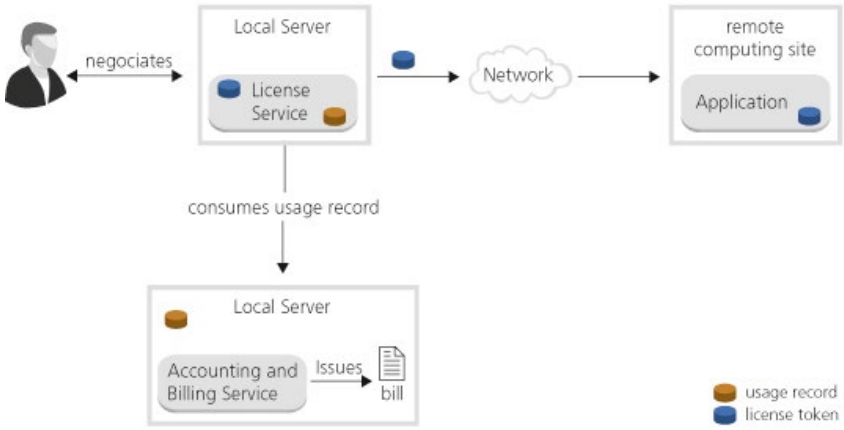
Grid-friendly software licensing for location independent application execution

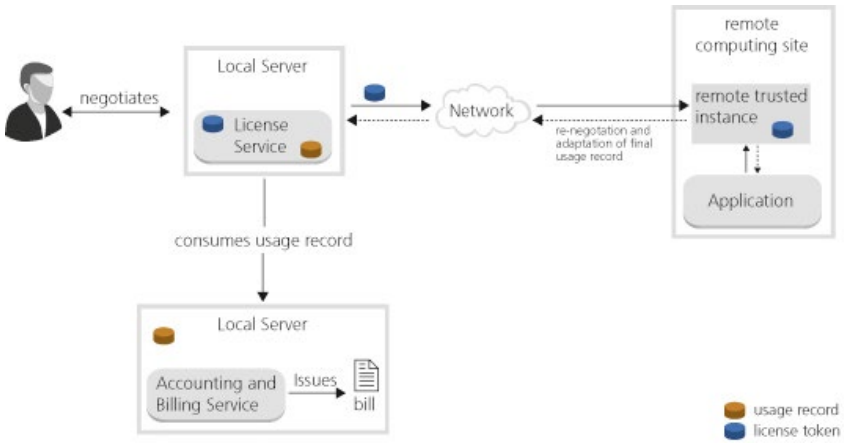
About SmartLM

Current software licensing practices are limiting the acceleration of Grid adoption. Indeed, the rapid emergence of service and virtualization environments requires a rapid evolution in licensing models. *SmartLM* will provide a generic and flexible licensing virtualization technology for new service-oriented business models across organization boundaries.

Basic Scenario
No bi-directional network link available at run-time



Advanced Senario
Bi-directional network link available at run-time



For further information:

Software & Service Architectures and Infrastructures
European Commission - Information Society and Media DG
Office: BU25 3/134 B-1049 Brussels

At a Glance

Project
SmartLM - Grid-friendly software licensing for location independent application execution

Contract number
216759

Project coordinator
Mr. Josep Martrat
josep.martrat(at)atosorigin.com
Atos Research & Innovation.

Partners
Atos Origin - Spain
Fraunhofer SCAI - Germany
FZ-Jülich - Germany
CINECA - Italy
The 451 Group - UK
INTES - Germany
ANSYS Germany - Germany
LMS International - Belgium
T-Systems - Germany
CESGA - Spain
Gridcore Aktiebolag - Sweden

Duration
30 months (starting on February 2008)

Total cost
4.012.070 EUR

Programme
Information and Communication Technologies

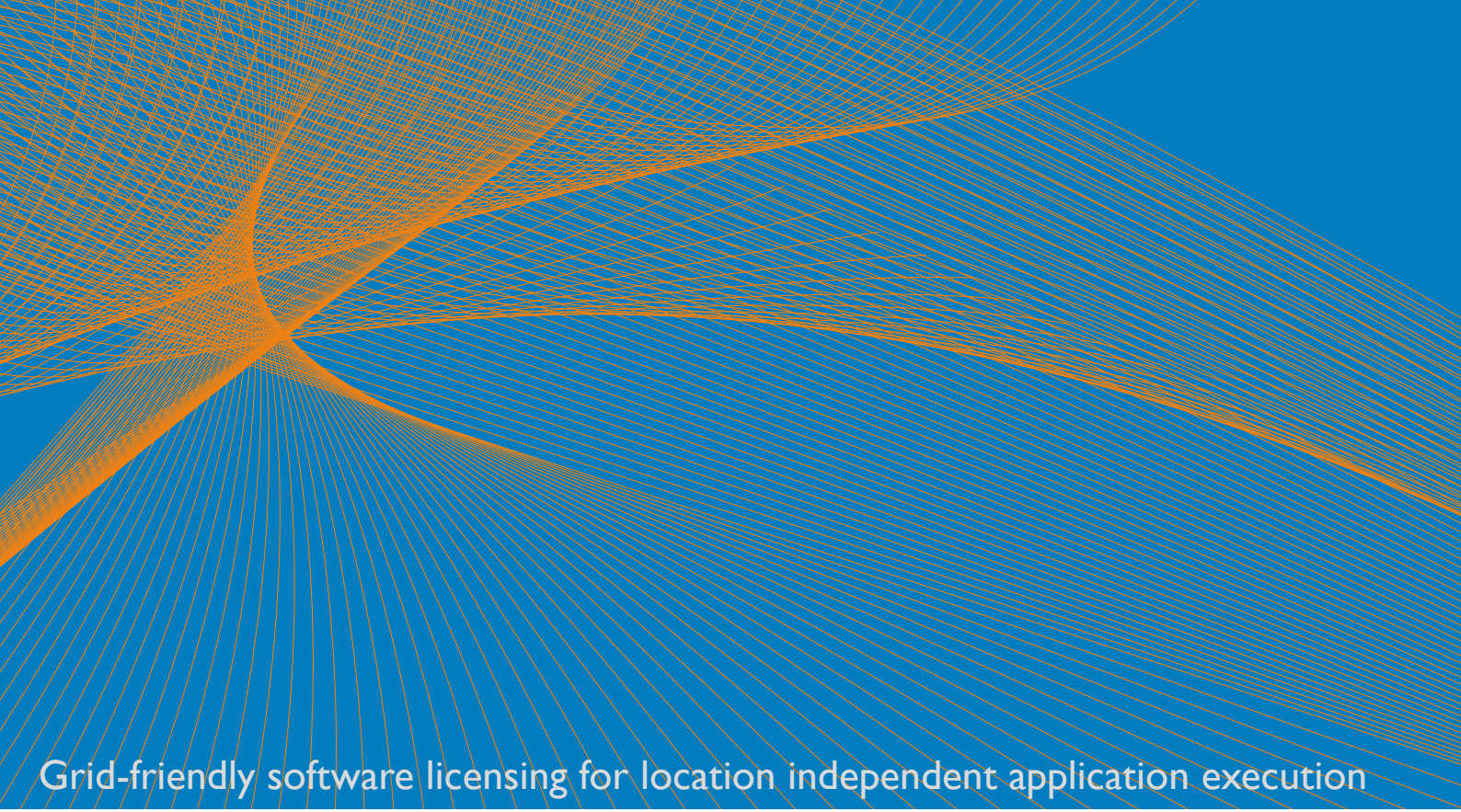
Flyer SmartLM (PDF 2.5MB)

Further Information

Tel: +32 2 298 93 02
Fax: +32 2 296 70 18
infso-st(at)ec.europa.eu
http://cordis.europa.eu/fp7/ict/ssai/

SmartLM receives research funding of the European Commission under the 7th Framework Programme.

SmartLM is part of the European Commission Programme »Information and Communication Technologies«.


Grid-friendly software licensing for location independent application execution

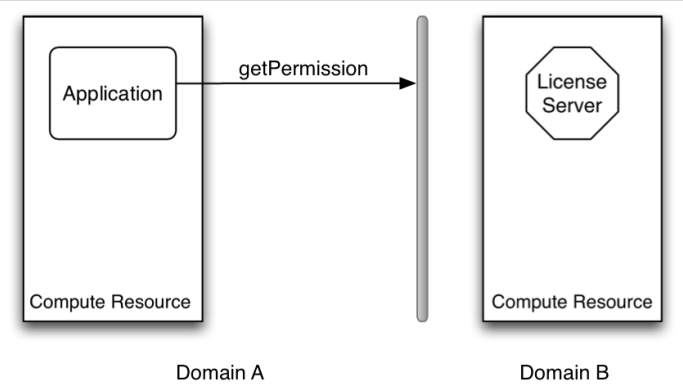
SmartLM

The Problem

IT Infrastructure paradigms have been changing over the last years to support more flexibility and reduce costs at the same time. In parallel, computer-based simulations became more important and tend to become more complex and demanding with respect to the computational requirements. Several approaches to address these issues evolved, driven by academia and industry. Grid computing aims at providing infrastructure for sharing or pooling resources in a collaborative manner. Clouds, which appeared more recently on the scene, focus on resource provisioning, e.g. for peak demand or when customer owned infrastructure is overloaded or its use is not appropriate for any reason.

However, extending a company's business or a research institution's information processing beyond the borders of the respective administrative domain raises a number of issues, one of them is the use of license-protected software. Software protection and licensing are important topics for both the independent software vendors and software users. In Grid and Cloud environments, the use of license-protected applications is almost impossible and becomes a challenging task. The reasons are twofold: (i) there are - with a few exceptions for the Amazon EC2 environment that have been introduced recently - no business models of the independent software vendors for Grids or Clouds and (ii) there is no licensing technology suitable for Grid environments.

Figure 1: Firewall blocking the communication of an application with the license server

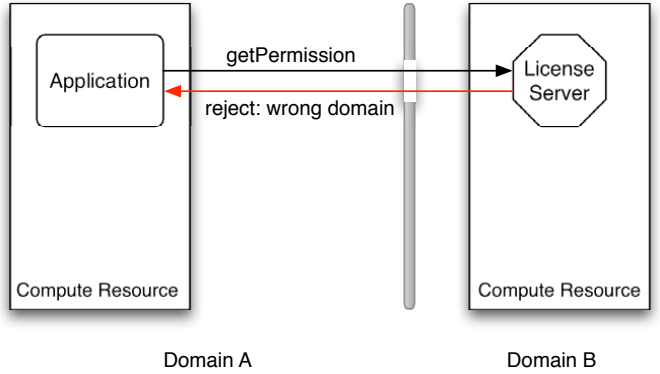


The figures above depict the usual situation: In figure 1 the application is executed in another domain than the license server. Due to the firewall the communication between the application and license server is blocked and the application will be aborted due to the missing authorisation. Figure 2 shows the same situation after the firewall has been opened for the communication between application and license server. However, the license server detects that the request is not coming from an application running in the local domain and rejects the use of the license and the application is aborted again.

Moreover, the current business models of the ISVs most often result in contracts restricting the use of licenses for application execution on resources in a computing centre of the company or institution that bought the license, thus rendering the use of paid licenses for application execution on remote Grid or Cloud resources a breach of contract.

For that reason the 451 group concluded in an extensive survey on licensing issues in Grids that current software licensing practices are limiting the acceleration of Grid adoption already in 2005. Indeed, the rapid emergence of service and virtualization environments requires a rapid evolution in licensing models.

Figure 2: Firewall allowing an application to communicate with the license server



The SmartLM Project

SmartLM aims at rendering mechanisms for managing and using software licenses in a more fair and flexible way. SmartLM licenses may be used seamlessly in local cluster environments, as well as in local or remote Grid and Cloud environments, and under circumstances that the SOA concept presents.

The development of Grid-aware software licensing integrated into service-oriented architectures (SOA) will significantly bolster Grid deployment generally – but specially into new areas exploiting a broad range of commercial software, beyond boundaries of technical and high performance computing.

The ability to effectively and dynamically manage the use of software licenses based on business objectives is not only an issue in Grids, but Grids are an important inflection point in this transformation.
SmartLM is clearly aiming to aid the creation of new industrial opportunities based on the creation of the emerging Service market that uses the service-oriented infrastructure as a means to deliver new software services in great many fields like mechanical industries, Finance, Entertainment, Retail, Pharmaceutics, etc.

SmartLM is contributing to the technology convergence (virtualization, Grid, SOA, etc.) and interoperability with focused contributions to Web Services standards and specifications of the Open Grid Forum (OGF).

The open nature of the new license management software is enabling distributed applications by removing one of the major obstacles for its deployment. Well-known applications can be adapted to this new networked environment, helping organizations to get dynamic access to the right to use applications. This supports not only large enterprises but also Small and Medium Enterprises (SMEs) who do not have the negotiating power to obtain non-standard pricing or licensing from ISVs.

The challenge is to compensate the potential revenue loss on one side with greater business values on the other side. Customer needs and vendor requirements must be balanced.

Software licensing as a No. 1 barrier in commercial grid deployments

Software licensing

Traditionally, software licenses have been provided on the basis of a named user, node-locked host, number of concurrent jobs or possibly a floating site license. These models are not sufficiently flexible to support commercial applications that access resources beyond the current administrative domain – possibly as a Utility-like service outside the organization or Software as a Service (SaaS) model.

Software licensing is identified as a particular concern for enterprise IT managers as they start to deploy virtual Grids in any meaningful way. For all the potential benefits of Grids, IT departments cannot afford to buy software licenses for every device in the service-oriented infrastructure that by nature consumes resources dynamically.

In addition, the Grid-based Service-Oriented Architecture (SOA) solutions together with other technology trends such as multi-core and virtualization environments are forcing Independent Software Vendors (ISV) to move away from traditional software licensing models.

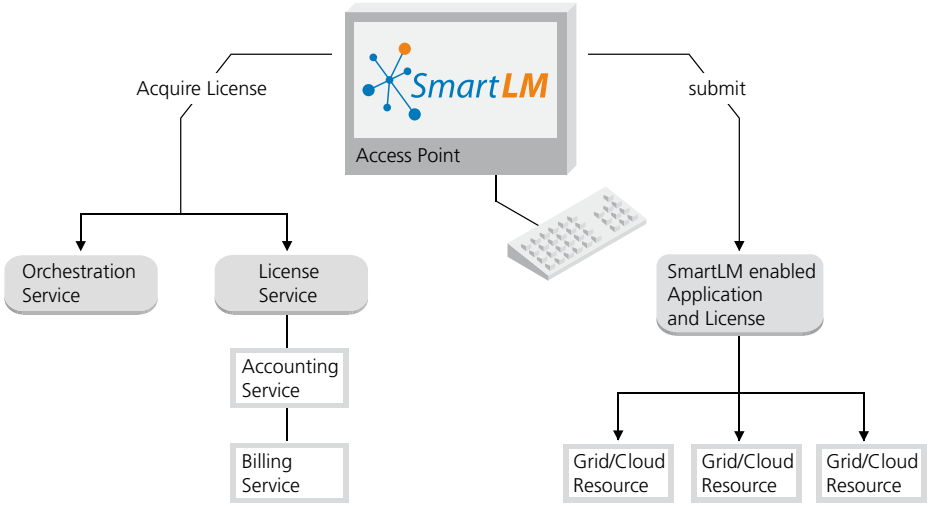
The SmartLM goals

The objective of SmartLM is to provide a new generic licensing virtualization framework based on standards, integrated in major Grid middleware solutions, ready for Cloud environments and Service Oriented Architectures.

The strategic objectives of SmartLM are:

- To understand the licensing requirements for Grid use and deployment in the commercial environment, involving software vendors, application service providers, IT integrators, resource providers as well as end users.
- To identify service-oriented business models for distributed scenarios across organizations.
- To design and build a secure, platform-independent license management framework.
- To provide specific models and technologies for accounting and billing of licenses.
- To enable and validate the license management tools with commercial applications deployed in Grids.

SmartLM´s distributed architecture



SmartLM – Grid-friendly software licensing for location independent application execution, contract number 216759. Project coordinator: Josep Martrat (josep.martrat@atosorigin.com) Atos Research & Innovation. Scientific coordinator: Wolfgang Ziegler (wolfgang.ziegler@scai.fraunhofer.de), Fraunhofer Institute SCAI. Partners: Atos Origin – Spain, Fraunhofer SCAI – Germany, Forschungszentrum Jülich – Germany, CINECA – Italy, The 451 Group – UK, INTES – Germany, ANSYS Germany – Germany, LMS International – Belgium, T-Systems – Germany, CESGA – Spain, Gridcore Aktiebolag -Sweden Duration: 30 months (starting in February 2008) Total cost: 4.012.070 EUR Programme: Information and Communication Technologies.

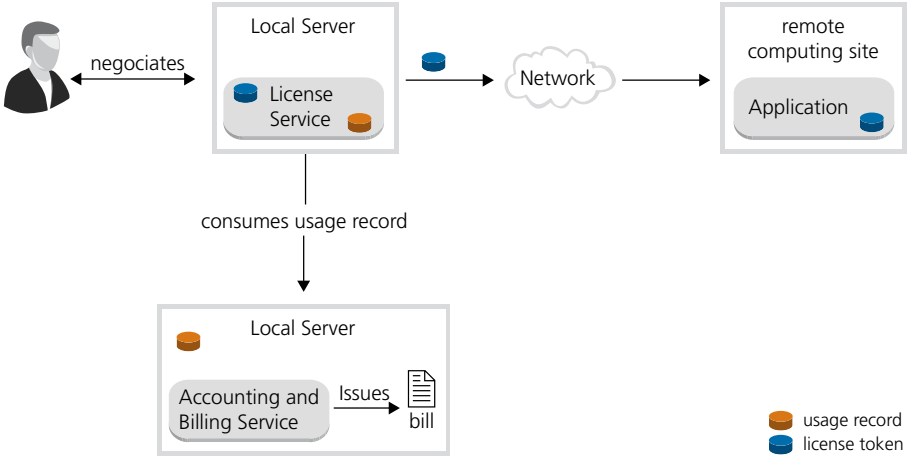
elasticLM – The Product

The overall approach consists in treating and implementing software licenses as Grid services, thus providing platform independent access just like any other virtualized resources.

- Licenses will become Grid services; a promising approach to overcome the limitations of current monolithic licensing models.
- Licenses will be managed as agreements, extending the conventional Service Level Agreements (SLAs) which are made today between sellers and buyers in the market.
- Licenses will be dynamic in order to support agreements that may change over time and where the dynamic negotiation between service provider and consumer is needed.

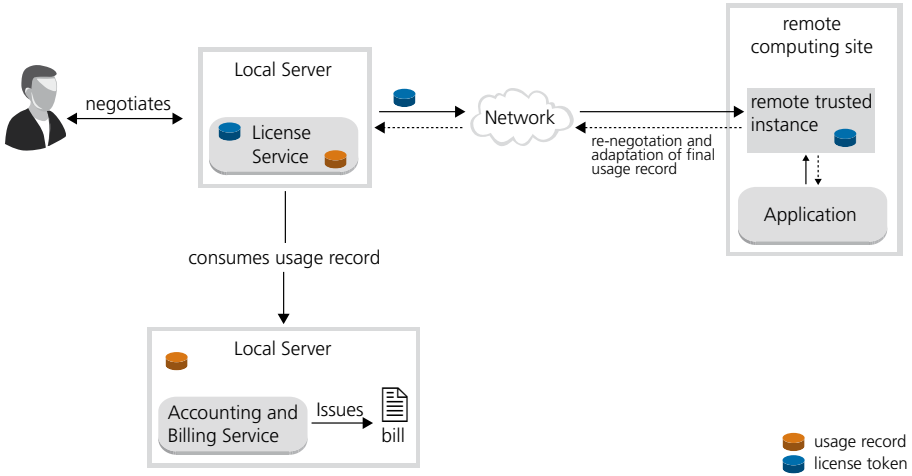
Basic Scenario

No bi-directional network link available at run-time



Advanced Senario

Bi-directional network link available at run-time



elasticLM is implemented as a framework of Web Services. Together with the built-in mechanisms to evaluate different policies prior to taking a decision on the user's license request elasticLM achieves a maximum flexibility. Thus, adapting the functionality and the behaviour of elasticLM to the needs of the respective environments is easy.

elasticLM is based on open standards to ease the integration into existing environments and leveraging interoperability. Moreover, this approach renders the elasticLM system extensible with components adding site-specific functionality, e.g. billing.

In contrast to most of the other existing license management systems, elasticLM comes with an integrated, modular solution for accounting & billing, supporting comprehensive analysis of license usage. Moreover, the integrated approach allows to determine the costs for license usage based on the

users' requests and other parameters like previous usage or department-specific pricing.
Finally, this integration allows checking users' license requests against predefined budgets per user, department etc. License usage is granted only if the request does not lead to a violation of the budget constraints.

Security

Aspects of security have been examined with special care. We identified relevant issues related to both the different actors and the license mechanism itself. These issues relate to

- authentication and authorization of users, services, and servers,
- security and confidentiality of the communication between different actors or components,
- security of the delegation process, when using a portal or an orchestrator, for example,
- disclosure of sensitive information, e.g. compromise of licenses
- integrity of the process to inhibit non-repudiation
- security of the licensing mechanism itself, e.g. the license generators, manipulation of the executables, or clock tweaking

Sophisticated measures have been taken either employing standards like X.509 certificates or XACML and other state-of-the-art technologies for code protection.

Main innovations of elasticLM

elasticLM licenses are mobile objects that may move as applications to different execution environments. Use of protected applications is granted through Service Level Agreements resulting from negotiation of license terms prior to application execution.
Using elasticLM allows advance reservation of licenses. Thus, licenses are available when needed but not blocked when the application is waiting for execution.

All authorization for the use of a license is done locally at the home organisation of a user, taking into account policies of the ISV, site-specific policies defined locally or user-specific attributes as e.g. retrieved from a Virtual Organisation. Signed and encrypted terms of a license are scheduled to the (remote) execution environment.

Integration of an Accounting and Billing System allows price determination and budget control when the license is requested.

Overcoming current limitations by switching to elasticLM

In the following table the major features of the elasticLM product are shown compared to the limitations in current license management:

Current Situation	Innovation of elastictLM
Software licenses allow little flexibility in terms of location independent use. Thus, license protected applications may hardly be used in Grid or Cloud environments.	With elasticLM, licenses may be used to run applications in Grid and Cloud environments no matter whether during the application run there is network connectivity to access the site that hosts the license server that issued the license.
Licenses are often spread across departments making it difficult to track license usage.	elasticLM provides access to and management of all licenses owned by a site.
All license usage policies are embedded in the license of the ISV.	elasticLM allows the definition of local policies for license usage addressing the site-specific needs. These policies are evaluated in addition to the embedded policies of the ISVs.
Before starting an application a user has only limited information about the cost incurred most often estimated based on wall-clock time of usage only.	With elasticLM, an accurate, user-specific price is calculated beforehand based on a large number of configurable parameters, like the time of usage, the features, the history of usage, local policies that define different prices for different users or user groups.
Accounting of license usage more often than not is statically bound to usage times.	elasticLM comes with an advanced accounting and billing system that allows to adapt the accounting information after license usage, taking into account the effective usage, e.g. runtime information, hardware capabilities.

Current Situation	Innovation of elastictLM
License usage control in terms of budget for different users or groups is done independently from the process of granting licenses.	In elasticLM budget limitations are checked and enforced when a user requests a license.
Illegal usage of licenses can be achieved through hacked license servers or hacked versions of the license supplied by the ISV.	elasticLM realises a number of sophisticated, state-of-the-art security mechanisms that render illegal use almost impossible.
License terms are immutable once checked out from the license server.	elasticLM offers re-negotiation of license terms at run-time, e.g. giving up a license before the reservation period is over, trying to extend a reservation period, or adding new features.
License servers only support first come first served schema.	elasticLM allows advance reservation of licences for later use, e.g. coordinated with the availability of computational resources.
Customer owned licenses managed in his administrative domain usually can not be used for running applications using an ASP's computational resources.	Through elasticLM an ASP can temporarily host the customer's licenses allowing the execution of applications using the customer's own licenses. Customers' licences may be combined with ASP owned licenses for running complex jobs, e.g. exceeding the number of processors a single license grants to use, with different applications or application features.

As described in the table above, elasticLM will introduce a number of innovative features compared to existing technology for software licensing. Basically all parties involved can take advantage of using the elasticLM solution:

Independent Software Vendors

Technical characteristics	Benefits
Extended trust management that uses standardized authentication and authorisation technologies.	Providing more flexibility to the trusted customer while increasing the level of security. Easy to deploy in Grid, Cloud and SOA environments.
Capability for defining arbitrary local policies that regulates license usage.	More fine-grained local policies might be defined on top of the policies laid down in the contract between ISV and customer. Thus, ISV policies embedded in the license may be kept simple.
Increased flexibility through token mechanisms for the implementation of new usage models, e.g. pay per use or SaaS.	ISVs may extend their customer base by supporting new business models, thus generating additional economical benefit.

Computing Centres

Technical characteristics	Benefits
Single point for managing licenses.	Better control of the license portfolio in an institution or company. Always up-to-date information on all purchased licenses, e.g. used and free licenses. Thus, less unused or barely used licenses, no duplicate licenses in different departments.
Capability for defining arbitrary local policies that regulate license usage.	Fine grained steering of license usage for all groups and individual users.
License reservation	Licenses are available when needed by an application at a later time. Thus, there is no need to guarantee availability through over-provisioning. This allows the centres to operate more cost-efficiently.
License co-scheduling with resources	Licenses can be used to run applications on the most appropriate or idle resources.

Application Service Providers

Technical characteristics	Benefits
Extended trust management that uses standardized authentication and authorisation technologies.	Providing more flexibility to the trusted customer while increasing the level of security. Easy to deploy in Grid, Cloud and SOA environments.
Capability for defining arbitrary local policies that regulates license usage.	More fine-grained local policies might be defined on top of the policies laid down in the contract between ISV and customer. Thus, ISV policies embedded in the license may be kept simple.
Increased flexibility through token mechanisms for the implementation of new usage models, e.g. pay per use or SaaS.	ISVs may extend their customer base by supporting new business models, thus generating additional economical benefit.

Computing Centres

Technical characteristics	Benefits
Single point for managing licenses.	Better control of the license portfolio in an institution or company. Always up-to-date information on all purchased licenses, e.g. used and free licenses. Thus, less unused or barely used licenses, no duplicate licenses in different departments.
License reservation	Licenses are available when needed by an application at a later time. Thus, there is no need to guarantee availability through over-provisioning. This allows the centres to operate more cost-efficiently.
License co-scheduling with resources	Licenses can be used to run applications on the most appropriate or idle resources.

Application Service Providers

Technical characteristics	Benefits
Single point for managing licenses.	Better control of the license portfolio in an institution or company. Always up-to-date information on all purchased licenses, e.g. used and free licenses. Thus, less unused or barely used licenses.
Temporarily including customers' licenses into own pool of licenses	ASP can provide the customer with access to applications without buying additional licenses. Customers' licences may be combined with the ones owned by the ASP for running complex jobs with different applications. This allows making better usage of the ASPs' computational resources while reducing the overhead of maintaining additional licenses for customers.
Capability for defining arbitrary local policies that regulate license usage.	Fine grained steering of license usage for all groups and individual users.
License reservation	Licenses are available when needed by an application at a later time.
License co-scheduling with resources	Licenses can be used to run applications on most appropriate resources, either taking into account the requirements of the customer/the application or the actual situation of the ASP's computational environments, e.g. idle resources.

End users

Technical characteristics	Benefits
Multiple ways to request a license, e.g. portal, Grid scheduler, by the application, a command-line interface.	Flexibility for the user embedding the licensing mechanism into his workflows.
Co-scheduling of licenses and computational resources	Select the most appropriate resources depending on actual requirements and have the necessary licenses available.
Reservation in advance	Licenses can be reserved for later use, and coordinated with the availability of computational resources.
License token	Allows execution completely decoupled from the site that hosts the license server.
Trusted entity available at execution site	Allows re-negotiation at run-time, e.g. giving up a license before the reservation period is over, trying to extend a reservation period, or adding new features. Thus, the license usage may be adapted to the real need. This allows reducing the cost either by not paying for times where the license is unused, or by avoiding an application crashing because the license was no longer valid.
Increased flexibility through token mechanism for the implementation of new usage models, e.g. pay-per-use.	Pay-per-use schemas allowing end users, in particular SMEs, to use software they could not afford with the traditional licensing models.

Use-cases

The following list depicts selected use-cases for elasticLM:

- Run license protected applications on (remote) Grid or Cloud resources using licenses from your local license pool
- ASP outsourcing
  - Outsource application execution to an ASP using licenses of the local license pool
  - One ASP is forwarding large jobs to another ASP
  - Reuse of existing licenses
- Use of Test licenses in virtualised environments
  - Provide infrastructure for freelance software developers
- Aggregation of licenses from different license pools,
  - e.g. local ones, ASP ones or from a Broker to run an application exceeding the locally available licenses
- License brokering
- Local use in (multi-)cluster environments without Grid or Cloud infrastructure
- Advance reservation of licenses, co-scheduling of licenses and other computational resources
- Extend/Reduce license terms when job is running.

Business models for ISVs and ASPs

In the SmartLM project we addressed both new business models and the technology supporting these business models for the emerging service and virtualization environments as well as in traditional cluster environments. elasticLM picks up both focal points of the SmartLM development extending the prototype to a product.

elasticLM is designed to support new business models, e.g. through aggregation or extension of licenses, along with service-oriented business models, e.g. budget-controlled pay per use, SaaS, use of remote resources for application execution. Naturally, elasticLM supports traditional license usage scenarios also.

Our analysis of existing business models, the analysis of current vendor and user issues and the examination of the impact of new license mechanisms result in the paradigm governing the development process: provide features allowing to create a Win-Win Situation for ISVs and users with elasticLM.

Integration in Applications

A single module implements the application's elasticLM interface. In the basic scenario, this API acts as a policy decision point decrypting the license terms, verifying the signature and analysing the terms. The result is provided to the application for further processing as usual.

In the advanced scenario the trusted entity is in charge of decrypting the license terms, verifying the signature and analysing the terms and forwards the results to the API. Moreover, the elasticLM API may provide advanced capabilities when connected to a trusted entity, e.g.

- trusted clock
- re-negotiation of license terms at run-time
- providing actual usage information to update the initial usage record

Since the elasticLM API implements the interface of the existing policy enforcement point in the application, there is no need to change the existing policy enforcement point in the application. elasticLM provides different language bindings depending on the application.

Contact for further information:

Josep Martrat
ATOS ORIGIN

SmartLM Project Coordinator
Av Diagonal 200
08018 Barcelona, Spain
Tel: +34 93 486 1818
Fax: +34 93 486 0766
email: josep.martrat@atosresearch.eu

Wolfgang Ziegler
Fraunhofer Institute SCAI

SmartLM Scientific Coordinator
Schloss Birlinghoven
53754 Sankt Augustin, Germany
Tel: +49 2241 2258
Fax: +49 2241 42258
email: Wolfgang.Ziegler@scai.fraunhofer.de

The SmartLM project is funded by the European Commission´s ICT programme under grant #216759

SmartLM Partner

Looking at the impact

The SmartLM Project

The development of a Grid-aware software licensing integrated into service-oriented architectures (SOA) will significantly bolster grid deployment generally - but specially into new areas exploiting a broad range of commercial software, beyond boundaries of technical computing and high performance computing.

The ability to effectively and dynamically manage the use of software licenses based on business objectives is not a grid-only issue, but grids are an important inflection point in this transformation.

It is with clarity that SmartLM is aiming to aid the creation of new industrial opportunities based on the creation of the emerging Service market that uses the service-oriented infrastructure as a means to deliver new software services in great many fields like Finance, Entertainment, Retail, Pharma, etc.

SmartLM will contribute to the technology convergence (virtualization, Grid, SOA, etc) -and large interoperability objective with focused contributions to WS standards and specifications at the Open Grid Forum (OGF).

The open nature of the new license management software will enable distributed applications by solving one of the major shortcuts for its deployment. Well-known applications can be adapted to this new multinomial networked environment, helping organizations to get dynamic access to the right of usage applications. This will support not only large enterprises but also Small and Medium Businesses (SMBs) who do not have the negotiating power to obtain non-standard pricing or licensing from ISVs.

The challenge is to compensate the potential revenue loss on one side with greater business values on the other side. Customer needs and vendor requirements must be balanced.

Product

Competitive Advantages of SmartLM

SmartLM aims at rendering mechanisms for managing and using software licenses in a more fair and flexible way. SmartLM licenses may be used seamlessly in local cluster environments, as well as in local or remote Grid and Cloud environments, and under circumstances that the SOA concept presents.
In the following table the main features and unique selling points of the SmartLM product are shown compared to the current situation:

Current situation vs. SmartLM

Current Situation	Innovation of SmartLM
Software licenses allow little flexibility in terms of location independent use. Thus, license protected applications may barely be used in Grid or Cloud environments.	With SmartLM, licenses may be used to run applications in Grid and Cloud environments no matter whether during the application run there is network connectivity to access the site that hosts the license server that issued the license.
Licenses are often spread across departments making it difficult to track license usage.	SmartLM provides access to and management of all licenses owned by a site.
All license usage policies are embedded in the license of the ISV.	SmartLM allows the definition of local policies for license usage addressing the site-specific needs. These policies are evaluated in addition to the embedded policies of the ISVs.
Before starting an application a user has only limited information about the cost incurred most often estimated based on wall-clock time of usage only.	With SmartLM an accurate, user-specific price is calculated beforehand based on a large number of configurable parameters, like the time of usage, the features, the history of usage, local policies that define different prices for different users or user groups.
Accounting of license usage more often than not is statically bound to usage times.	SmartLM comes with an advanced accounting and billing system that allows to adapt the accounting information after license usage, taking into account the effective usage, e.g. run-time information, hardware capabilities.
License usage control in terms budget for different users or groups is done independently from the process of granting licenses.	In SmartLM budget limitations are checked and enforced when a user requests a license.
Illegal usage of licenses can be achieved through hacked license servers or hacked versions of the license supplied by the ISV.	SmartLM realises a number of sophisticated, state-of-the-art security mechanisms that render illegal use almost impossible.
License terms are immutable once checked out from the license server.	SmartLM offers re-negotiation of license terms at run-time, e.g. giving up a license before the reservation period is over, trying to extend a reservation period, or adding new features.
License servers only support first come first served schema.	SmartLM allows advance reservation of licences for later use, e.g. coordinated with the availability of computational resources.
Customer owned licenses managed in his administrative domain usually cannot be used for running applications using an ASP's computational resources.	Through SmartLM an ASP can temporarily host the customer's licenses allowing the execution of applications using the customer's own licenses. Customers' licences may be combined with ASP owned licenses for running complex jobs, e.g. exceeding the number of processors a single license grants to use, with different applications or application features.
Traditional business models are selling licenses trough a large contract (annual, from 5 to 10 years, forever...)	SmartLM introduces a new business model 'Extension of License' that enables users to extend their license on demand. Users can use the software and only pay per what they use.
Current business models are fixed by time, penalizing the slow hardware.	With SmartLM the user will pay an effective license price, independent of the hardware.

As described in the table above, SmartLM will introduce a number of innovative features compared to existing technology for software licensing. Basically all parties involved can take advantage of using the SmartLM solution:

Software licensing as a #1 barrier in commercial grid deployments

Software licensing

Software licensing as a #1 barrier in commercial grid deployments

Traditionally, software licenses have been provided on the basis of a named user, node-locked host, number of concurrent jobs or possibly a floating site license. These models are not sufficiently flexible to support commercial applications that access resources beyond the current administrative domain - possibly as a Utility-like service outside the organization or Software as a Service (SaaS) model.

Software licensing is identified as a particular concern for enterprise IT managers as they start to deploy virtual Grids in any meaningful way. For all the potential benefits of Grids, IT departments cannot afford to buy software licenses for every device in the service oriented infrastructure that by nature consumes resources dynamically.

In addition, the Grid-based Service-Oriented Architecture (SOA) solutions together with other technologies trends such as multi-core and virtualization environment are forcing Independent Software Vendors (ISV) to move away from traditional software licensing models.

The SmartLM ambitions

The mission of SmartLM is to provide a new generic licensing virtualization framework based on standards, and integrate it in major Grid middleware solutions.

The strategic objectives of SmartLM are:

- To understand the licensing requirements for Grid use and deployment in the commercial environment, involving software vendors, application service providers, IT integrators, resource providers and end users.
- To identify service-oriented business models for distributed scenarios across organizations.
- To design and build a secured platform-independent licensing management framework.
- To provide models and technologies for accounting and billing of licenses.
- To enable and validate the licensing management tools with commercial applications deployed in Grids.

Treating and implementing software licenses as Grid services

The »Smart« LM approach

The overall approach consists in treating and implementing software licenses as Grid services thus providing platform independent access just like other virtualized resource.

- Licenses will become Grid services; a promising approach to overcome the limitations of current monolithic licensing models.
- Licenses will be managed as agreements, extending the conventional Service Level Agreements (SLAs) which are made today between sellers and buyers in the market.
- Licenses will be dynamic in order to support agreements that may change over time and where the dynamic negotiation between service provider and consumer is needed.

SmartLM Partner

Atos Origin - Spain

Fraunhofer SCAI - Germany

FZ Jülich - Germany

CINECA - Italy

The 451 Group

INTES - Germany

ANSYS - Germany

LMS International - Belgium

T-Systems - Germany

CESGA - Spain

Gridcore Aktiebolag - Sweden

Publications

Whitepapers

- The Business Side of Software Licensing (PDF, 296 KB)
- Architectural overview of the SmartLM innovative license management system (PDF, 3,3 MB)

Executive Summaries of Deliverables

- D2.1 Software Licensing Panorama Today (PDF, 180 K)
- D2.2 Report on business impact in adoption of Grid licensing mechanisms and suggested Business Models (PDF, 184 K)
- D3.1 Design of the license service, specification of WS-Negotiation protocol (PDF, 251 K)
- D4.1 Process-Model for Accounting (PDF, 165 K)
- D6.1 Scenarios, criteria and methodology for evaluation V1.0.pdf (PDF, 191 K)
- D7.3 Product description and Market context analysis for exploitation_v1.0.pdf (PDF, 193 K)
- D7.4 Preliminary exploitation plan v1.0.pdf (PDF, 246 K)

Public Deliverables

- D1.1 Functional requirements of the new licensing architecture for Grids (PDF, 605 K)

Booklets

- SmartLM - Grid-friendly software licensing for location independent application execution (PDF, 2.5 M)

Posters

- SmartLM Innovation (PDF, 750K)
- SmartLM Overview (PDF, 2,2M)
- elasticLM - The Product (PDF, 330K)

Presentations

Wolfgang **Ziegler**, elasticLM - A novel approach for software licensing in distributed computing infrastructures. 2nd IEEE International Conference on Cloud Computing Technology and Science, Indianapolis, USA, November 30, 2010

Björn **Hagemeier**, SLA-based management of software licenses s web service resources in distributed environments. Euro-Par Conference, GECON Workshop, Iscia, Naples, Italy, August 31st 2010**.**

Wolfgang, **Ziegler**: Software licenses as mobile objects in distributed computing infrastructures. Euro-Par Conference, CoreGRID Workshop, Iscia, Naples, Italy, August 31st 2010**.**

Wolfgang **Ziegler**: elasticLM - Flexible Software License Technology for Distributed Computing Infrastructures. Exhibitor Forum at the International Supercomputing Conference, Hamburg, Germany, June 2nd, 2010.

Angela **Rumpl**, Wolfgang **Ziegler**: Booth at the at the International Supercomputing Conference, Hamburg, Germany, June 1st - 3rd, 2010.

John **Barr**: 451 CloudScape - Success Stories, Vendors and Directions. Keynote at the Open Grid Forum 28, Munich, Germany, March 18, 2010

John **Barr**: An industrial perspective on current Innovations in HPC. HPC Consortium meeting, Munich, Germany, March 15, 2010

Wolfgang **Ziegler**, WS-Agreement: Foundations for standardized QoS negotiations. Workshop Managing Quality of Service in Distributed Computing Infrastructures, Open Grid Forum 28, Munich, Germany, March 15-18, 2010

Wolfgang **Ziegler**: Extending WS-Agreement with multi-round negotiation capability. IEEE Workshop Service Level Agreements in Grids, Grid 2010, Banff, Canada, October 12-16, 2009.

Scientific Publications

Cacciari, Claudio; **D'andria**, Francesco; **Hagemeier**, Björn; **Mallmann**, Daniel; **Martrat**, Josep; **Rumpl**, Angela; **Ziegler**, Wolfgang; **Zsigri**, Csilla: elasticLM - Software License Management for Distributed Computing Infrastructures. In: ERCIM News, volume 83.

Cacciari, Claudio; **D'andria**, Francesco; **Hagemeier**, Björn; **Mallmann**, Daniel; **Martrat**, Josep; **Peréz**, David; **Rumpl**, Angela; **Ziegler**, Wolfgang; **Zsigri**, Csilla: elasticLM - A novel approach for software licensing in distributed computing infrastructures. In: Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science, Indianapolis, USA, Nov. 30-Dec. 3, 2010.

Battré, Dominic; **Brazier**, Frances M.T.; **Clark**, Kassidy P.; **Oey**, Michael; **Papaspyrou**, Alexander; **Wäldrich**, Oliver; **Wieder**, Philipp; **Ziegler**, Wolfgang: **A proposal for WS-Agreement Negotiation.** In: Association for Computing Machinery u.a.: Grid 2010 : 11th ACM/IEEE International Conference on Grid Computing, 25-29 October 2010 in Brussels, Belgium. 2010

Cacciari, Claudio; **D'andria**, Francesco; **Hagemeier**, Björn; **Mallmann**, Daniel; **Martrat**, Josep; **Rumpl**, Angela; **Ziegler**, Wolfgang; **Zsigri**, Csilla: **SLA-based management of software licenses as web service resources in distributed environments.** In: Euro-Par 2010 : Ischia, Italien, 31.8.-3.9. 2010. Berlin [u.a.] : Springer, 2010

Cacciari, Claudio; **D'andria**, Francesco; **Hagemeier**, Björn; **Mallmann**, Daniel; **Martrat**, Josep; **García Peréz**, David; **Rumpl**, Angela; **Ziegler**, Wolfgang; **Zsigri**, Csilla: **Software licenses as mobile objects in distributed computing environments.** In: Euro-Par 2010 : Ischia, Italien, 31.8.-3.9. 2010. Berlin [u.a.] : Springer, 2010

Rumpl, Angela; **Wäldrich**, Oliver; **Ziegler**, Wolfgang: **Extending WS-AGREEMENT with multi-round negotiation capability.** In: Institute of Electrical and Electronics Engineers u.a.: Grid 2009 : the 10th IEEE /ACM International Conference on Grid Computing, October 12-16, 2009 in Banff, Alberta, Canada. New York, NY [u.a.] : Springer, 2010 (CoreGrid series).

Battré. Domonic (Ed.); **Wieder**, Philipp (Ed.); **Ziegler**, Wolfgang (Ed.); Open Grid Forum: **WS-Agreement Specification Version 1.0 Experience Document.** Open Grid Forum, 2010 (OGF document series)

Francesco **D'Andria**: An Enhanced Strategy for License Management in a Dynamic Grid Environment. Echallenges e-2009 Conference, Istanbul, Turkey, October 22, 2009.

Björn **Hagemeier**: Securing Software Licenses in a Location Independent Manner. Krakow Grid Workshop, Krakow, Poland, October 12, 2009.

Angela **Rumpl**, Wolfgang **Ziegler**: Booth at the at the International Supercomputing Conference, Hamburg, Germany, June 23rd - 25th, 2009.

Daniel **Mallmann**, Wolfgang **Ziegler**: Software License Issues in Grids, Clouds & SOA. BoF session at the International Supercomputing Conference, Hamburg, Germany, June 23rd, 2009.

Wolfgang **Ziegler**: elasticLM - Software License Technology for Grids, Clouds, SOA. Exhibitor Forum at the International Supercomputing Conference, Hamburg, Germany, June 25th, 2009.

Wolfgang **Ziegler**: SmartLM - Software License Technology for Grids, Clouds, SOA. D-Grid Workshop Software-Lizenzmanagement in Grid Umgebungen, Sankt Augustin, Germany, May 4th, 2009.

Wolfgang **Ziegler**: SmartLM - Software License Technology for Grids, Clouds, SOA. GridEcon Workshop, Athens, Greece, April 23rd, 2009

Karl **Catewicz**, Wolfgang **Ziegler**: SmartLM demonstrations. Internet of Service Collaboration Meeting, Brussels, Belgium, June 10th - 11th, 2009

Wolfgang **Ziegler**: SLA Negotiation in SmartLM. Internet of Service Collaboration Meeting, Session of the Technical Working Group on QoS and SLA, Brussels, Belgium, June 10th, 2009

Wolfgang **Ziegler**: SmartLM Business Models and Technology. Internet of Service Collaboration Meeting, Session of the Technical Working Group on Business Models and SLAs, June 11th, 2009

SmartLM launch event: Contribution to the European Commission Event Contribution towards the Internet of Services - INFRASTRUCTURE/ VIRTUALISATION

John **Barr**: This gig is bigger than grid: from HPC to service orientation. OGF22, Boston, Financial Workshop, 27 February 2008.

Wolfgang **Ziegler**: Service Level Agreements in the SmartLM Project. Dynamic SLA Workshop, Open Grid Forum 23, Barcelona, Spain, 2. June, 2008

Wolfgang **Ziegler**: SmartLM - Grid-friendly software licensing for location independent application execution. CCGrid2008 European Projects Showcase, Lyon, France, 22. May 2008

Wolfgang **Ziegler**: Towards SLA-based Software Licenses and License Management in Grid Computing. CoreGRID Symposium, Las Palmas de Gran Canaria, Gran Canaria, Spain, 25. August 2008

Wolfgang **Ziegler**: SmartLM Software License Technology for Grids, Clouds, SOA. Open Grid Forum 25, Catania, Sicily, Italy, 6. March 2009

John **Barr**: Smart License Management. The 451 Group's Enterprise Computing Strategies Summit, London, 5 June 2008.

M. **Riedel**, D. **Mallmann**, A. **Streit**: Usage of the UNICORE Grid Technology in Scientific und Economic Domains, Grid & Sea, Bulgaria, 11th June 2008

Björn **Hagemeier**, Wolfgang **Ziegler**: SmartLM - Grid-friendly License Management for Location Independent Application Execution. D-Grid License Workshop, Hannover, Germany, November 4th, 2008.

Björn **Hagemeier**, Wolfgang **Ziegler**: SmartLM - Grid-friendly License Management for Location Independent Application Execution. D-Grid All Hands Meeting, Göttingen, Germany, March 23rd, 2009.

Video

The following video present a high-level view of the resulting product elasticLM:
elasticLM video (mp4, 59 M)

Rana, Omar; **Ziegler**, Wolfgang: **Research challenges in managing and using Service Level Agreements.** In: Desprez, F. (Hrsg.) u.a.; European Research Consortium for Informatics and Mathematics: Grids, P2P and services computing : based on the CoreGrid ERCIM Working Group Workshop on Grids, P2P and Service Computing in Conjunction with EuroPar 2009, August 24th, 2009 in Delft, The Netherlands. New York, NY [u.a.] : Springer, 2010 (CoreGrid series 12).

Francesco **D'andria**, Josep **Martrat**, Csilla **Zsigri**, Daniel **Mallmann**, Björn **Hagemeier**, Claudio **Cacciari**: An Enhanced Strategy for License Management in a Dynamic Grid Environment. eChallenges e-2009 Conference Proceedings ISBN: 978-1-905824-13-7

Hagemeier, Björn; **Mallmann**, Daniel; **Ziegler,** Wolfgang: **Securing software licenses in a location independent manner.** In: Bubak, Marian (Hrsg.) u.a.: Cracow Grid Workshop '09 : October 12 -14, 2009, Cracow, Poland ; proceedings. Cracow : Academic Computer Centre CYFRONET AGH, 2010, S. 112-119

Csilla **Zsigri** : SmartLM's flexible licensing virtualization technology makes licenses mobile objects The 451 Group, July 2009,

PDF

Daniel **Mallmann**, Josep **Martrat**, Wolfgang **Ziegler**: Smart LM: Grid-friendly Software Licensing for Location independent Application Execution. inSiDE, Vol. 6 No. 1, Spring 2008,
Link

Jiadao **Li**; Oliver **Wäldrich**; Wolfgang **Ziegler**: Towards SLA-based Software Licenses and License Management in Grid Computing. CoreGRID Symposium, Las Palmas de Gran Canaria, Gran Canaria, Spain, August 2008, Springer, 2008, CoreGRID Series, p. 139 - 152.

M. **Riedel**, D. **Mallmann**, A. **Streit**: Usage of the UNICORE Grid Technology in Scientific und Economic Domains, Proceedings of the Workshop on Grid and Scientific and Engineering Applications (Grid&SEA) at Engineering and Economics (AMEE'08), June 2008, Sozopol, Bulgaria, AIP Conference Proceedings 1067, pp. 559-566,

Link

Csilla **Zsigri**: SmartLM. 451 MIS 2008 Review EU Research, 5 December 2008

Events

2010

Event	*Date / Place*	*www*
FIREweek 2010	30 June - 1 July Barcelona	fireweek2010.upf.edu/index.htm
Future Internet Conference Week	13-17 December Ghent, Belgium	www.fi-week.eu
Future Internet Assembly (FIA)	16-17 December Ghent, Belgium	www.fi-week.eu
ServiceWave 2010	13-15 December Ghent, Belgium	servicewave.eu/2010/
Future Internet Research and Experimentation (FIRE) Conference	15 December Ghent, Belgium	cordis.europa.eu/fp7/ict/fire/
3rd Future Internet Symposium 2010	20-22 September Berlin	www.fis2010.org
ICT 2010 Event	27-29 September Brussels	ec.europa.eu/information_society/events/ict/2010/index_en.htm
ICT2010 Networking session on Internet Science	29 September Brussels	www.paradiso-fp7.eu
Future Internet Assembly (FIA)	14-16 April Valencia	www.fi-valencia.eu
Collaboration Meeting FP7 ICT projects funded by Challenge 1.2 (Call1 and Call5)	19-20 October	Brussels ec.europa.eu/information_society/events/ssai/ios/index_en.htm
Future Networks 2010 Concertation meeting for FP7 projects	18-20 October Brussels	ec.europa.eu/information_society/events/future_networks/concertation/index_en.htm
2009		
Event	***Date / Place***	***www***
ACUM	18-20 November 2009 Germany	
ACUM	21-22 October 2009 France	
International Supercomputing Conference (ISC)	23-26 June 2009 Hamburg, Germany	Link
CEBIT	3-8 March 2009 Hannover, Germany	Link
2008		
Event	*Date / Place*	*www*

ICSOC 2008	5 December 2008 Sydney, Australia	Link
eChallenges 2008	22-24 October 2008 Stockholm, Sweden	Link
GRID 2008 IEEE/ACM International Conference on Grid Computing	29 September 2008 Tsukuba, Japan	
Open Grid Forum - OGF24	15- 19 September 2008 Singapore	Link
UNICORE Summit 2008	26 August 2008 Gran Canaria, Spain	Link
CoreGRID Symposium @ EuroPar	25-26 August, 2008 Gran Canaria, Spain	Link
451 Enterprise Computing Strategy Summit	5 June 2008 London, UK	Link
BEinGRID Industrial event	3-5 June 2008 Barcelona, Spain	Link
The 23rd Open Grid Forum - OGF23	2-6 June 2008 Barcelona, Spain	Link
Open Grid Forum - OGF22	25- 28 February 2008 Boston, USA	Link

Publishing Notes

The Fraunhofer Institute for Algorithms
and Scientific Computing SCAI

Schloss Birlinghoven
53754 Sankt Augustin
Germany

Phone +49 2241 14-2500
Fax +49 2241 14-2460
http://www.scai.fraunhofer.de
is a constituent entity of the Fraunhofer-Gesellschaft, and as such has no separate legal status.

Fraunhofer-Gesellschaft
zur Förderung der angewandten Forschung e.V.
Hansastraße 27 c
80686 München
Germany
Phone: +49 89 1205- 0
Fax: +49 89 1205-7531
www.fraunhofer.de

VAT Identification Number in accordance with §27 a VAT Tax Act: DE 129515865

Court of jurisdiction
Amtsgericht München (district court)
Registered nonprofit association
Registration no. VR 4461

Responsible editor:
Diplom-Journalist Michael Krapp
michael.krapp(at)scai.fraunhofer.de
Phone +49 2241 14-2935

Executive Board
Prof. Dr.-Ing. Reimund Neugebauer, President, Corporate Policy and Research Management,
Technology Marketing and Business Models
Prof. Dr. Alexander Kurz, Human Resources, Legal Affairs and IP Management
Prof. (Univ. Stellenbosch) Dr. Alfred Gossner, Finance, Controlling
(incl. Business Administration, Purchasing and Real Estate) and Information Systems

Schloss Birlinghoven
53754 Sankt Augustin

Notwithstanding this requirement, material may be downloaded or printed for use in connection with press reports on the activities of the Fraunhofer-Gesellschaft and its constituent institutes, on condition that the following terms are complied with:

No alterations may be made to pictorial content, with the exception of framing modifications to emphasize the central motif. The source must be quoted and two free reference copies must be sent to the above-mentioned address. Such usage is free of charge.

Disclaimer
We cannot assume any liability for the content of external pages. Solely the operators of those linked pages are responsible for their content.
We make every reasonable effort to ensure that the content of this Web site is kept up to date, and that it is accurate and complete. Nevertheless, the possibility of errors cannot be entirely ruled out. We do not give any warranty in respect of the timeliness, accuracy or completeness of material published on this Web site, and disclaim all liability for (material or non-material) loss or damage incurred by third parties arising from the use of content obtained from the Web site.
Registered trademarks and proprietary names, and copyrighted text and images, are not generally indicated as such on our Web pages. But the absence of such indications in no way implies that these names, images or text belong to the public domain in the context of trademark or copyright law.

Contact

Project Coordinator:

Mr. Josep Martrat
josep.martrat(at)atosorigin.com
Atos Research & Innovation.

For further Information:

Software & Service Architectures and Infrastructures European Commission - Information Society and Media DG
Office: BU25 3/134 B-1049 Brussels
Tel: +32 2 298 93 02
Fax: +32 2 296 70 18
infso-st(at)ec.europa.eu
http://cordis.europa.eu/fp7/ict/ssai/

Competitive Advantages (Part 2)

Independent Software Vendors

Benefits for the independent software vendors

Technical characteristics	Benefits
Extended trust management that uses standardized authentication and authorisation technologies.	Providing more flexibility to the trusted customer while increasing the level of security. Easy to deploy in Grid, Cloud and SOA environments.
Capability for defining arbitrary local policies that regulates license usage.	More fine-grained local policies might be defined on top of the policies laid down in the contract between ISV and customer. Thus, ISV policies embedded in the license may be kept simple.
Increased flexibility through token mechanisms for the implementation of new usage models, e.g. pay-per-use, SaaS.	ISVs may extend their customer base by supporting new business models, thus generating additional economical benefit.

Benefits for the end user

End users

Technical characteristics	Benefits
Multiple ways to request a license, e.g. portal, Grid scheduler, by the application, a command-line interface, or scripts.	Flexibility for the user embedding the licensing mechanism into his workflows.
Co-scheduling of licenses and computational resources	Select the most appropriate resources depending on actual requirements and have the necessary licenses available
Reservation in advance	Licenses can be reserved for latter use, and coordinated with the availability of computational resources.
License token	Allows execution completely decoupled from the site that hosts the license server.
Trusted entity available at execution site	Allows re-negotiation at run-time, e.g. giving up a license before the reservation period is over, trying to extend a reservation period, or adding new features. Thus, the license usage may be adapted to the real need. This allows reducing the cost either by not paying for times where the license is unused, or by avoiding an application crashing because the license was no longer valid.

Competitive Advantages (Part 3)

Benefits for the application service providers

Application Service Providers

Technical characteristics	Technical characteristics
Single point for managing licenses.	Better control of the license portfolio in an institution or company. Always up-to-date information on all purchased licenses, e.g. used and free licenses. Thus, less unused or barely used licenses.
Temporarily including customers licenses into own pool of licenses	ASP can provide the customer with access to applications without buying additional licenses. Customers' licences may be combined with the ones owned by the ASP for running complex jobs with different applications. This allows making better usage of the ASPs' computational resources while reducing the overhead of maintaining additional licenses for customers.
Capability for defining arbitrary local policies that regulate license usage.	Fine grain steering of license usage for all groups and individual users.
License reservation	Licenses are available when needed by an application at a later time.
License co-scheduling with resources	Licenses can be used to run applications on most appropriate resources, either taking into account the requirements of the customer/the application or the actual situation of the ASP's computational environments, e.g. idle resources.

Benefits for the computing centers

Computing Centres

Technical characteristics	Benefits
Single point for managing licenses.	Better control of the license portfolio in an institution or company. Always up-to-date information on all purchased licenses, e.g. used and free licenses. Thus, less unused or barely used licenses, no duplicate licenses in different departments.
Capability for defining arbitrary local policies that regulate license usage.	Fine grain steering of license usage for all groups and individual users.
License reservation	Licenses are available when needed by an application at a later time. Thus, there is no need to guarantee availability through over-provisioning. This allows the centres to operate more cost-efficient.
License co-scheduling with resources	Licenses can be used to run applications on the most appropriate resources, idle resources.

Competitive environment and SmartLM

Alternative solutions

This section presents the most relevant products and solutions that are in use for license management today or have the potential to become alternatives to SmartLM in the future. An attempt is made here to compare the features of each of these solutions. This is not an exhaustive list of all the solutions in the market or under development. We also name a few other known license managers that are used for single applications, but that do not take part of our benchmarking exercise. Before looking into the different license management systems we highlight some of the commonly used licensing models:

- Trial - license runs for a predetermined number of days following the initial use of the product
- Network (concurrent)/Floating - license and its usage (seat or count) managed by central server
- Standalone or Named User (personal license) - license a specific machine or user
- Pay-per-use - license usage (for payment) managed by actual usage, not seat limit or count
- Perpetual - license is granted to an individual by a centralized server for use on and off the network
- Evaluation - Licenses have a fixed expiration date
- License Roaming - allow a floating license to roam to a system which will subsequently be disconnected from the network
- Node Locked - The license is tied to a specific machine
- WAN/time zone restricted - license restricted for use in a specific time zone
- Time-limited - licenses restrict usage periods and allow subscription-based licensing models
- Token-based/product suite - licenses let you bundle products in many ways using token counts

Comparison of features

The following tables provide a comparison of features between the five most relevant license managers and SmartLM.

Product	Grid middleware support	Indepen- dence from academic Grid middleware	License Reservation	Multi vendor single manager	Access Control	Provides Statistical Information	Interface to Distributed Resource manager
FlexNET [1]	X	X		X	X(*)	X (LSF, Moab, PBSpro **)	
Sentinel RMS (Safe-Net Inc.) [2]	X			X	X		
Reprise License Manager [9]	X	X		X	X		
LM-X [10]	X			X	X		
GenLM	X	X		X	X	X	
SmartLM	X	X(***)	X	X	X	X	X

Table 1: Comparison of features I

(*)The user has to purchase a tool for this or has to create his statistic from the license log file (detects just check out and check in of licenses).
(LSF, Moab, PBSpro **): LSF, Moab and PBSpro are commercial add to the customer has to purchase.
(***)All others except SmartLM don't offer any kind of integration with Grid middleware, hence the independence for them from academic Grid middleware.

Product	Integrated Accounting and Billing	Support for Authentication based on Grid standards	Built-in aggregation of licenses from different administrative domains	Feature based licensing	Support for negotiation of license usage	Support for re-negotiation of license terms at run-time
FlexNET [1]		(*)	X			
Sentinel (SafeNet Inc.) [2]		X	X			
Reprise License Manager [9]			X			
LM-X [10]			X			
GenLM		X		X		
SmartLM	X	X	X	X	X	X

Table 2: Comparison of features II

(*)Aggregation is only possible by the application itself. It is possible to search different locations for licenses during one start and get licenses from different locations. Aggregation is not possible by a license server.

Product	Temporarily host and use ISV licenses at another site	Effective user-specific price available beforehand	Final accounting based on effective usage	Pay-Per-Use Model
FlexNET [1]		X		
Sentinel (SafeNet Inc.) [2]		X	X	
Reprise License Manager [9]		X		
LM-X [10]		X	X	
GenLM			X	X
SmartLM	X	X	X(*)	X

Table 3: Comparison of features III

(*) Effective usage for all the other license management systems is elapsed time, however, SmartLM goes way beyond this, being able to provide many more types of effective usage.

Positioning of SmartLM

Positioning of SmartLM

SmartLM is an advanced license enforcement product with a strong selling proposition for software licensing. SmartLM covers current features of available licensing software and meets the requirements of commercial Grid and Cloud deployments. The most important selling points are:

- Improved security, authentication and authorization
- Embedding in Grid environments
- Automated Accounting and Billing (100% trustworthy)

As SmartLM addresses the security issues of license tokens too, SmartLM covers also the traditional market for software licensing in LANs, and WANs. The feature comparison from Table 10 to Table 12 is used to describe the positioning of SmartLM in comparison to the most relevant five alternative solutions.

Improved Security, Authentication and Authorization

SmartLM addresses the needs of increased security for license protected applications and advanced authentication and authorization mechanisms for software access in Grid environments. These needs come from both the Independent Software Vendors (ISVs) and the end users. SmartLM prevents the misuse of license tokens like license cracking and license cloning by signing. The integrity of the license data and transported job data is guaranteed. The improved protection of license tokens is important for software licensing in local environments like LAN too.

To avoid the misuse of license in Grids by non-authorized users, SmartLM supports state of the art authentication and authorisation technologies like PKI (Public Key Infrastructure) methods and SAML assertions [5].

At the moment there is only one alternative solution on the market that uses state of the art Grid authentication and authorization technologies. But SmartLM is the only product supporting negotiation and re-negotiation at run-time.

Embedding in Grid environments

SmartLM is ready to use in Grid environments. SmartLM is embedded in Grid middleware like Globus Toolkit 4 [3] and Unicore [4]. The middleware integration includes advanced authentication and authorization (see above) and a license reservation system. The advanced reservation system is necessary to deal with the latency between process initiation and process start in the Grid environment.

SmartLM is the only licensing software supporting Grid middleware and the full range of licensing models described in chapter 5.1.

Automated Accounting and Billing

An accounting and billing interface is included in SmartLM. This accounting and billing interface allows the easy adoption of advanced business models like pay-per-use in Grid environments. The negotiation interface ensures WS-agreements (Web Service Agreements) and SLAs (Service Level Agreements). On the basis of these agreements it is possible to charge end users for the license usage. ISVs administrate the accounting and billing interface themselves. Sensitive accounting and billing data are protected by encryption.

SmartLM is the only product with an integrated accounting and billing interface. The unique build in accounting and billing interface and the additional functionalities like license aggregation and license extension allow SmartLM to offer new business models [6] like on demand combination of rented licenses with pay-per-use licenses.

Perceptual mapping

Figure 1 shows a perceptual map to illustrate the SmartLM positioning. Two criterions were used to position the licensing software solutions. The first criterion is the support or Grid environments in terms of support of Grid middleware, accounting and billing, license aggregation, license reservation. The second is the security. The highest level is the Grid state of the art authentication and authorization.

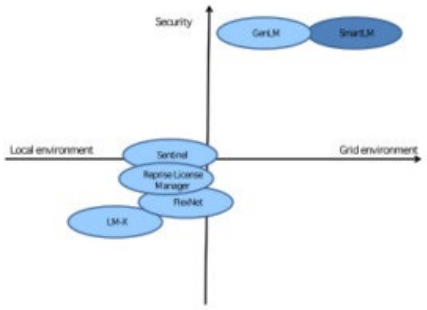


Figure 1: Perceptual map for Smart LM positioning

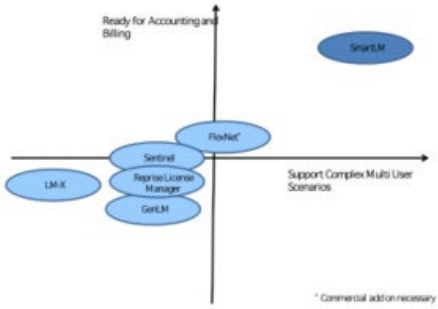


Figure 2: Perceptual map for SmartLM positioning in local environments with multiple clients

The perceptual map illustrates the excellent positioning of SmartLM in the market for software licensing in Grid environments.

Figure 2 shows a perceptual map showing the positioning of SmartLM in local environments with multiple users. Multiple users mean different companies, different applications and so on. This is a typical scenario from ASPs. The first criterion is the support of complex multiple user scenarios. To support complex scenarios like this you need features like license reservation, license aggregation, interface to Distributed Resource Managers (DRM), multiple vendor support. The second criterion is the connection to accounting and billing interfaces.

This figure show again the excellent position of SmartLM, as SmartLM is the only solution offering all features like license reservation, license aggregation, and interface to Distributed Resource Managers (DRM), multiple vendor support and built-in accounting and billing interface. It is difficult to determine the exact position of the alternative solutions regarding accounting and billing. FlexNet was positioned at second rank, as there are a lot of commercial tools (from Arcesso as FlexNet vendor) and other vendors available around FlexNet. The position of the rest should be similar. For better visibility they were ordered in an arbitrary order.

To conclude, SmartLM is a flexible Grid ready state of the art licensing software. SmartLM delivers highest security in combination with highly automated business processes. All market actors benefit, the ISVs from business expansion to the Grid, the End Users from increased flexibility and the ASPs from supplying more resources in the Grid.
As analyzed in Deliverable 2.2. [6] SmartLM offers new business models of which all market actors benefit. Four scenarios are accessible in SmartLM to fill in current gaps in the software licensing market (Featuring the ASP, Extension of licenses, Aggregation of licenses and Hardware-independent pricing) and therefore offering added value, such as pay-per-use model that opens the door to many new clients.

elasticLM

elasticLM - The Product

elasticLM is a novel technology for management of software licenses designed for distributed computing environments like Grids, Clouds or SOA.
elasticLM overcomes the limitations of existing software license management solutions making the use of license protected applications on resources outside of the own administrative domain as easy as running them locally.

Learn more about elasticLM: www.elasticlm.com

Atos Origin

Atos Origin

Atos Research & Innovation, the Atos Origin Spain's R&D hub, is a worldwide reference point in innovation for the whole Atos Origin group. Our work is focused on performing projects, usually within an international scope, that combine the most advanced technological developments with the economic exploitation of the R&D outcomes, based on an updated knowledge of the European policies and a deep awareness on human and societal aspects.

Thanks to our extensive background on R&D&I, Atos Research & Innovation leverages, inside and outside Atos Origin, research activities that are being performed on the newest technologies, and takes the outcomes of this research to specific projects with customers, introducing innovative elements in their business processes.

Main GRID projects in recent years are:

- BEinGRID - The BEINGRID project, a 16-million Euro funded initiative, is the largest investment of the European Commission in Grid technologies applied to the business world. www.beingrid.com
- TRUSTCOM - A Trust and Contract Management framework enabling secure collaborative business processing in on-demand created, self-managed, scalable, and highly dynamic Virtual Organizations www.eu-trustcom.com
- ASSESSGRID - The AssessGrid project's main objective is to address obstacles of a wide adoption of Grid technologies by bringing risk management and assessment to this field, enabling use of Grid computing in business and society.
- AKOGRIMO - The Akogrimo framework is targeted to scenarios where mobile dynamic virtual organizations (MDVO) require the ability to dynamically adapt the organisational structure to changing local situations, to dynamically establish and process complex workflows, and to access data and compute intensive services from distributed, sometimes even mobile resources.

Atos

http://www.atosresearch.eu
http://www.atosorigin.com

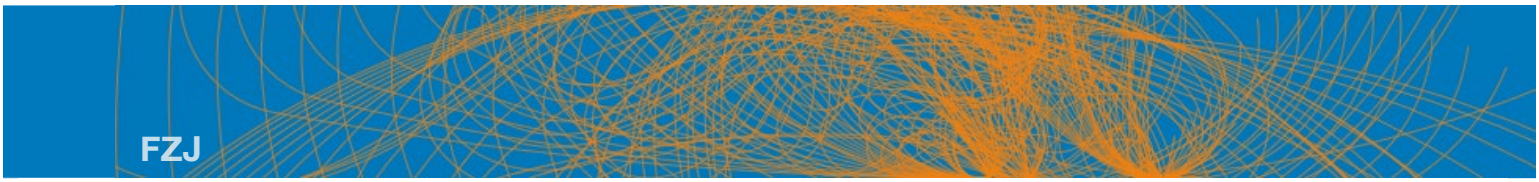
Fraunhofer SCAI

Fraunhofer Institute for Algorithms and Scientific Computing (SCAI)

Fraunhofer-Gesellschaft is the leading organization of institutes of applied research and development in Europe. Fraunhofer-Gesellschaft is a non-profit registered association, which currently maintains 58 research institutes in locations throughout Germany.
The Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), Sankt Augustin, led by Prof. Ulrich Trottenberg is working in the areas of Simulation, Numerical Software, Bioinformatics, and Optimization. The Department for Bioinformatics hosts a Grid Group that is working in the area of job scheduling for massive parallel machines since 1994 and does research and development in the domain of Resource Management and Scheduling for Grids since 2000.
Since 2003 the Grid-related work is focusing on: co-allocation of arbitrary resources for running complex jobs in the Grid, implementation of a resource allocation protocol supporting multi-site SLAs and advance reservation; development and evaluation of a flexible local scheduling system implemented as Grid-service.

Fraunhofer SCAI

http://www.scai.fraunhofer.de



Forschungszentrum Jülich GmbH

The Research Centre Jülich (FZJ) is the largest national German research facility and part of the German Helmholtz Association. The Jülich Supercomputing Centre (JSC) is part of FZJ and operates a national HPC centre. FZJ has been involved in Distributed and Grid Computing for many years and earned a proven track record. Through ZAM, FZJ co-ordinated the German UNICORE and UNICORE Plus projects, the EU-funded projects GRIP, OpenMolGRID, and UniGrids, participated in the EUROGRID project, and is currently participating in German Grid projects, like D-Grid and VIOLA, and European projects like OMII-Europe, DEISA; EGEE-II, A-WARE, Chemomentum, CoreGRID, NextGRID, and PHOSPHORUS. They managed the standardisation efforts of projects like GRIDSTART, GRIP, and NextGRID, and actively contribute to and lead standardisation activities in the Open Grid Forum and OASIS. Approximately 20 researches and technical personnel are involved in Grid-related activities among which are UNICORE user support for production systems and coordination of and support for the UNICORE Open Source development.

JÜLICH FORSCHUNGSZENTRUM

http://www.fz-juelich.de

 Publishing Notes | Contact

CINECA

CINECA Consorzio Interuniversitario

CINECA is a non profit Consortium, made up of 32 Italian universities, The National Institute of Oceanography and Experimental Geophysics - OGS, the CNR (National Research Council), and the Ministry of University and Research.

Today it is the largest Italian scientific computing centre and one of the most important worldwide. The present staff consists of more than 350 employees. CINECA is active in the area of technology transfer through high performance scientific computing, development and management of networks and web based services, development and amangement of complex information systems for large amounts of data.
It develops advanced Information Technology applications and services for italian academia, national research institution, industry and Public Administration.
Main EU projects involving CINECA are:

- DEISA-DEISA is one of the two GRID-empowered infrastructures funded by the EU. CINECA leads the Service Activity on Resource Management and Middleware.
- UniGrids - CINECA leaded the Work Package on applications, worked with Fujitsu Lab Europe in the work package on the technological foundations and with ICM on the high level services.
- A-Ware - CINECA leads the project management and the technical testing and participates to all the activities of the project.
- BEinGRID - CINECA is technical coordinator for BE12 and it's involved in the BE17 as a Grid Expert .

CINECA consorzio interuniversitario

http://www.cineca.it

451 Group

The 451 Group

The 451 Group is an independent technology industry analyst company focused on the business of enterprise IT innovation. The company's analysts provide critical and timely insight into the market and competitive dynamics of innovation in emerging technology segments.
Clients of the company - at vendor, investor, service-provider and end-user organizations - rely on 451 insights to support both strategic and tactical decision-making for competitive advantage.

The 451 Grid Adoption Research Service (GARS) - an investigation into user experiences and vendor strategies - extends The 451 Group's proven expertise in analyzing the grid technology market. The 451 Group believes that as grid technology is being absorbed into overall enterprise computing strategies, the discussion has moved beyond the grid itself.

The 451 Enterprise Computing Strategies (ECS) Research Service is the logical evolution of the GARS. This service offers in-depth, timely perspectives on enterprise architectures that incorporate grid techniques to support computing strategies such as enterprise utilities, shared services infrastructures and eco-efficient IT. The 451 ECS examines the effectiveness of the strategies of both existing and new vendor companies evolving their grid computing technologies to support enterprise computing strategies and applying virtualization and service-oriented architecture.
It also delivers frontline intelligence on customer adoption issues and market dynamics and provides relevant case studies and 'lessons learned' from early adopters and vendors.

the 451 group

http://www.the451group.com

INTES

Ingenieurgesellschaft für technische Software

INTES GmbH is an engineering company focussed on its product PERMAS, a general purpose finite element analysis system. INTES offers development, training, hotline and engineering services to its clients. It is located in Stuttgart, Germany with a subsidiary in France (INTES France). For the PERMAS development, INTES has participated to several national and EC projects (i.e. for the parallelization of the software).
Besides functionality and computational efficiency, a flexible software licensing system is important for the software business. For this reason, an own solution for network licensing has been developed. The experiences with this system form the basis to participate in the proposed project.

INTES

http://www.intes.de

ANSYS

ANSYS Germany GmbH

ANSYS, Inc., founded in 1970, develops and globally markets engineering simulation software and technologies widely used by engineers and designers across a broad spectrum of industries. The Company focuses on the development of open and flexible solutions that enable users to analyze designs directly on the desktop, providing a common platform for fast, efficient and cost-conscious product development, from design concept to final-stage testing and validation. ANSYS employs approximately 1,400 people worldwide and supports CAE tools such as ANSYS FEM, ANSYS CFX, and Fluent which are recognized as market and technology leaders. As a member of the SmartLM project the main objective of ANSYS is to contribute to software requirements for SmartLM from perspective of a software vendor (technical and business point of view), to build up test systems for SmartLM and evaluate these systems.

www.cfx-germany.com

Ansys Germany GmbH
Staudenfeldweg 12
83624 Otterfing
Germany

Name: Henning Eickenbusch
Phone: +49 (8024) 9054 24
Fax: +49 (8024) 9054 17
E-Mail: henning.eickenbusch@ansys.com
Web: www.cfx-germany.com
www.ansys.com

LMS

LMS International

LMS International N.V. and its affiliated subsidiaries (LMS Solutions, LMS Numerical Technologies and LMS Belgium) is a Belgian company active in the field of noise, vibration and durability engineering. LMS was founded in 1980, as a spin-off of the Katholieke Universiteit Leuven, Belgium.

A core activity of LMS is in the area of Process Integration and Design Optimization through the development and promotion of the Software tool OPTIMUS. OPTIMUS consists of modules that allow the definition and execution of Workflows as well as modules to address meta-modeling and optimization.

LMS actively participates in a multitude of Flemish and European R&D projects for the development of new technologies, both as project leader and as partner. To fuel this strong R&D activity, the company invests a considerable part of the annual turn-over in new developments.
The LMS software development process is ISO 9001 certified.

In the area of GRID computing LMS is an active member in the SIMDAT FP6 EC project (Project no.: 511438), BRIDGE (Project no: 97106) and PEGASUS (Project no.: 026673-2)

LMS ENGINEERING INNOVATION

http://www.lmsintl.com

 |

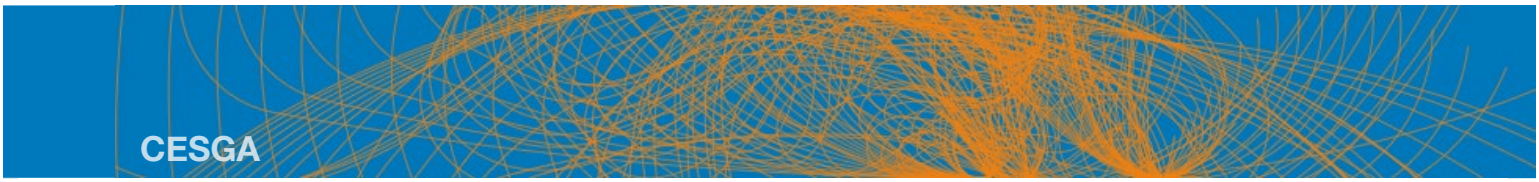
T-Systems

T-Systems-SfR

T-Systems-SfR is a joint-venture between T-Systems International (75%), one of Europe's top 5 providers of ITC-services and DLR, the German Centre for Aerospace Research (25%). TS-SfR has a focus on services for industrial and public Research and Development.
In the context of the EC-funded projects EUROGRID, UniGrids and BEinGrid and the UNICORE project funded by the German ministry for science and education the base-technologies for Grid-service environments were and are integrated and exploited for pilot scenarios.
License Management is a critical issue for the success of commercial Grid-environments. In the context of EUROGRID and the internal HPCPortal project, T-Systems has already implemented rudimentary methods for the management and accounting of software-licenses. These and T-Systems« internal license-management experiences will be exploited and further developed in the SmartLM project. So, T-Systems brings the asset of several years of production-experience in license-management into the project.
T-Systems will directly integrate the results of SmartLM into its services. T-Systems is also the responsible partner for license-management in the D-Grid project InGrid funded by the German ministry for science and education. The context of the SmartLM project is one brick in the development of management and business structures for a more general deployment of Grid-technology in service-environments.

T · · Systems · · ·

http://www.t-systems.de



Fundación Centro Tecnológico de Supercomputación de Galicia

CESGA (Galicia Supercomputing Centre) was founded in 1993 by the regional government of Galicia (Xunta de Galicia) and the National Research Council (CSIC) to promote and provide supercomputing facilities to the regional and national researchers. Nowadays, they are more than 20 people (6 of them are PhD). They provide more than 20.0 Tflops and more than 40 scientific applications, some of them commercial as ANSYS, MATLAB or Gaussian. CESGA is also participating in Grid projects since 1999. Since then, they have participated in several regional, national and international projects (Crossgrid, Irisgrid, EGEE, ProductionGrid, int.eu.grid, Ibergrid, e-IMRT, etc) where they have developed activities mainly in integration, testbed and testing of the software. CESGA main expertise today is in the deployment of Grid solutions, testing and validation, accounting and monitoring. In these two last topics, they are in charge of them in the SW federation of EGEE and they are now the responsible partner for accounting enforcement in EGEE-II. Also they have developed their own tools for monitoring and accounting for their production systems.

CESGA

http://www.cesga.es

Gridcore

Gridcore Aktiebolag

Gridcore AB is a system integrator with focus on Technical Computing. Girdocre AB delivers turn key solutions for Industry using Numerical Simulations (Fluid Dynamics, Structural analysis, Electromagnetics, etc...). Gridcore provides everything from Hardware to full Computational Portals for large corporations.
Gridcore has a large experience on designing and providing Computational environments for their customers. Our special competences are knowledge: about numerical methods, applications integration on Linux clusters, general performance issues, computational portal design and applied Grid computing.
Gridcore AB provides since 6 years back High Performance Computing on Demand for Scandinavian customers within the field of Technical and Scientific Applications together with major vendors of CAE applications.
Some customers of Gridcore are: TetraPak, Rolls Royce, Volvo Penta, Volvo Aero, Chalmers University, Lunds University, SAAB Bofors Dynamics, BAE Bofors, Astra Zeneca, Forsmarks Nuclear Power Station, Swedish Defence Department, etc. For some of the aforementioned customers Gridcore has total responsibility for: the systems architecture, applications integration and general simulation environment.

#GRIDCORE

http://www.gridcore.se

THE BUSINESS SIDE OF SOFTWARE LICENSING

Although licensing models have evolved with technology innovations, they do not fully satisfy the business issues faced by today's enterprises. The focus of successful licensing and support has to extend beyond cost and technology issues, the goal is to achieve software licensing based on business objectives balancing customer needs and vendor business models.

SmartLM Project White Paper

About the SmartLM project

SmartLM: Grid-friendly software licensing for location independent application execution. Contract number 216759. Project coordinator: Atos Origin - Spain Partners: Fraunhofer SCAI – Germany, Jülich Research Centre – Germany, CINECA – Italy, The 451 Group – UK, INTES – Germany, ANSYS – Germany, LMS International – Belgium, T-Systems – Germany, CESGA – Spain, Gridcore AB - Sweden Duration: 30 months (starting in February 2008) Total cost: 4.012.070 EUR Programme: European Commission - FP7 Information and Communication Technologies.

About The 451 Group

The 451 Group is a technology analyst company. We publish market analysis focused on innovation in enterprise IT, and support our clients through a range of syndicated research and advisory services. Clients of the company — at vendor, investor, service-provider and end-user organizations — rely on 451 insights to do business better.

About the author

This white paper was written by Csilla Zsigri, The 451 Group, based on the work done by the SmartLM Consortium in business modeling.

SOFTWARE IN BUSINESS

Software plays a critical role in business. Traditional software licensing models are under pressure as they do not satisfy the changing business needs of today's enterprises. Enterprise IT infrastructure is evolving towards hybrid models which harness in-house and third party resources (cloud, managed services, co-location). However, licenses are usually bound to hardware rather than fungible resources and are provided on the basis of named users, hostnames, or as a site license for the same admin domain of an organization. With Clouds notwithstanding, or when it comes to distributed environments and virtualized infrastructures, we run into trouble. Common software licensing terms are often too restrictive or expensive to run databases and applications on virtualized servers. Grids were an important inflection point in the transformation, but the lack of flexibility for running commercial software licenses in Clouds is still a bottleneck.

The software licensing issue is a complex one because transformation is going on at a macro level where a lot of money is involved. What has been happening and what can be expected is an evolution of license models, rather than a revolution. The goal and challenge is to balance customer needs and vendor requirements. In the past, end-user companies which wanted to extend the use of software to grid environments, either paid up or found a workaround. The ability to proactively manage the use of software licenses based on business objectives is not a grid-only issue. Virtualized infrastructures and distributed environments (including the Cloud) call for flexible and non-hardware based license models that support service-oriented business models. Software manufacturers need to change the way licensing works and use flexible and non-hardware based licensing solutions that better fit into a virtual environment.

THE SMARTLM PROJECT

Within the frameworks of the SmartLM project, we have addressed the licensing problem by working on a framework which delivers improved customer choice, but that will also keep the vendors happy. In the first half of 2008 we interviewed 30 companies, both software vendors and software buyers (end users), specifically for SmartLM, and we revised existing reporting in order to see the software licensing market clearly and address the real problems and challenges stakeholders are facing these days.

SOFTWARE LICENSE COMPLIANCE

The software industry employs the ´right to use´ model that ensures that the ownership and control of software usage remains in hands of the vendors. This right to use is granted through a license agreement, resulting in a contractual obligation that can end in a termination of the license if the licensee does not adhere to it.

The first obvious challenge we bump into is software license compliance. When it comes to compliance, we may find three behavioral patterns:

*"**The Good**"*- when your IT department knows exactly what software is installed on all systems and there are no concerns about what an audit would uncover.

"The Bad"- when your IT department thinks you are in good shape regarding compliance, they hope there are no renegade applications installed, and they are optimistic that an audit would not bring negative consequences.

"The Ugly"- when your IT department doesn't really have a clear picture of what applications are installed, and they have no idea about what an audit might uncover.

The unfortunate situation is that for most medium-sized organizations, *"the bad"* and *"the ugly"* tend to be the rule of the day when it comes to ensuring accurate software license compliance. Both software vendors and end users face a challenge here: vendors want their intellectual property to be protected and maximize revenue; end users need to easily track and manage the licenses that they are using.

VENDOR AND END-USER ISSUES

Obviously, non-compliance is not the way to go, but let's put this issue aside for a little while and think about what is wrong with licensing in light of emerging IT trends and companies' business drivers. As simple as it sounds and as complicated as it is, what users want is to control their expenses and get more value, while vendors do not vote for a reduction in revenue. While conventional, revenue-based business models are still dominating licensing mechanisms, but it is evident that the market is getting restless, and the demanding for more flexible licensing solutions from customers is growing.

It is clear that the focus of successful licensing and support has to extend beyond cost and technology issues, the goal is to achieve software licensing based on business objectives that balances customer needs and vendor business models. The achievement of a win-win situation between software vendors and users can be seen as the main requirement for a change.

Based on the conversations we had with vendors and users, software licensing is both a technology and a business concern, but the business issues are the most problematic ones. There are a few basic rules that licensing should comply with. We have called these *SmartLR*, SmartLM's Licensing Rules:

I. Licensing must offer reliability.
II. The cost of licensing must be low compared to the value of the license.
III. Users deserve fair conditions and maximum value.
IV. Licensing must offer flexibility to the user.
V. Keep the license model simple.

IT departments look to reduce software costs, as done with hardware and services in the past. Traditional licensing models are under pressure from a variety of alternative options that can tighten vendors´ profit margins and push down software licensing costs. These changes give more negotiating power to users. Customers want flexibility and they want vendors they can partner with. Models supporting distributed and virtualized technologies might vary, but some form of measured usage will likely be employed.

UNDERSTANDING USAGE, MAXIMIZING VALUE

The licensing landscape is quite chaotic with many licensing and pricing models around and providers randomly introducing new ones: node-locked license, flexible single-user license, floating license, score-based or token-based model, perpetual license, per-seat, per-CPU, per-concurrent-user models, pay-per-use and subscription pricing, hybrid license models, custom-contract based models and value-oriented pricing. I guess we all heard or mentioned a few of them in a conversation about software licensing and many times we would be talking about the same things using different words.

We believe that what we can talk about is an evolution of license models, rather than a revolution. ISVs can´t afford a revolution, as licenses are in the books, this way customers could question the value of their current licenses and put pressure on them. Vendors do not vote for a reduction in revenue, however, there must be some additional value to the user, such as being able to move a license around.

The SaaS delivery model has been offering some relief, as for the offered business services, instead of a single (usually) large licensing fee, customers pay recurring subscription fees. This subscription model most typically follows a relatively simple time-based approach (e.g. monthly fee). However, many SaaS providers realize that there is a need for far greater range of subscription models that may also take other factors into account such as usage, specific features and functions, service transactions, advertising funded revenue models, etc. But, it's important to point out that the development of the functional aspects of a SaaS application needs heavy investments on the provider's side.

Open Source is also a very important enrichment of the software world. Open Source is a development and distribution model, it´s attached to the software industry in the same way e-commerce is attached to the retail industry. If users can find an application that suits their needs and does not require a cumbersome license, they will use it. License flexibility seems to be a major driver to Open Source solutions. Software vendors can get a sound competitive edge by

exploiting Open Source strategies, as we are not talking about an ´all or nothing´ issue, hybrid models offer many advantages. Defining the line between what is free and what is paid for, is the critical component of any Open Source strategy. When a product is too niche-oriented, then it is not convenient to make it Open Source. When the software needs to generate its own revenue, dual licensing might be a very good option, and when the purpose of the software is to stimulate other sales, selling services around the software may be a good strategy to go.

To enhance customer satisfaction and value, vendors need to adapt to changes in surrounding technology and offer a wider array of licensing options. The evolution of licensing should move along understanding usage and maximizing value. Traditional licensing represents multiple restrictions, high risk and a lot of 'shelfware' for users and low commitment for vendors. Vendors need to work on offering an improved customer choice (increased value and more flexibility), which at the same time would result in improved customer relationships.

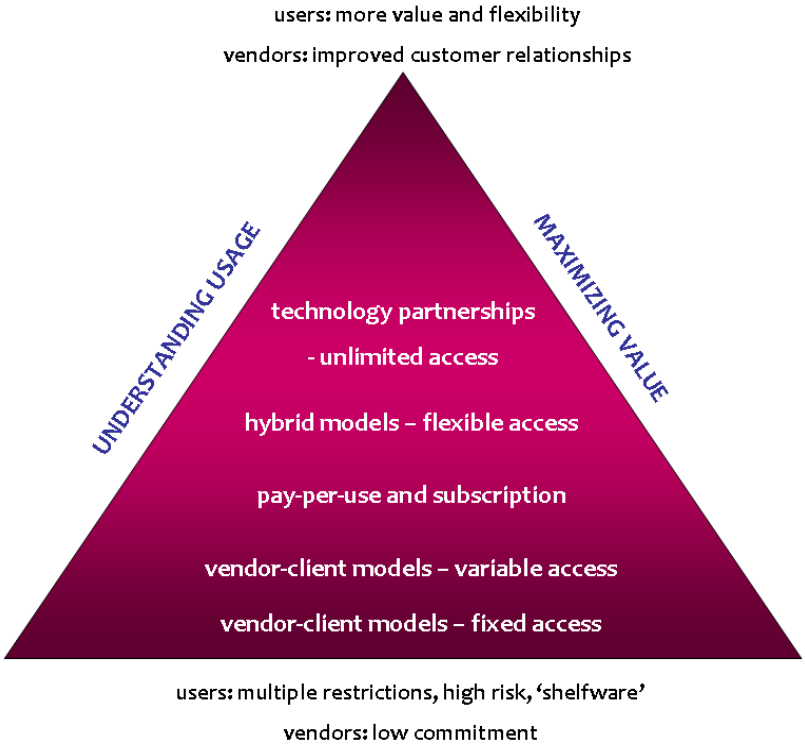


Figure 1: Evolution of software licensing

Although licensing models have evolved with technology innovations (from traditional vendor-client models through pay-per-use and hybrid models to technology partnerships), they do not fully satisfy the business issues faced by enterprises of all kinds and sizes when it comes to balancing productivity and efficiency, adjusting to changing needs or dealing with new requirements.

SERVICE-ORIENTED BUSINESS MODELS

In the SmartLM project, we are developing a flexible licensing virtualization technology which integrates new service-oriented business models. Through close collaboration with a wide range of stakeholders - software vendors, application providers, end users –, we identified some real licensing gaps and have developed new models that would help fill them in.

'Featuring the ASP' – In this model we find the Application Service Provider (ASP) offering various solutions to various problems. We highlight the following cases:

1) Customer license hosting: the customer owns a license for a specific application and deploys the license in the ASP´s environment. This case also allows for aggregation of licenses.
2) Embedded license: a dependant software vendor (DSV) commercializes its templates through the ASP. This case implies a license dependency to be solved, accounted and billed by the ASP.
3) License redirection: a third party (external consultant) owns a license and deploys the full license or part of it (sub-license) to carry out a specific project for the customer. Proper accounting is needed.
4) License reselling: the ASP resells the ISV's licenses for third-party use. ISVs may prefer to minimize the number of contacts they sell directly to and eventually minimize the risk for non-payments. Also for small software vendors, the ASP makes the access to market easier.

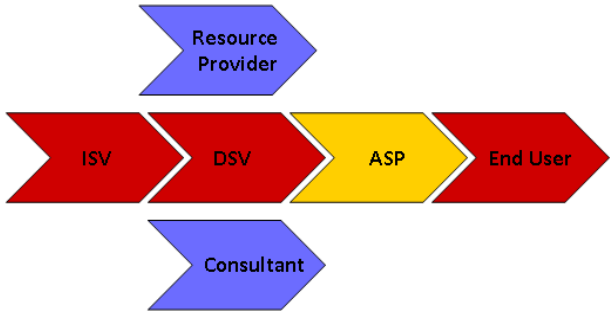


Figure 2: 'Featuring the ASP' business model

As illustrated in the figure above, the ASP plays a central role in all of these cases, being a reseller of hardware, software and services. The introduction of the ASP can be very advantageous for both the ISVs and the end users. From the ISVs' point of view, the ASPs can generate additional business offering licenses and hardware resources for end users on-demand (competitive resource provision). Making use of economies of scale and SmartLM features, the ASP provides resources in a competitive way, makes existing models (e.g. short-term licenses) more attractive to customers, and introduces new ones the ISV is not willing to offer, such as pay-per-use for license reselling.

License extension - The license extension model allows end users to extend their licenses in both their Local Area Network (LAN) and distributed environments on-demand, e.g. for workload peaks. The license server manages the process of the extension of licenses, e.g. in terms of accounting and license administration. These mechanisms give end users more flexibility and value and at the same time generate additional revenue for ISVs and ASPs. A proper pricing model is of paramount importance for the success of this business model: a price that is high enough so license extension does not affect the overall business model of the vendor and total revenue is increased; at the same time, a price that is low enough so the extension of licenses is interesting for the end user. The question is: at which point it is cheaper for a customer to extend its existing license instead of paying for an additional one? The extension of licenses is more attractive for the end user if he automatically gets an additional license after reaching the break even point.

License aggregation - Most contracts between ISVs and end users restrict the license usage to LAN. The license aggregation model allows the use of licenses that belong to different sites and brings them together to form a single license token. These licenses can come from either the ISV or the ASP. End users gain more flexibility and value and get access to huge hardware resources. The ASP provides these hardware resources to the end user and generates additional business for the ISV. In this case, besides proper pricing, correct identification of end user location and license source is essential. The key element is the location of the license to be aggregated. There are special agreements between software vendors and local software distributors. These software distributors often have exclusive distribution agreements with an ISV and license aggregation should aim for a win-win situation that supports this business network.

Hardware-independent pricing model and feature-based accounting – The hardware-independent pricing model makes the license price become effectively independent of the underlying hardware, enabling this way a cost-efficient usage of licenses. With the introduction of a set of micro-benchmarks, the user is not tied to hardware anymore, hence is not punished for slower hardware. All we need is a set of pre-defined micro-benchmarks that measure different performance elements of the platform on which the application is running. A price can be fixed on a linear scale against a pre-defined reference point – the profiling system. The reference price model is negotiated between end-user and ISV or end-user and ASP. The profiling is done by the ISV to formulate the application performance in terms of the micro-benchmarks. The final price will always be a weighted reference price.

This benchmark model leads us to a more general approach, to a feature-based accounting. The core issue here is letting the application define the features and set what it wants to charge for. As opposed to the time approach, the feature-based approach is really independent of the machine where the application is being executed.

CONCLUSIONS

BOTTOM LINE
Although licensing models have evolved with technology innovations, they do not fully satisfy the business issues faced by today's enterprises. The focus of successful licensing and support has to extend beyond cost and technology issues, the goal is to achieve software licensing based on business objectives balancing customer needs and vendor business models.

Software plays a critical role in business. Traditional software licensing models are under pressure as they do not satisfy today's enterprises changing business needs. **PAGE 3**

Virtualized infrastructures and distributed environments (incl. the Cloud) call for flexible and non-hardware based license models that support service-oriented business models. **PAGE 3**

To enhance customer satisfaction and value, vendors need to adapt to changes in surrounding technology and offer a wider array of licensing options. The evolution of licensing should move along understanding usage and maximizing value. **PAGE 5**

The software licensing issue is a complex one because transformation is going on at a macro level where a lot of money is involved. **PAGE 3**

Software licensing is both a technology and a business concern, but the business issues are the most problematic ones. There seem to be a few basic rules that licensing should comply with. (SmartLR) **PAGE 4**

Through close collaboration with a wide range of stakeholders - software vendors, application providers, end users –, we identified some real licensing gaps and worked out new models that would help fill them in. **PAGE 6**

SOURCES

The 451 Group: Grid Computing – The Impact of Software Licensing. 451 Grid Adoption Research Service Report (GARS) Report Four, William Fellows, March 2005
SmartLM D2.1 Software Licensing Panorama Today, June 2008
SmartLM D2.2 Business impact on the adoption of Grid licensing mechanisms and suggested business models, October 2008
Software License Compliance: The Good, the Bad, and the Ugly. KACE Networks, 2008. http://www.kace.com/about/newsletter/2008/software-license-compliance-the-good-the-bad-and-the-ugly.php
The Evolution of Software Licensing Models http://licensetracker.ca/
The 451 Group: SmartLM's flexible licensing virtualization technology makes licenses mobile objects, 451 Market Insight Service, EURO Report, Csilla Zsigri, July 2009
http://www.the451group.com/report_view/report_view.php?entity_id=59125

Architectural Overview of the SmartLM Innovative License Management System

SmartLM Whitepaper

SmartLM Consortium July 2010

Abstract

This Whitepaper describes the architecture of the SmartLM solution for creating and managing a new type of software licenses. While SmartLM licenses have been designed with a focus on distributed computing infrastructures like Grids or Clouds the SmartLM solution leverages its benefits also in other computing environments based on for example resources of an application service provider, local workstations or clusters .

Contents

1 Introduction

Software plays a critical role in business. While conventional, revenue-based business models are still dominating licensing mechanisms, it is evident that the market is getting restless, and the demand for more flexible licensing solutions from customers is growing. Enterprise IT infrastructures are evolving and software licensing needs to evolve with them.

Software manufacturers need to change the way licensing works and use flexible and non-hardware based licensing solutions that better fit into distributed and virtual environments. Grids were an important inflection point in the transformation, but the lack of flexibility for running commercial software licenses in Clouds is still a bottleneck.

The focus of successful licensing and support has to extend beyond cost and technology issues, the goal is to achieve software licensing based on business objectives that balances customer needs and vendor business models. The SmartLM project has addressed the licensing problem by working on a framework which delivers improved customer choice and that also keeps software vendors happy. SmartLM's offering brings along a model that makes licenses mobile objects. The main approach here is to provide platform-independent access and treat software licenses as services.

The rest of the whitepaper is organised as follows. Section 2 gives a global view on the architecture of the License Service and the Accounting and Billing Service. Section 3 presents the License Service in more detail, Section 4 does the same for the Accounting and Billing Service. The Orchestration Service is described in Section 5 and the SmartLM application programming interface presented in Section 6. The Conclusions in Section 7 and Acronyms and a Glossary in Sections 8 and 9 complement the whitepaper.

2 Overall Architecture

The SmartLM architecture has been designed as foundation an innovative License Management Service where license resources are treated as web service resources that the end-user can use on-demand, book in advance, aggregate with other license resources and more, while having complete control over the costs thanks to an integrated Accounting and Billing Service.

One important design goal is the adaptability of the SmartLM solution to the needs of different customers. This allows delivering SmartLM in three flavours basic, extended and full offering increasing functionality, e.g. depending on the customer's infrastructure or the customer's use-cases. Therefore, the License Service providing the core functionality and the Accounting and Billing Service were designed and implemented as a loosely coupled systems, which allows the Accounting and Billing Service to be included in the extended and full version but not in the basic version.

2.1 License Service Architecture

The SmartLM License Service follows a layered architecture comprising 6 layers: Coallocation, Authentication, Administration, Management, Business, and Persistency. The services,

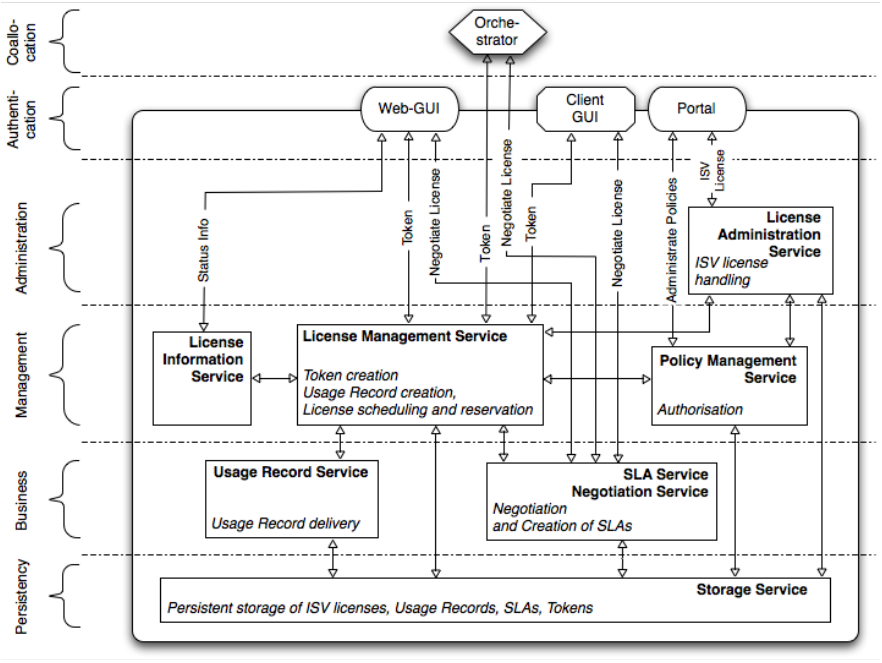


Figure 1: License Service architecture.

components and interfaces described below are the building blocks of this architecture. Figure 1 depicts the layered architecture of SmartLM highlighting the major communication paths between the components. The following paragraphs describe the different layers, the components inside a layer, and their interaction with other components. Finally, since security affects all layers, the fundamentals of the orthogonal SmartLM security are described.

2.2 Accounting and Billing Service Architecture

The Accounting and Billing (AB) Service within SmartLM is realised as an additional feature that allows ISV, ASP or end users to have global control and overview of their license usage. Thanks to the SmartLM Accounting and Billing Service, detailed statistics based on license resource usage is provided to the final user as graphs, tables or printable files.

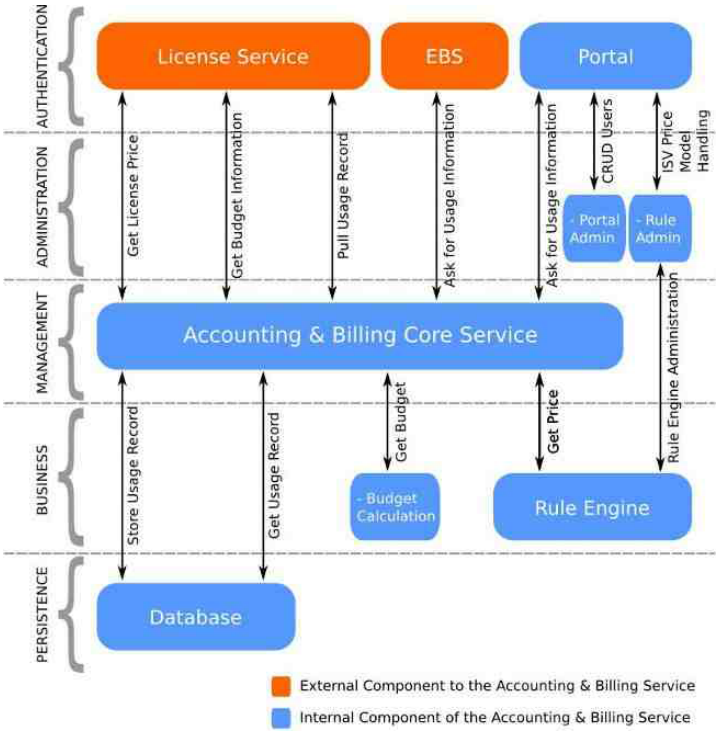


Figure 2: Accounting and Billing Service architecture.

As an overall view, the Accounting and Billing Service is a monolithic service with four main components (see Figure 2): AB Core, AB Database (DB), AB Portal, and the Rule Engine. The Database, Portal, and Rule Engine are coupled together with the AB core where administrative modules are included. At a glance, the DB stores the Usage Records while

the Rule Engine is used to set up license resource pricing models and the Portal provides the user with his/her accounting and billing license usage information.

The last section presents the coupling of the License Service and the Accounting and Billing Service, describing interaction and interfaces.

2.3 Interaction and Interfaces

The integration of the Accounting and Billing Service (ABS) with the License Service is based on web service communication that makes it possible to send end user's license usage information back and forth. An XML file (based on the Usage Record proposed recommendation of the Open Grid Forum) is used as the transfer key as it includes all the relevant information about individual license resources usage.

Figure 3 shows the interfaces involved in the communication and integration between the License Service and the Accounting and Billing Service.

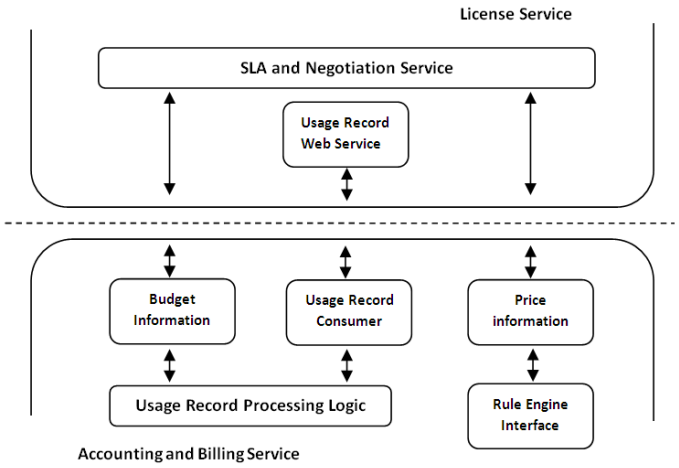


Figure 3: Communication interfaces between License Service and Accounting and Billing Service.

3 License Service

This section first briefly describes the internal components and their interaction according to the six layers of the overall architecture depicted in Figure 1. The second part of this section is dedicated to more detailed descriptions of the individual services.

The **Coallocation Layer** actually is not part of the SmartLM License Service but sits on top of the License Service. It comprises a single component: the *Orchestration Service*.

The *Orchestration Service* is responsible for assuring that computational resources and licenses for the execution of applications are available at the same time. This is achieved through negotiation with the resource management systems and the *SLA and Negotiation Service* for licenses. If successful, the negotiation results in both the reservation of computational resources and the licenses required for the application execution. Service Level Agreements are used to describe the reservations and the corresponding guarantees.

The *Orchestration Service* directly interacts with the *License Management Service* and the *SLA and Negotiation Service*. The user accesses the Orchestration Service through a client. This client has the same graphical user interface as the Client GUI in the **Authentication Layer** and requests the user to authenticate himself prior to accessing the *Orchestration Service*.

The **Authentication Layer** comprises three components, which allow the user to authenticate himself and - after successful authentication - to access the core services of the License Service. All user interfaces of the License Service are located in this layer: the interface to the Web-GUI, the graphical eclipse Client GUI and the interface to the Portal.

The Portal is used to access the *Policy Management Service*, the *License Information Service* and the *License Administration Service*. The Portal connects through two own interfaces to the *Policy Management Service* and the *License Administration Service*.

The other two interfaces - Web-GUI and Client GUI - connect both to the *License Management Service* and to the *SLA and Negotiation Service*. The Web-GUI additionally allows retrieving information from the *License Information Service*.

In the **Administration Layer** the *License Administration Service* for administrating the licenses received from an ISV is located.

The *License Administration Service* allows including new licenses to the License Service, removing licenses, and adding features to already installed licenses. This includes other vendor specific artefacts like special keys generated by the ISV, which become part of the license tokens.

The *License Administration Service* connects to the Portal, to the License Management Service, to the Policy Management Service and the Storage Service.

The **Management Layer** comprises three core services of the License Service: the *License Management Service*, the *License Information Service* and the *Policy Management Service*. The main responsibility of the *License Management Service* is (i) creation of License Tokens

taking into account the authorization of the user (including signing and - if required - encryption of the Token), (ii) creation of the initial version of the Usage Record and - if possible - creation of updated Usage Records based on the actual license usage, and (iii) license scheduling and reservation of licenses.

The *License Information Service* provides detailed information on existing licenses (i.e. licenses of a site that are administered by the *License Administration Service*), the actual use of licenses and license reservations.

The *Policy Management Service* is the central entity for managing end evaluating policies that define the access to and the usage rights of a certain license or license feature for a user or a group of users. The policies can include such defined by the ISV, policies defined locally by the site owning the license, and policies determined from the membership of the user in a Virtual Organisation.

The *License Management Service* as the central component of the License Service interacts with the *License Information Service*, the *Policy Management Service*, the *Usage Record Service*, the *SLA and Negotiation Service*, the *Policy Management Service*, and the *Storage Service.* Additionally, the *License Management Service* connects to the Web-GUI, the Client GUI and the external *Orchestration Service*. The *License Information Service* interacts with the Web-GUI and the *License Management Service*. The *Policy Management Service* interacts with the Portal, the *License Management Service* and the *Storage Service*.

The **Business Layer** hosts two services: the *Usage Record Service* and the *Service Level Agreement and Negotiation Service*. The *Usage Record Service* manages the Usage Records received from the *License Management Service*. It notifies the external *Accounting and Billing Service* that records are available for further processing by this service. The *Accounting and Billing Service* in turn pulls the available Usage Records whenever this is suitable.

The *SLA and Negotiation Service* is a crucial component of the License Service. It is responsible for creating the agreements for using licenses based on the requirements of the user and the availability of licenses for this user. Moreover, the Negotiation Service allows negotiating the terms of a license, especially the time when a license will be available for the user. While processing a license request the service also communicates with the external *Accounting and Billing Service* for retrieving a price for the requested license and the budget already used by this user or user group.

The two services of this layer interact both with internal components of the License Service and external components. The *Usage Record Service* interacts with the *License Management Service* and the *Storage Service*. The service also communicates with the external *Accounting and Billing Service*. The *SLA and Negotiation Service* interacts with the *License Management Service* and the *Storage Service* and with the external *Orchestration Service* and *Accounting and Billing Service*.

The **Persistency Layer** provides a single service: the *Storage Service*, a service that stores arbitrary XML data on disk.

All artefacts of the License Service, either imported - like ISV licence keys - or produced by the License Service itself - like License Tokens, Usage Records or Service Level Agreements - which need to be stored persistently for a given time are passed to the *Storage Service*. The *Storage Service* in turn is responsible for physically storing the data on disk and to retrieve it later for further processing by components or services of the License service.

The *Storage Service* interacts with all components of the License Service but the *License Information Service* and the interfaces on the **Authentication Layer** (since it is an internal service only).

3.1 License Management Service

The License Management Service is used as a central service for license administration, license storage and scheduling as well as token processing and usage record creation. It is splitted into several components, each responsible for a specific part of the functionality. Since all components, which have a direct interaction with an actor (e.g. a administrator or another service) are implemented as separate webservices, it is possible to migrate a (sub-) set of these to another application server instance or even another host system. All other components are implemented as libraries, which are used by the webservices and other components.

For interaction with a SmartLM administrator the Admin Service is used. On the one hand this service provides a set of operations, which allow the administrator to manage the set of handled licenses, for instance add new licenses or remove existing licenses. On the other hand it is responsible for license reservation creation and management. It processes the requests and delegates the scheduling decision to the Licenses Scheduler component, then parses and returns the schedulers' result as a webservice-compatible output.

The License Scheduler is a database-driven component, which manages all license reservation related operations. It is the elementary component of the License Management Service, since it is the only component maintaining information about all licenses, their states, the current utilization of the licenses and their features, etc. Since we use a relational-database at the scheduler level, all of these informations are automatically persisted and exist during runtime and after a restart of the service (except you decide to use a In-memory database). After creating a reservation we use the client side of the Usage Record Creations component to create a new Usage Record.

During the process of adding a new or removing an existing license as well as creating a new token for a reservation, the License Management Service has to interact with the different kinds of storages, to persist the tokens and the license documents. Therefore we introduced the Resourcemanager component, which is a abstraction layer for first the local file system, where the created tokens are cached/stored, and the Storage Service, where the licenses and all other global accasible documents are stored. The Resource Manager is

responsible to find, read and write the required documents and knows how to interact with these different storage types.

3.2 SLA and Negotiation Service

The SLA and Negotiation service provides license mechanisms based on WS-Agreement/WS-Agreement Negotiation. The Service is responsible of creating Service Level Agreements as a result of a user request for a license addressed to the license server. The created SLA describes all specific conditions of the application usage the user is entitled to, e.g. application id, duration, number of processors and guarantees like the maximum cost, etc.

The WS-Agreement Negotiation protocol is used if the agreement may not be obtained in a single step (e.g. because the initial request cannot be fully satisfied) or the agreement has to be changed during lifetime.

In order to implement WS-Agreement and WS-Agreement Negotiation, the SmartLM component SLA and Negotiation Service uses the WS-Agreement Framework for Java (WSAG4J). WSAG4J implements the basic features of the WS-Agreement protocol and also the WS-Agreement Negotiation extension developed in collaboration with the GRAAP working group of the Open Grid Forum. Furthermore, it uses a number of standards in conjunction with WS-Agreement to provide a complete development framework for SLA based services.

The compatibility with external Orchestrator's also implementing WS-Agreement and Negotiation is easily accomplished and user driven co-allocation of licenses and compute resources (plus data, network, and other resources as necessary) is provided.

3.3 License Information Service

The License Information Service (LIS) is a unique point of aggregation to collect pieces of information from different sources, which are the other components of the License Service. It has been implemented as a Java Web Service with a common front-end and a modular back-end, based on Quartz (http://www.quartz-scheduler.org), a Java job scheduler, which can be embedded as a library. In fact it's possible to plug-in new modules in the back-end in order to collect data from new sources. In this way the other components which need information are decoupled from the specific SOAP message schema of the source. They are required to know just the LIS front-end interface, the name of the data source and the optional parameters of the query. The mapping between the name of the source and the back-end module is defined in the configuration properties of the LIS, which are file based and can be modified without the need to restart the service. In the same file are included some security parameters which allow the LIS to rely on the authentication and authorization framework of the License Service. Moreover all the documents, exchanged among the License Service's components, are XML based, therefore the LIS supports XQuery (http://www.w3.org/XML/Query) and

adopts Saxon (http://saxon.sourceforge.net) as XQuery engine. This is valid also for the main data source, that is the License Management Service, but in addition all the messages exchanged between LIS and License Management Service are base64 encoded.

3.4 License Administration Service

The License Administration Service (LAS) is the main contact point for the Independent Software Vendor (ISV) and the License Service administrators in general. It acts as a proxy to forward the requests to the various internal components of the License Service and to report back the responses. It has been written in Java and adopts a Web Service interface, implementing a factory pattern to deal with multiple modules, which allow the LAS to interact with the other components. In particular the License Management Service (LMS), the License Information Service (LIS) and the Policy Engine. The actions performed through the LAS interface are related to license, authorization and policy management, and resemble a CRUD (Create, Read, Update and Delete) paradigm since each of those three groups of operations is based on an XML document as information unit. Given the aforementioned loosely coupled architecture, it's possible to set up a single LAS as gateway for multiple License Services or to use two LAS instances pointing to a single License Service for redundancy. The LAS relies on the security framework of the License Service, by means of the configuration properties stored in a file. File which contains also the addresses of the other components and can be dynamically updated. Thanks to the delegation capability of the security framework, the LAS can act on behalf of the real administrator without breaking the authorization policies enforced by the Policy Engine. In fact each request keeps trace of the original user who has sent it. There are two clients that are able to interact with the LAS: a command line interface and a Web portlet (JSR286 compliant, http://jcp.org/en/jsr/detail?id=286). See below for further details.

3.5 Storage Service

The Storage Service is responsible for storing XML data permanently in files or databases. This includes e.g. individual usage record documents, licenses and SLAs.

The architecture of the service is composed by two modules: a front-end module that ensures soap interactions between the backend and the other services belonging to the license server, and the backend that implements the file-system logic.

The web-service is able to manage multiple independent storages (data bases or folders), with disparate access policies and backend types. The reason to separate the storages is to increase flexibility, allowing some critical components to manage its own storage while keeping the same interface.

Each stored document will have a unique identification, and will be hosted in a single storage. The available functionalities (both with the web-service and the library are:

- Store - to add a new document
- Get - to retrieve the document content
- Append / Update - to add or replace the content of a previously stored document
- Delete - to remove a document from permanent storage

3.6 Exception Handling

The License Service communicates with the ABS through two main interfaces, the Budget Information Service and the Price Information Service. If the ABS stops due to an internal failure the SLA and Negotiation Service will handle the exceptions thrown by one of these two components.

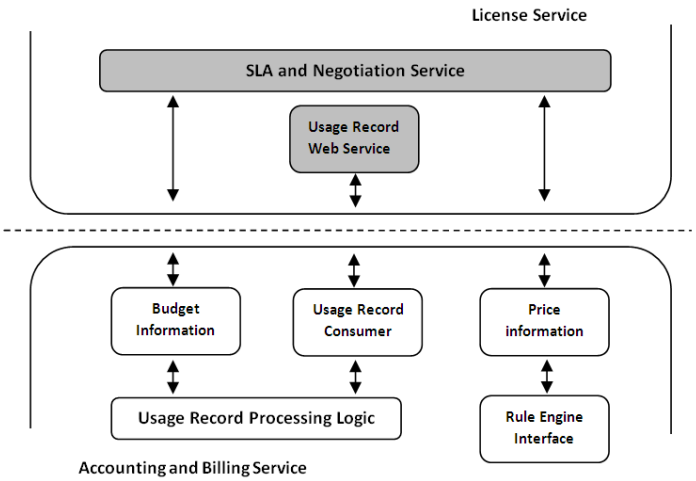


Figure 4: SmartLM exceptions handling.

3.6.1 CASE 1: No budget information available

The SLA and Negotiation Service queries the Budget Information Service in order to get the available budget of a user. If there was an error in any ABS component involved during budget calculation (UR corrupted, DB connection failed, etc.), the Budget Information Service will

throw an exception including in the error message the origin of the failure. The SLA-Service will catch the exception and check in the local policies if it is ok to ignore this error and go on with creating tokens. Otherwise the user will be notified that there was a problem retrieving the budget and that there is an error in the accounting and billing service.

3.6.2 CASE 2: No price information available

The SLA and Negotiation Service queries the price for a license execution in order to know if the current price exceeds the available budget. The Price Information Service is called and any exceptions thrown by this service will be caught. The local policies will be checked in order to know if the user is allowed to continue without an available price. If yes, the token will be created anyway. Otherwise the user will be notified that there was a problem retrieving the price and that there is an error in the accounting and billing service.

4 Accounting and Billing Service

The Accounting and Billing Service is a monolithic service integrated within the License Service Manager through web service. The Accounting and Billing Service architecture can be split into two main levels: an external layer that confers the service with the required communication within the License Service and the end users and an internal layer that allows the service to treat, send and filter information regarding the external layer requirements.

This section briefly describes the main functionalities and the main components inside the different layers depicted in Figure 2 as well as in Figure 5. As a final section an introduction about exceptions handling into the Accounting and Billing Service has been included.

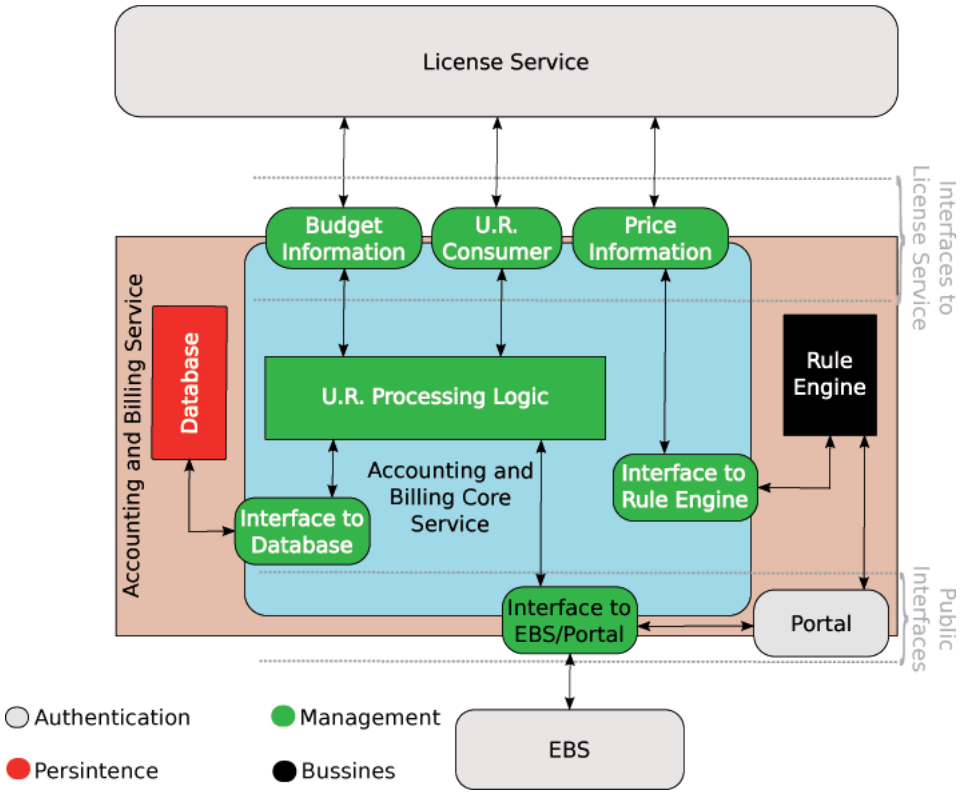


Figure 5: The layered architecture of the Accounting and Billing Service.

The **Authentication Layer** is composed by three different modules, two external to the ABS, the License Service and the External Billing Service (EBS), and one internal to the ABS, the *ABS Portal*. Users of the Accounting and Billing Service are authenticated through the Portal using a liferay-based login portlet that relies on UVOS. Once the user gets the authorisation, access to the Portal's features is allowed based in the user's role.

The **Authentication Layer** is also responsible to authenticate the communications between the License Service or an External Billing System (EBS) and the Accounting and Billing Service.

The **Persistence Layer** includes only one module, the *Database*, that stores all the Usage Records that are retrieved from the *Storage Service* inside the License Service and that contain information about licenses features usage.

Inside the **Business Layer** we find the *Budget Information Service* and the *Rule Engine Interface*. The main responsibility for the Business layer is to provide the License Service with the required licenses prices for the execution of SmartLM licensed applications.

The *Budget Information Service* provides to the License Service the consumed budget for a specific end user. This information will be used by the License Service to allow or reject the use of the requested license.

The *Rule Engine* is used to map the different prices for the different license's features and the different users. Both the administrator and the ISVs have access to this tool through the Portal and its connection to the *Rule Engine Interface*.

The **Management Layer** comprises five modules: The *Usage Record Consumer*, the *Price Information Service*, the *Usage Record Processing Logic*, the *Database interface* and the *EBS/Portal Interface*. The main responsibility of this layer is to coordinate the different components inside the ABS. Some of the main tasks are: (i) retrieval of Usage Records (URs) from the License Service, (ii) Storage of the URs into the Database, (iii) retrieval of the URs from the database based in Portal queries, (iv) retrieval of license features prices.

The **Administration Layer** hosts only one module, the *Rule Engine Interface* and it is responsible to configure the Rule engine components.

Description of the internal components and their interaction in more detail

4.1 Usage Record Processing Logic

The *Usage Record Processing Logic* is implemented as a Java library and a separate web application service that automatically starts when the application server starts.

The responsibility of the *Usage Record Processing Logic* is to publish usage records using the database interface and act as a single point of entry to the database interface. It will publish usage records to the database when it has received a specified number of notifications from the *Usage Record Consumer*. If not enough notifications have been sent for a certain time (also configurable) it will force the *Usage Record Consumer* to pass all records currently in the *Storage Service* to the *Usage Record Processing Logic* for publication. Notifications from the *Usage Record Consumer* to the *Usage Record Processing Logic* are sent through a JMS queue. This queue has to be setup before the server is started.

4.2 Database

PostgreSQL was selected as database server to store all the accounting and billing information generated by the use of the licenses. It is a database server that runs in any of the operating systems required or recommended for the project and initially, it should cover the necessary performance and scalability requirements.

The table structure uses standard SQL so, it should be easy to move it from a different database server. This SQL script is a direct translation of the SmartLM Usage Record Document.

4.3 Rule Engine

The *Rule Engine* (currently Drools is used) is responsible for price calculation for the requested licenses. The only actors who interact with it are the ISV and the ABS. The ISV will use the Business Rule Management System (BRMS) for interacting with the *Rule Engine*. Excel rules created by the ISV can be uploaded directly by using the web interface.

In SmartLM, the rule engine *Drools* is preferred, because it is easy to handle through a web interface and delivers spreadsheet support for decision tables.

Drools provides a component called *rule agent*. It is for interacting with the BRMS. The rule agent is a component which is embedded in the core runtime of the Rules Engine. To use this rule agent, no extra components are required. In fact, the only components which would be required by the ABS to interact with the Rule Engine are the Drools-core dependencies in its classpath (Drools and MVEL libraries only). The ABS however, needs to pass attributes pertaining to a particular license to the Rule Engine when firing the pricing model rules. These attributes are the ones which will be accessible in the Excel based pricing model rules which are uploaded by the ISVs into the BRMS. To make this task easier, a defined class called as LicenseDetails will be passed by the ABS to the BRMS using the rule agent.

4.4 Interfaces

4.4.1 Price Information Web Service Interface

Through this interface the license service can connect to accounting and billing components to query for a price of a given SmartLM licensed application. If the price returned raises over the available budget, the license use will be denied.

The price information is calculated using Drools, which is integrated in the SmartLM release. But it is possible to remove the drools engine and use any other rule engine to calculate the license price.

The web service is based on Axis2 and can be deployed on any servlet container like tomcat or jboss.

The service messages are secured trough the Web Service Security protocol (WSS). In particular, the project relies on the Apache implementation, called Rampart.

4.4.2 Usage Record Consumer Web Service Interface

Through the *Usage Record Consumer Web Service* usage records flow from the License Service to the Accounting and Billing Service.

The service is composed by two different modules: a front-end (Web Service) that ensures soap interactions, a backend (library) that implements a messaging system supported by the Sun Java System Message Queue.

It provides two functionalities:

- Whenever a new usage record is available to be accounted the *Usage Record Consumer* alerts the *Usage Record Processing Logic* by a notification message in a JMS queue.
- Periodically the Accounting and Billing Service needs to download all usage record documents stored in the Storage Service active repository. *Usage Record Service Consumer* allows to manage (retrieve, update, delete) the Usage Record Document in the Active Repository.

4.4.3 External Billing System (EBS) Web Service Interface

The EBS interface is a web service which connects the EBS portal to the Usage Record Service. The EBS service exposes out the accounting and billing information retreived from the database by the Usage Record Processing Logic (URPL) component. The EBS web service is developed using Apache Axis2 framework and therefore can be deployed in an application server like Tomcat or JBoss. The objective of this interface is to expose out the Accounting and Billing System information retreived by the URPL from the database and forward to the EBS portal on request.

When the web service interface gets a request from EBS for retrieving Usage Records from the database, it forwards the request to the URPL component and afterwards sends back the requested information to the EBS portal. This way the EBS interface connects an EBS System indirectly to the UR Service. The EBS web service interface provies all the required operations to retrieve the necessary information stored in the database which are generally used in accounting and billing.

4.4.4 XACML Policy Document Editing Interface

The XACML Policy Document Editing Interface (Policy Editor) is a web based java application providing the feature of creating complex xacml policy documents very easily. Policy

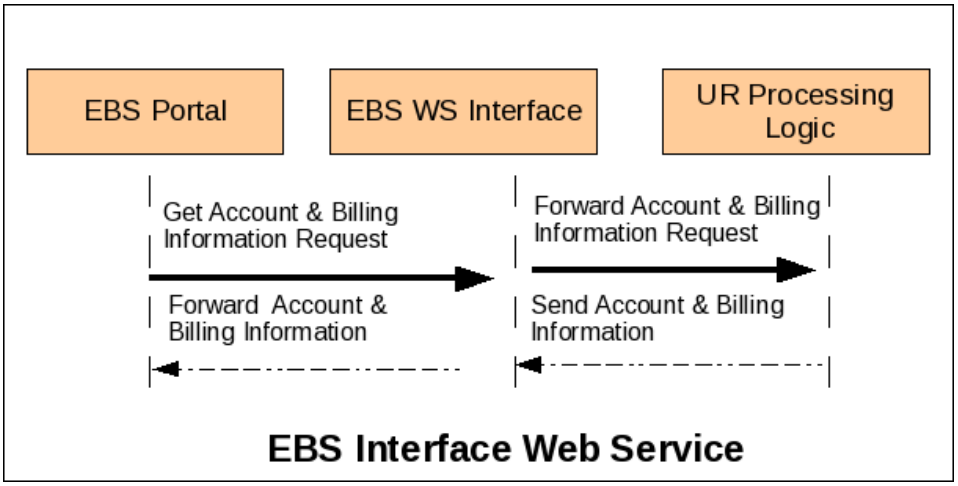


Figure 6: Communication of the External Billing System Interface.

editor is an Apache Maven web application developed by using Apache-Click framework. The policy editor web application can be deployed in an application server for example Tomcat or JBoss.

The policy document can be created by just a few clicks in the graphical document editor shown in the portal. The policy editor takes care of the xacml document to be well formed and valid. The editor also provides tips and hints to proceed with creating the document. When the policy document is generated, the document can be viewed instantly directly in the browser or can be copied or downloaded to the local disk from the browser.

4.4.5 Budget Information Web Service Interface

The Budget Information Web Service Interface is the interface between the license service and the accounting and billing service.

The purpose of this interface is to provide the available budget of a user group (project/department/company etc. are managed as user groups) to the license service. This information is needed in order to calculate if the current license request would exceed the available budget. As a result the server will reject or allow the use of the license.

The interface uses the *Usage Record Processing Logic* which calculates the used budget of a user group within a time frame.

It is implemented as a web-service based on Axis2. The service can be deployed on any servlet container like tomcat or jboss.

The security implemented uses UVOS for authentication and XACMLight for authorisation. The service messages are secured trough the Web Service Security protocol (WSS). In particular, the project relies on the Apache implementation, called Rampart.

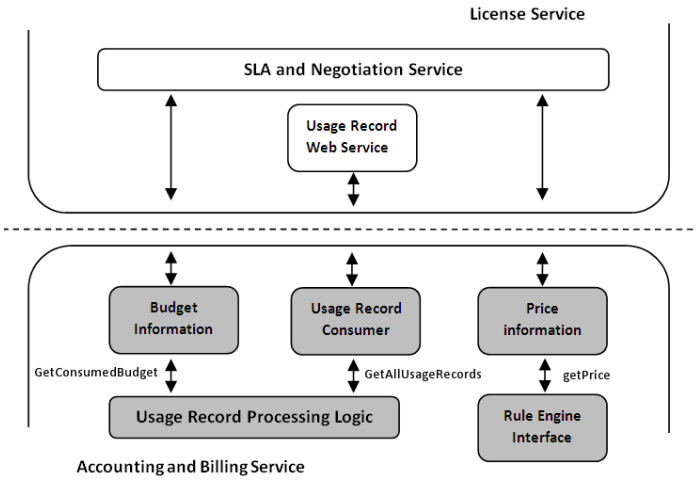


Figure 7: Exception handling of the Accounting and Billing Service.

4.4.6 Portal

The Portal was implemented as a set of Java Portlets (JSR-286) working in a personalized distribution of the Liferay Portal. In concrete two Portlets where implemented, the Query Portlet, that has built-in the necessary functionality to construct queries to retrieve the accounting and billing information from the database. The other Portlet is the Result Portlet that gets from the Query Portlet the usage information stored in the accounting and billing database. The Result Portlet formats this information in a human readable way, giving to the user the option to save it to a file.

The Portal does not access directly to the Accounting and Billing Database. It loads its information using the Interface to External Billing System/Portal.

4.5 Exception Handling

To briefly explain how exceptions are handled between the Accouting and Billing Service and the License Service, we will use the following schema with the interfaces and modules involved.

4.5.1 CASE 1: ABS stopped due to an internal problem

The License Service communicates with the ABS through two main interfaces, the Budget Information Service and the Price Information Service. If the ABS stops due to an internal failure the License Service will get a notification through one of these two components.

- Budget information service: This web service calls the URPL asking for the consumed budget every time a token is created. If there was an error inside the ABS (UR corrupted, DB connection failed, etc.), the URPL will launch an exception including in the error message, the origin of the failure. This exception may be launched by the Budget Information Service and treated a step farther by the SLA-Negotiation Service, as explained in the License Service section above.

- Price Information Service: This web service calls the Rule Engine interface to get the license features prices. If the Rule Engine interface does not answer or if it launches an exception because of a failure into the Rule Engine, it will be the Price Interface Service that will launch it a step further and the SLA-Negotiation service the component that will treat it. The Price Information Service will additionally write a proper message into the log file and it will send an email including the error found to the administrator.

4.5.2 CASE 2: No answer from the License Service

Once there is a budget request from the Budget Information Service, the URPL has to retry all the URs from the Storage Service. In that communication process if the URPL gets an exception from the Usage Record Consumer, the URPL will catch that exception and the ABS will stop working.

NOTE: apart for the communication and exceptions handling between the License Service and the Accounting and Billing Service, internally within the ABS, an email to the administrator and a proper error message into the log file are created every time the URPL gets an exception from any of the modules connected to it.

5 Orchestration Service

The Orchestrator provides automated mechanisms to support the user in co-allocating computational resources and the licenses both in time and space, allowing to use remote Grid resources for the execution of an application which needs a software license at run time, while being sure that the license required is available at the remote site when the application will start-up and during execution.

The availability of computational resources and licenses for a same time period is achieved through negotiation with the resource management systems and the license management service. If successful, the negotiation results in both the reservation of computational resources and the licenses required for the application execution. The token is created by the License Management Service and transferred to the Orchestrator, which in turn sends the token together with the user's job to the reserved computational resource. Service Level Agreements are used to describe the license terms, reservations and the corresponding guarantees.

The orchestrator client, an extended version of UNICORE rich client, is used to steer the license and computing resources negotiation, computing resources selection, data staging, and job submission and management.

Once the terms of license usage have been negotiated successfully the license token is created by the License Management Service and transferred to the Orchestrator, which in turn sends the token together with the user's job to the reserved computational resource.

6 API

SmartLM have designed a token enforcement API, which is the interface between the applications and SmartLM license system, therefore his purpose is to allow the licensed applications to access the relevant token information. The API is responsible to verify the certified documents (tokens) provided by the License Server (LS) and ensure that the above mentioned documents are validated and their contents evaluated.

The validation processes made by the API are decomposed bellow in three steps:

- Validity period verification: the token holds information about the start and end date when these documents are allowed to be used. Each part of the token has its own validity period, one for the token itself; other for the Authorization inside the token and each of these have associated an X509 certificate which has their own validity period.
- Integrity verification: the integrity of the documents is done using double signature verification. On the one hand, the API verifies that the token is signed by a valid SmartLM server instance. On the other hand, the second signature guaranties that the issuer SmartLM server is authorized by the appropriate ISV. In addition there is an additional verification; the subject of the authorization must be the license server which issued the token.
- Hash binding verification: the information in the token is structured in two levels, firstly the application features and values, finally the second level hold extra attributes for the features. The API is in charge to reject unauthorized usage of features and the associated files using the included hashes in the signed token.

SmartLM offers APIs for the C and Java programming languages. Additionally, there is a wrapper around the C API available for the FORTRAN programming language.

7 Conclusions

This whitepaper gives an overview of the global SmartLM architecture and technology. We pointed out how SmartLM developed a generic and flexible licensing virtualization technology based on standards and new service-oriented business models, through the implementation of software licenses as Grid services.

SmartLM overcomes the actual limitations of existing software license management solutions making the use of license protected applications on resources outside of the own administrative domain as easy as running them locally.

License usage is authorised through negotiation of Service Level Agreements between the license service and the user taking into account availability, policies defined locally, policies of the ISV or attributes defined for user in a virtual organisation. The integration of an Accounting and Billing System allows price determination and budget control when the license is requested. Aspects of security have also been examined through the implementation of a number of sophisticated, state-of-the-art security mechanisms that render illegal use almost impossible.

8 Acronyms

Acronym	Long Form
AA	Authentication and Authorisation / Assertion Authority
ADI	Administrator Interface
API	Application Programming Interface
ASP	Application Service Provider
CA	Certificate Authority
CAS	Central Authorisation Service
CLI	Command Line Interface
CPU	Central Processing Unit
CSR	Certificate Signing Request
DRM	Distributed Resource Manager
HPC	High Performance Computing
IANA	Internet Assigned Numbers Authority
ICT	Information and Communication Technologies
IDB	Incarnation Database
IP	Internet Protocol
IPR	Intellectual Property Rights
ISV	Independent Software Vendor
IT	Information Technologies
JDBC	Java DataBase Connector
JKS	Java KeyStore
JPA	Java Persistence API
LAN	Local Area Network
LAS	License Administration Service
LMS	License Management Service
LS	License Service
LSDL	License Submission Description Language
LSP	License Service Provider
OGF	Open Grid Forum
ORM	Object-Relational-Mapping
OSS	Open Source Software
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKI	Public Key Infrastructure
QoS	Quality of Service

Acronym	Long Form
RMS	Resource Management System
ROI	Return of Interest
SaaS	Software as a Service
SAML	Security Assertion Mark-up Language
SHA	Secure Hash Algorithm
SLA	Service Level Agreement
SME	Small and Medium Enterprises
SOA	Service Oriented Architectures
TCP	Transmission Control Protocol
UNICORE	Uniform Interface for Computing Resources
URL	Uniform Resource Locator
URPL	Usage Record Processing Logic
UVOS	UNICORE Virtual Organisation Service
VAR	Value Added Reseller
VO	Virtual Organisation
VOMS	VOMS is an acronym used for Virtual Organisation Membership Service in Grid computing
W3C	World Wide Web Consortium
WAN	Wide Area Network
WS Agreement	Web Services Agreement
X.509	A standard for Public-Key Infrastructures
XACML	eXtensible Access Control Mark-up Language
XML	Extensible Mark-up Language

9 Glossary

Term	Role	Description
Application Service Provider (ASP)		An *ASP* provides the use of its applications over the network. This software is hosted on central servers of the provider. The *ASP* also hosts the *license server* from which *users* can rent out *licenses* for use on the hardware resource (for fault-tolerance reasons multiple redundant license servers may be hosted).
Computational resource		A cluster, supercomputer or simple computer that is accessible through a local *Resource Management System (RMS)* or through a Grid middleware.
Distributed Resource Manager (DRM)		A species of a *resource management system* often used in multi-cluster environments. The *DRM* e.g. Sun Grid Engine, Windows CCs, is responsible for dispatching user jobs based on the resources requested to the right cluster nodes in the *High Performance Computing (HPC)* resource for execution.
SmartLM	License manager	*SmartLM* manages all licenses owned by a site. The SmartLM software includes the *license service*, accounting & billing service, orchestration service, accounting data storage & license data storage.
Feature		A part of an application which can be licensed separately.
Grid Orchestrator		When a *user* submits his job to the *Grid orchestrator*, the orchestrator cares for where and when the job is executed, allocates the *computing resources* and *licenses* for the execution.
HPC resource		A *computational resource* suitable for high performance computing. May be operated under different Operating Systems. The system may be accessible from outside the site that is hosting the *HPC resource*, but access from outside may also be restricted to a Web front-end or any other front-end (login) system usually not located in the same network as the *HPC resource*.
Independent Software Vendor (ISV)	License owner	An *ISV* is the owner and vendor of a *licensed application*.

Term	Role	Description
License Administration Service (LAS)		Through the License Administration Service all licenses of a site are managed, e.g. new ones added to the pool of licenses or outdated ones removed from the pool.
License		Right to use a *license protected software*. The license usually includes further rules under which the software might be used, e.g. restrictions with respect to the execution environment, the executing *user*, etc. the license may be spawned from a pool of licenses, and the rules of individual license issuing are governed by a *license contract framework*.
License contract framework		The contract between a licenser, usually the owner of the copyright, and a licensee, usually an organisation that buys the right to use the copyrighted software. The *license contract framework* sets the general rules and conditions of software usage and usually also the general cost agreement.
License protected software		A software (usually protected by copyrights) where the owner of the copyright grants the right to use the software through a *license*.
License server		A software process running at the site of the *user* controlling the use of one or *more license protected software* systems. The *license server* acts as license provider in the *negotiation* process when a *user* requests a *license* for using *license protected software*.
License service		Hosted by the *license server*, delivering the schedulable license from the *license server* to the end-user. If the *license protected software* is executed using local resources there might be a communication channel between the application and the *license service* during run-time. In case of using remote *computational resources* usually no communication channel exists.
License system administrator		The *license system administrator* manages the local *license server*, e.g. is responsible for loading license *features* of various *ISVs* onto the *elsticLM license server*, handling license *feature* renewals and controlling access to individual license *features* hosted on the *elsticLM license server*.
License virtualisation		The process of creating and probably reserving a schedulable license from the *license server*.
Licensed software		Short for *license protected software*.

Term	Role	Description
Negotiation		The process to agree on the terms of usage of a *license protected software*. Resulting in a *Service Level Agreement* if successfully completed.
Resource broker		The *resource broker* has to choose a site where all the available resources (compute, network, licenses, etc.) fit best to accomplish the job.
Resource Management system (RMS)		The local *RMS* (usually a batch queuing system like PBS-pro), which is responsible for dispatching user jobs to the local *computational resource*.
Service Level Agreement (SLA)		A template setting the terms of software usage the two parties agree upon. This includes the constraints of using the software, e.g. duration, user and may include guarantee terms and penalties for both parties.
Service provider		Short for *Application Service Provider*.
Simulation engineer	End-user	The *simulation engineer* starts the simulation software either from his working environment or submits the application to a Grid *computational resource*.
Software developer	License owner	A person acting as an *Independent Software Vendor (ISV)*.
User	End-user	A *user* who uses a *license protected* (SmartLM enabled) *software*. The end-user may come from different environments, with or without *computational resources* for the execution of her application, e.g. being a customer of an ASP, a member of a research institute, or an employee of a company. End-user is a synonym of user.
User-Group	End-user	User-Groups may be defined using attributes (e.g. defined in a Virtual Organisation) or simply through their relation to an organisation or company. User-groups may receive a dedicated (probably restricted) set of licenses from an ASPs license server. User-groups are also supported for environments without ASP.

D2.1 Software Licensing

Panorama Today

Version 1.0

WP 2 New Business Models

Dissemination Level: Confidential

Lead Editor: Csilla Zsigri (451 Group)

27/06/2008

Status: Final

Seventh Framework Programme of the European Community

Proposal/Contract no.: 216759

Abbreviations

ASP	Application Service Provider
COTS	Commercial, off-the-shelf
CPU	Central Processing Unit
EDA	Electronic Design Automation
ISV	Independent Software Vendor
HaaS	Hardware as a service
HPC	High Performance Computing
ICT	Information and Communication Technologies
IT	Information Technologies
LAN	Local Area Network
LSP	License Service Provider
Q	Question
SaaS	Software as a service
SmartLM	Grid-friendly software licensing for location independent application execution
SOA	Service Oriented Architecture
WAN	Wide Area Network
WP	Work package

1. Executive Summary

In the context of current IT trends, the inadequacy of the existing licensing mechanisms have become evident especially in their geographical restrictions and pricing models. Current licensing mechanisms are not just difficult to adapt to new trends but, in fact, are hindering the wider adoption of these technologies. Software licensing has been identified as a particular concern for users as for the potential benefits of Grid-like technologies (Grids, virtualization, multi-core, clouds, etc.).

The software industry employs the ´right to use´ model that ensures that the ownership and control of software usage remains in hands of the vendors. This right to use is granted through a license agreement. This license agreement, with the help of a license enforcement mechanism, should ensure the protection of intellectual property and result in license compliance by tracking and managing the licenses in use.

With respect to these licenses, what users want is to control their expenses and have flexibility, while vendors do not vote for a reduction in revenue. The achievement of a consensus, a win-win situation between software vendors and users is the main requirement for a change. For now, users either pay up or find a workaround.

The main players of the software licensing playground are the Independent Software Vendors (ISVs) who own the software and provide licenses for its use, the End Users who use the software in their local or external environments, and the Application Service Providers (ASPs) who provide access to an application over a network. Depending on the relations between these players, we have identified four main licensing scenarios that exist currently. The ASPs are not always part of the licensing scenarios. Nevertheless, they can often offer a good cost-to-performance ratio for on-demand hardware resources for End Users and bring additional business to ISVs.

When it comes to license compliance, there is insecurity at user organizations regarding what a license audit might uncover. For the moment, the most widely used license enforcement product is Acresso Software´s FLEXlm (from version 10.0 it is called FLEXnet) and Platform Computing has recently come up with its Licensing Scheduler, that is a complementary product to FLEXlm, enhancing its functionalities. The future SmartLM product comes to the picture here, offering enhanced capabilities and functionalities that would supersede or add value to current solutions.

From a technical point of view most applications are licensed by client-server license technology or hardware dongles. In addition, other used models are the node-locked licenses that are 'per named node' license models and score-based licenses where the different functionalities of the software have score values and each user can use all functionalities up to a total score. Open source solutions are also popular among users. If Grid users can find an application that suits their needs and that does not require a cumbersome license, they will use it. Software vendors can´t ignore the open source movement. Hybrid models offer many advantages, defining the line between what is free and what is paid for, is the critical component of any open source strategy.

As of pricing, users usually pay for an annual one-time license and an 18-25% maintenance fee (but there is an increasing demand for the ´pay per use´ option for overloads). The most widely used pricing structures are per-seat (system, server), per-CPU and per-concurrent-user models. Conventional per-CPU, per-seat and per-job license management models and pricing structures are problematic and quite expensive for users seeking to run commercial applications on top of Grid, web services or similar technologies, so custom-contract based models have become the

opted solution. Now, vendors are more and more heading to a customer value-oriented pricing model based on (and limited to) reliable metrics in HPC and Grid. Pure utility pricing is still a complex issue since there is no one clear utility metric that is of universal value to customers. However, some vendors implement utility pricing on a case by case basis.

As an important input for this report and a starting point for the next one about new business models for software licensing, we have briefed several large and small software vendors, application service providers and end user organizations to help us understand the current licensing practices and market needs. Different business sectors were covered, such as IT industries, mechanical industries (automotive, aerospace), electronics, financial service industries and media. The findings of these questionnaires are used all along the report, nevertheless, sections 5 and 6 explicitly provide a summary of the information obtained for each of the questions. As a general observation, the answers obtained were highly diverse and sometimes even contradictory. Often there is a mix of licensing and pricing models used that makes the panorama even more confusing. The situation also varies depending on the industry vertical we have under analysis. What we can say is that there are some basic licensing rules that can be considered as general on the market.

Software licensing is both a technology and a business concern, but the business issues are the most problematic ones. Licensing must offer reliability and flexibility to the user and its cost must be low compared to the value of the license. Customers want flexibility and they want vendors they can partner with. Vendors do not vote for a reduction in revenue, however, there must be some additional value to the user, such as being able to move a license around. So, what is expected is an evolution, more than a revolution. Where major vendors go, the market will follow. Models supporting Grid-like technologies might vary, but some form of measured usage will likely be employed.

This report is the basis for deliverable D2.2 about new licensing scenarios and related business models, and also an important part of the commercial exploitation of smartLM.

D2.2

Business impact on the adoption of Grid licensing mechanisms and suggested business models

Version 1.0

WP 2 New Business Models

Dissemination Level: Confidential

Lead Editor: Csilla Zsigri 451Group

24/10/2008

Status: Final

Seventh Framework Programme of the European Community

Information Society

Proposal/Contract no.: 216759

Abbreviations

API	Application Programming Interface
ASP	Application Service Provider
CC	Consulting company
CPU	Central Processing Unit
D1.1	SmartLM Deliverable 1.1
D2.1	SmartLM Deliverable 2.1
DSW	Dependant Software Vendor
FC	Final customer
ISV	Independent Software Vendor
HWP	Hardware Provider
HPC	High Performance Computing
IT	Information Technologies
IP	Internet Protocol
IPR	Intellectual Property Rights
LAN	Local Area Network
LSP	License Service Provider
MPI	Message Passing Interface
OSS	Open Source Software
SaaS	Software as a service
SmartLM	Grid-friendly software licensing for location independent application execution
UC	Use case
VAR	Value Added Reseller
WAN	Wide Area Network
WP	Work package

1. Executive Summary

Traditional licensing practices are under pressure from a variety of alternative options (SaaS, open source, low-cost development environments, Chinese software companies, etc.) and are tightening vendors´ profit margins, pushing down licensing costs and giving more negotiating power to users.

Licensors may expect licensees to buy additional licenses for each processor that executes the licensed software (multiplied software fees). This is definitely not viable in a Grid or Cloud context. Why should users pay many times for the same software? It is clear that new usage-based models are needed where customers are charged for useful, measurable units, relevant to the software they are using, and are allowed to flexibly select the best model for their environment.

Naturally we can´t disregard the fact that software vendors´ will fight tooth and nail in order to maintain (and increase) their revenue, so we won´t revolutionize the entire software licensing market, but we can try to fill in some gaps and improve current conditions favouring Grid-like technologies and trying to reach a win-win situation between vendors and users.

We think that the role of the ASP can be very important from both angles (vendors´ and users´), a role that is evolving and expanding, while bringing along solutions to specific shortcomings, reselling hardware, software and services. We have identified five cases that companies may most frequently encounter in real operations.

When customers do not want to fight for budget to acquire new hardware, they can choose to transfer their private application license through a secure mechanism to the environment of the ASP, while still physically owning what they paid for but with the capability to dynamically reassign the usage grant. Another case is, when the final customer needs special software for a specific task. This situation calls for the appearance of a DSV (Dependant Software Vendor) who embeds the ISV´s license in its new template (creating new special software) and commercializes it through the ASP, in a way that the ISV trust the DSV and be assured that the DSV can access only those parts of the application for which they are granted access. In the third case, a consultant owns a license and deploys it at the ASP for the realization of a specific project for the final customer. The original license needs to be re-directed. In the fourth ASP case, the customer license housing- is enhanced in a way that it allow users to complement the licenses they have bought already from the ISV with additional licenses from the ASP. Furthermore, it may also happen that the ASP simply resells the ISV´s license for external use, in some cases providing different contract terms. This generates additional business to the ISV. It can occur because the ISV wants to reduce the number of its direct accounts, or wants to reduce the risk of non-payment, among others. Also, small ISVs can access the market easier, this way, with the confidence to host their applications in an ASP environment without having to worry about losing their intellectual property. Making use of economies of scale and SmartLM features, the ASP can make existing models attractive to end users as well as introduce new ones that ISVs are not willing to offer.

In addition to the ASP scenarios, we propose further business models that may or may not involve the ASP, but they do fill in current gaps in software licensing.

In our second scenario, the proposed license extension model enables end users to use their rented licenses with additional pay-per-use licenses in their local environments. This business model allows users to extend their licenses in time, in their Local Area Network (LAN) on

demand (e.g. for workload peaks). The new SmartLM licensing service addresses the weaknesses of the existing processes for license extension. The License Server takes care of the extension of the license automatically. It enables a budget control strategy that limits the total costs but allows a flexible extension of licenses under specific circumstances. Security mechanisms are implemented based on high level security web standards. SmartLM also takes care of the 'breakeven point' issue, accounting the point when it is cheaper for a customer to rent an additional license instead of paying for an additional one.

One of the major problems today is that most contracts between ISVs and end users limit the license usage to LAN. It is obvious that this approach contradicts the Grid idea as this way we are not able to use licenses from different locations. Naturally, a worldwide acting company needs licenses that can be dynamically used all over the world. SmartLM aims at overcoming the legal and technical limitations of current licensing mechanisms and show the benefits for all involved parties. We call this model ´license aggregation´, a model that we also mention among the ASP scenarios, but from a different perspective. The correct identification of the end user´s location and the license source is critical for this new business model.

In our last scenario we try to overcome a major paradigm in software licensing. With this new approach, the license price becomes effectively independent of the underlying hardware. It is a pricing/accounting model that specifies a way to charge for the use of a license. Within the currently established paradigm, on-demand licensing would benefit those with faster hardware (as for a given amount of time, prices are fixed in advance). What we propose is to fix a price on a linear scale against a pre-defined reference point – the profiling system. The profiling is done by the ISV to formulate the application performance in terms of micro-benchmarks. The reference price model is negotiated between end user and ISV, DSV or ASP. The final price will always be a weighted reference price. This way, vendors get their revenue by charging for the real use of their applications while offering more fair conditions to the users enabling a cost-efficient usage of the sold licenses. Based on the initial micro-benchmark idea, we go further and define a more general approach called feature-based accounting.

As said before, we don´t offer a revolution here, but certainly an evolution –in some cases quite disruptive- of the current situation. For this, we have taken into account the different requirements of the parties involved. In the context of current technology and market trends, we try to offer flexibility and fair conditions to the users without forcing the vendors to reduce their revenue.

D3.1

Design of the license service, specification of WS-Negotiation protocol

Version 1.0

WP 3 - Implementation of Basic Technology and Security

Dissemination Level: Restricted to Programme Participants

Lead Editor: Wolfgang Ziegler, Fraunhofer SCAI

24/10/2008

Status: Final

Seventh Framework Programme of the European Community

Information Society

Proposal/Contract no.: 216759

Acronyms

ADI	Administrator Interface
API	Application Programming Interface
ARS	Accounting Record Service
ASP	Application Service Provider
CAS	Globus Community Authorization Service
DRM	Distributed Resource Manager
GRAAP-WG	Grid Resource Allocation Agreement Protocol Working Group
IM	Instant Messenger
ISV	Independent Software Vendor
LIS	License Information Service
OASIS	Advancing open standards for the information society
OGF	Open Grid Forum
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PKI	Public Key Infrastructure
PMI	Privilege Management Infrastructure
SAML	Security Assertion Mark-up Language
SLA	Service Level Agreement
SmartLM:	Grid-friendly software licensing for location independent application execution
SOA	Service Oriented Architecture
UNICORE	Uniform Interface to Computing Resources
UVOS	UNICORE Virtual Organization Service
VO	Virtual Organization

VOMS	VOMS is an acronym used for Virtual Organization Membership Service in grid computing.
WS Agreement	Web Services Agreement
WSDL	Web Service Description Language
WSRF	OASIS Web Services Resource Framework
X.509	Public-Key Infrastructure
XACML	eXtensible Access Control Mark-up Language
XKMS	XML Key Management Specification
XML	Extensible Mark-up Language

1. Summary

The objective of this deliverable is twofold:

1. To present architecture and design of the SmartLM license service,
2. To describe the WS-Agreement-Negotiation protocol.

Accordingly, the document is structured in two major parts: Section 2 focuses on the license service while section 3 addresses the negotiation protocol.

Architecture and design of the SmartLM license service are based on three sources:

1. The outcome of work package 1 described in D1.1 [1], namely functional requirements and use-cases.
2. The results of work package 2 described in D2.1 [2].
3. State-of-the-art in web-service technologies.

Gathering of the requirements was driven by technical and functional aspects as well as market perspectives and business models, which are elicited in WP2. The business relevance of the requirements was analyzed and rated in order to distinguish between State of the Art features and new ones, which are either required or optional for new business models.

For the initial prototype of the license service, non-functional requirements will not be taken into account except for security requirements.

The negotiation protocol has been specified in close collaboration with the Grid Resource Allocation Agreement Protocol working group (GRAAP-WG) [3] of the Open Grid Forum (OGF) [4]. The protocol currently described is in the process of becoming a proposed recommendation of the OGF. Members of the SmartLM project actively contribute to this process participating in the GRAAP-WG of the OGF. The collaboration within the GRAAP-WG ensures that the requirements of the SmartLM project find their way into the OGF specification of the WS-Agreement-Negotiation protocol.

D4.1

Process-Model for Acounting

Version 1.0

WP 4 Implementation of Accounting and Billing

Dissemination Level: Confidential

Lead Editor: Christian Simmendinger, T-Systems SfR

Status: Final

Seventh Framework Programme of the European Community

Information Society

Proposal/Contract no.: 216759

Abbreviations

AA	Assertion Authority
API	Application Programming Interface
BES	Basic Execution Service
CAS	Community Authorization Framework
CIM	Common Information Model
CIS	Credential Issuing Service
CVS	Credential Validation Service
DMTF	Distributed Management Task Force
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve DSA
GLUE	Grid Laboratory Uniform Environment
ISV	Independent Software Vendor
JDD	Job Description Document
JSDL	Job Submission Description Language
JSR	Java Specification Request
LRM	Local Resource Manager
MD5	Message Digest algorithm 5
OASIS	Organization for the Advancement of Structured Information Standards
OGF	Open Grid Forum
OGSA	Open Grid Service Architecture
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
RSA	Rivest, Shamir, Adleman
SAM	Software Asset Management

SAML	Security Assertion Mark-up Language
SHA	Secure Hash Algorithm
SP	Service Provider
SPL	Simple Policy Language
UVOS	UNICORE Virtual Organization Service
VOMS	VOMS is an acronym used for Virtual Organization Membership Service in grid computing
W3C	World Wide Web Consortium
WSN	Web Service Notification
WSRF	Web Service Resource Framework
X.509	A standard for Public-Key Infrastructures
XACML	eXtensible Access Control Mark-up Language

1. Summary

This deliverable addresses the process-model for accounting and billing. It is structured in four main parts. The first chapter gives a general overview of the accounting and billing architecture. Two main scenarios are discussed. The second chapter details the actual accounting process. SmartLM will be a token-based system. The accounting process hence will regard the token both as a record and also as a symbolical receipt with an associated price. The pricing models and the associated billing then will be discussed in chapter three. The final chapter discusses the interface to the accounting and billing systems. A portal-based aggregation over the respective license services will be employed. The document concludes with a summary.

D6.1

Scenarios, Criteria and Methodology

Version 1.0

WP.6 Evaluation

Dissemination Level: Confidential

Lead Editor: Henning Eickenbusch, ANSYS Germany GmbH

28/01/2009

Status: Approved by QM

Seventh Framework Programme of the European Community

Information Society

Proposal/Contract no.: 216759

Acronyms

ASP	Application Service Provider
DRM	Distributed Resource Manager
HPC	High Performance Computing
ISV	Independent Software Vendor
LAN	Local Area Network
MTTF	Mean Time To Failure
SmartLM	:Grid-friendly software licensing for location independent application execution
WP	Work Package

1. Executive Summary

The purpose of this deliverable is the description of the evaluation scenarios, criteria and methodology for the SmartLM software.

SmartLM is designed as licensing software tool of the future. It covers the functionality of different existing software licensing tools as well as the requirements of increasing grid deployment. These requirements are improved security identification, security authentication, security authorization and security integrity, integration into Grid and Cloud environments and an automatic accounting and billing interface. The requirements for SmartLM are described in D1.1 [1]. These requirements allow the adoption of new business models in the growing market of Grid and Cloud environments as described in D2.2 [2].

Chapter 2 gives an introduction to WP6 "Evaluation" of the SmartLM project, its connection to WP1 to WP5 and a first outlook to the tested requirements and the content of the document. To check, if the complex functional requirements are achieved, five different evaluation scenarios were developed in chapter 3 on the basis of the used cases defined in D1.1 [1]. The evaluation testbeds cover local environments as well as European Grid environments.

Chapter 4 lists the important additional non-functional requirements like usability, portability and security. Chapter 5 describes the evaluation methods how to check these requirements. The content of chapter 3 to chapter 5 results in Annex A. Here all evaluation platforms and evaluation tests are specified in detail.

The clear description of tasks, responsibilities of different project partners and time schedules in chapter 6 ensures the observance of the development plan given in the DoW [3].

In the future the content of this deliverable is used to perform the following tasks of WP6:

- Task 6.2: Evaluation testbed set-up and tools development (responsible CESGA)
- Task 6.3: Local evaluation (responsible T-Systems)
- Task 6.4: Final evaluation (responsible Gridcore)

If SmartLM passes all the tests and criteria defined in this deliverable, SmartLM shows its potential as licensing software tool of the future.

This 1st version of D6.1 is delivered to the European Commission by end of January 2009. This document will see changes after the SmartLM integration meeting end of January 2009. During this meeting the SmartLM developers put together the single components into a first version. The 2nd version of D6.1 is scheduled for end of April 2009.

D7.3

Product description and Market context analysis for exploitation

Version 1.0

WP.7 Dissemination and Exploitation

Dissemination Level: Confidential

Lead Editor: Belén Serrabou, Atos Origin

30/01/2009

Status: Final

Seventh Framework Programme of the European Community

Information Society

Proposal/Contract no.: 216759

Abbreviations

ASP	Application Service Provider
CAD	Computer-Aided Design
CPU	Central Processing Unit
ISV	Independent Software Vendor
IT	Information Technologies
LAN	Local Area Network
ROI	Return On Investment
SaaS	Software as a service
SmartLM	Grid-friendly software licensing for location independent application execution
SME	Small and Medium Enterprise
SOA	Service Oriented Architecture
SWOT	Strengths, Weaknesses, Opportunities, Threats
WAN	Wide Area Network
WP	Work Package

1. Executive Summary

The purpose of this deliverable is to provide an initial market study and product analysis for the future exploitation of the SmartLM product.

In the context of current IT trends, existing software licensing mechanisms have become inadequate, especially in their geographical restrictions and pricing models. Traditional software licensing mechanisms present limitations for users and especially for Grid and Cloud environments. Software manufacturers need to change the way licensing works and use non-hardware based licensing solutions that can work more easily within a virtual environment. Software as a Service, where the final users only pay for what they use, has been around for a few years now, we can say with certainty that it has already gone mainstream.

Since software vendors do not bet for a reduction in their revenue, a win-win situation with vendors and users must be achieved to obtain exploitation success. The SmartLM product overcomes current limitations of the market and provides manifold benefits for all parties involved.

A market analysis has been carried out to identify potential target markets for our product. In mechanical industries, the simulation software market is dominated by several big independent software vendors. The business model of these vendors is based on selling licenses. The licensed software of these independent software vendors is in general protected by software/hardware licensing products. Besides the large companies, many medium and small software vendors need license enforcement software too. The number of users in companies varies from a single user to hundreds of users. What vendors most commonly sell are annual licenses. Most of the times, the software is used in in-house machines and clusters, not in external Grid environments. Access to Grid environments is not yet automated. When Grid environments are involved, ASPs usually have to guarantee access to specific hardware.

In the Telecom sector there are two main scenarios that are worth differentiating: internal services and customer services. Internal services, due to its special characteristics, is not a suitable target sector for SmartLM. On the contrary, customer services is an area that SmartLM should definitely take into account. In the areas as marketing, technical support, administration or computational resources, the telecom companies will play the major role.

The Financial sector has experienced a tremendous change in the last decade and many financial institutions have relied on Grid technologies as well as Service-Oriented Architectures. Many times they develop their own software but for specific activities, they use commercial applications, where they find themselves in front of a pricing restriction when they need to buy a license for every device in the Grid. Software vendors use traditional models where the user is bound to a large contract. In the financial institutions it is common to have workload peaks, and with traditional models, they are forced to pay up during the entire year. A solution as SmartLM solves these limitations to efficiently run commercial applications in Grid environments and boost the usage of SaaS in this sector.

The introduction of a new license management service as SmartLM could be very beneficial in the Digital Media sector too. Companies in this market usually have workloads that vary relevantly throughout the year, and a flexible model that allows them to add or remove licences on the fly could help them reduce costs. At the same time, SmartLM could help them monitor who, when and where has access to the digital content.

Grid technologies in the Pharmaceutical sector allow researchers to access external data. In this case, highly secure mechanisms are a must, as data represents intellectual property. The current

license models are floating or node locked licenses that are mostly limited to the companies' local networks. There is demand for pay-per-use license management to enable and optimise the usage of software worldwide, across networks.

SmartLM aims at rendering mechanisms for managing and using software licenses in a more flexible and fair way. SmartLM provides multiple improvements versus the current situation as SmartLM licenses may be used seamlessly in cluster environments, as well as in local or remote Grid, Cloud and SOA environments. SmartLM allows the definition of local policies for license usage on top of the embedded policies and provides usage information with the help of a 100% trustworthy accounting mechanism. SmartLM realises a number of sophisticated state-of-the-art security mechanisms. It also offers reservation and re-negotiation mechanisms, among others. There are advantages for all the parties involved, independent software vendors, computing centres, application service providers and end users.

A SWOT analysis has been carried out to reveal important internal and external factors of the product. The main strength is the manifold features that enable to manage licenses in the IT trends environments. But it will be a new unknown product in the market which will have to gain the market. A clear opportunity is that there is an increasing use of Grid and Cloud environments and there is an unfulfilled customer need. A threat to take into account is the emergence of substitute products.

There are other similar products to SmartLM, as FlexNET, Sentinel RMS, Reprise License Manager, LM-X License Manager, GenLM and others that serve a specific application and are provided by the application vendor itself. After a comparison of features, we can conclude that SmartLM outstands distinctly, with features as it supports Grid middleware, has a 100% trustworthy accounting and billing system built in, support for negotiation of license usage, support for re-negotiation of license terms at run-time, temporarily host and use ISV licenses at another site, and support pay-per use model.

SmartLM is an advanced license enforcement product with a strong selling proposition for software licensing. SmartLM covers current features of available licensing software and meets the requirements of commercial Grid and Cloud deployments like improved security, authorization, authentication, embedding and automated accounting and billing. Two perceptual maps have been drawn to show the positioning of the software licensing solutions. The first one is using security and Grid capabilities criteria and the second one is to show the positioning in local environments with multiple users. In both maps SmartLM receives a privileged position.

The multiple benefits of this new product, as we stated before, are definitely noticeable for all the players involved, however a fundamental change is needed in the way vendors sell licenses now and this is not a simple process. Nonetheless, the benefits are clear and a win-win situation can be achieved. Current IT trends definitely support a solution like SmartLM, but timing is crucial. SmartLM is a new product, its launch is not evident as there are established products in the market, so proper distribution channels and communication are very important for SmartLM to succeed.

This report is the basis for the next deliverables 7.4 and 7.5 where we will present our exploitation strategies and agreement that will result in a final exploitation plan. As soon as the project has an operable prototype, we will define a sound business strategy for the exploitation of the future SmartLM product.

D7.4

Preliminary Exploitation Plan

Version 1.0

WP 7 Dissemination and Exploitation
Dissemination Level: Confidential

Lead Editor: Belén Serrabou, Atos Origin

17 February 2010

Status: Final

Seventh Framework Programme of the European Community

Information Society

Proposal/Contract no.: 216759

Table of Contents

Abbreviations

API	Application Interface
ASP	Application Service Provider
CPU	Central Processing Unit
ISV	Independent Software Vendor
IT	Information Technologies
LAN	Local Area Network
ROI	Return On Investment
SaaS	Software as a service
SmartLM	Grid-friendly software licensing for location independent application execution
SME	Small and Medium Enterprise
SOA	Service Oriented Architecture
SWOT	Strengths, Weaknesses, Opportunities, Threats
SP	Service Provider
WP	Work Package
MEUR	Millions of Euros
CAGR	Compound Annual Growth Rate
USP	Unique Selling Point

1. Executive Summary

The purpose of this deliverable is to provide the preliminary SmartLM exploitation plan. Taking into consideration the recommendations of the last review, a comprehensive series of actions have been followed in order to maximise the exploitation potential of the project a completed exploitation plan has been executed. This deliverable is the result of many discussions during which a number of key decisions have been made within the entire consortium regarding the future exploitation of the project results.

First of all we analysed the market size. Pierre Audoin consultants sized the European Software industry in 2008 at more than 100 billion Euros and expect it to grow at a 4% CAGR[1] until 2012. In terms of software revenue models, the traditional revenue model will remain stable, in absolute terms, showing lower growth with only 2% CAGR until 2012 whilst the paid web-based model is expected to grow up to 23% CAGAR until 2012. Other analysts have placed higher expectations on this market. For instance, according to The 451 Group, the cloud market will grow at a CAGR of 68% to reach 2.9billions € in 2013. Regardless of which exact set of figures one chooses to accept, this is a market in strong expansion and there is a great opportunity to SmartLM.

We analysed the elasticLM marketing Mix: product, price, place and promotion. We have decided elasticLM as the commercial name for the future SmartLM product. elasticLM aims at rendering mechanisms for managing and using software licenses in a more fair and flexible way. The packaging consists in 3 components: License service, Accounting and billing service and Application interface, all available for download from the website. In order to accommodate our product to the client's needs, we will produce three different options to acquire elasticLM:

- Basic - no accounting and billing, no re-negotiation → for small companies, for first evaluation.
- Extended - includes accounting and billing, no renegotiation → for companies where accounting and billing is important, for productive use.
- Full - for productive use, includes accounting and billing and re-negotiation

In addition to elasticLM product, we identify some additional services that could be interesting for our customers: We can provide trained personnel to implement the solution, helping companies to integrate elasticLM into their products and business consultants with experience in SaaS and knowledge of our product to enable companies to make the most of elasticLM or even design new business models for ISV companies.

elasticLM components are not open source as this would make attacks against the elasticLM security mechanisms easier and our customers, the ISVs, could suffer from cracked license mechanisms. The project has a social impact since offer new and affordable license model for start-ups, entrepreneurs and SMEs. They will reduce large investments increasing the competitive position of European companies. It enables the SaaS model, creating new business opportunities for ISV, ASP and end user. In addition, we will release some components open source beneficially to the community.

[1] Compound annual growth rate is a business and investing specific term for the geometric mean growth rate on an annualized basis.

A penetration pricing strategy will be used to the initial stages of entering to the market. We should set a relatively low price to attract early adopters and whilst gaining experience ourselves. This will help us to introduce a new product and to start building customer loyalty and appreciation for it. elasticLM pricing will be based on an initial price consisting of the 1% of the company revenue of the software that include elasticLM. From the second year onwards only the maintenance has to be paid, with a fee of 20% of elasticLM cost. The additional services will be paid in a flat rate.

The distribution of the product will be through direct selling from the future owners of elasticLM. The final software solution and manuals can be downloaded securely from the website. Firstly we target Europe, and a next step could be to expand to other regions. Project partners could act as resellers by receiving their commission, and afterwards we will have external resellers.

Our marketing goals in the first year are: creation of brand, launch a new product and positioning of new product. To do so we will use promotional tools and tactics as the elasticLM website, press release, trade shows, print advertising and other marketing collateral. For the initial launch the promotional budget will be the 20% of the investment made for the further development of the commercial product. In subsequent years the promotional budget will be the 5% of the revenues forecast.

In terms of financials the costs and expected revenues have been calculated in detail. There is needed an important initial investment to commercialize the SmartLM prototype into a product and yearly costs associated to the correct performance of the product. Revenue forecasts have been produced according to the initial interest received from some potential customers and the market size. As the market may be unpredictable, three different scenarios have been explored to forecast the revenues under different conditions. These are: the expected, the pessimistic and the optimistic scenario. We expect to achieve the break-even point in the middle of 2013, in the pessimistic scenario this will not happen until 2015 and in the optimistic scenario, it will occur in 2012.

The consortium has realized the business impact elasticLM can have in the market and we have identified the need to develop an exploitation agreement. The objective of it is to formalise the modalities and the conditions that will govern the commercial exploitation of the project results after the end of the project. The development of an exploitation agreement is complex, but we have created a roadmap and we have been progressing in the right direction. After several discussions, we have a SmartLM prototype ownership map between all the partners. According to their contribution to the development of the prototype each partner has assigned a percentage of ownership. We have decided that the partners of the consortium are going to exploit the results itself and we will not create a spin-off or start-up for this purpose. We have identified the roles that may appear after the end of the project, describing each role with rights and obligations enclosed in each role: future elasticLM owner, internal use and research, integrator, re-seller and other options. Partners freely decide which role they are willing to participate. Three partners are willing to be future owners of elasticLM. All have described their decision jointly with their exploitation interest in their preliminary individual exploitation plan.

D1.1

Functional requirements of the new licensing architecture for Grids

Version 1.3

WP 1 Requirements Analysis

Dissemination Level: Public

Lead Editor: Daniel Mallmann, Forschungszentrum Jülich

30/06/2008

Status: Approved by QM

Seventh Framework Programme of the European Community

Information Society

Proposal/Contract no.: 216759

Context

WP 1	**Requirements Analysis**
Dependencies	**This deliverables specifies the technical requirements of the SmartLM licensing architecture. It will be the basis for the developments in WP 3 and WP4.**

Contributors: **Sergio Bernardi (CINECA), Claudio Cacciari (CINECA), Francesco D'Andria (ATOS), Roberto d'Ippolito (LMS), Henning Eickenbusch (ANSYS), Naji El Masri (LMS), Andrés Gómez (CESGA), Björn Hagemeier (FZJ), Jiadao Li (FZJ), Daniel Mallmann (FZJ), Josep Matrat Sotil (ATOS), Jose Carlos Mouriño Gallego (CESGA), Angela Rumpl (SCAI), Eberhard Sekler (INTES), Christian Simmendinger (T-Systems), Devarajan Subramanian (Gridcore), Wolfgang Ziegler (SCAI), Csilla Zsigri (451 GROUP)**

Reviewers: **Roberto d'Ippolito (LMS), Francesco D'Andria (ATOS), Csilla Zsigri (451 GROUP)**

Approved by: **QM**

Version	Date	Authors	Sections Affected
0.1	29/05/2008	Sergio Bernardi, Claudio Cacciari, Francesco D'Andria, Roberto d'Ippolito, Henning Eickenbusch, Naji El Masri, Andrés Gómez, Björn Hagemeier, Daniel Mallmann, Josep Matrat Sotil, Jose Carlos Mouriño Gallego, Angela Rumpl, Eberhard Sekler, Christian Simmendinger , Devarajan Subramanian Wolfgang Ziegler, Csilla Zsigri	All
0.2	02/06/2008	Jiadao Li, Daniel Mallmann	All, Chapter 2
0.3	03/06/2008	Henning Eickenbusch , Eberhard Sekler, Wolfgang Ziegler	Summary, Chapter 3, Annex A
0.4	08/06/2008	Henning Eickenbusch, Björn Hagemeier, Daniel Mallmann, Devarajan Subramanian, Wolfgang Ziegler	Summary, Chapter 3, Annex A
0.5	09/06/2008	Francesco D'Andria	Acronyms, Glossary, Chapter 3
0.6	09/06/2008	Christian Simmendinger	Chapter 3
0.7	10/06/2008	Daniel Mallmann	All, Chapter 3
0.8	12/06/2008	Henning Eickenbusch, Björn Hagemeier, Daniel Mallmann, Wolfgang Ziegler	Summary, Chapter 3, Annex A
0.9	13/06/2008	Daniel Mallmann , Christian Simmendinger	Chapter 3, Annex A
1.0	13/06/2008	Daniel Mallmann	All (format)
1.1	16/06/2008	Daniel Mallmann, Eberhard Sekler	All (typos, rewording)
1.2	27/06/2008	Daniel Mallmann	All (changes suggested by reviewers)
1.3	30/06/2008	Daniel Mallmann	Annex A, Annex B table and figure captions

Full license

License

Definitions

a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.

c. "Licensor" means 1all partners of the SmartLM consortium that have participated in the production of this text.

d. "Original Author" means the individual or entity who created the Work.

e. "Work" means the copyrightable work of authorship offered under the terms of this License.

f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

Fair Use Rights

Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

License Grant

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.
c. For the avoidance of doubt, where the work is a musical composition:
d. Performance Royalties Under Blanket Licenses. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
e. Mechanical Rights and Statutory Royalties. Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).
f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved.

Restrictions

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by clause 4(b), as requested.
b. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or Collective Works, You must keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

Limitation on Liability

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

Miscellaneous

a. Each time You distribute or publicly digitally perform the Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Table of Contents

List of Tables

List of Figures

Terminology

Keywords to indicate requirement levels

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 to indicate requirement levels.

MUST

This word, or the terms "REQUIRED" (abbreviation “REQUIR”) or "SHALL", mean that the definition is an absolute requirement of the specification.

MUST NOT

This phrase, or the phrase "SHALL NOT" (abbreviation “SH-NOT”), mean that the definition is an absolute prohibition of the specification.

SHOULD

This word, or the adjective "RECOMMENDED" (abbreviation “RECOM”), mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

SHOULD NOT

This phrase, or the phrase "NOT RECOMMENDED" (abbreviation “NOT-REC”) mean that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour described with this label.

MAY

This word, or the adjective "OPTIONAL" (abbreviation “OPTION”), mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Acronyms

Table 1. Table of acronyms used in this deliverable

API	Application Programming Interface
ASP	Application Service Provider
CAS	Central Authentication Service
DRM	Distributed Resource Manager
HPC	High Performance Computing
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
ISV	Independent Software Vendor
LSF	Load Sharing Facility – a resource management system from Platform Computing Inc.
PKI	public key infrastructure
PMI	Privilege Management Infrastructure
QoS	Quality of Service
RMS	Resource Management System
SAML	Security Assertion Markup Language
SLA	Service Level Agreement
SmartLM	Grid-friendly software licensing for location independent application execution
TCP	Transmission Control Protocol
VO	Virtual Organization
VOMS	VOMS is an acronym used for Virtual Organization Membership Service in grid computing
WS Agreement	Web Services Agreement
X.509	Public-Key Infrastructure
XML	Extensible Markup Language

Glossary

Table 2. Definition of terms used in this deliverable

Term	Role	Description
Application Service Provider (ASP)		An *ASP* provides the use of their applications over the network. This software is hosted on central servers of the provider. The *ASP* also hosts the *license server* from which *users* can rent out *licenses* for use on the hardware resource (for fault-tolerance reasons multiple redundant license servers may be hosted). In this project the provided software is SmartLM enabled.
Computational resource		A cluster, supercomputer or simple computer that is accessible through a local *Resource Management System (RMS)* or through a Grid middleware.
Distributed Resource Manager (DRM)		A species of a *resource management system* often used in multi-cluster environments. The *DRM* e.g. Sun Grid Engine, Windows CCs, is responsible for dispatching user jobs based on the resources requested to the right cluster nodes in the *High Performance Computing* (*HPC) resource* for execution.
Feature		A part of an application which can be licensed separately.
Grid Orchestrator		When a *user* submits his job to the *Grid orchestrator*, the orchestrator cares for where and when the job is executed, allocates the *computing resources* and *licenses* for the execution.
HPC resource		A *computational resource* suitable for high performance computing. May be operated under different Operating Systems. The system may be accessible from outside the site that is hosting the *HPC resource*, but access from outside may also be restricted to a Web front-end or any other front-end (login) system usually not located in the same network as the *HPC resource.*
Independent Software Vendor (ISV)	License owner	An *ISV* is the owner and vendor of a *licensed application*. In this project the application is *SmartLM* enabled.

License		Right to use a *license protected software*. The *license* usually includes further rules under which the software might be used, e.g. restrictions with respect to the execution environment, the executing *user*, etc. the license may be spawned from a pool of licenses, and the rules of individual license issuing are governed by a *license contract framework*.
License contract framework		The contract between a licenser – usually the owner of the copyright - and a licensee – usually an organization that buys the right to use the copyrighted software. The *license contract framework* sets the general rules and conditions of software usage and usually also the general cost agreement.
License protected software		A software (usually protected by copyrights) where the owner of the copyright grants the right to use the software through a *license*.
License server		A software process running at the site of the *user* controlling the use of one or more *license protected software* systems. The *license server* acts as license provider in the *negotiation* process when a *user* requests a *license* for using *license protected software*.
License service		Hosted by the *license server*, delivering the schedulable license from the *license server* to the end-user. If the *license protected software* is executed using local resources there might be a communication channel between the application and the *license service* during run-time. In case of using remote *computational resources* usually no communication channel exists.
License system administrator		The *license system administrator* manages the local *license server*, e.g. is responsible for loading license *features* of various *ISVs* onto the *SmartLM license server*, handling license *feature* renewals and controlling access to individual license *features* hosted on the *SmartLM license server*.
License virtualization		The process of creating and probably reserving a schedulable license from the *license server*.
Licensed software		Short for *license protected software*.
Negotiation		The process to agree on the terms of usage of a *license protected software*. Resulting in a *Service Level Agreement* if successfully completed.

Resource broker		The *resource broker* has to choose a site where all the available resources (compute, network, licenses …) fit best to accomplish the job.
Resource Management System (RMS)		The local *RMS* (usually a batch queuing system like PBS-pro), which is responsible for dispatching user jobs to the local *computational resource*.
Service Level Agreement (SLA)		A template setting the terms of software usage the two parties agree upon. This includes the constraints of using the software, e.g. duration, user and may include guarantee terms and penalties for both parties.
Service provider		Short for *Application Service Provider*.
Simulation engineer	End-user	The *simulation engineer* starts the simulation software either from his working environment or submits the application to a Grid *computational resource*.
SmartLM	License manager	*SmartLM* manages all licenses owned by a site. The SmartLM software includes the *license service*, accounting & billing service, orchestration service, accounting data storage & license data storage.
Software developer	License owner	A person acting as an *Independent Software Vendor (ISV)*.
User	End-user	A *user* who uses a *license protected* (SmartLM enabled) *software*. The end-user may come from different environments, with or without *computational resources* for the execution of her application, e.g. being a customer of an *ASP*, a member of a research institute, or an employee of a company. End-user is a synonym of user.
User-Group	End-user	*User-Groups* may be defined using attributes (e.g. defined in a Virtual Organization) or simply through their relation to an organization or company. *User-group*s may receive a dedicated (probably restricted) set of licenses from an *ASPs license server. User-group*s are also supported for environments without *ASP*.

1. Summary

The objective of this deliverable is the identification of functional requirements for the new licensing architecture considering the perspectives of the different players and systems involved.

The actors are:

- Independent Software Vendors (ISV)
- Application Service Providers (ASP)
- Academic Partners and Public Centres
- End Users
- Grid Middleware Providers

We analysed existing licensing mechanisms (chapter 2) and gathered potential requirements from all actors. An initial list of requirements was gathered in a brainstorming activity, to which all project partners contributed. This rather rough list was then matched with the requirements derived from detailed use cases (Annex A) that the project partners collected. The resulting requirements were levelled according to the terminology of RFC 2119 (see Terminology on page 10). We received some non-functional requirements as well, which are summarized in Annex B.

The requirements were driven by technical and functional aspects as well as market perspectives and business models, which are elicited in WP2. The business relevance of the requirements was analysed and rated in order to distinguish State of the Art and new features, that are either required or optional usable for new business models.

2. Existing licensing mechanisms

In this chapter we give an overview on existing license mechanisms. Currently the license management is restricted to a single administrative domain in which the centralized license management paradigm is suitable and efficient to manage the licenses. We use the licensing models and terminology of FLEXnet, as this is the technology for software licensing most often used by ISVs and the other licensing technology providers offer either subsets of models supported by FLEXnet or quite similar models. Section 2.2 "License Scheduler" provides a short introduction to time scheduling of licenses and the Platform Computing Inc. LSF license scheduler as one example.

2.1. Typical License Models

The basic license models provided by FLEXnet presented next can be combined to create new license models [1], [2], [3], [4].

- Node-locked licenses: Node-locking means the software can only be used on one machine or a set of machines. There are two types of node-locked licenses: uncounted and counted. For the counted node locked licenses, a *license server* and a vendor daemon are necessary.
- Floating (concurrent) licenses: Anyone on the network can use the licensed application, up to the limit specified in the license file (also referred to as concurrent usage or network licensing).
- Mixed node-locked and floating licenses: Uncounted node-locked and concurrent usage licenses can be mixed in the same license file, therefore more flexible usage models can be derived.
- Demo licenses/evaluation licenses: Properties of an evaluation license may include: (i) Limited product functionalities or *features*, (ii) Limited number of uses, (iii) Expiration date.
- Usage-based licensing: A quite important licensing strategy in which the actual usage patterns are monitored by the license management system, and billing or auditing are based on the actual usage data. FLEXnet Licensing supports several usage-based models, e.g.:
  - Overdraft: allowing the *ISV* to specify a number of additional licenses which customers are allowed to use in addition to the licenses purchased;
  - Pay-per-use: allowing the customers to pay for the effective usage of the licenses, which can be audited based on time, the number of transactions, etc.
- Mobile licensing: Used when *users* want to run an application on a machine that does not have a continuous connection to a *license server* system. These situations can include:
  - Working on a laptop; or using a computer both at work and at home or off-site; or working from several different computers not connected to a *license server* system
  - Fulfilled from a prepaid license pool: The license is fulfilled from a prepaid number of license-days for the usage period.
  - Node-locked to a user name: If a license is to be used exclusively by one *user* on different machines, that license can be node-locked to the *user's* user name.
  - License rehosting: if an end-user want to move a license without using one of the other mobile licensing methods. In this model, a new node-locked license certificate for each new machine should be generated.

- Hard-mobile: Mobile license usage is controlled by a FLEXid. If the FLEXid is attached to a *license server* system, then the use floats on the network. To temporarily transfer the license, the *user* moves the FLEXid from the server to a standalone machine.
- Soft-mobile: Licenses are temporarily transferred to a *license server* system on the mobile laptop. The FLEX enabled product uses an encrypted local file, placed there by the *license server* system, to do checkouts during the usage period.
- License borrowing: A license can be borrowed from a *license server* system via a special checkout and used later to run an application on a computer that is no longer connected to the *license server*.

2.2. License Scheduler

If the number of licenses available for an enterprise is limited, e.g., due to the cost factor, it is necessary that these licenses are efficiently managed and highly utilised since even if an enterprise can apply for additional licenses from the *ISV*, it has to pay for the extra licenses. A local license scheduler could help scheduling the licenses of a site efficiently. However, while in most cases the local license management system provides information on the licenses already in use and still available there are no built-in queuing or reservation mechanisms. While an external scheduler might create an efficient schedule for the available licenses based on the *users'* requests, monitoring the use of the licenses and enforcing the schedule is difficult due to the usually encrypted communication between *license server* and application. Co-operations between license technology providers and license scheduler implementations could be a way to overcome this limitation. For instance, platform computing offers a product called LSF license scheduler [5] which is a local license scheduler restricted in a single administration domain and manages the license tokens instead of controlling the licenses directly. The current available number of licenses can be obtained by the FLEXnet manager. There are several license scheduling polices provided, e.g., fair share, round robin, pre-emption. The licenses can also be checked out for non-LSF jobs. In this way, the licenses can be scheduled and co-allocated with other resources/services. Dong et al. [6] developed a software sharing system in the grid environment which is not restricted to a single domain. The system adopts the constellation model for resource management and combines the sharing and scheduling of both hardware and software license resources. However, there is no support for SLAs and *QoS* in this system.

3. Functional requirements

This section describes the requirements for the different components and services of the SmartLM architecture. Most of the requirements are derived from the use-cases that were contributed by the SmartLM partners (see Annex A for details). The process can be summarized as follows. In a first step, the existing licensing mechanisms were analysed and potential requirements were gathered from all actors in the Grid licensing scenarios in a brainstorming activity. Then use cases that describe the various Grid licensing scenarios in detail, were written down by individual participants and mapped to the list of requirements in a subsequent step. This led to a consolidation of the requirements list. The use cases reference the requirements that had been gathered in the first step. As a consequence it was possible to divide the list into hard and soft requirements, thus identifying the set of *features* to be implemented during the first phase of development. The less important and non-functional requirements are left for a later phase, after the initial prototype of the Grid licensing infrastructure has been put in place. The non-functional requirements are summarized in Annex B. The final decision, however, will be taken in the WP3 - Implementation of Basic Technology and Security, WP4 - Implementation Accounting & Billing, and WP5 - Integration in middleware, applications, *ASP* environment.

The requirements are clustered according to the main components and services identified when analysing the use-cases:

- *License server*
- License mechanisms, format, and content
- License token transfer and security
- Authentication and authorisation
- Policy enforcement in the applications
- Middleware and orchestration service
- Accounting and billing

To distinguish the relevance of the requirements we assigned a requirement level to the individual requirements. The keywords used follow the terminology defined in the IETF RFC 2119 and are to be interpreted as described in this RFC. The semantic of the keywords is described in detail in section Terminology (page 10) and the terms used in column "Level" are the abbreviations introduced there:

REQUIR: REQUIRED

RECOMM: RECOMMENDED

OPTION: OPTIONAL

The requirements were analysed according to their business relevance. The result is a rating that classifies the requirements as follows:

SOTA: State of the Art, i.e. existing licensing mechanisms offer this already

N-REQ: New *feature*/requirement, that is not yet available in existing licensing mechanisms but is needed for new business models.

N-OPT: New *feature*/requirement, that is not yet available in existing licensing mechanisms and could be usable for new business models.

The rating can be found in the column business relevance. Requirements that are seen as a technical detail without relevance to business scenarios were not rated.

3.1. License server

The *license server* is expected to manage all licenses of a site, institution or company. Based on the *license contract framework* between the licenser and the licensee the *license system administrator* defines the local policies for using the license-protected software. The *license server* may

- directly manage tokens provided by the licenser, which a *user* or application may request for executing an application,
- create such tokens on the fly depending on the policies specified e.g. number of concurrent *users*, or
- a combination of both depending on what the *license contract framework* with the licenser specifies.

Table 3. Requirements for the license server

No	Description	Level	Business relevance	Referring use case
1.	Two companies hosted by one *ASP*: properly separate the license usage	REQUIR	SOTA	UC01, UC14
2.	The license manager must support two models of accounting: per permanent (long time) license and per usage	REQUIR		UC01
3.	*Negotiation*: Functionality that allows (to a *Service provider* and a Client) negotiating an *SLA* Contract starting by an *SLA* Template.	REQUIR		UC05
4.	*Negotiation*: The SmartLM must allow *negotiation*. The rules/policies for *negotiation* should be expressed in *XML* format (probably as creation constraints) inside the *SLA* template and each *VO*/vendor admin must be able to administrate them. It is mandatory to deny the *negotiation* for *users* of some countries.	REQUIR		UC05
5.	*Negotiation*: The *negotiation* interface must work using WS-Agreement	REQUIR		UC05
6.	*Negotiation*: There must exist a signed record of the final agreement (one copy is available at the orchestrator, and additional copies might be available at the license server	REQUIR		UC05
7.	*Negotiation*: The SmartLM service must be able to access a blacklist or whitelist of *users*	REQUIR		UC05

8.	License reservation: Licenses can be reserved.	REQUIR		UC05
9.	It should be possible to set an expiration date for license reservation.	REQUIR		UC05
10.	License manager monitoring: The *license server* should be able to provide information on license availability, reservations, usage at any point in time. This information must be provided in human readable, *XML* format to be usable in other applications.	REQUIR		UC05, UC14, UC15
11.	Trusted service for *SLA* verification: Any information record should be signed by the parties – *Grid orchestrator*/resource provider/license provider	REQUIR		UC05
12.	Ability to host licenses and *features* from multiple vendors in one instance of the *License server*	REQUIR		UC14
13.	SmartLM can provide usage statistics about each set of similar license *features*.	REQUIR		UC14, UC12
14.	SmartLM can handle access permissions for each set of similar license *feature* sets individually, e.g. SmartLM can host multiple sets of the same license *features* owned by different groups on the same *license service* instance.	REQUIR	SOTA	UC14
15.	Ability to update licenses for a particular *feature* without having to restart the *license server*.	REQUIR	SOTA	UC15
16.	Statistical information must include at least the following information about the license checkouts and reservations: • Who: Name, Company, localization of the *user* (to estimate where he asks for support) • When • Where (*IP*/Hostname) • *Features* (ANSYS checks out different *features* at the same time, e.g. solver, parallel, number of processes, combustion models, multiphase models …) • Quantity	REQUIR	SOTA	UC15, UC08, UC09, UC12
17.	Possible notification based interface to tell the *Distributed Resource Manager* when a license is checkout/reserved or checked back in., either a standards based interface (WS-Notification) or the DRM must poll	REQUIR		UC16

18.	License server interface exposed should ideally be platform independent meaning interface bindings should exist for the Linux world and the Microsoft world.	REQUIR	SOTA	UC16
19.	Possibility to run the *license server* as a standalone application listening on a certain port, for single cluster use.	REQUIR	SOTA	UC16
20.	Default hosting environment will be bundled with license server for lightweight installation	REQUIR		UC16
21.	The *license server* in standalone mode i.e. listening on just a single port, should be able to handle concurrent incoming connections.	REQUIR	SOTA	UC16
22.	Content: Modules (functionality i.e. linear static, ...) • Start date/time • Expiration date/time • Number of CPUs used • Features used • Duration of granted usage, in case of flexible start date/time	REQUIR	SOTA	UC02-2, UC13, UC07, UC08, UC09, UC10

3.2. License token transfer and security

This section summarises the requirements derived from the use cases that are related to the transfer of license tokens and usage information as well as trust and security aspects in the *license server* and the policy enforcement in the application.

Table 4. Requirements for the license token transfer and security

No	Description	Level	Business relevance	Referring use case
23.	Security: Any information record should be signed by the parties – *user*/resource provider/license provider	RECOM	N-OPT	UC01
24.	Transfer: Execution cluster may not have direct external access. If not direct access, any communication must go through a proxy	REQUIR	N-REQ	UC01
25.	Transfer: Secure transfer of license token	REQUIR	N-REQ	UC02-2, UC02-3, UC07, UC13
26.	Transfer: Human readable form of license token stored locally	RECOM	N-OPT	UC02-2, UC02-3, UC07, UC13

27.	Security: Transfer of checksum information	REQUIR	N-REQ	UC02-2, UC02-3, UC07, UC13
28.	Transfer: Latency of data transfer must be taken into account	REQUIR	N-OPT	UC02-2, UC07, UC13
29.	Security: Prevent license token from misuse • Coupling of data and license token by signing • License token can only be used once • Guarantee of integrity of license token and transferred job data	REQUIR	N-REQ	UC02-2, UC02-3, UC07, UC13

3.3. Authentication and authorisation

In this section the focus is on the requirements for the authentication and authorisation infrastructure of the SmartLM framework. The objective of SmartLM is to support state –of-the-art authentication and authorisation technologies. This includes those coming from *PKI*-based environments like Globus Toolkit 4 [7] or UNICORE [8] but also those coming from environments where attributes of a *user* are embedded in *SAML* assertions like in Shibboleth federations.

If a *user* accesses a *license service* hosted at his site he usually already provided credentials to identify himself, thus there is no need to provide again a proof of his identity for the *license service*. However, the *user* has to supply information that allows the service to determine what kind of license the *user* is authorised to request from the *license service*.

3.3.1. Requirements

No	Description	Level	Business relevance	Referring use case
30.	Should support usage of *VOMS* attribute certificates provided by a *user* for authorisation of license usage.	OPTION	N-OPT	UC01, UC06
31.	Should support *user* attributes provided as *SAML* for authorisation of license usage.	OPTION	N-OPT	UC01, UC06
32.	Support of identification and authentication using *PKI* certificates	REQUIR	N-OPT	UC03, UC05
33.	Authorization using *PMI* certificates	OPTION	N-OPT	UC03, UC05
34.	Should support *CAS* (Globus Community Authorization Service)	OPTION	N-OPT	UC06

35.	X509 certificates can be used to authenticate the consumer to the *license server*.	REQUIR	N-OPT	UC01
36.	Authentication and authorisation based on local uids	REQUIR	N-OPT	UC16

3.4. Policy enforcement in the applications

This section summarises the requirements that derive from the use cases related to the applications (i.e. the *licensed software*). The application is the point where the *ISV*'s license policy is enforced.

Table 5. Requirements for the policy enforcement in the applications

No	Description	Level	Business relevance	Referring use case
37.	X509 certificates can be used to authenticate the consumer to the *license server*.	REQUIR	N-REQ	UC01
38.	Web services exposed that talk to the *License server* must be usable directly on Globus Containers and UNICORE containers. It might not be feasible to install Tomcat everywhere, and it just adds up to the services if Globus or UNICORE already exist.	REQUIR		UC01
39.	Web Services should provide functionality not only to checkout/checkin licenses but also to reserve/get current license usage status.	RECOM	N-OPT	UC01
40.	If external access is mandatory/needed, It must support access through intermediary/proxy/SOCKS	OPTION	N-OPT	UC01
41.	It can run on private networks	REQUIR	SOTA	UC01, UC10
42.	It must use a well-known port range	REQUIR	SOTA	UC01
43.	Location and Discovery: The consumer must be able to make a decision on the best available license resource	OPTION	N-REQ	UC02-2, UC13, UC07
44.	Network: Connection to LM	REQUIR	SOTA	UC02-2, UC13, UC10
45.	Performance, scalability, redundancy: Application must confirm the sending of the accounting record	RECOM	N-REQ	UC02-2, UC13

46.	Performance, scalability, redundancy: The accounting record must also be stored locally in a file if demanded	OPTION	N-OPT	UC02-2
47.	License renewal through SmartLM API on the *computational resource*	RECOM	N-REQ	UC03
48.	Language bindings: FORTRAN 77, 90, 95, 2003	REQUIR	SOTA	UC03, UC06, UC07, UC12
49.	Language bindings: C89, C99	REQUIR	SOTA	UC03, UC06
50.	Language bindings: C++	REQUIR	SOTA	UC03, UC06
51.	Language bindings: Java	REQUIR		UC03, UC06
52.	Language bindings:.net	OPTION		UC03, UC06
53.	API for use in solver scripts (Mixture Shell and Perl) to return license if the solver itself crashes	REQUIR	SOTA	
54.	Code integration: Transfer of license token to check for expiration	REQUIR		UC07
55.	Code integration: No other logic to be included in existing code	RECOM		UC07
56.	Code integration: Business logic must be done by container	RECOM		UC07

3.5. Middleware and orchestration service

In this section we summarize the functional requirements concerning the Grid middleware and the orchestration service.

Table 6. Requirements for the middleware and the orchestration service

No	Description	Level	Business relevance	Referring use case
57.	The orchestrator must be able to aggregate licenses from different instances of SmartLM, e.g. from a local *license server* within the *computational resource*s administrative domain and a SmartLM service of an *ASP*	REQUIR	N-REQ	UC01, UC16

58.	The orchestrator cares for prolongation of licenses if jobs run longer than expected	REQUIR	N-REQ	UC05, UC07, UC08, UC09, UC11
59.	The Grid middleware offers job submit, job monitoring and controlling, retrieving intermediate and final results, and cancelling unsuccessful jobs	REQUIR	N-REQ	UC05, UC07, UC11
60.	Grid middleware offers advanced reservation	RECOM	N-REQ	UC05
61.	Grid middleware publishes information about available software, operating system, hardware, ... (all necessary information that is needed for the license *negotiation*)	REQUIR	N-REQ	UC05
62.	*Negotiation*: should support *X.509*v4 certificates to check if *user* is allowed to make the contract.	REQUIR		UC05
63.	*SLA* Translator: allows the physical access to the documents (*SLA*-Template and *SLA*-Contract) and get (or set) information from them. For each document it will know the associated ID and it will contact the repositories in order to retrieve the appropriate document.	REQUIR		UC05
64.	Contract/Template-Repository: The *SLA* Template Repository allows storing and retrieving *SLA* Template documents. This service is used as a storage facility by the *SLA*-Translator.	REQUIR		UC05
65.	*SLA* Evaluation: After the *negotiation* phase, once instances of our License have been created, it is necessary to ensure that who provides the license observes the contractual terms (*WS Agreement [9]*).	RECOM		UC05

3.6. Accounting and billing

This section covers aspects of billing and accounting. SmartLM aims to provide accounting and billing in a flexible way, suitable for scenarios ranging from local accounting to multi-source billing and accounting.

Table 7. Requirements for accounting and billing

No	Description	Level		Referring use case
66.	Secure and trusted	REQUIR	N-REQ	UC01
67.	Redundant	OPTION	N-OPT	UC01

68.	Usage statistic • Who/ when/ where • Functional modules used • Date/time • Number of CPUs used • Model Size (unknowns, nodes)	REQUIR	N-REQ	UC01, UC07, UC08, UC10
69.	Billing statistic • Who/when/where/ *user* localisation (to estimate where he asks for support) • Date/time • Amount • Balance	REQUIR	N-REQ	UC01, UC07
70.	Real time accounting and billing	REQUIR	N-REQ	UC03, UC08
71.	Accounting based on historical records, e.g. discounts based on previous usage	RECOM	SOTA	UC18
72.	Rule engine to allow for flexible pricing policies	REQUIR	SOTA	UC18
73.	Application calls SmartLM API and submits accounting information to SmartLM License manager – possibly via the grid orchestration service - after job completion if no duration is negotiated.	REQUIR	N-REQ	UC01, UC02-2, UC03, UC13, UC07, UC04, UC18
74.	The accounting record must also be stored locally in a file when demanded	OPTION	N-REQ	UC13
75.	Fine grained usage statistics per group and per cost-unit (context based accounting, user-defined)	REQUIR	N-REQ	UC14, UC18
76.	Accounting based on local UID	REQUIR	SOTA	UC16
77.	Multi Source Billing and Accounting in order to support ISVs without commercial infrastructure.	REQUIR	N-REQ	UC13, UC02
78.	Multi Source Billing and Accounting in order to support aggregation of accounting records in an ASP context.	REQUIR	N-REQ	UC18
79.	Budget control for end users necessary, e.g. limit costs per run, per department of end user company, per month	REQUIR	N-REQ	UC08, UC03, UC18
80.	Budget information (Original budget, amount left) must be available for user, ASP and ISV	REQUIR	N-REQ	UC08, UC03

81.	ASP creates one bill (license cost and hardware resources cost) for the user's company	REQUIR	SOTA	UC08
82.	Usage statistics need protection against manipulation	REQUIR	N-REQ	UC08
83.	The *ISV* must be able to view/download license usage records	RECOM	N-OPT	UC05

References

[1] *FLEXnet Licensing 11.4 Programming and Preference Guide for Trusted Storage-Based Licensing.* Macrovision, 2006.

[2] *FLEXnet Licensing End User Guide.* Macrovision, 2006. Product Version 11.4, Document Revision 01.

[3] *FLEXnet Licensing Programming and Reference Guide for License File-Based Licensing.* Macrovision, 2006.

[4] *Getting Started With the Licensing Toolkit for License File-based Licensing.* Macrovision, 2006. Product Version 11.4, Document Revision 01.

[5] *Platform LSF License Scheduler.* http://www.platform.com/Products/platform-lsf-license-scheduler/

[6] Xiaoshe Dong, YinfengWang, Fang Zheng, Zhongsheng Qin, Hua Guo, and Guofu Feng. *Key techniques of software sharing for on demand service-oriented computing.* In Yeh-Ching Chung and Jos'e E. Moreira, editors, GPC, volume 3947 of Lecture Notes in Computer Science, pages 557–566. Springer, 2006.

[7] *Globus Toolkit 4*: http://www.globus.org/toolkit/docs/4.0/

[8] *UNICORE*: http://www.unicore.eu/documentation/

[9] *WS Agreement Specification*: OGF Grid Final Document GFD.107 http://www.ogf.org/documents/GFD.107.pdf

[10] *ANSYS CFX*: computational fluid dynamics software http://www.ansys.com/products/cfx.asp

Annex A. Use cases

A.1. UC01: case study ASP outsourcing

This use case is the case study – example 1 from the SmartLM Description of Work.

A.1.1. Description

Computational Science Provider (CSP) is a European SME which has several applications in its portfolio to be serviced on-demand (*First level ASP*). It has contracted the provision of the service with few big companies including strong *Service Level Agreements*. To provide such a service, it runs a small cluster with all applications tested and validated and one internal *license server*. This infrastructure is enough for usual workload. An engineer at one company (*Customer*) utilizes a Web-Services enabled Process Integration Workflow tool (OPTIMUS) to integrate a Simulation based engineering process. This process consists of several Web-Services enabled Simulation tasks. The Workflow tool can either be accessed in the *user* local computer infrastructure or as a Web-Service. The Workflow uses the services of CSP when it needs. However, due to an unexpected demand, CSP own resources are not enough to fulfil the contracted SLAs, so it has to buy resources from a broker company (*Broker*). It demands from the broker both external computer resources and temporal licenses for running the demanded applications with the same SLAs. The broker finds and books these external resources (*Second level ASP*) and finally, CSP can submit the jobs. The input information for each job includes the information to check out the license. When the applications start on the contracted cluster, the software gets the license on behalf of the final *user* at the beginning and checks it periodically during the execution. Because it uses standard ports and protocols, there is no need to change security rules on the firewall of any company. If the license expires during execution, the software demands automatically a new license, waiting for it. Because it expands the initial contract, the *license server* asks the broker for an authorization. The broker Server, which is the only one who knows the final *user* identity, asks to the client for permission to extend the lifespan of the license. Real time accounting and billing are available for all parties at any time. Billing may have a variable dependency with CPU hour and number of simultaneous CPUs involved on each run.

In addition to the actors mentioned in the glossary this use case has the following definitions:

Table 8. Additional actors of use case UC01: case study ASP outsourcing

Actor	Role	Description
Customer		CSP customers are large companies, which use the application services provided by CSP whenever they run out of internal resources.
First level *ASP*		Computational Science Provider (CSP) is an SME providing applications serviced on demand. Services are provided to its customers, with which CSP has pre-established SLAs.
Second level *ASP*		A second level *ASP* is providing services to other ASPs (and possibly end *users* as well).
Broker		If CSP runs out of resources to run the jobs of their

		customers, they need to extend resources on demand and establish relationships with other ASPs, where they can run the jobs. This is mediated by a broker, searching for and negotiating on-demand resources including licenses.

Table 9. Use case table UC01: case study ASP outsourcing

Use Case ID	UC01
Use Case Name	DoW Case Study – Example 1 (*ASP* outsourcing)
Purpose	An *ASP* needs additional resources on demand to fulfil contractual requirements and service the previously negotiated SLAs towards its customers.
Initiator	Workflow tool at CSP customers' site.
Primary Actor	First level *ASP*
Additional Actors	Customer, Broker, Second Level *ASP*
Description	A First level *ASP* has pre-established contracts with a small number of big companies for the provision of on-demand application services. The *ASP* also maintains a cluster, which is sufficient to cover the average demand. However, situations may arise where the in-house cluster may not be sufficient to serve all SLAs. Thus the first level *ASP* needs to forward jobs to another *ASP*, called second level *ASP* in this scenario. In order to find the second level *ASP*, it employs a third party Broker to find appropriate resources and licenses.
Pre-condition	ANSYS CFX [10] is adapted for the usage of the SmartLM license manager and available on all resources Resources are accessible for the *user's* of the organisation and have a network connection to the SmartLM license manager.
Post-condition	The application executes, the results are made available to the *user*, and accounting information is available at the accounting and billing service.
Use Case Functionality	
Sequence	1. Customer submits jobs to first level *ASP*, which exceeds current availability of resources (machines and licenses) at the *ASP*. 2. *ASP* requests additional licenses and resources from a Broker. 3. *ASP* submits jobs to selected second level *ASP*.

	4. Second level *ASP* gathers licenses on job start. Licenses com from First level *ASP* and other providers, which were booked by the Broker. 5. If the licenses need to be extended, because of a longer than expected runtime of the jobs, the Second level *ASP* needs to try and extend the licenses. For those licenses issued by a third party license provider, this is done through a proxy license issuing service co-located with the broker. For those licenses issued by the First level *ASP*, extensions are retrieved directly from there. 6. Once the job is done, still valid licenses are returned. The job outcome is returned to the First level *ASP*, from where the customer can finally pick it up.
Alternatives	5. Instead of extending the licenses that originate from the First level *ASP* directly, this could also be done via the Broker, thus making the refreshing more generic. It may not be a big difference considering the technical side.
Non-functional Requirements	Connections between parties directly communicating with each other, need to be permanently available. • Second level *ASP* -> Broker • First level *ASP* -> Broker • Broker -> License issuer • Broker -> First level *ASP*
Exceptions	Broker negotiated licenses can't be extended: • Find new ones and return those. Needs authorization from First level *ASP*, if not part of the contract between First level *ASP* and Broker. No additional licenses available (at the moment): • Can't run job. Serious violation of *SLA* between Customer and First level *ASP*. • Depending on timing constraints. No resource available to run large job: • Can't run job. Violation of contract.
Use Cases used	UC08 Pay per Use Scenario UC17: License aggregation
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/	NONE

security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	Refreshing of licenses goes via the Broker, because otherwise the parties could bypass him after the initial setup of licenses and resources. This is clearly not in the broker's interest.
Further Information	
Particular Requirements	• Two companies hosted by one *ASP*: how to properly separate the license usage • Any information record should be signed by the parties – *user*/resource provider/license provider • Execution cluster may not have direct external access. If direct access is not available, any communication must go through a proxy. • X509 certificates can be used to authenticate the consumer to the *license server*. • Should support *VOMS* certificates. • Should support *SAML* assertions. • The *license server* must be able to aggregate *user* licenses from different instances of SmartLM (see license aggregation use case A.13 UC17: License aggregation) • Internet access • The license manager must support two models of accounting: per license or per real usage • Accounting & Billing
Assumptions	All services (*license server*, broker, First level *ASP*, Second level *ASP*) are available at all times for each of the partners that need it.
Open Issues	NONE
Information Requirements	An *ASP* or piece of Software wanting to renew a license needs to know the service that issued that license. Thus, a link to that service needs to come along with the license information.

A.2. UC02: software developer

This use case is the case study – example 2 from the SmartLM Description of Work.

A.2.1. Description

Javier Pérez Pérez (*Software developer*) is a freelance *software developer* who has produced new solution for Financial Risk Analysis as a plugging of a well-known package. The plugging can be installed automatically downloading it from an internet server. Because he has no infrastructure for selling it directly, he decides to include the new standard solution - SmartLM - and register it in a license broker. This license broker is a European SMEs which allows to buy temporal, permanent or pay-per-use licenses. Bank of Wonderful Land (Final *User*), who has a technological surveillance department, detects in Internet the new plugging. Because the internal rules of the Bank do not allow installing untested software, the employees decide to test the software in an external test infrastructure based on Grid. After performing a complete set of tests externally using a pay-per-use model, they decide that the new software solves their analysis needs and buy 1000 permanent licenses to the broker to install inside the organization on their standard *license server*. Javier Pérez Pérez, finally, receives the receipts from the license broker without creating a commercial infrastructure.

A.2.2. Use Case Diagram

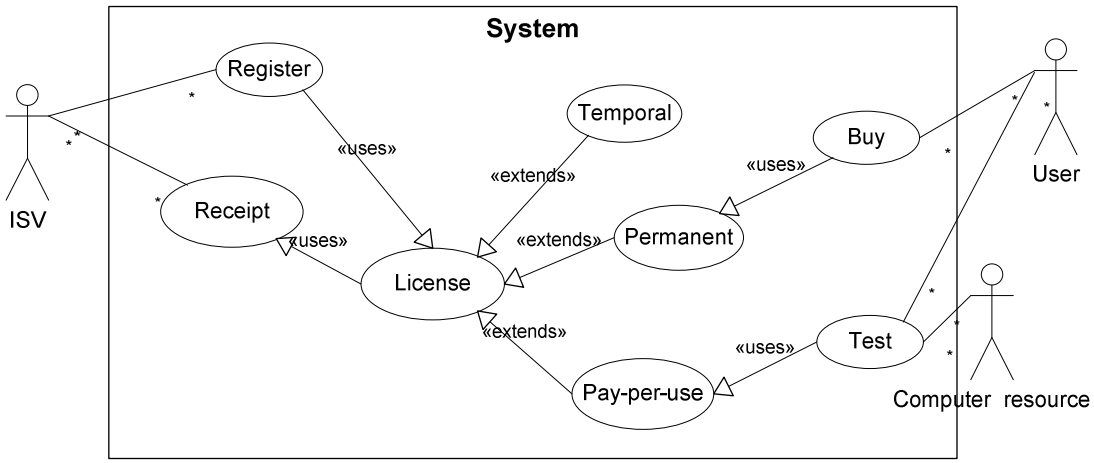


Figure 1. Use case diagram UC02: software developer

A.2.3. Actors

In addition to the actors mentioned in the glossary this use case has the following definitions:

Table 10. Additional actors of use case UC02: software developer

Actor	Role	Description
User (Bank)	End user	A *user* who uses a *license protected* (SmartLM enabled) *software*.

A.2.4. Use case 02 step 1: software developer registers the plug-in in license system

A.2.4.1. Use case diagram

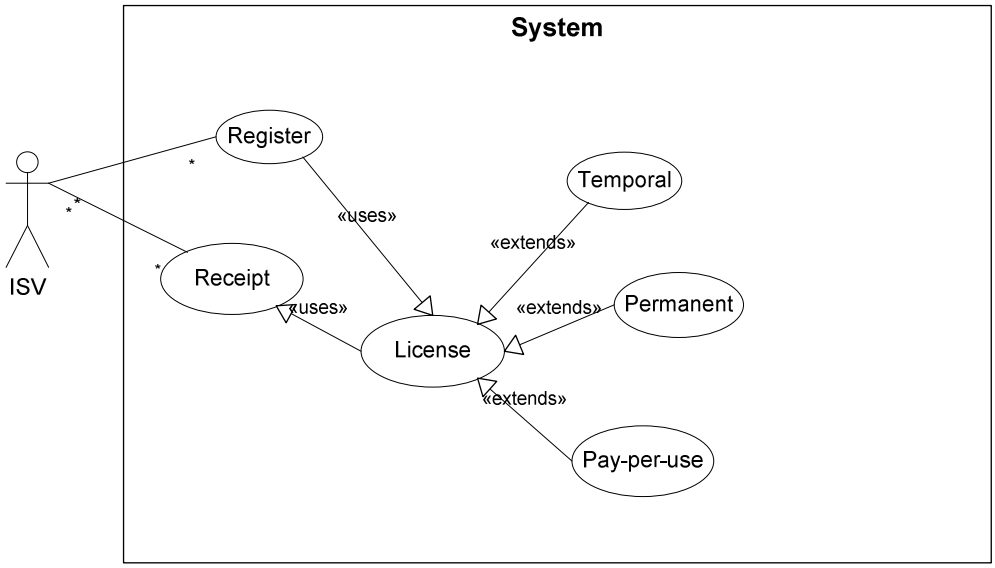


Figure 2. Use case diagram UC02 step 1: software developer registers the plug-in in license system

Table 11. Use case table UC02 step 1: software developer registers the plug-in in license system

Use Case ID	UC02-1
Use Case Name	Plug-in registration.
Purpose	Register the plug-in in the System
Initiator	*Software developer* (*ISV*)
Primary Actor	*Software developer* (*ISV*)
Additional Actors	SmartLM license manager
Description	The *Software developer* registers the plug-in in the System
Pre-condition	• *Software developer* already knows where the System is (the access point to the registration form) • A license for the plug-in exists • The plug-in can interact with the SmartLM Interface
Post-condition	the developer receives incomings for the use of his plug-in
Use Case Functionality	

Sequence	1. The *Software developer* logs on the System 2. The *Software developer* start the registration providing: 3. Plug-in software information Information about the utilization of the Licenses: max numbers of available license, license typology (Temporal, Permanent, Pay for Use) etc. 4. The *Software developer* publish the plug-in software license 5. The registration finishes successfully 6. The System returns: 1. The registration ID 2. A formal contract
Alternatives	
Non-functional Requirements	
Exceptions	• The registration form is not filled out correctly • The System doesn't support the published license • One of the parts does accept the contract
Use Cases used	
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	
Assumptions	

Open Issues	
Information Requirements	

A.2.5. Use case 02 step 2: Bank tests the plug-in in a Grid environment

A.2.5.1. Use case diagram

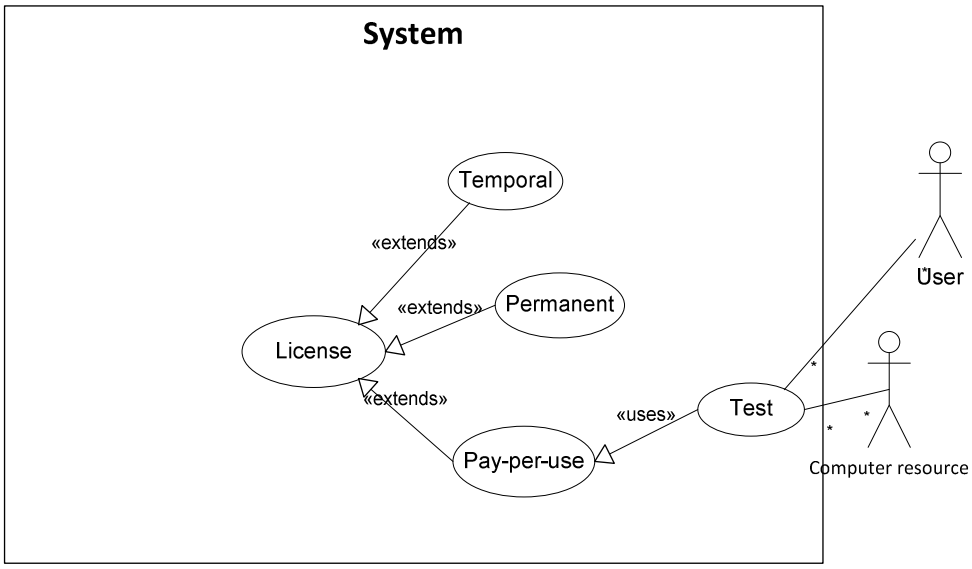


Figure 3. Use case diagram UC02 step 2: Bank tests the plug-in in a Grid environment

Table 12. Use case table UC02 step 2: Bank tests the plug-in in a Grid environment

Use Case ID	UC02-2
Use Case Name	Bank tests the plug-in in a Grid Environment
Purpose	Test the plug-in
Initiator	Bank (*user*)
Primary Actor	Bank (*user*)
Additional Actors	*Computational resource*, SmartLM license manager
Description	The Bank wants to test the plug-in in a pay per use scenario in a Grid environment
Pre-condition	A license agreement between the *Software developer* and the bank wants test the plug-in exists.

	The *License service* is located at the System. A "resource use" agreement between the Grid environment and the bank that wants to test the plug-in exists The Grid *user* has a certificate
Post-condition	Accounting information is available
Use Case Functionality	
Sequence	1. The plug-in is installed in the Grid environment (on request of the Bank). 2. The Bank contact the system and acquires the license 3. The System registers the license usage 4. The Bank log on to the Grid 5. The Grid provides information about the available resource to the Bank 6. Bank submits a job and license on the GRID 7. The Bank starts the test on the GRID 8. While the test runs the License is evaluated 9. The test finishes and inform the system about accounting information 10. The system registers the accounting information
Alternatives	
Non-functional Requirements	
Exceptions	Sequence 2: Error during application runtime: • Check in license to the "server" • Error while the simulation runs • Notify Simulation to the Bank • the demo license token is not valid or cannot be verified • the *user* is not allowed to use the license • the bank is not allowed to execute on Grid • The test application can not inform the system about accounting information • The system cannot check out the license because there are no

	tokens available.
Use Cases used	UC08 Pay per Use Scenario
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	**License consumer (application)** Location and Discovery: The consumer must be able to make a decision on the best available license resource Network: Connection to LM Performance, scalability, redundancy: Application must confirm the sending of the accounting record The accounting record must also be stored locally in a file when demanded **License** Transfer: Secure transfer of license token Human readable form of license token stored locally Transfer of checksum information Latency of data transfer must be taken into account Security: Prevent license token from misuse • Coupling of data and license token by signing • License token can only be used once • Guarantee of integrity of license token and transferred job data Content: Modules (functionality i.e. Linear static ...)

	• Start date/time • Expiration date/time • Number of CPUs used • Duration of granted usage, in case of flexible start date/time • Accounting information should be send back after job completion in the token
Assumptions	
Open Issues	
Information Requirements	

A.2.6. Use case 02 step 3: the bank buys the license

A.2.6.1. Use case diagram

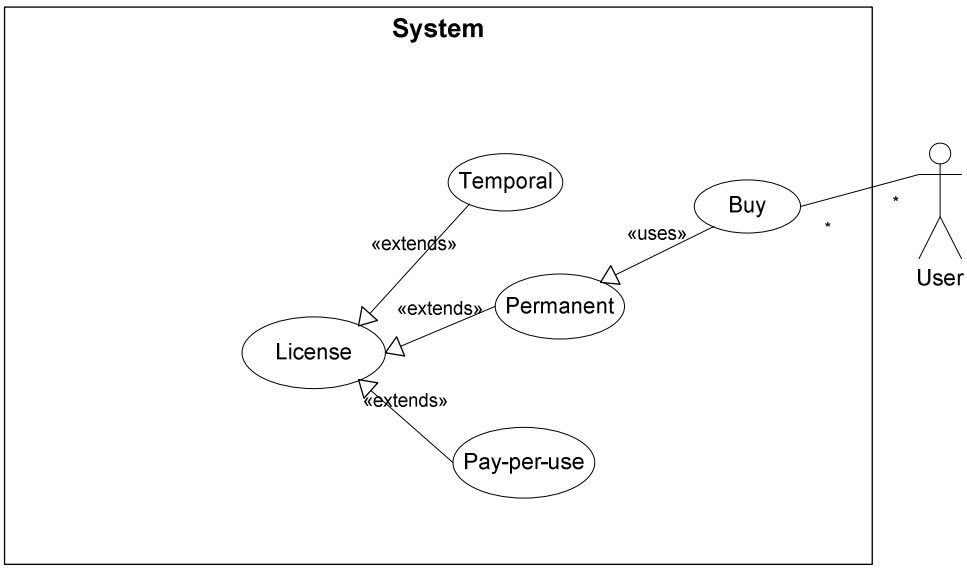


Figure 4. Use case diagram UC02 step 3: the bank buys the license

Table 13. Use case table UC02 step 3: the bank buys the license

Use Case ID	UC02-3
Use Case Name	Bank buy the license
Purpose	The Bank buy some permanent licenses
Initiator	Bank (*user*)
Primary Actor	Bank (*user*)

Additional Actors	SmartLM license manager
Description	The Bank after tested the plug-in software wants to buy some permanent licenses
Pre-condition	The plugin is registered into the License Broker and it have been tested
Post-condition	Account the license
Use Case Functionality	
Sequence	1. The bank contact the system to acquire the permanent licenses 2. The bank store the licenses locally in their own *license server*
Alternatives	
Non-functional Requirements	
Exceptions	The licenses cannot be acquired
Use Cases used	UC16: local scenario without Grid
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	**License** Transfer: Secure transfer of license token Human readable form of license token stored locally Transfer of checksum information Latency of data transfer must be taken into account

	Security: Prevent license token from misuse
Assumptions	
Open Issues	
Information Requirements	

A.3. UC03: case study multidomain access

This use case is the case study – example 3 from the SmartLM Description of Work.

A.3.1. Description

A national research agency AeroSpaceLab ASL has decided to at least partially outsource its *HPC* hardware resources to a Computational Service provider (CSP), but is still maintaining computational resources like smaller scale Linux cluster computers on department level in their own organization and company network. Furthermore research staff is using ANSYS CFX [10] on personal workstations for smaller applications and testing. On the other hand side ASL likes to maintain software licenses of ANSYS CFX in a centralized location on one single *license server* for the entire research organization and for both internal and external (CSP provided) use of the CFD software in order to ensure a most efficient use of all obtained ANSYS software licenses and high flexibility in the license maintenance related to license renewal and software version changes. Integration of the grid-based licensing mechanism from SmartLM into the ANSYS CFX software allows ASL to install all ANSYS licenses obtained by ASL on a single centralized license manager resource and to check-out necessary licenses for the use on the CSP computational resources in whatever network location from this centralized license manager. At the same time these licenses can be used on ASL's own computational facilities on a by demand basis. Overlapping software license use for in-house demands on staff workstations and small-scale clusters and on CSP's large-scale Linux clusters allows for a better and more efficient overall use of ANSYS software licenses and adds additional flexibility to outsource large computational tasks to the CSP's facilities more easily. Furthermore ASL is no longer bound to a single CSP, where a local license manager resource was installed in the local CSP's network, but ASL is now able to use on a very flexible basis services from different CSP's offering *HPC* computational services and thereby taking advantage from market competition between CSP's.

Table 14. Use case table UC03: case study multidomain access

Use Case ID	UC03
Use Case Name	DoW case study #3
Purpose	The SmartLM license manager serves applications running on different resources in different locations and domains.
Initiator	The organisation running their own SmartLM license manager
Primary Actor	SmartLM license manager
Additional Actors	*Users*, *computational resources*, SmartLM API
Description	Different members of an organisation use *licensed software* on different resources located inside and outside the organisations domain. The organisations SmartLM license manager offers license to all these resources.

Pre-condition	ANSYS CFX is adapted for the usage of the SmartLM license manager and available on all resources Resources are accessible for the *users* of the organisation and have a network connection to the SmartLM license manager.
Post-condition	The application executes, the results are made available to the *user* and accounting information is available at the accounting and billing service.
Use Case Functionality	
Sequence	1. *User* A transfers his input data to the resource at the CSP 2. *User* A starts the ANSYS CFX application on the resource at the CSP 3. ANSYS CFX requests a license from the SmartLM *license server* hosted in the domain of *user* A's organisation 4. ANSYS CFX receives a license token, validates the license token, and executes 5. *User* A's job finishes, the application calls the SmartLM API and submits the accounting information to the SmartLM license manager
Alternatives	*User* A can be replaced by *user* B and *user* C, the procedure should be the same.
Non-functional Requirements	The location/domain where the license is used does not affect the licensing mechanism. Of course, the usage of licenses must be restricted to *user*s of the organisation and resources available to the organisation.
Exceptions	At step 3 the SmartLM license manager might run out of licenses. Then the application should not execute. At step 4 the execution might take longer than the license token is valid, thus the license token needs to be renewed. This should be done either by the *licensed software* (i.e. ANSYS CFX) or a service at the *computational resource*, both will use the SmartLM API.
Use Cases used	UC06: generic application execution
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ …	

issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	Security: X509 certificates can be used to authenticate the consumer to the *license server*. support of identification and authentication using *PKI* certificates Authorization using *PKI+PMI* certificates It should support *VOMS* certificates should support *CAS* (Globus Community Authorization Service) should support *SAML* assertions License authentication only based on the userid Site policy is deciding on use of certificates or enforcing the use of certificates License renewal through SmartLM API on the *computational resource* Language bindings: FORTRAN 77, 90, 95, 2003 C89, C99 C++ Java .net Accounting and billing: The *user* receives one bill for the usage of ANSYS CFX
Assumptions	The SmartLM license manager is able to offer licenses to resources regardless of their location. The application or a SmartLM API is able to obtain a license from the SmartLM license manager.
Open Issues	
Information Requirements	

A.4. UC04: multiple license servers scenario *ASP* + user licenses

A.4.1. Description

This use case for SmartLM describes a *User* that wants use resources that are beyond the boundaries of an *Application Service Provider* (*ASP*) domain. The *ASP* is build on top of a Grid Environment therefore it runs its applications inside its grid system. Although the *ASP* has his *license server* it has limited available licenses.

Table 15. Use case table UC04: multiple license servers scenario ASP + user licenses

Use Case ID	UC04
Use Case Name	Multiple *license server*s scenario *ASP*
Purpose	The *ASP* doesn't have available license, therefore its Server License contacts another *license server* to obtain the additional licenses
Initiator	*User*
Primary Actor	*User*
Additional Actors	*User*, *ASP*, *License server*, *Grid orchestrator*
Description	
Pre-condition	The *User* has already registered to the *ASP*. The *ASP License server* knows (it has the contact point and the credential to access) where other *License server*s are. ANSYS CFX is adapted for the usage of the SmartLM license manager and available on all resources Resources are accessible for the *user's* of the organisation and have a network connection to the SmartLM license manager.
Post-condition	The application executes, the results are made available to the *user*, and accounting information is available at the accounting and billing service.
Use Case Functionality	
Sequence	1. The *User* logs to the *ASP* 2. The *User* submits jobs to the *ASP* 3. The *ASP* realizes it doesn't have available license

	4. The *ASP* requests additional license a. The *ASP*'s *License server* contacts a different *License server* b. The *ASP*'s *License server* negotiates new (maybe pay per use) licences c. It obtains new license tokens 5. *ASP* submit jobs to the *Grid orchestrator* 6. the *Grid orchestrator* finds a suitable resource and reserves a timeslot 7. the *Grid orchestrator* submits the job and the license token to the chosen resource on behalf of the *user* 8. job starts, the application verifies the license token and executes 9. job finishes, the accounting subsystem retrieve the information it need to account the execution and sends it to the license manager
Alternatives	
Non-functional Requirements	
Exceptions	The new *License server* is not available. No additional licenses available (at the moment): Can't run job.
Use Cases used	UC05: Grid submission UC01: case study ASP outsourcing UC10 ANSYS CFX Local Usage
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	

Assumptions	
Open Issues	
Information Requirements	

A.5. UC05: Grid submission

A.5.1. Description

This use case describes the scenario where a *user* submits a job to the Grid and the *user's* job executes a licensed application.

The *user* submits his job to the *Grid orchestrator*. The orchestrator cares for where and when the job is executed, allocates the computing resources and licenses for the execution.

Table 16. Use case table UC05: Grid submission

Use Case ID	UC05
Use Case Name	Grid submission
Purpose	A *user* executes a licensed application on a resource through the Grid.
Initiator	Grid *user*
Primary Actor	*Grid orchestrator*
Additional Actors	*User*, *license service*, Grid middleware, application, SmartLM license manager
Description	A *user* submits his job through a Grid user interface to a *Grid orchestrator*. The *Grid orchestrator* asks the *resource broker* for a suitable resource to execute the job, and asks the *license service* for a license token. Then the orchestrator submits the job and the license token to the chosen resource.
Pre-condition	*User* has a valid certificate for authentication at the *Grid orchestrator* and access to at least one computing resource in the Grid. The orchestrator is authorised to contact the license manager and to submit jobs on behalf of the *user* to the resource. The *user* is authorised to use the application on the resource.
Post-condition	The application executes, the results are made available to the *user*, the accounting information is available at the SmartLM license manager.
Use Case Functionality	
Sequence	1. *user* submits the job to the *Grid orchestrator* 2. *Grid orchestrator* contacts the *resource broker* 3. *resource broker* queries the resources where the *user* has access and

	where the application is available 4. *resource broker* finds a suitable resource and reserves a timeslot 5. *Grid orchestrator* negotiates with the license manager a license token for the usage of the application on behalf of the *user* 6. *Grid orchestrator* submits the job and the license token to the chosen resource on behalf of the *user* 7. job starts, the application verifies the license token and executes 8. job finishes, the *Grid orchestrator* submits the accounting information to the SmartLM license manager
Alternatives	
Non-functional Requirements	
Exceptions	
Use Cases used	UC06: generic application execution
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	*Negotiation*: Functionality that allows (to a Service Provider and a Client) negotiating an *SLA* Contract starting by an *SLA* Template. The SmartLM must allow *negotiation*. The rules/policies for *negotiation* should be expressed in *XML* format and each *VO*/vendor admin must be able to administrate them. It is mandatory to deny the *negotiation* for *users* of some countries. The *negotiation* interface must work using WS-Agreement There must exists a signed record of the final agreement The *VO*/vendor admin must be able to view/download the records. The *negotiation* should support *X.509*v4 certificates to check if *user* is

	allowed to make the contract. *SLA* Translator: allows the physically access to the documents (*SLA*-Template and *SLA*-Contract) and get (or set) information from them. For each document it will know the associated ID and it will contact the repositories in order to retrieve the appropriate document. Contract/Template-Repository: The *SLA* Template Repository allows storing and retrieving *SLA* Template documents. This service is used as a storage facility by the *SLA*-Translator. License reservation: Licenses can be reserved. License reservation should have an expiration date License manager monitoring: The *license server* should be able to provide information on license availability, reservations, usage at any point in time. This information must be provided in human readable, *XML* format to be usable in other applications. *SLA* Evaluation: After the *negotiation* phase, once instances of our License have been created, it is necessary to ensure that who provides the license observes the contractual terms (*WS Agreement*). Trusted service for *SLA* verification: Any information record should be signed by the parties – *user*/resource provider/license provider Orchestration: Job submit, job monitoring and controlling, get intermediate results, cancel unsuccessful jobs, prolongation if necessary, get final results Security: X509 certificates can be used to authenticate the consumer to the *license server*. support of identification and authentication using *PKI* certificates Authorization using *PKI+PMI* certificates It should support *VOMS* certificates should support *CAS* should support *SAML* assertions Grid middleware: Grid middleware offers advanced reservation Grid middleware publishes information about available software, operating system, hardware, ... (all necessary information that is needed for the license *negotiation*)
Assumptions	License manager has licenses available

Open Issues	
Information Requirements	

A.6. UC06: generic application execution

A.6.1. Description

This use case describes the execution of a generic licensed application on a resource. The application needs to validate a license token.

It is part of the use case UC05 (Grid submission).

Table 17. Use case table UC06: generic application execution

Use Case ID	UC06
Use Case Name	Generic application execution
Purpose	Validation of a license token
Initiator	Grid *user*
Primary Actor	Application
Additional Actors	*User*, SmartLM license manager, SmartLM API
Description	A licensed application is requested to execute. It needs to check whether the provided license token is valid or not.
Pre-condition	SmartLM license manager signed a license token for the application. The license token is available on the *computational resource* where the application should be executed.
Post-condition	The application executed successfully, the output data is available to the *user*, the SmartLM license manager is informed about the license usage.
Use Case Functionality	
Sequence	1. application starts execution with access to the license token 2. application calls the SmartLM API to validate the license token 3. the license token is valid, the application proceeds execution 4. application finishes execution and calls SmartLM API to inform the license manager about the license usage
Alternatives	1a: before the start of the application, the *RMS* checks the availability of the license token and if none is available it calls the SmartLM API to obtain a license token from the SmartLM license manager

Non-functional Requirements	
Exceptions	1. The execution fails (e.g. because of missing or wrong input data); the license manager should be immediately informed. 2. the license token is not valid; the application does not execute 3. the license token is valid only for a certain time; if the time is exceeded, the application should call the SmartLM API to get a new license token
Use Cases used	
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	**License consumer** Language bindings: FORTRAN 77, 90, 95, 2003 C89, C99 C++ Java.net
Assumptions	License manager offers license reservation The license token contains all necessary information that the application needs for execution
Open Issues	
Information Requirements	

A.7. UC07: topology optimization of rear axle

A.7.1. Description

A typical example of daily work should be used to demonstrate the SmartLM capabilities on grid resources. We propose the optimization of a truck rear axle. An application *user* has a mixture of own soft- and hardware at different locations which are often not powerful enough to full fill the time schedule.

Table 18. Use case table UC07: topology optimization of rear axle

Use Case ID	UC07
Use Case Name	Topology Optimization of rear axle
Purpose	Validation of SmartLM framework and SmartLM enabled application
Initiator	Grid *user*
Primary Actor	SmartLM framework
Additional Actors	*User*, license manager, SmartLM API, application
Description	The following possibilities are available: 1. use of own resources 2. get additional license from *ISV* 3. use *ASP* resources
Pre-condition	Assist decision • Offer must be based on job characteristics and requested resources • Collection of possible solutions (own, *ISV*, *ASP*) • Best price/ fast turnaround/ mixed compromise • Assistance in decision making for various scenarios • Sign a contract with the involved partners
Post-condition	
Use Case Functionality	
Sequence	Transparent decision • Make decision parameters transparent - what is the influence of each parameter

	• Decide on best price • Best performance ratio • Elapsed time • Turnaround time Submit calculation • Generate license token based on contract • License token transfer only if contract is verified • Submit job • Latency due to data transfer (data amount typically <500 MByte) • Job monitoring and controlling, detailed information is needed • Get intermediate results • Cancel unsuccessful jobs • Prolongation if necessary (container task) • Get final results (data amount some GByte) Get accounting/ billing information • Job statistics for all involved partners • Information on resources used
Alternatives	
Non-functional Requirements	
Exceptions	
Use Cases used	UC05: Grid submission UC06: generic application execution
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by	

SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	

Particular Requirements	*License server*
	• Orchestration: Job submit, job monitoring and controlling, get intermediate results, cancel unsuccessful jobs, prolongation if necessary, get final results
	• Statistical information must include at least the following information about the license checkouts and reservations:
	  • Who
	  • When
	  • Where (*IP*/Hostname)
	  • *Features* (ANSYS checks out different *features* at the same time, e.g. solver, parallel, number of processes, combustion models, multiphase models …)
	  • Quantity
	
	Content:
	• Modules (functionality i.e. linear static, ...)
	  • Start date/time
	  • Expiration date/time
	  • Number of CPUs used
	  • Duration of granted usage, in case of flexible start date/time
	• Accounting information should be send back after job completion in the token
	
	Transfer
	• Secure transfer of license token
	• Human readable form of license token stored locally
	• transfer of checksum information
	• Latency of data transfer must be taken into account
	
	Security
	• Prevent license token from misuse
	  o coupling of data and license token by signing

	○ License token can only be used once
	○ Guarantee of integrity of license token and transferred job data
	
	Policy enforcement in the applications
	• Location and Discovery: The user must be able to make a decision on the best available license resource
	
	Middleware
	• Orchestration - Job submit, job monitoring and controlling, get intermediate results, cancel unsuccessful jobs, prolongation if necessary, get final results
	
	Accounting and billing
	• Usage statistic
	  • Who/ when/ where
	  • Functional modules used
	  • Date/time
	  • Number of CPUs used
	  • Model Size (unknowns, nodes
	• Accounting information should be send back after job completion in the token
	  • Billing statistic
	  • Who/ when/ where
	  • Date/time
	  • Amount
	  • Balance

Assumptions	Code integration • Minimize *ISV* effort for software integration • Transfer of license token to check for expiration • No other logic to be included in existing code • Business logic must be done by container Language Binding • FORTRAN 77 binding
Open Issues	

Information Requirements	

A.8. UC08-UC12: ANSYS CFX use cases

A.8.1. Description

The ANSYS CFX [10] Use Cases describes the usage of the new SmartLM system. The use cases "A.8.2 UC08 Pay per Use Scenario", "A.8.3 UC09 "Standard License Approach" in Grid Environments" and "A.8.4 UC10 ANSYS CFX Local Usage" cover the ANSYS CFX usage with respect to different business models like "pay per use in Grid environment" and "Annual & Perpetual Licenses in Grid and local environments". Use case "A.8.5 UC11 Job Submission" is a (short) description of the requirements for the ANSYS CFX job submission in Grid environments. Use case "A.8.6 UC12 License System Administration" is a (short) description for the requirements for the license administration. The use cases are:

A.8.2 UC08 Pay per Use Scenario

A.8.3 UC09 "Standard License Approach" in Grid Environments

A.8.4 UC10 ANSYS CFX Local Usage

A.8.5 UC11 Job Submission

A.8.6 UC12 License System Administration

A.8.2. UC08 Pay per Use Scenario

Table 19. Use case table UC08: ANSYS Pay per Use Scenario

Use Case ID	UC08
Use Case Name	ANSYS Pay per Use Scenario
Purpose	Run a pay per use scenario in a Grid environment
Initiator	*User*
Primary Actor	*User*
Additional Actors	*Independent Software Vendor* (*ISV*), *ASP*, *Licensed software*, *Grid orchestrator*, Resource Manager, *HPC resource*, SmartLM License Manager
Description	The *Simulation engineer* wants to use licenses provided directly from *ISV* in a Grid environment
Pre-condition	• A license agreement between the *ISV* and the *User's* Company exists • User company buys ordered licenses for a specific time duration • An agreement between the *ASP* and the *user's* company about the maximum budget for resource usage exists • An account for the *user* at *ASP* exists

	• User is on whitelist of ISV
Post-condition	
Use Case Functionality	
Sequence	1. *User* log on to the server of an *ASP* 2. License manager provides information (optional) about license availability (features, remaining time …) from *Independent Software Vendor* and available budget on HPC resource 3. *User* submits a job 4. *Grid orchestrator* sends the job with valid license to the *HPC resource* 5. Resource manager starts the *licensed software* with the booked resources. Time booking at start time of the *licensed software. User* Company does not want to pay for data copying time 6. *Licensed software* finished normally: a. Check in license to the license manager b. Book 'real' license usage time, necessary for accounting and billing c. Recalculate the remaining time for the available licenses d. Book 'real' hardware resource time, necessary for accounting and billing e. Notify *Simulation engineer*
Alternatives	
Non-functional Requirements	• Support multiple *features* at the same time, e.g. solver + combustion + multiphase • Support FLEXnet licenses also • Support multiple *license server* locations • Allow monitoring of overdraft licenses. Overdraft licenses allow *User* Companies more flexibility, if a time limited license shortage occurs. Example: a. Company buys 5 licenses b. Company get additional 2 licenses c. *License service* should be able to monitor the usage of the two additional licenses. If the additional licenses are used more than specified percentage, the *User* Company will pay for it.

Exceptions	Sequence 1: Not enough licenses available • Notify *user* Sequence 2: Error during application runtime: • Check in license to the license manager • Book 'real' license usage time • Book 'real' hardware resource time • Notify *user*
Use Cases used	UC11 Job Submission
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	Scenario 1 for Accounting and Billing : ASP has access to 'unlimited number' of licenses (Hosted at ISV or ASP): • ASP offers application licenses and HPC resources • SmartLM License Manager monitors usage • ASP creates one bill (license cost and hardware resources cost) and sends this to the user's company • Budget control for end users necessary, e.g. limit costs per run, per department of end user company, per month • Budget information (Original budget, left amount …) must be available for user, ASP and ISV • Allow definition of discounts for different user companies • ASP pays ISV on the basis of real usage o If ISV hosts the license, he directly gets information from SmartLM license manager about real usage o If ASP hosts the SmartLM license server, ASP has to provide detailed usage data. Usage statistics needs

	protection against manipulation o Redundant system to avoid real usage statistic data loss Scenario 2 for Accounting and Billing : ASP as reseller: • ASP buys licenses from ISV • ASP offers application licenses and HPC resources • SmartLM License Manager monitors usage • ASP creates one bill (license cost and hardware resources cost) to the user's company • Budget control for end users necessary, e.g. limit costs per run, per department of end user company, per month
Further Information	
Particular Requirements	• SmartLM API interface to Fortran 77 and Fortran 90 • Exact localization of *user* that starts the job
Assumptions	
Open Issues	
Information Requirements	

A.8.3. UC09 "Standard License Approach" in Grid Environments

Table 20. Use case table UC09: ANSYS Standard License Approach in Grid Environments

Use Case ID	UC09
Use Case Name	ANSYS "Standard License Approach" in Grid Environments
Purpose	Run application software in a Grid Environment with licenses from Customer Engineer's company
Initiator	*User*
Primary Actor	*User*
Additional Actors	*Licensed software*, *Grid orchestrator*, Resource Manager, *HPC resource*, SmartLM License Manager
Description	The *user* wants to use licenses from his LAN in a Grid environment.
Pre-condition	• A license agreement between the *ISV* and the *User* Company of the *user* exists. The *license service* is located at the company of the *Simulation engineer* • An agreement between the *ASP* and the company of *user* about the maximum budget for resource usage exists • An account for the *user* at *ASP* exists
Post-condition	
Use Case Functionality	
Sequence	1. *User* log on to the server of an *ASP* 2. License manager provides information (optional) about license availability from *Independent Software Vendor* and available budget 3. *User* submits a job 4. *Grid orchestrator* sends the job with valid license to the *HPC resource* 5. Resource manager starts the *licensed software* with the booked resources. Time booking at start time of the *licensed software*. User Company does not want to pay for data copying time 6. *Licensed software* finished normally: a. Check in license to the license manager b. Book 'real' license usage time, necessary for accounting

	and billing c. Book 'real' hardware resource time, necessary for accounting and billing d. Notify *Simulation engineer*
Alternatives	
Non-functional Requirements	• Support multiple *features* at the same time, e.g. solver + combustion + multiphase • Support FLEXnet licenses also • Support multiple *license server* locations • Allow monitoring of overdraft licenses. Overdraft licenses allow User Companies more flexibility, if a time limited license shortage occurs. Example: a. Company buys 5 licenses b. Company get additional 2 licenses c. *License service* should be able to monitor the usage of the two additional licenses. Dependent on usage of the 2 additional licenses the User Company
Exceptions	Sequence 1: Not enough licenses available • Notify *user* Sequence 2: Error during application runtime: • Check in license to the license manager • Book 'real' license usage time • Book 'real' hardware resource time • Notify *user*
Use Cases used	UC11 Job Submission UC12 License System Administration
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by	

SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	• SmartLM API interface to Fortran 77 and Fortran 90 • Exact localization of *user* that starts the job
Assumptions	
Open Issues	
Information Requirements	

A.8.4. UC10 ANSYS CFX Local Usage

Table 21. Use case table UC10: ANSYS CFX Local Usage

Use Case ID	UC10
Use Case Name	ANSYS CFX Local Usage
Purpose	Run application software in a local environment
Initiator	*User*
Primary Actor	*User*
Additional Actors	*Licensed software*, SmartLM License Manager
Description	The *user* wants to use licenses in his local LAN on internal workstations or clusters
Pre-condition	A license agreement between the *ISV* and the company of the *Simulation engineer* exists. The *license service* is located at the company of the *user*
Post-condition	
Use Case Functionality	
Sequence	1. *User* directly starts the *licensed software* 2. *Licensed software* checks out license from license manager at start time 3. Application finished normally: a. Check in license to the SmartLM License Manager b. Book 'real' license usage time
Alternatives	
Non-functional Requirements	Support multiple features at the same time, e.g. solver + combustion + multiphase
Exceptions	Sequence 1: Not enough licenses available • Notify *user* Sequence 2: Error during application runtime:

	• Check in license to the license manager • Book 'real' license usage time • Notify *user*
Use Cases used	UC12 License System Administration
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	SmartLM API interface to Fortran 77 and Fortran 90
Assumptions	
Open Issues	
Information Requirements	

A.8.5. UC11 Job Submission

Table 22. Use case table UC11: ANSYS Job Submission

Use Case ID	UC11
Use Case Name	ANSYS Job Submission
Purpose	Describe requirements for ANSYS CFX Job Submission
Initiator	*Grid orchestrator*
Primary Actor	*Grid orchestrator*
Additional Actors	*Licensed software*, *HPC resource*
Description	*Grid orchestrator* submits an ANSYS CFX job to a Grid Resource
Pre-condition	• Licenses are available • Hardware resources are available • *Grid orchestrator* send a job to the resource manager
Post-condition	
Use Case Functionality	
Sequence	1. Copy data from local *user* directory to *HPC resource*: a. ANSYS CFX Definition File or ANSYS CFX Result File b. ANSYS CFX Result File for Initialisation (optional) 2. Translate start options into ANSYS CFX (*Licensed software*) start command • cfx5solve - 3. Copy intermediate data during run time data from hardware resource to local *user* directory (necessary for solution monitoring) 4. After ANSYS CFX finished, copy result data to local *user* directory
Alternatives	Alternative 1: Request ANSYS CFX stop (Change runtime parameters). Continue with 4. Alternative 2: 3.1 Request ANSYS Backup file. Continue with 3. and 4.

Non-functional Requirements	
Exceptions	
Use Cases used	
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	SmartLM API interface to Fortran 77 and Fortran 90
Assumptions	
Open Issues	
Information Requirements	

A.8.6. UC12 License System Administration

Table 23. Use case table UC12: ANSYS License System Administration

Use Case ID	UC12
Use Case Name	License System Administration
Purpose	Describe Requirements for License Administration
Initiator	*License system administrator*
Primary Actor	*License system administrator*
Additional Actors	
Description	License Administration via Web Based Interface
Pre-condition	• Valid licenses are available
Post-condition	
Use Case Functionality	
Sequence	1. Start *license service* 2. Check status 3. License reservation for different *users*, groups … 4. Stop *license service* 5. Get usage statistics
Alternatives	
Non-functional Requirements	
Exceptions	
Use Cases used	
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/	

security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	
Assumptions	
Open Issues	
Information Requirements	

A.9. UC13: demo license scenario

A.9.1. Description

This document describes the use cases for a Demo License scenario and it is included into the Application *ASP* scenarios.

A Generic *user* uses a Test or Evaluation license of an application.

Table 24. Use case table UC13: demo license scenario

Use Case ID	UC13
Use Case Name	Demo license scenario
Purpose	Test a licensed application
Initiator	*User*
Primary Actor	Application
Additional Actors	SmartLM license manager, Computer resource
Description	A generic *user* uses an evaluation or trial license of an application
Pre-condition	The application is registered in the SmartLM License Manager The application has tokens for trial execution
Post-condition	
Use Case Functionality	
Sequence	1. *User* contact the SmartLM License Manager and acquires a trial license 2. The SmartLM License Manager registers the license usage. 3. *User* submits the application with license to a computer resource for execution 4. Application starts execution 5. application verifies license trial token 6. application executes normally 7. application finishes and inform the SmartLM License Manager about accounting information 8. The SmartLM License Manager registers the accounting information.

Alternatives	
Non-functional Requirements	
Exceptions	• the demo license token is not valid or cannot be verified • the *user* is not allowed to use the license • the demo license is only valid for a certain time, and this time expires while the application is still running or has not started. • The application needs more resources that allowed by the trial license. • The application can not inform the SmartLM License Manager about accounting information • The SmartLM License Manager cannot check out the license because there are no tokens available.
Use Cases used	
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business Driven sequence variations	
Business driven sequence variations	
Further Information	
Particular Requirements	**License consumer (application)** Location and Discovery: The consumer must be able to make a decision on the best available license resource Network: Connection to LM Performance, scalability, redundancy: Application must confirm the sending of the accounting record

	The accounting record must also be stored locally in a file when demanded **License** Transfer: Secure transfer of license token Human readable form of license token stored locally Transfer of checksum information Latency of data transfer must be taken into account Security: Prevent license token from misuse • Coupling of data and license token by signing • License token can only be used once • Guarantee of integrity of license token and transferred job data Content: Modules (functionality i.e. Linear static ...) • Start date/time • Expiration date/time • Number of CPUs used • Duration of granted usage, in case of flexible start date/time • Accounting information should be send back after job completion in the token
Assumptions	The demo license is free of charge
Open Issues	
Information Requirements	

A.10. UC14: ASP environment

A.10.1. Description

This Use Case for SmartLM describes the use of the SmartLM in an Application Service Provider (*ASP*) environment. The *ASP* in this case is only a provider of on-demand hardware resources for customers. The customers in the scenario are responsible for obtaining their own licenses from the vendor of the product for use at the *ASP* location. This use case describes the situation where the *ASP* can have multiple customers who want to use the same application with their private license sets provided by the application vendor.

Customers of the *ASP* who belong to the same organization obtain their own private license sets from an application vendor. These licenses are hosted at the *ASP* location for their private use. These license sets are for utilizing the hardware/environment of the *ASP* to run the software product. The *user groups* are only provided with a URL/host:port to access their private license sets. Even if they obtain the license access point information of another group, the SmartLM service will not allow them to use others licenses

Table 25. Use case table UC14: ASP environment

Use Case ID	UC14
Use Case Name	Multiple groups scenario - one *license server* hosts licenses for the same application belonging to different *user groups*
Purpose	License *feature* sets for a particular application belonging to different groups, must be able to coexist in their own security context on the same *License service* without getting aggregated.
Initiator	*User* or a Group of *Users* who are authorized to use the same type of license features.
Primary Actor	*User* or a Group of *Users* who are authorized to use the same type of license features.
Additional Actors	*ISV* of the *license protected software.* *ASP* which hosts *user* license features and applications for running on their *computational resource.*
Description	1. Different Groups Of *users* must be able to host their license features on the same physical machine (preferably on the same *license service* process), even if these license features are for the same *license protected software.* 2. One Group of *Users* must not be able to access the license features belonging to another *user group* and vice-versa. 3. If two groups host license features for the same *license protected software* on the same *License service*, the license features must not be

	aggregated but remain separated for use by the group which owns each set of features.
Pre-condition	• SmartLM up and running. • Two *user groups* not related to each other in the system where the license features will be used. • Each *user group* having their own license *feature* set, but provided by the same *ISV* for the same *license protected software*, i.e. each *feature* has the same *feature* name and could have the same or different *feature* counts.
Post-condition	• Both *user groups* can run the software product • Both *user groups* get their licenses from the same *license service*. • The *license service* prevents one group from accessing license features belonging to the other group and vice-versa. • Only one *license service* instance runs on the physical machine designated to be the *license server* i.e. no individual *license server* for each group is started. • The *License server* can at any point in time provide fine grained usage statistics for each group of users accessing their private license *feature* sets.
Use Case Functionality	
Sequence	1. *ISV* develops a product which uses SmartLM licensing. 2. *ISV* allows ASPs to host their product for hardware on demand *users* and generate license *feature* keys for customers in a way that it can be hosted at the *ASP* location. 3. *User Group* A decides to use the product at an *ASP* location and sends their license file to the *ASP*. 4. *User Group* B also decides to use the same application with the same set of features as user group A at the same *ASP* location. Their license with the same set of features is also sent to the *ASP*. 5. The *ASP* has one instance of the SmartLM which hosts both groups' license features in their own security context. 6. *User Groups* A and B use the same *license protected software* at the same *ASP* location, use the same set of license features from the same SmartLM *license service* but access their own private license *feature* sets. 7. SmartLM generates fine grained *feature* usage statistics for each *user group* individually which is used by the accounting system.
Alternatives	NONE

Non-functional Requirements	NONE
Exceptions	
Use Cases used	NONE
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	The need to get private license *feature* sets which are identical but for different groups from the same vendor would be unnecessary if the ISVs are able to provide the *ASP* with a pool of licenses which can be rented out by the *users*, instead of the *users* having to obtain/buy a license directly from the *ISV*.
Business Driven sequence variations	
Business driven sequence variations	NONE
Further Information	
Particular Requirements	• Ability to host licenses and features from multiple vendors in one instance of the *License server* • Two companies hosted by one *ASP*: how to properly separate the license usage • The *license server* should be able to any point in time provide information on license availability, reservations, usage. This information must be provided in human readable, *XML* format to be usable in other applications. • Ability to have redundant *license server*s serving the same set of features.
Assumptions	• SmartLM can host multiple sets of the same license features owned by different groups on the same *license service* instance. • SmartLM can provide usage statistics about each set of similar license features. • SmartLM can handle access permissions for each set of similar license *feature* sets individually.
Open Issues	NONE

Information Requirements	NONE

A.11. UC15: multiple applications on one license server

A.11.1. Description

This use case describes the use of the SmartLM for hosting license *feature* sets belonging to different *Independent Software Vendors* (ISVs) on the same service.

Table 26. Use case table UC15: multiple applications on one license server

Use Case ID	UC15
Use Case Name	Multiple application licenses from different vendors on one *license server*
Purpose	The SmartLM *license service* must be able to host license *feature* sets belonging to different ISVs on the same service.
Initiator	SmartLM license administrator.
Primary Actor	SmartLM license administrator.
Additional Actors	*ISV* of the *license protected software*.
Description	ISVs provide license *feature* sets for their applications to be used. All these *feature* sets must be able to coexist on the same SmartLM *license service* and must be controllable and accountable individually without interfering with the functioning of the *license service*.
Pre-condition	• SmartLM up and running. • Multiple ISVs providing SmartLM compatible license *feature* sets. • A SmartLM *license service* administrator who wants to host these license *feature* sets on the same SmartLM *license service*.
Post-condition	• The license *feature* sets belonging to different ISVs can be renewed without interfering with the SmartLM *license service* i.e. without having to start/stop the *license service* and without interrupting other applications which have already checked out licenses from the SmartLM *license service*. • New license *feature* sets belonging to different ISVs can be introduced for hosting on the SmartLM *license service* without interrupting it, i.e. without having to start/stop the *license service* and without interrupting other applications which have already checked out licenses from the SmartLM *license service*. • Access control can be established for individual license features on

	the SmartLM *license service* even if they belong to different ISVs. • Only one *license service* instance runs on the physical machine designated to be the *license server* i.e. no individual *license service* for each *ISV* is started. • The *License service* can at any point in time provide fine grained usage statistics for each license *feature* it hosts.
Use Case Functionality	
Sequence	1. *ISV* develops a product which uses SmartLM licensing. 2. *ISV* provides a license *feature* set for its *license protected software* to a SmartLM license administrator. 3. The SmartLM license administrator loads these new license features into the organization's SmartLM *license service* without having to interrupt the service. 4. The license *feature* set expires after a specific time period and the usage of the product is stalled. 5. The *ISV* provides the license administrator with a new license which is used to renew the license *feature* set on the SmartLM service without interrupting it. 6. The *license system administrator* is able to obtain license *feature* set usage information for different *license protected software*, belonging to different ISVs from the same service.
Alternatives	NONE
Non-functional Requirements	NONE
Exceptions	NONE
Use Cases used	NONE
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	NONE
Business Driven sequence variations	

Business driven sequence variations	NONE
Further Information	
Particular Requirements	• Ability to update licenses for a particular *feature* without having to restart the *license server*. • The *license server* should be able to any point in time provide information on license availability, reservations, usage. This information must be provided in human readable, *XML* format to be usable in other applications. • Statistical information must include at least the following information about the license checkouts and reservations: • Who • When • Where (*IP*/Hostname) • Features (ANSYS checks out different features at the same time, e.g. solver, parallel, number of processes, combustion models, multiphase models …) • Quantity • Ability to have redundant *license server*s serving the same set of features.
Assumptions	NONE
Open Issues	NONE
Information Requirements	NONE

A.12. UC16: local scenario without Grid

A.12.1. Description

This Use Case 3 for SmartLM describes the use of the SmartLM for hosting license *feature* sets in a standalone *HPC* environment. This means that there is no grid middleware involved in this Use case. It is important to note that all computation nodes are on a separate internal *HPC* network only accessible from the login node.

Table 27. Use case table UC16: local scenario without Grid

Use Case ID	GRIDOCRE_03
Use Case Name	Local scenario without grid.
Purpose	The SmartLM *license service* must be able to function on a standalone *HPC resource* without using any grid middleware or connection to any service external to the *HPC resource*.
Initiator	*User* of the *HPC resource*
Primary Actor	*User* of the *HPC resource*
Additional Actors	*DRM*, SmartLM *license service*, *HPC resource*
Description	In a typical enterprise *HPC* system, *users* submit jobs directly to a local *DRM* and have their *user* accounts local to the *HPC resource* i.e. the *user* is identified by his/her login name into the system, it doesn't matter how the operating system authenticates the *user*. The authentication mechanism employed by the operating system could be /etc/passwd, LDAP, ADS etc.
Pre-condition	• SmartLM *License service* running on a designated *license server* machine internal to the cluster and hosting license *feature* sets for all applications installed on the cluster. • A *HPC resource* having the applications installed typically in a shared file system, This resource is internal to the organization with no grid middleware or access to external networks. • A *DRM* running on the *HPC resource* responsible for job dispatch on the *HPC resource*. • *Users* having login accounts to the *HPC resource*.
Post-condition	• The *DRM* scheduler can keep track of license availability from the SmartLM service and use that information when going through a job dispatch iteration.

	• The SmartLM service can provide fine grained statistical usage information based on local *user* account names.
Use Case Functionality	
Sequence	1. An organization sets up an internal *HPC resource* with one master node, one designated *license server* machine and multiple computation nodes. 2. A *DRM* is setup on the *HPC resource* to handle job submissions. 3. The SmartLM *license service* is set up on the designated *license server* machine serving licenses for all applications installed on the shared file system of the *HPC resource*. 4. *Users* login directly into the master node of the *HPC resource* using their local cluster username and password. 5. *Users* create a *DRM* compatible script which launches a simulation software they want to use and request for specific license features which those applications need. They submit the script to the *DRM* using *DRM* specific commands like qsub, bsub etc. 6. The *DRM* checks with the SmartLM *license service* about availability of the requested license features and performs either sequence a or b a. Requested resources available • Schedules the job if the resources are available. • The application runs and checks out licenses from the SmartLM service • Application terminates releasing the licenses • Licenses get checked back to the SmartLM service. • *User* job leaves the queue b. Requested resources unavailable • The job is queued till the next scheduler run. The sequence then loops through till sequence a. above is satisfied. 7. The SmartLM *license service* administrator obtains license *feature* set usage information for different software products, sorted by local *user* account names.
Alternatives	NONE
Non-functional Requirements	NONE
Exceptions	NONE

Use Cases used	NONE
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	NONE
Business Driven sequence variations	
Business driven sequence variations	NONE
Further Information	
Particular Requirements	• Possible notification based interface to tell the *Distributed Resource Manager* when a license is checkout/reserved or checked back in. • Interface exposed should ideally be platform independent meaning interface bindings should exist for the Linux world and the Microsoft world. • Possibility to run the *license server* as a standalone application listening on a certain port, for single cluster use. • Will it be possible to use the *license server* without having to set up the Tomcat Servlet container? • If the *license server* will have the *feature* of just listening on a *TCP* port , we should think of getting the port registered as an *IANA* port. • The *license server* in standalone mode i.e. listening on just a single port, should be able to handle concurrent incoming connections. • Ability to have redundant *license server*s serving the same set of features. • The *license server* should be able to handle clients which have crashed. If the client crashes without checking the license back in, the *license server* within a reasonable amount of time should determine this and mark the license as checked in. As an additional option, some tool must be provided which either the administrator or the *user* whose crashed application had checked out a license can use to put the license back in. In the case where the license is checked back in, the *license server* must again make sure that process is actually force killed in case it still exists.
Assumptions	It is assumed that there will be an interface in SmartLM to connect

	DRMs. However this assumption does not reduce the credibility of this use case. Even if no such interface exists, the *users* must be able to run applications on local *HPC resource*s using SmartLM licensing without using any grid middleware or external services. The only requirement should be the *user* having an account on the master node which is exported to the computation nodes.
Open Issues	NONE
Information Requirements	NONE

A.13. UC17: License aggregation

A.13.1. Description

This use case describes the use of licenses aggregated from an *ASP*'s SmartLM *license server* and local private licenses which a *user* might already possess.

Table 28. Use case table UC17: License aggregation

Use Case ID	UC17
Use Case Name	License Aggregation
Purpose	The *user* must be able to use part of his/her license *feature* counts at an *ASP* location, and rent the additional license *feature* counts required from the *ASP*'s public *license service*.
Initiator	*ASP* SmartLM service
Primary Actor	*ASP* SmartLM service
Additional Actors	*User*, *ASP*, *User's License service*, *ASP*'s public SmartLM service
Description	A *user* might have a license to run an application on a limited number of processors. At some point the *user* might need to use a very high number of CPUs to run a job. This is possible at an *ASP*'s site. However the licenses which the *user* has, are not enough for all the processors which the job will use on the *ASP* resource. The *user* might rent out the difference in the number of license tokens required, from the *ASP*. The *user* wants to pay only for the license tokens required for using the extra processors. The *ISV* wants to make sure the *user* cannot make use of their private licenses at two locations simultaneously.
Pre-condition	• The *ASP* has a public *license service* and a contract with an *ISV* to rent out licenses to *users* of the *ASP* hardware. • A *user* having a private *license server* for running the *ISV*'s *license protected software*. • The *user* wants access to *HPC resources* which are not available at the *user* location.
Post-condition	• The private license *feature* sets of the *user* are locked during the period the application is running at the *ASP* location. • The *user* only needs to pay for the extra licenses used at the *ASP* location meaning if the *user* already had n licenses and used m

	licenses at the *ASP* location where m>n ,the *user* only pays for m-n.
Use Case Functionality	
Sequence	1. *User* obtains **n** licenses to run an *ISV* application and sets up a private *license server*. This entitles the *user* to run the *license protected software* on **n** processors. 2. *User* is working on a large project and requires a large number of processors to run the application. These processors are unavailable at the *user* location. Let's say the number of processors required is **m** where **m>n.** This means the *user* requires **m-n** extra licenses to run the *license protected software*. 3. The *user* decides to use the *ASP*'s on demand hardware and rent out licenses to run the *license protected software*. 4. The *user* runs the n on the *ASP* resource using **m** licenses and **m** processors. 5. The **m** licenses are checked out from the *ASP*'s SmartLM *license service*. 6. The *ASP*'s SmartLM *license service* contacts the *user's license service* and locks all the **n** licenses from use, making them unusable. 7. The *user's* job terminates releasing all the **m** licenses. 8. The *ASP*'s SmartLM *license service* unlocks the **n** licenses from the *user's license service* making them usable. 9. The *user* only pays for utilizing the hardware and using **m-n** licenses at the *ASP* site.
Alternatives	NONE
Non-functional Requirements	NONE
Exceptions	NONE
Use Cases used	NONE
Technologically Driven sequence variations	
Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	NONE

Business Driven sequence variations	
Business driven sequence variations	NONE
Further Information	
Particular Requirements	• The *license server* must be able to aggregate user licenses from different instances of SmartLM and/or FLEXnet. For example, a user must be able to use a certain number of licenses from a private *license server* at some other site, at the same time getting those licenses invalidated for use during that time period. This way the ISVs can allow the user to use their personal licenses at different sites without the fear of the user having access to more licenses than what they paid for. Also the users can use their existing licenses at an *ASP* location and pay only for the extra number of licences they require to run on more CPUs.
Assumptions	• It is assumed that there is some way for the *ASP* SmartLM service to contact the user *license service*. Even if this is not true there can be some kind of agent which can be run at the User location to lock the licenses. Maybe a dummy job?
Open Issues	NONE
Information Requirements	NONE

A.14. UC18: ASP end-user

A.14.1. Description

The T-SYSTEMS Use Case describes the usage of the new SmartLM system. The use covers the T-SYSTEMS usage with respect to the end-users view in an Application Service Provider (ASP) scenario.

The use case is:

- T-SYSTEMS ASP scenario

Table 29. Use case table UC18: ASP end-user

Use Case ID	UC18
Use Case Name	T-SYSTEMS ASP Scenario
Purpose	Run an ASP scenario in a Grid environment
Initiator	Simulation Engineer
Primary Actor	Simulation Engineer
Additional Actors	User company, Independent Software Vendor, Application Service Provider, License Administrator, System
Description	The Simulation Engineer wants to use licenses, software and resources provided from the ASP in a Grid environment
Pre-condition	• A license agreement between the Independent Software Vendor and the User Company of the Simulation Engineer exists • An agreement between the Application Service Provider and the company of the Simulation Engineer about the maximum budget for resource usage exists (where resource here can refer to both hardware and licenses) • Typically also a Service Level Agreement exists between ASP and user company.
Post-condition	•
Sequence	1. Simulation Engineer logs on to the system 2. Simulation Engineer submits a job. The job description has to include an accounting context in order to support a cost-unit base

	accounting. 3. System checks whether user is allowed to use the ISV software. 4. System checks if there is enough budget available for the stated accounting context (license budget check only). 5. System iterates over a predefined list of license servers (or asks a license broker) until the required licenses are booked for the timeslot agreed upon by the orchestration service. 6. System starts the application with the booked resources (licenses and hardware resources) 7. Application finished normally: a. Check in license to the system b. Write accounting record: 'real' license usage time, necessary for accounting and billing, localization, features, model size, iterations. c. Notify Simulation Engineer 8. System aggregates license accounting information and consolidates the raw accounting information based upon agreed upon policies (Accounting context, SLA, historical usage records, etc) 9. System returns billing information to simulation engineer, including amount and current balance. 10. System issues bill for user company upon request. 11. System issues payment for
Alternatives	
Non-functional Requirements	A rule engine is used in order to support different policies in a flexible way.
Exceptions	Sequence 1: Not enough licenses available • Notify Simulation Engineer Sequence 2: Error during application runtime: • Check in license to the "server" • Book 'real' license usage time • Book 'real' hardware resource time Notify Simulation Engineer
Use Cases used	

Sequence variations based on unsolved administrative/ security/ architectural/ … issues that are NOT addressed by SmartLM	
Business driven sequence variations	
Particular Requirements	•
Assumptions	•
Open Issues	
Information Requirements	

Annex B. Non-functional requirements

Table 30. Non-functional requirements

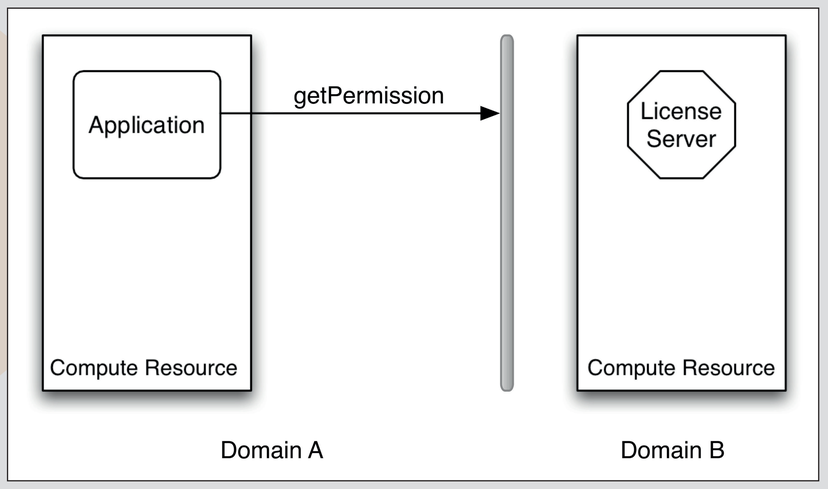
No	Description	Level	Business relevance	Referring use case
84.	Ability to have redundant *license server*s serving the same set of *features*.			UC14, UC15
85.	If the license server will have the feature of just listening on a TCP port, we should think of getting the port registered as an IANA port.			UC16
86.	Code integration: Minimize ISV effort for software integration			UC07
87.	It must run on LINUX. It should run on Debian, Scientific Linux, Suse and RedHat. It should run in MacOSX, UNIX and Windows.			
88.	It should be compiled with GNU compilers			
89.	It should be fault-tolerance (i.e., it must recovery from faults and/or allow replicated servers – in this case, with coordination)			
90.	It should run on virtual machines (including on-the-fly migration but without cloning). It should run at least in Xen and VMware.			
91.	Stand-alone solution: bundle it e.g. Jetty like UNICORE 6			

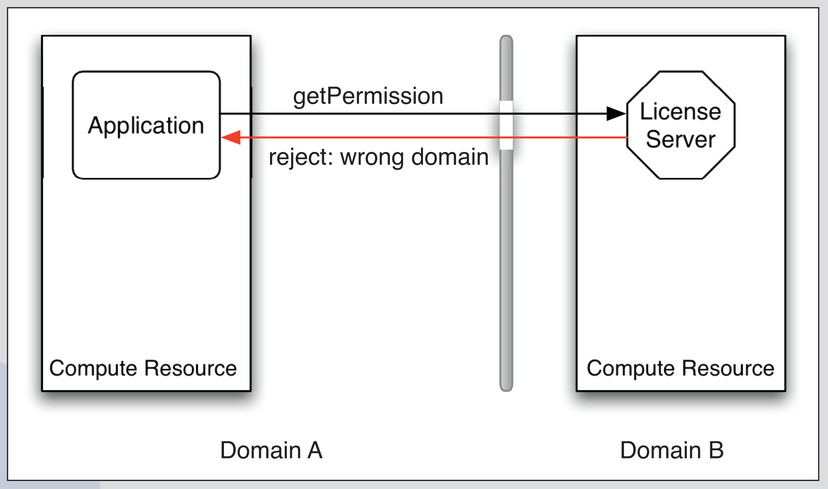
Managing and Providing Software Licenses for Grids and Clouds

SmartLM

The Problem

- IT Infrastructure paradigms have been changing over the last years to support more flexibility and reduce costs at the same time.
  - Grid computing aims at providing infrastructure for sharing or pooling resources including HPC resources for increased demand of computational simulations.
  - Clouds focus on resource provisioning, e.g. for peak demand or when customer owned infrastructure is overloaded or its use is not appropriate for any reason.
- However, extending a company's business or a research institution's information processing beyond the borders of the respective administrative domain raises a number of issues, one of them being the use of license-protected software.
- Software protection and licensing are important topics for both the independent software vendors and software users.
- In Grid and Cloud environments, the use of license-protected applications is almost impossible and becomes a challenging task for two major reasons:
  - there are – with a few exceptions for the Amazon EC2 environment that have been introduced recently – no business models of the independent software vendors for Grids or Clouds and
  - there is no licensing technology suitable for Grid and Cloud environments.
- The 451 group concluded in a survey on licensing issues in Grids that current software licensing practices are limiting the acceleration of Grid adoption already in 2005.
- Just recently the 451 group published a survey on Cloud adoption where software license technology was ranked on the second place of limiting factors for Cloud adoption.



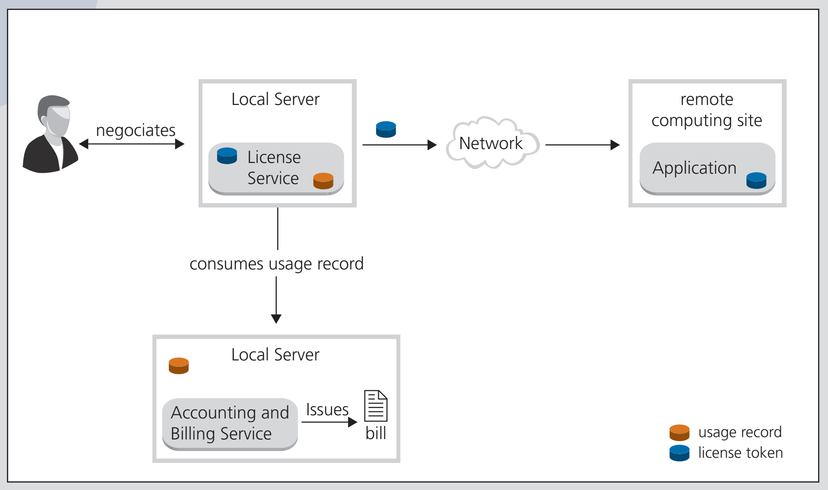


The Solution

The European project SmartLM explored and implemented new mechanisms for managing and using software licenses in a more fair and flexible way. SmartLM licenses may be used seamlessly in local cluster environments, as well as in local or remote Grid and Cloud environments.

Goals achieved

- New framework for software licenses
- SmartLM Policy Decision Point for evaluating the license token to be linked into the ISV application
- License usage terms embedded in mobile tokens for off-site and off-line authorisation
- New ISV License Models for »mobile« licenses
- Additional features like accounting & billing
- Sophisticated mechanisms to secure the token



The Product

- Based on the prototype developed in SmartLM a product is under development: elasticLM
- The basic version of elasticLM is now available for early adopters and for evaluation.

elasticLM

Main Innovations of elasticLM

elasticLM Licenses are mobile objects that may move as applications move to different execution environments. Use of protected applications is granted through Service Level Agreements resulting from negotiation of license terms prior to application execution.

Using elasticLM allows **advance reservation of licenses.** Thus, licenses are available when needed but not blocked when the application is waiting for execution.

All authorization for the use of a license is done locally at the home organisation of a user, taking into account policies of the ISV, site-specific policies defined locally or user-specific attributes as e.g. retrieved from a Virtual Organisation.

Signed and encrypted terms of a license are scheduled to the (remote) execution environment.

Integration of an Accounting and Billing System allows **price determination and budget control when the license is requested.**

Visit us at booth 154 on the exhibition floor.

JÜLICH FORSCHUNGSZENTRUM

Fraunhofer SCAI

B. Hagemeier, D. Mallmann, W. Ziegler
Research Centre Jülich, Jülich Supercomputing Centre, Jülich, Germany
Fraunhofer Institute SCAI, Schloss Birlinghoven, St. Augustin, Germany

The SmartLM project is funded by the European Commission's ICT programme under grant #216759.

SmartLM

FRAUNHOFER INSTITUTE FOR ALGORITHMS AND SCIENTIFIC COMPUTING SCAI

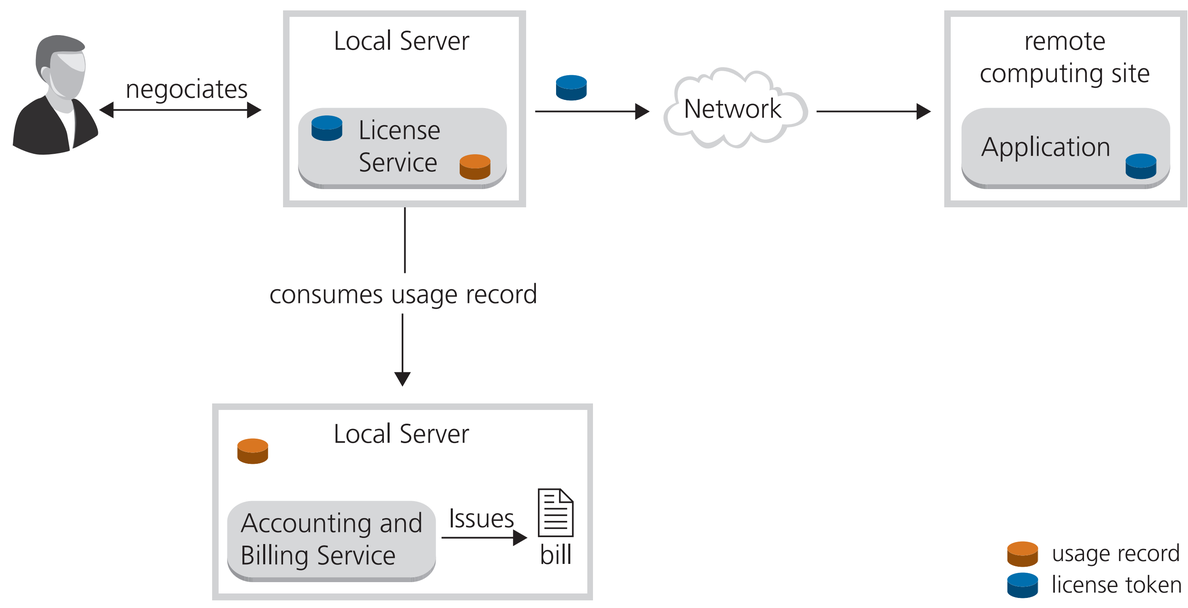
NOT HAPPY WITH YOUR CURRENT LIMITED SOFTWARE LICENSING SOLUTION? WE CAN HELP YOU: elasticLM

IT INFRASTRUCTURE PARADIGMS CHANGE

- Offering more flexibility: Grid computing, Clouds and SOA may reduce costs while increasing flexibility.
- **Computer-based simulations become more complex:** Increased computational peak requirements result in higher license costs today.

CUSTOMERS DEMAND MORE FLEXIBILITY

- **elasticLM** is designed to offer the missing flexibility for your customer.
- **elasticLM** is designed to create win-win situations with your customers. Don't lose customers or revenues.



CHANGING MARKETS

- **Answer with new Business Models:** elasticLM allows seamless combination of old and new models, migration is very easy.
- **Foster your business with innovative technology:** elasticLM is a flexible framework of policy-driven web-services, comprehensive security by design.

www.smartlm.eu

The SmartLM project is funded by the European Commission´s ICT programme under grant #216759

Fraunhofer SCAI

elasticLM: ELASTIC LICENSE MANAGEMENT

elasticLM

Contact: Wolfgang Ziegler
wolfgang.ziegler@scai.fraunhofer.de
+49 2241 14-2258

Not happy with your current limited software licensing solution? We can help you: elasticLM

IT INFRASTRUCTURE PARADIGMS CHANGE

Offering more flexibility

Grid computing, Clouds and SOA may reduce costs while increasing flexibility.

Computer-based simulations become more complex

Increased computational peak requirements result in higher license costs today.

CUSTOMERS DEMAND MORE FLEXIBILITY

elasticLM is designed to

offer the missing flexibility for your customer.

elasticLM is designed to

create win-win situations with your customers. Don’t lose customers or revenues.

CHANGING MARKETS

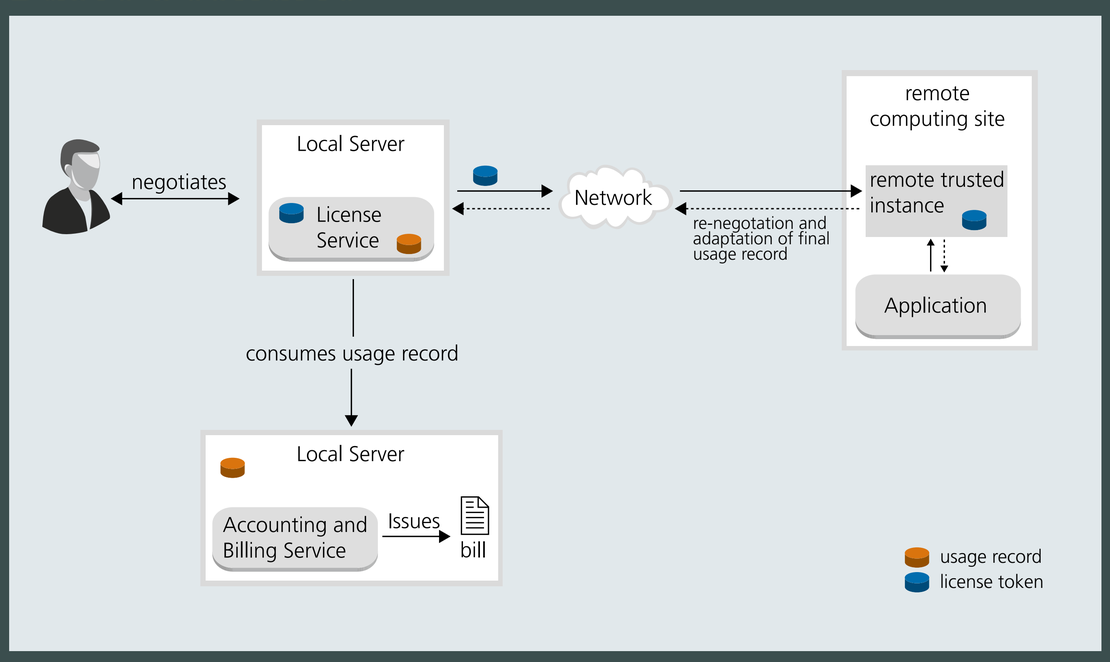
Answer with new business models

elasticLM allows seamless combination of old and new models, migration is very easy.

Foster your business with innovative technology

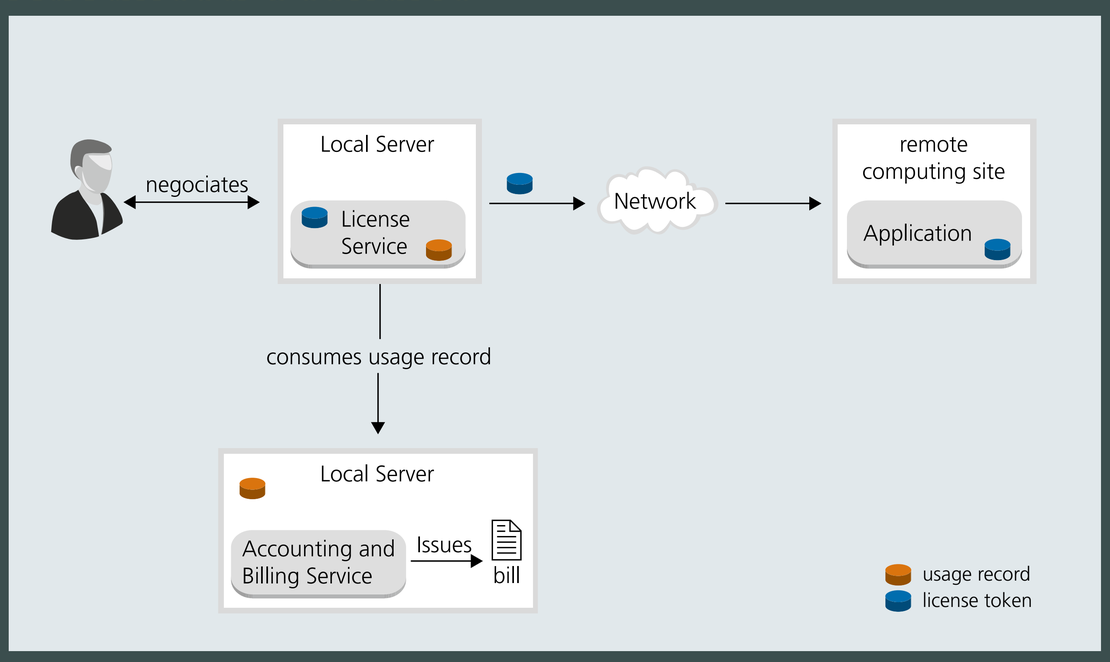
elasticLM is a flexible framework of policy-driven web-services, comprehensive security by design.

Basic Scenario



No bi-directional network link available at run-time

Advanced Scenario



Bi-directional network link available at run-time

www.elasticlm.de

Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen SCAI
Schloss Birlinghoven
53754 Sankt Augustin

451 Market Insight Service	Euro Report - 29 July 2009
SmartLM's flexible licensing virtualization technology makes licenses mobile objects	
Analyst: Csilla Zsigri	**Sector: Application software Infrastructure management / Virtualization**

Traditional licensing models focus on software used on computing resources that belong to a single organization. Licenses are usually bound to hardware, and are provided on the basis of named users, hostnames, or sometimes as a site license for the same admin domain of an organization. When it comes to distributed environments and virtualized infrastructures, we run into trouble. Software manufacturers need to change the way licensing works and use flexible and non-hardware-based licensing solutions that better fit into a virtual environment.

The EU-funded SmartLM project is developing a generic and flexible licensing virtualization technology based on standards, and integrating new service-oriented business models. One of the main goals, and the biggest challenge, for SmartLM is to balance customer needs and vendor requirements. The organizations that run the project are Atos Origin, Fraunhofer SCAI, Jülich Research Centre, Cineca, INTES GmbH, Ansys, LMS International, T-Systems-SfR, Centro de Supercomputación de Galicia, Gridcore AB and The 451 Group. The project kicked off in February of 2008, with a duration of 30 months.

the SmartLM approach

The SmartLM approach consists of providing platform-independent access, treating and implementing software licenses as services. The core part of the SmartLM software is the license service. Licenses are managed through the license service, which is not a monolithic component, but a bag of services, based on tokens, that grant the required flexibility. The license service is able to adapt to distributed environments and works with loosely coupled systems, because the software licenses themselves, as Web service resources, become adaptable and really dynamic.

Licenses are managed as agreements, extending the conventional service-level agreements (SLAs) made between sellers and buyers (negotiation), and are dynamic to support agreements that change over time (renegotiation). SmartLM allows licenses to be reserved in advance. In practice, that means licenses are available when needed, but aren't blocked when the application is waiting for execution.

In contrast to most of the existing license management systems, SmartLM integrates a modular offering for accounting and billing, enabling a comprehensive analysis of license usage and allowing price determination and budget control when the license is requested. Aspects of security have also been examined with special care. These aspects are related to the market players involved as well as the license mechanism itself.

An important challenge was to develop a license mechanism able to generate tokens that decouple the execution environment from the site that hosts the license server. It means not only to extend the actual license management approach, but to make the license server implementation part of grid and cloud systems. That, in turn, implies some other key features – such as the capability to work with an unreliable network or with no connection at all during the application execution, an extended trust management able to deal with the different administrative domains, and flexibility to cope with the requirements of the various players involved in the process.

new business models

As already mentioned, the use of licensed software in distributed environments and virtualized infrastructures is limited in many ways. Traditional licensing practices are under pressure from a variety of alternative options and are tightening vendors' profit margins, pushing down licensing costs and giving more negotiating power to users. Currently, the landscape is quite chaotic, with providers randomly introducing new models. Through close collaboration with a wide range of stakeholders – software vendors, application providers, end users – a few potential usage-based models have come up. They look at different aspects of software licensing.

Featuring the ASP – In this model we find the ASP offering various solutions to various problems. We have pinpointed and analyzed five cases – customer license housing, embedded license (dependant software vendor), license re-direction (external consultant), license aggregation, and license reselling – that companies may most frequently encounter in real operations. In all of them, the ASP plays a central role, being a reseller of hardware, software and services. The introduction of the ASP can be very advantageous for both the ISVs and the end users. From the ISV's point of view, the ASP can generate additional business offering licenses and hardware resources for end users on demand (competitive resource provisioning). Making use of economies of scale and SmartLM features, the ASP can make existing models (e.g., short-term licenses) more attractive to customers, and introduce new ones the ISV is not willing to offer, such as pay-per-use for license reselling.

License extension – The license extension model allows end users to extend their licenses in their LAN and distributed environments on demand, e.g., for workload peaks. The license server takes care and simplifies the process of the extension of licenses; for example, in terms of accounting and license administration. These mechanisms give end users more flexibility and value, and at the same time generate additional revenue for ISVs and ASPs.

License aggregation – Most contracts between ISVs and end users restrict the license usage to LAN. The license aggregation model allows the use of licenses that belong to different sites and brings them together to form a single license token. These licenses can come from either the ISV or the ASP. End users gain more flexibility and value and get access to huge hardware resources. The ASP provides these hardware resources to the end user and generates additional business for the ISV.

Hardware-independent pricing model and feature-based accounting – The hardware-independent pricing model makes the license price effectively independent of the underlying hardware, enabling a cost-efficient use of licenses. With the introduction of a set of micro-benchmarks, the user is not tied to hardware anymore, so is not punished for slower hardware. All users can get the highest benefit from their licenses. This benchmark model leads us to a more general approach, to a feature-based accounting. The core issue here is letting the application define the features and set what it wants to charge for. As opposed to the time approach, the feature-based approach is really independent of the machine where the application is being executed.

elasticLM, the product

The SmartLM consortium has named its future 'elasticLM.' ElasticLM is an advanced license management product with a number of innovative features that differentiate it from current license practices.

elasticLM's major innovative features

Licenses can be used to run apps in grid and cloud environments no matter whether there is network connectivity –during application run- to access the site that hosts the license server that issued the license.	Budget limitations are checked and enforced when the user requests a license.
License usage can be easily and better tracked as elasticLM provides access to and management of all licenses owned by a site.	elasticLM uses sophisticated security mechanisms to combat illegal license use.
elasticLM supports the definition of local policies for license usage addressing site-specific needs. These policies are evaluated in addition to the embedded policies of the ISVs.	elasticLM supports co-scheduling of licenses and computational resources, and re-negotiation of license terms at run time.
An accurate and user-specific price is calculated beforehand based on a number of configurable parameters.	Licenses can be booked in advance for later use and are coordinated with the availability of resources.
elasticLM's advanced accounting & billing system, based on effective usage, adjusts the accounting information after license usage.	ASPs can provide the user with access to applications without buying additional licenses.

The competitive landscape includes Acresso Software's widely used FLEXnet, SafeNet's Sentinel RMS, Reprise Software's RLM and X-Formation's LM-X. Also, we find users employing their own management models, and ISVs that support their own management schemes.

The launch date and the terms under which elasticLM will be released are not known yet. There are ownership and business model issues to be agreed on and, potentially, another round of private funding is needed to launch elasticLM in the market.

the 451 take

The software licensing issue is a complex one because transformation is going on at a macro level where a lot of money is involved. What has been happening and what can be expected is an evolution of license models, rather than a revolution. The goal and challenge is to balance customer needs and vendor requirements. In the past, end-user companies that wanted to extend the use of software to grid environments either paid up, or found a workaround.

The ability to proactively manage the use of software licenses based on business objectives is not a grid-only issue. Virtualized infrastructures and distributed environments (including cloud computing) call for flexible and non-hardware-based license models that support service-oriented business models. SmartLM's offering brings along improved customer choice with a model that makes licenses mobile objects.