS project aims at designing and building a platform intended as a facility for SOA testing automation providing support for all testing activities:

- *test case generation* for the functional, interaction, security and quality of service aspects of a service architecture
- *test run execution*, allowing the automatic configuration and initialisation of test scenarios and the automated execution of test runs on a service architectures under test (SAUTs), residing outside the MIDAS platform
- *test run evaluation planning and scheduling*, providing intelligent methods and tools for the evaluation of test results and for the planning and scheduling of test campaigns.

The functional testing on a SOA application will determine the accuracy of a services orchestration. Security testing determines whether the SAUT protects data and maintains functionality as intended, relying on model-based fuzzing. The basic concepts covered by security testing are confidentiality, integrity, authentication, availability, authorisation, and non-repudiation. The usage-based testing within MIDAS is intended to augment the functional and security testing, by using a stochastic model of the SAUT to generate test cases.

The generation of test cases for functional and interaction testing, security testing, and usage-based testing is model-driven (Grabowski et al., 2003). Test cases are expressed in TTCN-3 languages (Grabowski et al., 2003), while the relative SAUT is described by an UTP/UML model (OMG, http://utp.omg.org) or a WS-* model.

The MIDAS platform is developed incrementally, and its functionality will be tested on two real-word use case pilots defined and implemented as SOA applications within the project: a healthcare services pilot for the management of patients affected by chronic diseases, and a GS1 logistic interoperability model supply chain pilot.

2.1 *The MIDAS users*

MIDAS TaaS has four classes of users, each one playing different roles in interacting with the platform, as reported in Figure 1.

- *End users*: they include both users responsible for planning, designing, and conducting test campaigns on service architectures (also known as *testers*), and users responsible for the creation, administration, and deployment of the service architecture under test (also known as *testees*).
- *Test method developers*: they consist of users responsible for designing and implementing test methods to be used for conducting test campaigns.
- *Administrators*: they are users responsible for managing both the identification and authentication of end users and test method developers, and the MIDAS TaaS facilities used by the administered users, including the accounting and billing of these facilities.
- *TaaS administrator*: it is the responsible entity for the entire TaaS platform, and for managing any interaction with the cloud provider selected for the MIDAS TaaS development and operation. As such, the TaaS administrator is responsible for the dynamic provisioning of all MIDAS public functionalities and for configuring the underlying cloud resources and services for the MIDAS TaaS users.

End users and test method developers are grouped in logical entities called respectively *tenancies* and *labs* that are separated user computing spaces managed by the corresponding administrators. Conceptually tenancies and labs represent units of:

a users identification and authentication

b cloud resources allocation, accounting and billing

c data and services ownership and access.

According to the project plan, only tenancies are included in the basic MIDAS platform on the cloud, since test method developers are the MIDAS technical partners in charge of developing test methods, and they directly contribute to the development of the platform itself. In fact, the test methods are integrated into the MIDAS platform, so becoming part of the complete platform deployed on the cloud, and updated, at each version increment, with the latest release of its components, as it will be detailed later on.

Each tenancy must be able to deal with its own cloud users, cloud resources, data and services in a private, sandboxed way. The MIDAS platform will contain several tenancies, each one composed of several end users. Each tenancy is managed by a tenancy administrator that interacts with the respective users, and it is responsible for creating user accounts and credentials for them.

2.2 *The MIDAS SOA design*

The MIDAS platform has been designed according to the SOA paradigm, so all its functionalities are exposed as services. The MIDAS services, also referred to as MIDAS components, are asynchronous and stateless web services accessed through well-defined APIs and communicating with each other. The APIs of such web services have been designed and implemented during the first phase of the MIDAS project, so all services are developed according to their specifications.

The MIDAS services are grouped in: *tenancy admin services*, *end users services* and *core services*, as shown in Figure 2.

The *tenancy admin services* are:

1 *identification and authentication service* (*I&A service*), that allows tenancy administrators to create and delete tenancy end users, as well as to list current members of a tenancy, and to verify that each member of a tenancy is authenticated before invoking the facilities of that tenancy

2 *accounting and billing service* (*A&B service*), that allows tenancy administrators to monitor the MIDAS cloud resources and services usage of a tenancy, and to get the corresponding updated and consolidated billing information.

Figure 1 The MIDAS users and their relationships

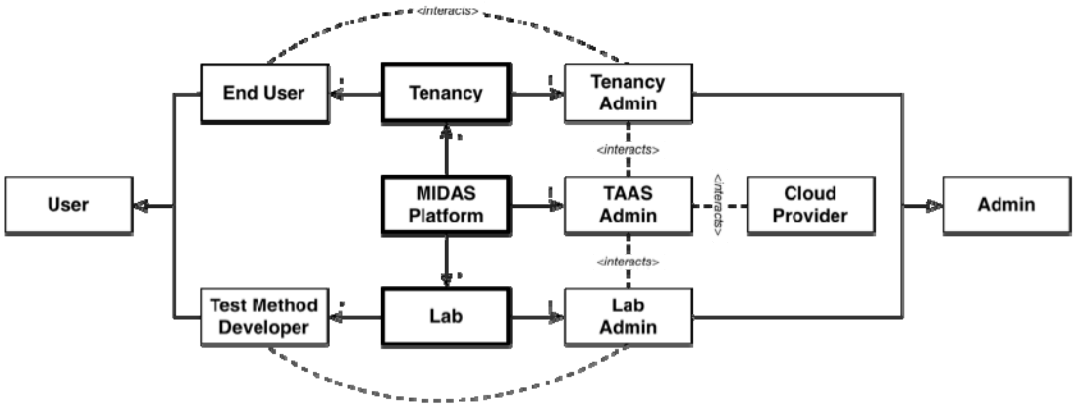
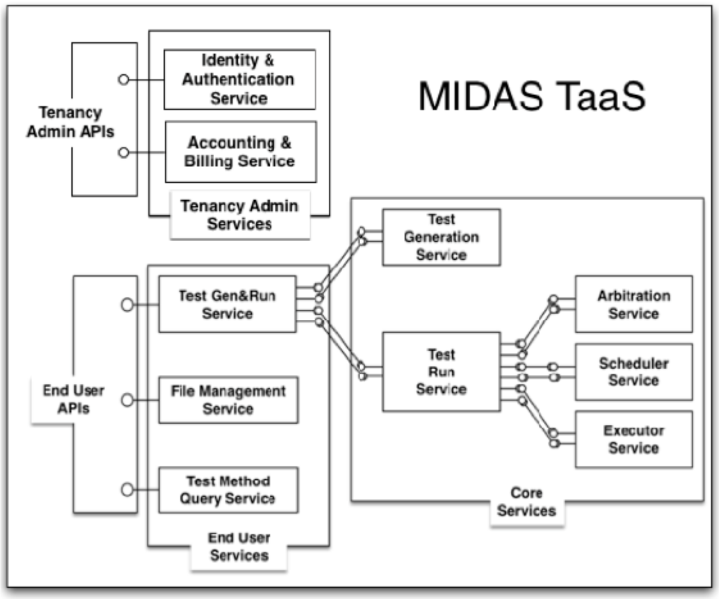


Figure 2 The MIDAS SOA design



The *end users services* are:

1 *test gen and run service*, that allows to asynchronously start the execution of a test task (either a test generation or a test execution task), and to actively poll it to inspect the status and the resulting output of any started test task

2 *test method query service*, that allows end users to list the test methods currently part of the MIDAS *portfolio*, and to retrieve the properties of any method in the portfolio

3 *file management service*, that allows end users to access the file system private to the tenancy they belong to, and to perform the usual operations supported by a file system.

The *core services*, organised in two levels (as shown in Figure 2), are:

1 *test generation service*, that is responsible for automatically generating test cases, test scripts and model transformations for testing

2 *test run service*, that coordinates the run of a specific test cycle, consisting in an optional scheduling phase, a mandatory execution phase, and an optional arbitration phase.

3 The cloud for software testing

There is a growing trend to deploy testing tools on cloud infrastructures, either public or private, in order to provide testing services through the coordinated use of cloud resources according to a pay-per-use business policy. This is due to several reasons. Software testing is seen as a necessary evil since it takes a large amount of human and dedicated infrastructure resources, although it is only one stage of the software development and maintenance lifecycle. Furthermore, business applications are becoming more and more dynamic, complex and distributed, so requiring increasingly sophisticated testing techniques and methods to deal with this complexity. In addition, for a growing number of companies maintaining in-house testing facilities that can mimic real operating environments is becoming prohibitive due to the high cost and technical difficulties, so the demand for new solutions where the testing process is outsourced and possibly automated is rapidly increasing.

The primary benefit of migrating software testing to the cloud is the cost: small and medium companies that cannot afford high capital expenditures in testing activities, can use cloud-based testing solutions in a pay-per-use manner, without spending money for acquiring and maintaining hardware facilities. The problem of having an in-house testing environment is that you have to

1 maintain it

2 buy and maintain hardware for it

3 re-purpose it for your target testing tasks

4 care about testing software/tools licenses.

An on-premise testing framework is thus a direct cost for companies. By using testing facilities provided as a cloud-based solution, it is possible to get rid of the hardware infrastructure and of commercial testing tools licenses, to streamline maintenance and to reduce repurposing efforts.

Beyond the cost there is more to cloud-based testing than expense savings. Several studies demonstrated that a high percentage of defecting testing activities is due to inaccurate test configurations and environments. Cloud-based testing service providers often offer a standardised infrastructure and pre-configured testing software images that are capable of reducing such errors considerably. This standardisation is achieved through the use of a testing software catalogue, which pushes tester users to learn a discipline of using a 'library of pre-configured and stable testing methods' and a commitment to meeting shared interfaces and configurations of testing environments. All these actions result in the faster provisioning of test environments and the ability to meet operational objectives. In conclusion, the possibility to test in the cloud leverage the cloud computing infrastructure reducing the cost of computing, while increasing testing effectiveness.

In order to set up the cloud computing resources needed to deploy the basic MIDAS platform on the cloud, the MIDAS TaaS administrator has to sign an agreement with the cloud provider reporting the pay-per-use computing resources that need to be allocated, their localisation (e.g., located in Europe), and other terms and conditions concerning their use. In addition, before a MIDAS tenancy is initialised, a service level agreement between the tenancy administrator and the TaaS administrator should be established defining terms and conditions on the use of cloud resources devoted to the specific tenancy together with additional requirements, e.g., security regulations, elastic resources demands, and so on.

3.1 Scalability requirements for MIDAS

The use of a cloud as the underlying computing infrastructure allows to address the scalability requirements of the MIDAS platform. In particular, the identified cloud resource scalability dimensions for MIDAS are:

- *Tenancy/lab space scalability.* Cloud resources are allocated to a tenancy/lab upon its creation, according to the agreement established with the MIDAS TaaS admin. In fact, the amount of allocated cloud resources is agreed according to an SLA between the MIDAS TaaS admin and the customer tenancy/lab administrator. Each time a new tenancy is created/allocated a new pool of cloud resources is assigned to it. Accordingly, upon tenancy/lab deletion all the currently associated cloud resource space has to be removed, that is all the virtual machine images (VMIs) hosting the MIDAS services have to be detached from the pool of allocated VMIs to be available as free resources for MIDAS TaaS.

- *Computing elasticity.* Inside a tenancy/lab space, initially allocated resources can scale up/down according to the current usage of the tenancy computing resources: when new resources are necessary, either because the number of registered users in a tenancy increases, or because end users testing activities increase, the cloud autoscaling facilities should be used to allocate new computing resources.

- *Storage scalability/redundancy.* Persistent storage for end users data (like test logs, journals, models, etc.) is designed to be physically shared, although logically separated (sandboxed), among different tenancy/lab spaces, i.e., among users belonging to different tenancies. Therefore, the cloud infrastructure has to provide a global storage for MIDAS, and mechanisms to ensure insulations/protection of data among different tenancies. Although we figure out a single storage entity, the cloud infrastructure should offer autoscaling, as well as redundancy of data storage to cope with failures.

- *Portfolio redundancy.* The MIDAS portfolio is the repository of test methods and test components developed by the MIDAS technological partners, and used by tenancy users to carry out their testing activities. As such, the portfolio storage is persistent as well as physically and logically shared among all MIDAS users. While tenancy users access the MIDAS portfolio in read-only mode (through the *test method query service*), test methods developers have read/write access to the portfolio (through the *test method management* service). In the deployment design we will rely only on cloud storage redundancy mechanisms in order to provide the MIDAS portfolio with the required fault tolerance features.

3.2 An IaaS cloud infrastructure for MIDAS: Amazon EC2

The cloud service model adopted for the MIDAS cloud deployment is the infrastructure as a service (IaaS) one, mainly because of the possibility it offers to developers to have full control of the entire software environment in which their applications run (see Figure 3). The benefits of this approach include support for legacy applications, and the ability to customise the environment to suit the application development needs. This is a very crucial requirement for the MIDAS platform development, since it relies on components that are developed and implemented by the MIDAS technical partners in an independent way, and that have to be integrated in a complete application. So, software development environments are required to fulfil different requirements and needs in terms of software support and customisation. Furthermore, MIDAS includes legacy commercial software, such as the TTworkbench execution engine (http://www.testingtech.com), that may require specific software support to be included in the MIDAS platform. Finally, since an IaaS cloud relies on virtualisation technologies, the portability of the MIDAS platform to a different cloud provider is guaranteed. The drawbacks of IaaS cloud solutions may be an additional complexity and effort required to setup and deploy the application.

In accordance to the project requirement of adopting a public cloud infrastructure, and to the analysis of available cloud providers, the Amazon Elastic Computing (EC2) Platform (http://aws.amazon.com/ec2/) has been adopted. It represents a good candidate solution since it currently offers the best compromise among cost, MIDAS development needs, and elasticity mechanisms. It provides advanced solutions concerning the possibility to achieve computing elasticity, and to tackle data storage reliability, persistency and fault tolerance.

4 The MIDAS development strategy

The MIDAS platform is designed according to the SOA paradigm that can fully leverage the web services paradigm to develop, deploy and operate on a cloud. Its components are asynchronous and stateless web services accessed through well-defined APIs and communicating with each other. The MIDAS services do not have tight dependencies on each other, so each service can be developed, deployed and managed by the cloud independently from the others. In theory, each MIDAS service could have its own VMI customised with the implementation-dependent software requirements (libraries, runtime environments, databases, ancillary services) as a basic deployment unit for the specific service implementation. All core services in MIDAS communicate asynchronously with the others and treat them as black boxes, without code or functional dependencies.

The overall architecture of the MIDAS integrated platform on the cloud is a layered one (as shown in Figure 4), and it is composed of three layers:

- the MIDAS front-end, allowing users to access all MIDAS functionalities made available as a TaaS
- a computing development environment available as a testing platform as a service (TPaaS), hosting all MIDAS services
- the underlying cloud computing infrastructure used according to the infrastructure as a service model.

The TPaaS layer is incrementally developed by integrating all MIDAS components independently developed by the partners step by step. Each step identifies a new release version of the MIDAS platform.

Figure 3 Layers of the cloud stack w.r.t the user control in IaaS, PaaS and SaaS (see online version for colours)

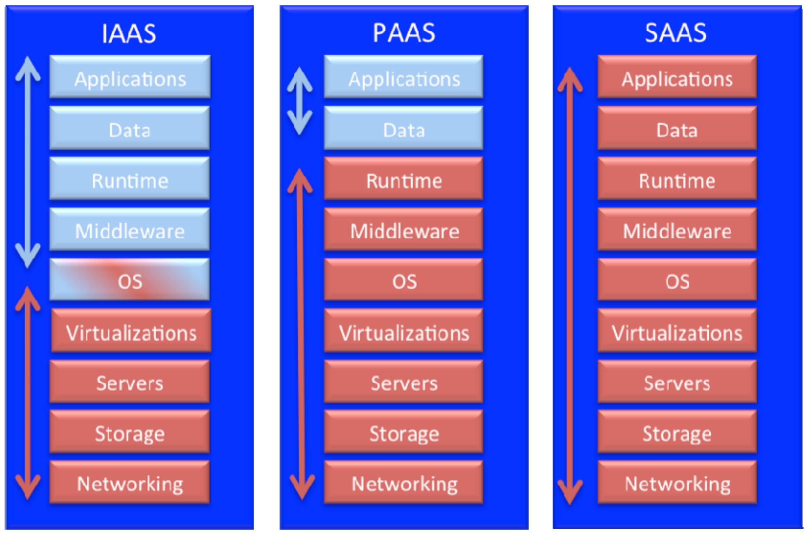


Figure 4 The MIDAS TaaS layered service architecture (see online version for colours)

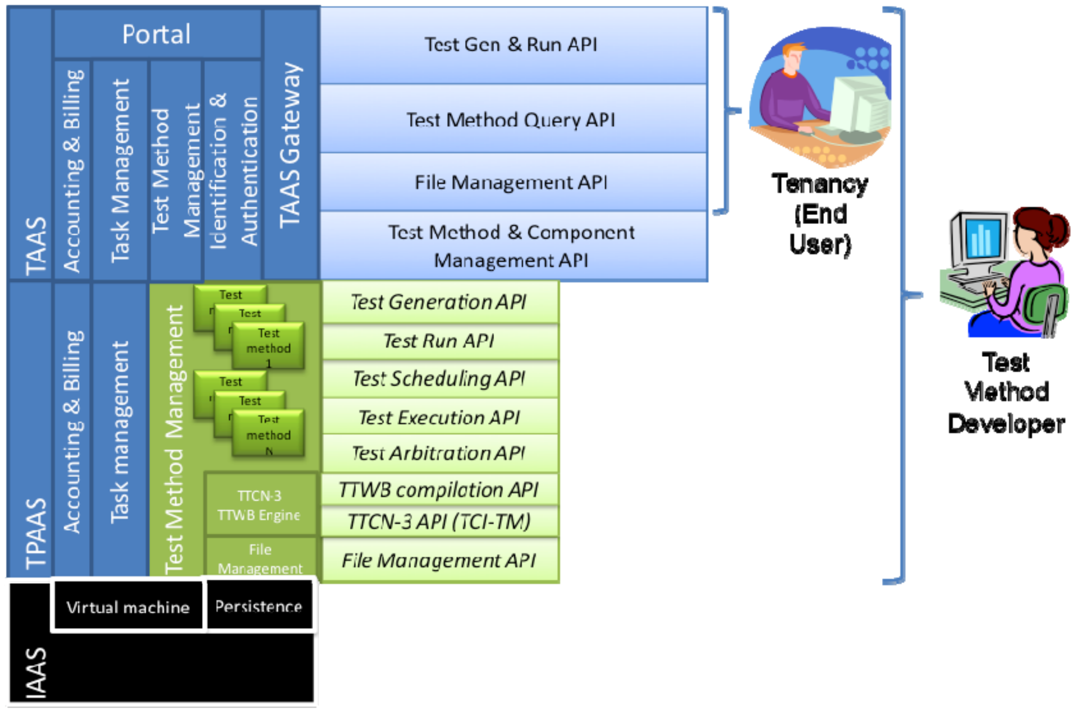
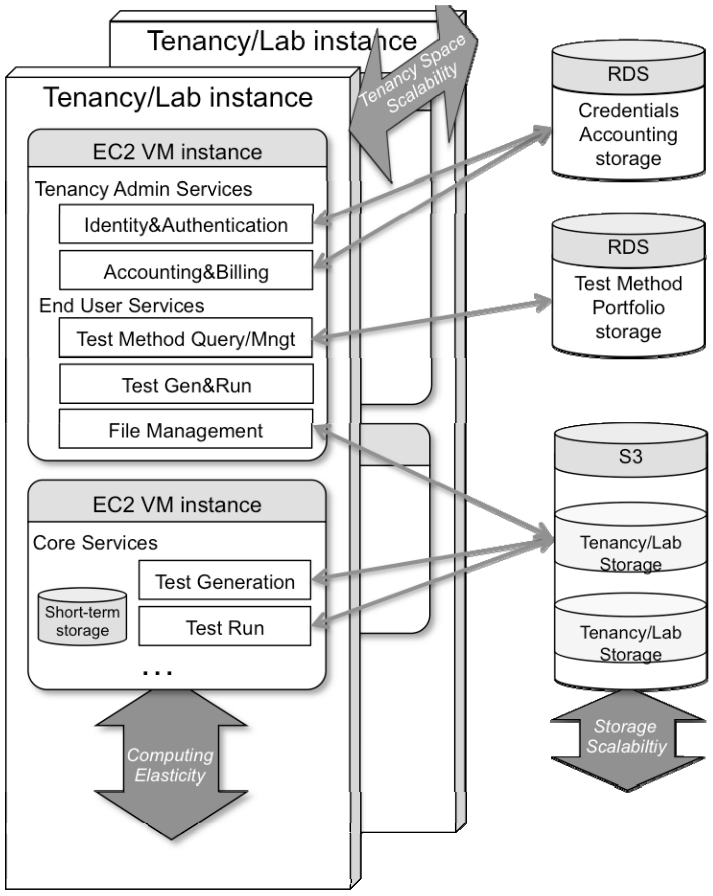


Figure 5 MIDAS deployment strategy on the cloud



4.1 The MIDAS prototype on the cloud

The MIDAS platform prototype, including the MIDAS components and the basic MIDAS user front-end (i.e., the MIDAS portal), is deployed on the cloud.

The deployment of the MIDAS TaaS platform on the cloud takes into account mainly the sandbox and scalability requirements of basic components, reported in MIDAS Consortium (2014). The MIDAS deployment strategy is depicted in Figure 5, where both labs and tenancies are reported to show their symmetric behaviour from the cloud deployment point of view. Each tenancy/lab instance has its own private cloud resources pool that can scale by exploiting the elasticity services of the underlying Amazon cloud infrastructure.

Each tenancy/lab instance is logically composed of two basic deployment units, i.e., two VMIs for each tenancy/lab, one hosting administration and end user services, one hosting core services. The rationale of this choice is due to the different scalability and elasticity requirements of the two groups of services. In fact, most of the workload is expected from the use of the core services that host the executor engine, the compiler of TTCN-3 scripts, and the other components developed by the MIDAS partners. According to these premises, end user services, tenancy admin services and the MIDAS portal reside on one VM (VM1), and the core services in another VM (VM2). In principle, there is not functional requirement that obliges to have core services in the same deployment unit. The rationale of grouping core services in a single deployment unit is twofold:

1 the entire pool of core services working instances and associated resources could be scaled as a whole through Amazon elasticity mechanisms

2 for the time being, there are no specific requirements to partition and distribute core services on separate deployment units.

The VMs computing resources can be resized to fit the CPUs, RAM, network I/O requirements of the MIDAS components.

To deploy the MIDAS platform on the cloud infrastructure with the added components integrated in it the Ansible (http://www.ansible.com) and Vagrant (Hashimoto, 2013) free software tools are used. Vagrant is used for creating customisable, lightweight, reproducible, and portable development environments made up of VMIs, while Ansible is used to automatically build and deploy on different virtual machine instances the VMIs designed for MIDAS. In the cloud, the VMs are configured with the Amazon EC2 Security Groups (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html) that enable the communication ports to send SOAP messages to test the SUTs and check for the results. Furthermore, Amazon EC2 automatically manages redundancy of deployment unit instances to deal with failures. The cloud resource allocation strategy can be fine-grained with the auto scaling and elastic load balancing facilities that will allow the MIDAS platform to scale up and down depending on the tenancy/lab resource usage demands.

The MIDAS platform requirements on the storage capabilities (see Figure 5), are three-fold. MIDAS services need *temporary disk space* for implementation-dependent service execution; each single running service may temporally write/read data whose persistence lasts from the service invocation time to the reply time. MIDAS needs a *short-term persistent disk* space to implement custom data sharing among services during single test method generation and run activities. In fact, end user services, like *test run*, are composite services orchestrating more atomic services, so although all MIDAS services are stateless, component services in an orchestration may communicate exchanging data files through a shared memory disk space. In that case, the persistence of these data must last from the invocation time to the reply time of the composite service. Finally, MIDAS must supply a *persistent disk space* for user data including models, test data, test logs, journals and documentation.

The store volumes associated to an Amazon EC2 instance, i.e., EBS volumes or ephemeral disks, provide a satisfactory solution for the temporary and short-term persistent disk spaces, since they are available until the instance is destroyed. Instead, for the user data persistent space shared among the different tenancies/labs, a suitable solution is Amazon S3 since its data model can be used to logically partition data among different tenancy/lab users in a sandboxed way. Furthermore, Amazon S3 is used to implement and manage the users persistent data storage scaling and replication independently from the lab/tenancy services temporary data storage. Finally, the Amazon RDS database solution is adopted for the implementation of the end user, core and tenancy admin services requiring to store structured information with frequent accesses.

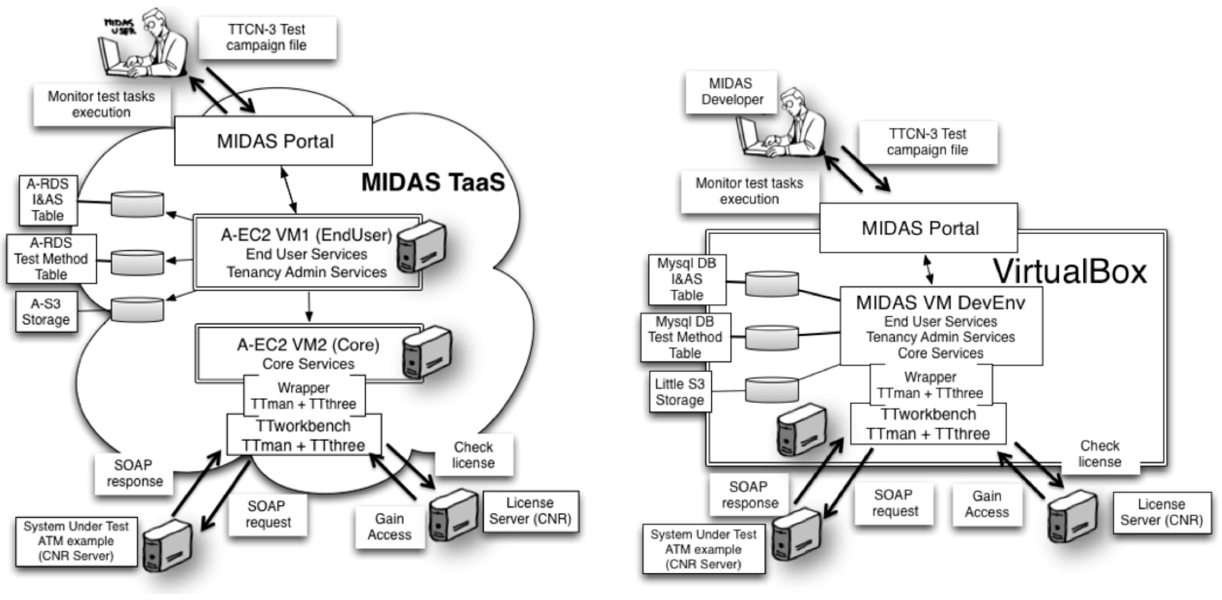
The adoption of Amazon S3 and Amazon RDS allows to rely on Amazon AWS facilities that make it easy to set up, operate, and scale both the relational database and the persistent storages in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks and storage backups/replications.

The first prototype of the MIDAS TaaS on the cloud has been implemented, and its components are mapped to Amazon AWS services as shown in Figure 6(a). The system under test represents the target of the testing activity and is outside the cloud, as well as the license server that is the license manager for the TTworkbench software suite.

4.2 The MIDAS development environment

To allow the developer partners to develop and integrate their components and test methods in the MIDAS TPaaS, a virtualisation approach was adopted already in the development phase of the MIDAS platform. The separation between resource provision and operating systems introduced by virtualisation technologies is a key enabler for cloud computing, specifically for IaaS clouds.

Figure 6 The MIDAS, (a) prototype on the cloud and (b) development environment


MIDAS User
TTCN-3 Test campaign file
Monitor test tasks execution
MIDAS Portal
MIDAS TaaS
A-RDS I&AS Table
A-RDS Test Method Table
A-S3 Storage
A-EC2 VM1 (EndUser) End User Services Tenancy Admin Services
A-EC2 VM2 (Core) Core Services
Wrapper TTman + TTthree
TTworkbench TTman + TTthree
SOAP response
SOAP request
Check license
Gain Access
License Server (CNR)
System Under Test ATM example (CNR Server)
MIDAS Developer
VirtualBox
Mysql DB I&AS Table
Mysql DB Test Method Table
Little S3 Storage
MIDAS VM DevEnv End User Services Tenancy Admin Services Core Services

(a) (b)

The MIDAS test developers have been provided with a seamless, loosely coupled development and integration platform, referred to as the MIDAS Development Environment supporting them in their implementation, debugging and testing activities. The MIDAS Development Environment, shown in Figure 6(b), relies on open-source and stable virtualisation technologies, and it is used to develop all the MIDAS components in a consolidated and conceptually shared VMI, and to configure and manage all the software packages required for the development of the MIDAS platform components, including third party software dependency.

The environment is based on Vagrant, Ansible, and the Oracle VM VirtualBox (http://www.virtualbox.org), that is the default virtual machine monitor (VMM) for Vagrant whose task is to create and run VMIs. The Vagrant and Ansible tools are used to set up the shared VMI, while virtual box is the virtual machine hypervisor. This VMI includes standard hardware architecture, operating system, developer tools, application containers, web servers, and libraries shared among all partners. It is deployed on a local machine by each developer partner, so allowing to locally provide an emulation of the basic MIDAS platform on the cloud. In fact, the MIDAS development Environment includes all the main building blocks of the MIDAS platform deployed on the cloud, as shown in Figure 6(b).

The S3 and DB storage facilities are emulated in the development environment respectively by installing the Little S3 server (an emulation of the Amazon S3 server service) and the MySQL engine server. Developers can carry out all their activities in an independent way, and they interact only when integrating different components in the common environment.

The adoption of a shared VMI to develop the components of the MIDAS platform allows:

- to avoid using cloud resources in the MIDAS development phase, so allowing for a cost-effective strategy to develop and deploy the MIDAS platform on the cloud without wasting cloud resources
- to guarantee the interoperability of the independently developed components since they are released only once they are stable and run on the same shared environment aligned among all partners.

The virtualised MIDAS development environment is deployed on the Amazon cloud infrastructure, at each increment.

5 Monitoring MIDAS usage

To MIDAS service that monitors the usage of MIDAS resources and services is the accounting and billing service (A&B). It is a tenancy admin service acting as a reporting front end for the tenancy administrator of the usage of cloud resources. It will work in tandem with the MIDAS-specific logging and tracing solutions to provide advanced information on the usage of the MIDAS resources and to exploit such information to design more advanced accounting information for the MIDAS tenancies.

The MIDAS accounting and billing is built on top of the Amazon account billing service, and it will be customised according to the MIDAS business model whose definition is still ongoing. The strategy adopted to implement it, is to associate an Amazon identity and access management (IAM) account to each tenancy admin created by the TaaS administrator to monitor the usage of Amazon EC2 resources (computing resources, elastic load balancer, autoscaling), Amazon RDS resources, Amazon S3 storage resources, I/O requests, and so on for each tenancy.

By enabling the consolidating billing of Amazon AWS, a monthly report of cloud resources consumption for each tenancy will be stored on Amazon S3 and processed to report all information about cloud resources consumption. In addition, we also keep track of resources consumption by each tenancy user, by keeping the history of user activities and the amount of cpu time spent in each testing task. All this information will be used to charge, according to a MIDAS business model, the tenancy users for cloud resources usage and MIDAS services usage as well.

5.1 The MIDAS Monitoring implementation

The current implementation of the A&B service is straightforward. Since there is a single A&B service per tenancy, and given that, according to the directives reported in MIDAS Deliverable D2.2, a tenancy is the atomic cost centre for the MIDAS platform (i.e., the MIDAS TaaS administrator will charge tenancies for cloud costs and profits, no single tenancy users), the A&B service has been preliminarily implemented as a facade of the Amazon A&B service. Hence, the current implementation of the A&B service reports to tenancy administrators the current monthly consolidated usage of CPU resources, disk space and data transfers of the users of the tenancy. To further improve the A&B service functionalities, we addressed the problem of monitoring resources usage in a generic software-as-a-service cloud. In general, the monitoring of events in a cloud can be performed on three levels:

1 Infrastructure level: events like changes in CPU load, memory consumption, disk occupancy, network transfer are monitored at physical resource level (i.e., OS kernel mechanisms) or virtual resource level (i.e., hypervisor mechanisms). These events are then exploited to derive resource usages during a given period of time (e.g., hour, day or month). Examples of these monitoring solutions include custom software deployed by cloud users such as Nagios, the industry standard for IT infrastructure monitoring, or proprietary software deployed by cloud providers such as CloudWatch, the Amazon monitoring service.

2 Platform level: the information we can obtain at this level for a SOA software depends on the deployment of the SOA, its runtime environment and its communications. Since MIDAS is implemented using web services WSDL/SOAP technologies in Java and the reference web application container for MIDAS services is Apache Tomcat, the events we are able to monitor are:

- all SOAP messages exchanged between web services, together with their platform-dependent and application-dependent information
- tomcat logs on all activities performed by the web application containers.

The services collecting and aggregating these events must be designed and implemented by the MIDAS platform developers, since the semantics of the messages and, partially, of the container logs are a domain knowledge described in detail in D2.2 and D6.2. Moreover, the storage, the processing, the analysis and the visualisation of these events should be implemented in a customised component(s) specifically designed for the MIDAS platform.

3 Application level: the information collected at application level strongly depends on the specific testing domain. In theory, any test method, along with its implementation, can be instrumented to collect a feature rich plethora of information on its behaviour and usage. In practice, this requires a deep knowledge and control of the functional implementation (i.e., source code) of single testing components, as well as specifically tailored monitoring and tracing mechanisms for each testing component.

The chosen monitoring solution adopted for the MIDAS platform is at platform level. Monitoring at infrastructure level provides detailed information on the resource usages that are difficult to map on the MIDAS architecture, composed of asynchronously communicating web services. Moreover, even if it could be possible to design ad-hoc solutions using a specific monitoring product like Nagios, its configuration, deployment and management will have a negative impact on the manageability of the whole MIDAS infrastructure. On the other side, monitoring at application level is not affordable, since we lack any access to the source code of the components deployed in the MIDAS architecture.

Even at platform level there are some caveats on the information we can collect and use. Logs are, in general, a vast amount of unstructured data. This information must be collected, stored and processed in order to extract useful events continuously, and this sequence of step must be continuously repeated. This can require a dedicated cloud platform just to process the data, and an additional engineering effort to design algorithms and solutions to implement log processors. Therefore, the reference log mechanism for the MIDAS platform is based on interception of all SOAP messages exchanged among MIDAS web services. This information is structured and with a clear semantic domain.

The basic element of the implementation of the MIDAS monitoring is an application server interceptor, a small software automatically deployed with the MIDAS platform on the cloud in any web application container, which is responsible to automatically process all incoming and outcoming SOAP messages, extract relevant timestamped information such as user id, test method id, target service that can be used by the A&B service to perform more elaborated operations. It is worth noting that MIDAS developers, as well as future test method developers, do not need to be explicitly aware of any monitoring configuration mechanism, as everything is automatically managed by the provided interceptor, even its own deployment.

However, since MIDAS is a testing-as-a-service platform implemented according to the SOA approach, its customers should not be aware of the underlying implementation of the services components of the platform, as well as any cloud-related details.

In this view, we aim for a billing model that is simple and transparent to its users, hiding from them all the implementation details such as the cloud provider hardware/software stack or the MIDAS internal architecture. The costs can then only be billed on metrics that are directly observable by the user, and should be charged based on the interactions that the user has with the platform.

Since all such interactions can only be performed through the end user services, we plan on instrumenting such services to log into a database (the usage DB) all the API interactions from the user together with metadata that can be included in the billing model; for example, for the file management service, these metadata could be the size of the files that are stored or retrieved.

The usage DB interaction with the end user services can be seen in Figure 7. Note that the Usage DB does not trace any interaction between the internal MIDAS services, since these are implementation details, thus invisible to the user; furthermore, many of these services are not owned by the MIDAS platform developer, so it would be necessary to agree on a common logging schema and have all the partners implement the logging instrumentation in their services.

It is however useful to log the interactions between the internal services during the execution of a test, both for developing/debugging/improving the services, and also to enable analytics that could identify bottlenecks or additional metrics that deserve to be added to the usage DB. To do this in a service-agnostic way, we plan to implement a SOAP interceptor that logs into a trace DB metadata about all the SOAP messages received by each service, specifically information that is present in all the messages of the MIDAS platform, such as the user that originated the request and the test tracking ID.

For the purposes of accounting, the information in the usage DB is augmented with the information in the test status DB, that tracks the status of a test task, in particular storing the invocation time, the user that invoked the test, and the completion time; this allows the cost model to include a usage metric for each user that accounts the overall time spent by the platform in executing tasks invoked by that user. This way, both load-independent infrastructural costs (base services, DBs, licenses...) and load-dependent costs (network traffic, test executor instances) can be factored in a per-minute billing rate, once all these costs have been estimated.

An example can be seen in Figure 8: tests invoked by user1 take overall eight minutes, by user2 ten minutes, and by user3 22 minutes, summing up to 38 minutes. The infrastructural cost, e.g., 10$, can then be accounted to the users in fractional parts of respectively 8/38, 10/38, and 22/38, plus the costs of the other operations as accounted by the usage DB. In this way, user1 will be charged for 2.1$, user2 for 2.6$ and user3 for 5.3$.

In Figures 9 and 10 we show an example of respectively accounting and billing information for users of a MIDAS tenancy.

Figure 7 The MIDAS monitoring interceptors (see online version for colours)

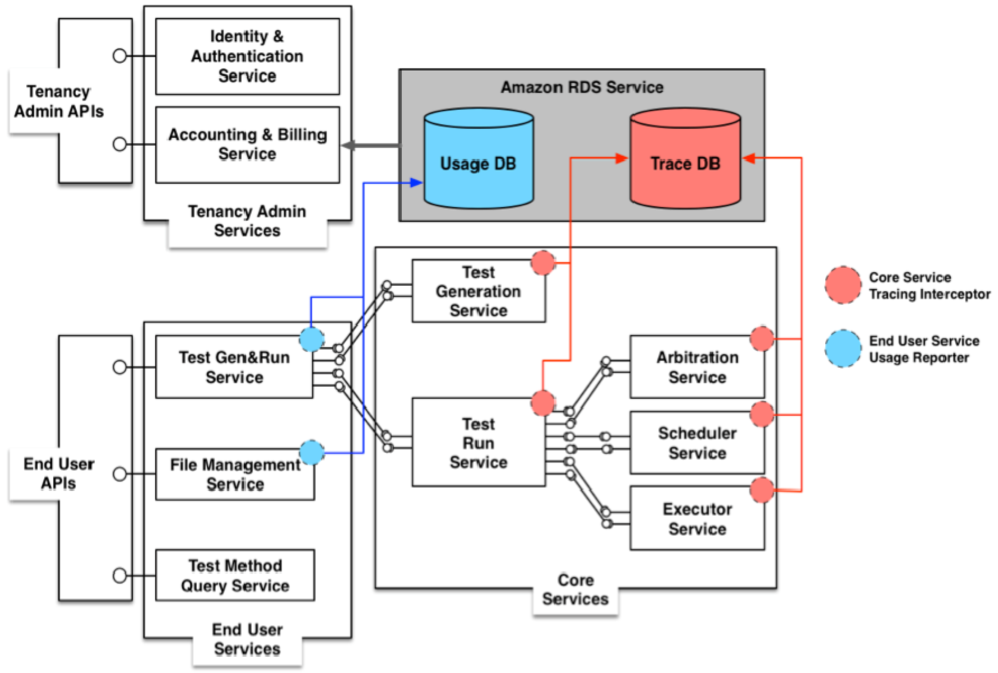
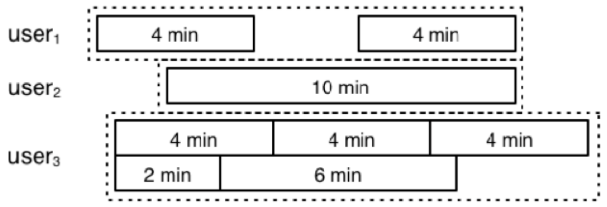


Figure 8 Time usage of tenancy users



6 MIDAS platform elasticity

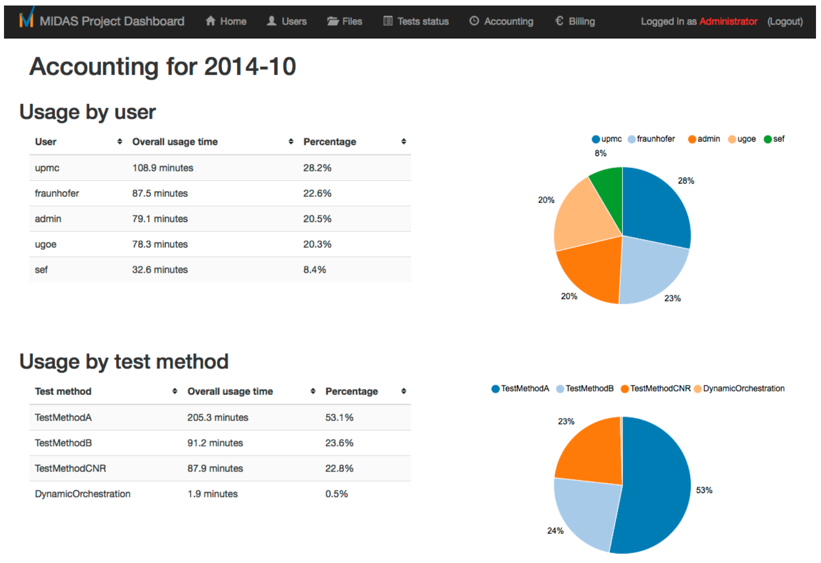
The SOA approach adopted for MIDAS platform makes it naturally suitable for automatic elasticity. The statelessness and loosely coupling of the MIDAS services represent the key enabling factors to provide automatic reconfiguration of MIDAS components depending on their usage. To fully implement automatic elasticity mechanisms in a cloud-based SOA application, such as MIDAS, the underlying cloud infrastructure must provide mechanisms both to monitor the resources usage of the applications, and to automatically scale up and down the virtual machines hosting running instances of the same service container of a tenancy/lab. Furthermore, the virtual machines allocated to the same tenancy/lab should be managed by a load balancer able to steer and schedule connection requests to the instances according to some predefined algorithm.

Amazon AWS provides components and services to address all the aforementioned elasticity requirements that will be used to implement the MIDAS auto scaling and elastic load balancing facilities to optimise the use of cloud resources for the MIDAS platform.

The Amazon AWS cloud building blocks for the MIDAS scaling and elasticity components are Amazon auto scaling, elastic load balancing and cloud watch. As already described, for each tenancy the VM1 is the Amazon EC2 instance where the MIDAS portal, the end user services and the tenancy admin services are deployed; while the VM2 is the Amazon EC2 instance hosting the core services. The core services are the services that contain the engines for the TTCN3 scripts execution, the inference logic, the usage profile scheduling, and so on. As we said, we expect that the CPU workload and/or network activity and/or disk utilisation of the VM2, depending on the users incoming requests of the core services, can increase, so requiring additional facilities to scale resources up/down according to some policies contracted by the tenancy administrator with the MIDAS TaaS administrator, and included in the SLA contracted between them.

To deal with scalability and elasticity requirements, the cloud infrastructure will be modified as shown in Figure 11. The auto scaling of the Amazon AWS enables us to satisfy the varying computational demands of MIDAS tenancies for the core components of the MIDAS platform, so reducing the need to provision Amazon EC2 capacity in advance. For example, we can set a condition to add new Amazon EC2 instances in increments up to five instances to the auto scaling group (see Figure 11) when the average CPU utilisation of the Amazon EC2 VM2 running instance goes above the 80% for five minutes; and similarly, we can set a condition to remove Amazon EC2 VM2 instances in the same increments when CPU utilisation falls below 20% for ten minutes. When the auto scaling group is created, we rely on the Amazon elastic load balancing service to distribute fairly the workload of incoming requests to the instances in the auto scaling group. For each tenancy, the CPU workload, network activities, disk utilisation will be monitored, in order to set up alarms to signal events according to the set conditions. These alarms will be used to trigger the auto scaling service so to scale up/down the Amazon EC2 VM2 instances accordingly, so to run the VM hosting the core services at optimal utilisation.

Figure 9 Accounting information of tenancy users (see online version for colours)


MIDAS Project Dashboard
Home
Users
Files
Tests status
Accounting
Billing
Logged in as Administrator (Logout)
● upmc ● fraunhofer ● admin ● ugoe ● sef
8%
28%
20%
20%
23%
● TestMethodA ● TestMethodB ● TestMethodCNR ● DynamicOrchestration
23%
53%
24%

Accounting for 2014-10

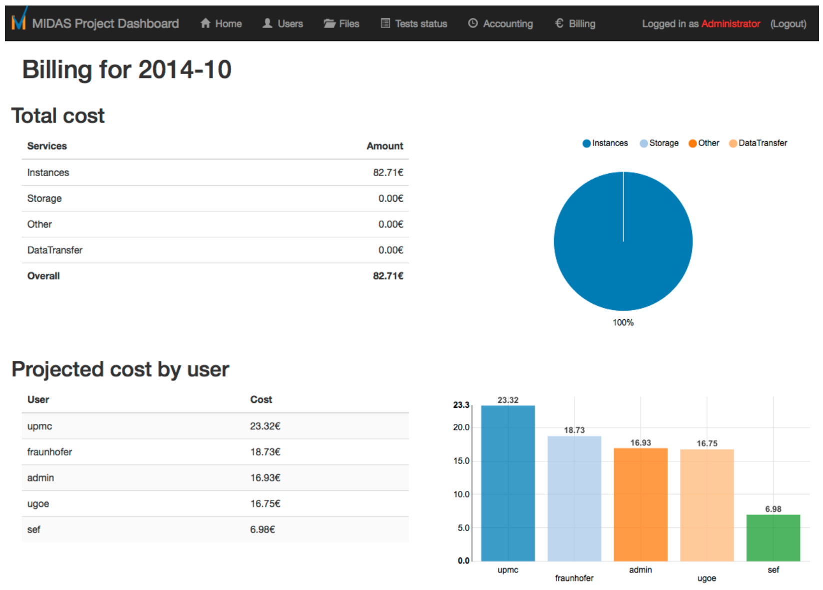
Usage by user

User	Overall usage time	Percentage
upmc	108.9 minutes	28.2%
fraunhofer	87.5 minutes	22.6%
admin	79.1 minutes	20.5%
ugoe	78.3 minutes	20.3%
sef	32.6 minutes	8.4%

Usage by test method

Test method	Overall usage time	Percentage
TestMethodA	205.3 minutes	53.1%
TestMethodB	91.2 minutes	23.6%
TestMethodCNR	87.9 minutes	22.8%
DynamicOrchestration	1.9 minutes	0.5%

Figure 10 Billing information of tenancy users (see online version for colours)



Billing for 2014-10

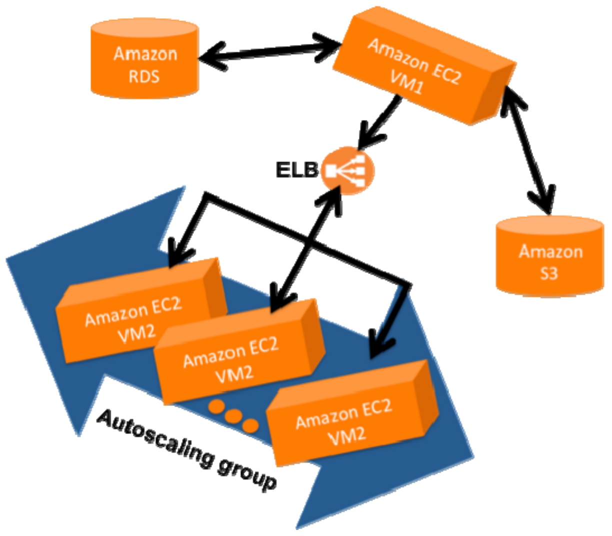
Total cost

Services	Amount
Instances	82.71€
Storage	0.00€
Other	0.00€
DataTransfer	0.00€
Overall	**82.71€**

Projected cost by user

User	Cost
upmc	23.32€
fraunhofer	18.73€
admin	16.93€
ugoe	16.75€
sef	6.98€

Figure 11 Autoscaling for a MIDAS tenancy (see online version for colours)



The elastic load balancing service allows also, for example, to make sure that the number of healthy Amazon EC2 instances behind an elastic load balancer is never fewer than two. We can use auto scaling to set this condition, and when auto scaling detects that this condition is verified, it automatically adds the requested amount of Amazon EC2 VM2 instances to the auto scaling group. Or, if we want to make sure that we add Amazon EC2 VM2 instances when the latency of any of the Amazon EC2 VM2 instances exceeds four seconds over any 15 minute period, we can set that condition, and auto scaling will take the appropriate action on the Amazon EC2 VM2 instances.

7 Commercial solutions related to MIDAS

The advent of cloud computing has changed the way software consumers use software as well as software vendors develop and deliver their products. This revolution is occurring also in the software testing area. While in the past testing tools and software where sold and delivered by licensing to run in-house, nowadays several companies and organisations provide testing facilities on demand, and they are billed per use.

Here a non-exhaustive list of some commercial software platforms providing testing capabilities as SaaS solution in cloud settings.

CloudTest (by SOASTA Inc., http://www.soasta.com/products/cloudtest) is cloud-based testing solution provided with a dynamic web application that combines Ajax-based user interfaces and distributed web services to support test creation, execution and test result analytics. CloudTest framework allows users to build, execute and analyse performance and functional tests and to share them with other CloudTest users. Professional test developers and performance engineers can use CloudTest to build tests that can be used in the CloudTest framework at web scale. CloudTest uses cloud resources quick availability and scalability to afford realistic load testing of web applications without hardware resource constraints. CloudTest uses persistent and scalable cloud storage to archive data and correlates data streams from a distributed load test in a single result on synchronised time-line to be used by the CloudTest analytics components.

Zephyr (by D Software Inc., http://www.getzephyr.com/zephyr) provides a framework for testing lifecycle management. Zephyr framework enables collaboration along the software development cycle: it allows teams to effectively manage test resources, testing projects, releases, requirements, test cases, scheduling, test execution, defects, automation, collaboration, metrics and reporting, and it provides global access, collaboration and management visibility. Zephyr's SaaS-based solution allows costumers to use a cloud-based dedicated test management system that relies on the Amazon 24/7 secure cloud computing platform. In addition to relieve users from the cost of infrastructure, backups and upgrades, Zephyr's SaaS offers a fast global access and sharing platform for testers, as well as uptime and global disaster recovery mechanisms.

TestMaker (by PushToTest^{TM}, http://www.pushtotest.com/cloudtesting) is a distributed test environment, that can run tests either in on-premise or on-demand cloud-based modes, or both. TestMaker is a console and runtime to operate functional tests, load and performance tests, and web services motoring in a distributed networks of TestNodes. TestMaker technology runs on different cloud infrastructures, like Amazon Web Services (EC2), GoGrid, and RackSpace. TestMaker defines its proprietary model, namely TestScenario, of orchestration of a test. TestScenario defines the locations of TestNodes to operate a test. The mapping of TestNodes to cloud resources is flexible, in the sense that you can remap TestNodes (and the associated test runs) on multiple types of cloud test equipments.

A software TaaS solution is provided by Riungu-Kalliosaari et al. (2013) including cloud-enabled test environments and test toolkits. Testing facilities are accessible using a central service portal, where you can select the testing service you require and pay-per-use.

SkyTap company provides SaaS-based environments as reported in SkyTap (http://www.skytap.com/product/intelligent-automation-platform) for developing and testing complex applications. Users can import existing virtualised applications or build new applications in the cloud. Environments can be accessed through any modern web browser, REST-based API, or command line interface (CLI). Skytap cloud uses a browser-based interface for all system management, and hosts a library of pre-configured virtual machine images. Using either these images or their own imported VMIs, users can create sharable configurations of one or more machines, and securely connect to active machines.

The aforementioned list of commercial products has not been here reported for comparison with the MIDAS cloud-based testing solution, but to show the potential of the MIDAS results with respect to what is currently available on the market. The main limit of commercial solutions is the difficulty (in some case the impossibility) to create/add new testing methods, and to freely compose them in new testing scenarios and share them among the user organisations. This is a peculiarity of MIDAS, which not only offers a portfolio of testing solutions, but provides also a framework in which test developers design, write and orchestrate their own test methods in new ways, relying on a common shared Domain Specific Language (DSL) as a *lingua franca* for the integration and communication of the heterogeneous components of the MIDAS framework (developed as services by different partners and with different technologies).

From the cloud perspective, MIDAS shares with the aforementioned commercial solutions the same objectives in the use of cloud resources. Indeed most of the discussed commercial products allows both on-premise and on-demand SaaS modes of testing capabilities provisioning. MIDAS was designed as a SOA composed of testing services plus administrative and storage management services to provide testing capabilities solely in an on-demand SaaS mode.

Like most of the mentioned commercial solutions, the MIDAS platform can be viewed as a provider of virtualised (cloud) resources to be distributed and delivered to test users, whereas cloud hardware infrastructures are acquired by Amazon EC2 and managed by MIDAS administrators to pursue the MIDAS business model in respect to the SLAs agreed with consumers. Like all cloud-based testing solutions MIDAS aims to transparently offer to test users efficient, flexible and large scale hardware infrastructures to support intensive and resource-consuming testing activities. In commercial solutions this is crucial in particular for load and stress testing of web applications and SOAs in realistic scenarios. Nevertheless, in MIDAS also the provisioning of test executors, generators and schedulers are resources to be distributed and delivered in pay-per-use mode. The fine-grain monitoring facilities of MIDAS allow to profile usage of both hardware and software resources and, consequently, to charge users for access to specific testing capabilities/technologies that could be MIDAS built-ins or made available in the MIDAS framework by test developers under payment or other forms of agreements.

8 Conclusions

MIDAS TaaS will provide companies with services to design, deploy and run their test cases without disclosing any information to the cloud provider, and without having to program the whole test procedures from scratch. The costs saving and easy accessibility of cloud's extremely large computing resources will make the MIDAS testing facility usage available to geographically distributed users, executing wide varieties of user test scenarios, with a scalability range previously unattainable in traditional testing environments.

In this paper the MIDAS TaaS platform deployment strategy on a cloud infrastructure is discussed. The adoption of an IaaS cloud infrastructure is mainly due to the flexibility it offers to developers in the control of cloud resources, so allowing to customise scalability strategies, according to the different requirements for the different MIDAS components. In addition, due to the rapid evolution of the cloud market together with the lack of standards, it is crucial to guarantee the portability to different cloud providers of a platform like MIDAS, that is intended to be industrialised in the future. Since an IaaS cloud provides cloud resources through virtualisation technologies, it allows to move to different cloud providers with minimum effort. For the time being, Amazon EC2 as the underlying cloud infrastructure represents a satisfying solution offering the best compromise among cost, development features and elasticity mechanisms.

The provision of a shared development environment based on the same virtualisation technologies adopted for its deployment on the cloud allows to easily integrate components independently developed by the partners and at the same time to address the different software requirements of the different MIDAS components.

The deployment strategy of MIDAS shows how the requirements of an automated SOA testing facility on the cloud could be met through the design of a SOA-based platform where the component services are loosely coupled and stateless, and how their cloud deployment is completely transparent to the developers of the MIDAS components.

The multi-tenancy design allows to tailor scalability and elasticity requirements of cloud resources coming from different users, so providing a real pay-per-use TaaS facility, accommodating different cost business model in a foreseen industrialisation phase of the MIDAS platform. The Accounting and Billing services will be used to support the cost business models adopted for different types of companies.

In conclusion, the MIDAS platform can be viewed as a provider of virtualised cloud resources to be distributed and delivered to test users, whereas cloud hardware infrastructures are acquired by Amazon EC2 and managed by MIDAS administrators to pursue the MIDAS business model in compliance with the SLAs agreed with consumers. More importantly, MIDAS also provides test executors, generators and schedulers as services to be delivered in pay-per-use mode. The fine-grain monitoring facilities of MIDAS allow to profile usage of both hardware resources and software services and, consequently, to charge users for accessing specific testing capabilities/technologies that could be MIDAS built-ins or made available in the MIDAS framework by test developers under payment or other forms of agreements. As future work, the platform will be tested on two real world case pilots that will provide on one hand an assessment of the developed test methods for SOA architectures, and on the other hand to collect information on the real cloud resources demands coming from different kinds of SOA-based applications.

Acknowledgements

This work is part of the MIDAS Project, funded by the EU FP7-ICT-2012-8, Grant Agreement No. 318786.

References

Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004) *Web Services – Concepts, Architectures and Applications*, Springer-Verlag, Berlin Heidelberg.

Amazon EC2 Security Groups [online] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html.

Amazon Elastic Computing (EC2) Platform, *AWS – Amazon Elastic Compute Cloud (EC2) – Scalable Cloud Hosting* [online] http://aws.amazon.com/ec2/ (accessed October 2014).

Ansible Documentation, http://docs.ansible.com.

Canfora, G. and Di Penta, M. (2009) 'Service-oriented architectures testing: a survey', *Software Engineering*, *LNCS*, Vol. 5413, pp.78–105.

D Software Inc., *Portfolio of Test Management Products from Zephyr*, http://www.getzephyr.com/zephyr.

Dustdar, S. and Haslinger, S. (2004) 'Testing of service-oriented architectures – a practical approach', *Object-Oriented and Internet-Based Technologies*, *LNCS*, Vol. 3263, pp.97–109.

European FP7 Project MIDAS, *MIDAS EU Project: Model and Inference Driven – Automated Testing of Service Architectures* [online] http://midas-project.eu (accessed October 2014).

Grabowski, J., Hogrefe, D., Rethy, G., Schieferdecker, I., Wiles, A. and Willcock, C. (2003) 'An introduction to the testing and test control notation (ttcn-3)', *Int. J. of Computer and Telecommunications Networking*, Vol. 42, No. 3, pp.375–403.

Hashimoto, M. (2013) *Vagrant: Up and Running*, O'Reilly.

Kalamegam, P. and Godandapani, Z. (2012) 'A survey on testing SOA built using web services', *Int. Journal of Software Engineering and its Applications*, Vol. 6, No. 4, pp.91–104.

MIDAS Consortium (xxxx) *Architecture and Specifications of the MIDAS Framework and Platform*, MIDAS Deliverable D2.2.

MIDAS Consortium (2014) *Specification and Design of the Basic MIDAS Platform as a Service on the Cloud*, MIDAS Deliverable D6.2.

OMG, *UML Testing Profile (UTP)* [online] http://utp.omg.org.

Oracle VM VirtualBox, *User Manual Version 4.3.18* [online] http://download.virtualbox.org/virtualbox/UserManual.pdf (accessed October 2014).

Papazoglou, M.P. and van den Heuvel, W-J. (2007) 'Service oriented architectures: approaches, technologies and research issues', *The VLDB Journal*, Vol. 16, No. 3, pp.389–415.

PushToTest^{TM}, *PushToTest: Cloud Testing* [online] http://www.pushtotest.com/cloudtesting.

Riungu-Kalliosaari, L., Taipale, O. and Smolander, K. (2013) *Software Testing as a Service: Perceptions from Practise*, pp.196–215, IGI Global, Hershey.

SOASTA Inc., *Cloud Testing with CloudTest* [online] http://www.soasta.com/products/cloudtest.

SkyTap, *Dev/Test Environments for Enterprise* [online] http://www.skytap.com/product/intelligent-automation-platform.

Testing Technology, TTworkbench (UTP) [online] http://www.testingtech.com/products/ttworkbench.php.

Wendland, M.F., De Francesco, A., Di Napoli, C. and De Rosa, F. (2013) 'MIDAS: automated SOA testing on the cloud', *ERCIM News*, Vol. 95, p.46.