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

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

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.

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

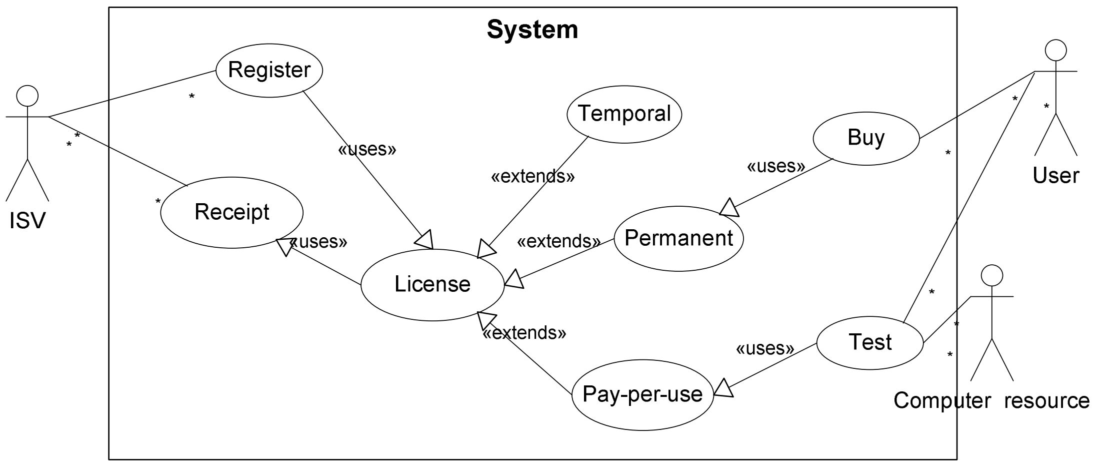

Figure 1. Use case diagram UC02: software developer

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

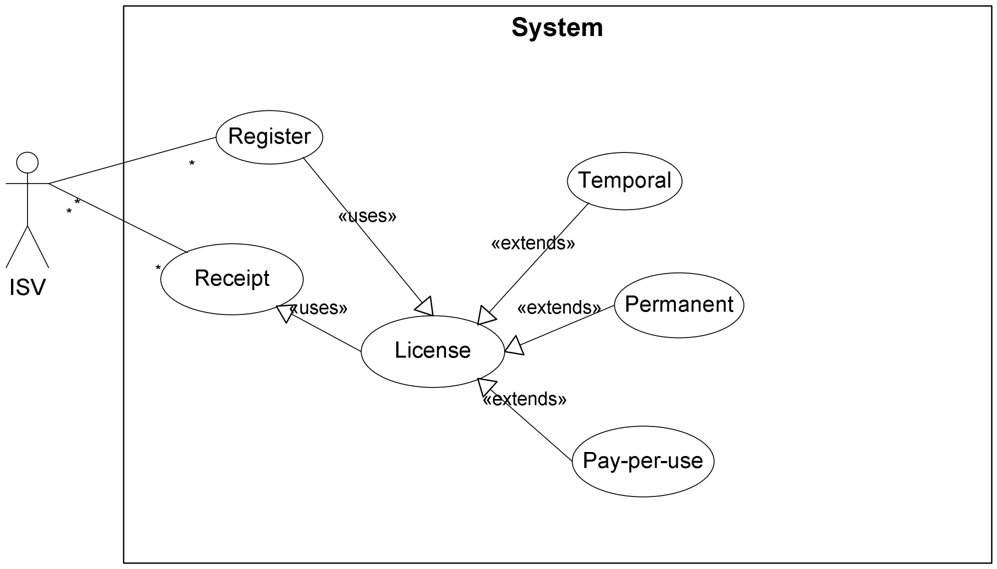

Figure 2. Use case diagram UC02 step 1: software developer registers the plug-in in license system

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

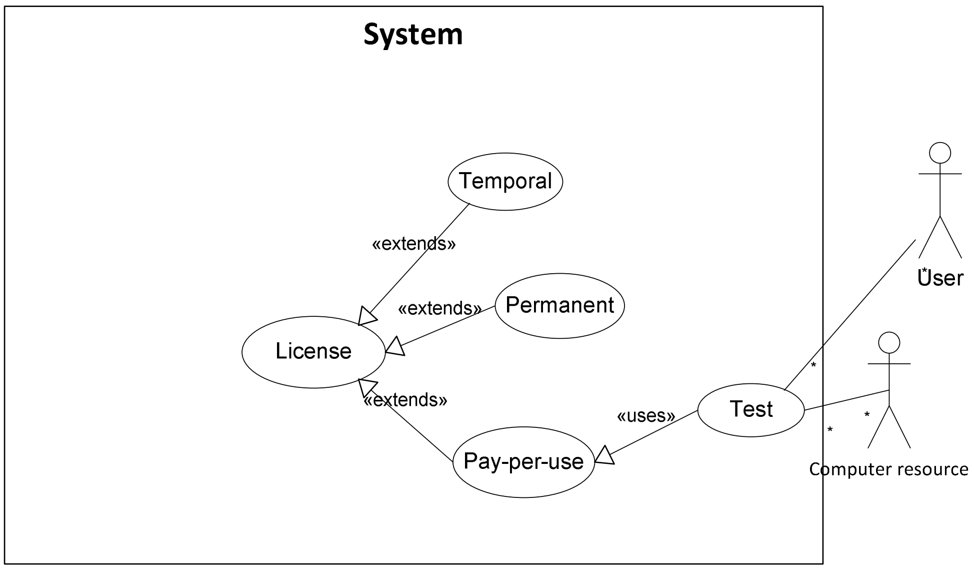

Figure 3. Use case diagram UC02 step 2: Bank tests the plug-in in a Grid environment

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

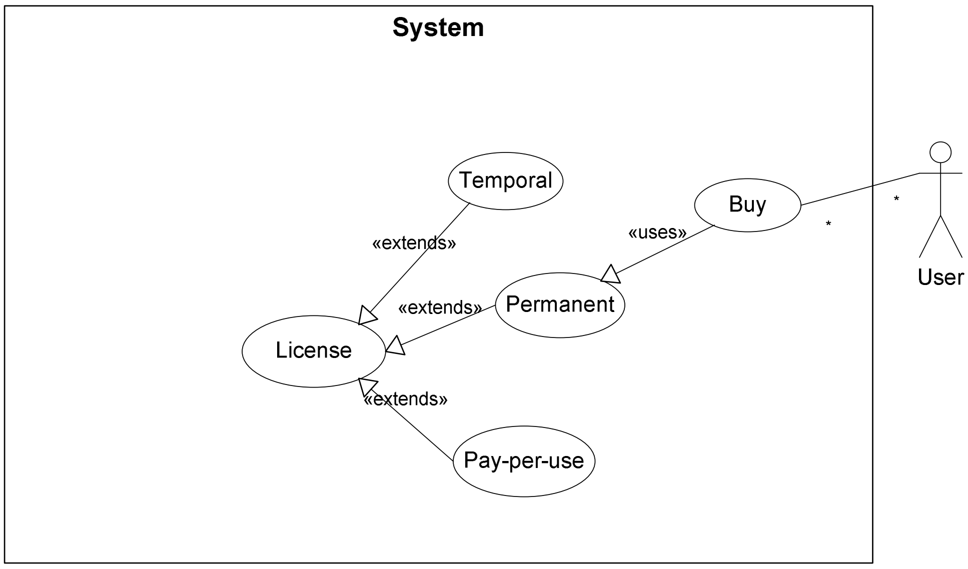

Figure 4. Use case diagram UC02 step 3: the bank buys the license

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

	o License token can only be used once o 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 resource*s 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			