Interactive Realtime Multimedia Applications on Service Oriented Infrastructures

ICT FP7-214777

WP3 – IRMOS Platform Specification

D3.1.1: Preliminary version of IRMOS Overall Architecture

IRMOS_WP3_D3_1_1_NTUA_v1_00

Scheduled Delivery:	**31/08/2008**
Actual Delivery:	**29/08/2008**
Version	**1.00**

Project co-funded by the European Commission within the 7th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Responsible Partner: NTUA

Revision history:

Date	Editor	Status	Version	Changes
18.04.2008	Andreas Menychtas	Draft	0.01	Initial Release
24.04.2008	Andreas Menychtas	Draft	0.02	TOC Refined
05.05.2008	Spyridon Gogouvitis	Draft	0.03	Unified Process Section added
20.05.2008	Andreas Menychtas	Draft	0.04	Initial User Requirements & Use Case Analysis added
25.06.2008	Spyridon Gogouvitis	Draft	0.05	Platform Design Principles section added
10.07.2008 - 15.07.2008	Andreas Menychtas	Draft	0.06-0.08	Contributions to the platform capabilities section
15.07.2008	Andreas Menychtas	Draft	0.09	ISONI section added
16.07.2008 - 22.07.2008	Andreas Menychtas	Draft	0.10-0.12	Contributions to the conceptual models section
24.07.2008	Andreas Menychtas	Draft	0.13	Security Section added
30.07.2008	Kleopatra Konstanteli	Draft	0.14	Framework Services section added
31.07.2008-05.08.2008	Andreas Menychtas	Draft	0.15-0.21	Various Contributions and changes based on comments from WP3 partners
05.08.2008	Andreas Menychtas	Draft	0.22	Internal QA version
27.08.2008	Andreas Menychtas	Draft	0.23-0.27	Various updates according to the internal reviewers' suggestions.
29.08.2008	Andreas Menychtas	Final	1.00	Final Version

Authors

Andreas Menychtas, Spyridon Gogouvitis, Kleopatra Konstanteli, Georgios Kousiouris, Dimosthenis Kyriazis, Theodoros Polychniatis (ICCS/NTUA); Jesus M. Movilla, Eduardo Oliveros (TID); Tim Courtney, Hugh Vanijprabha (XY); Karsten Oberle, Thomas Voith (ALUD); Matthew Addis, Bill Mitchell, Bassem M. Nasser (IT-Inn); Michael Braitmaier, Antonio Cuevas (USTUTT)

Internal Reviewers

Dominik Lamp (USTUTT); Tommaso Cucinotta (SSSA); Consolidating Reviewer: Matthew Addis (IT-Inn)

Acknowledgements

The research leading to these results has received funding from the EC Seventh Framework Programme FP7/2007-2011 under grant agreement n° 214777

More information

The most recent version of this document and all other public deliverables of IRMOS can be found at http://www.irmosproject.eu

Glossary of Acronyms

Acronym	Definition
API	Application Programming Interface
EE	Execution Environment
ICT	Information Communication Technology
ISONI	Intelligent Service Oriented Network Infrastructure
MAP	Modelling, Analysis and Planning
NR	Negotiation Resource
QoS	Quality of Service
RT	Real-time
SC	Service Component
SD	Service Discovery
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOI	Service Oriented Infrastructure
SP	Service Provider
UML	Unified Modelling Language
VMU	Virtual Machine Unit
VPN	Virtual Private Network
VSN	Virtual Service Network
WS	Web Service
WSDL	Web Service Description Language
XML	Extensible Markup Language

Table of Contents

List of Figures

List of Tables

1. Executive Summary

This report (D3.1.1) presents the preliminary version of the document describing the overall architecture of IRMOS platform. The main objective of IRMOS project is to build a Service Oriented Infrastructure (SOI) for interactive applications posing with real-time requirements. The work described in the report was carried out in the framework of WP3 -*IRMOS Platform Specification*- and provides the analysis, design and specification of the initial architecture of the IRMOS platform. This is the first out of the four reports, foreseen in WP3, which will describe, in more detail as the project advances, the functionality of the IRMOS platform, the functional and non-functional capabilities and the individual components as well as their interactions.

The output of WP3 is of high importance for the other WPs but also for the project itself, since these reports include the guidelines for developers to implement their components, associating, on the same time, the user's requirements and the scenarios with specific functionalities and building blocks. Therefore, from the beginning of the project, WP3 decided to follow, at the level this is possible, a well-established and successful methodology for the design of the IRMOS Platform, the *Unified Process*. This "off the shelf" process describes in detail how the user requirements and use cases are analyzed so as to identify the platform capabilities, conceptual model and sub-systems, and UML is used for their description. Using UML, the results of WP3 are more easily and effectively comprehensible and adopted.

Initially WP3 analyzed from the architectural point of view the user requirements, use cases and key parameters included in the D2.1.1 report [1], taking into account the technical and business prerequisites for the platform that consist mainly of two principles: *Service Oriented Architecture* and *real-time functionality*. Based on this analysis, the input was prioritized and associated with specific parts of the architecture. However, most of the requirements and use cases were application specific and further analysis was necessary to identify the functionality of the platform and associate them with specific blocks or components. The analysis was followed by the identification of the platform capabilities and the design and presentation of conceptual models for the IRMOS platform, which describe the main processes in the scenarios such as the Workflow and SLA Management. Part of the process was also aimed to confirm that the identified models and capabilities can be implemented with the technologies of the future trends/innovations acknowledged in the D2.3.1 report [2].

Finally an overview of the platform design and the main subsystems are presented to the maximum level of detail feasible for the time being, since there exist numerous open issues between the development WPs. Additionally, the expected role for each block and the interactions with the other blocks are explained. Partners provided input for this section, based on their expertise in the project and it is anticipated that in these reports, especially regarding the sections of the architecture overview, the complete IRMOS project design will come together and be presented. This is extremely useful for the development WPs to start their work in a consistent and efficient way. During the project months to follow, WP3 will focus on the analysis of subsystems and their interaction and hence this section will be considerably updated in the next versions of the report, describing in details all the components, interfaces but also the specifications about the hardware and software tools that will be adopted and used throughout the project.

2. Introduction

This report is the preliminary version of the IRMOS overall architecture, presenting the process and the results for the analysis and design of a Service Oriented Architecture, capable to support interactive applications with real-time requirements. In this process the user requirements and use cases have been analyzed in order to identify the key functionality and capabilities for the platform and thereafter design the main architectural blocks. The document is structured in four sections.

Chapter 3 describes the design principles that were used for the design of the IRMOS platform architecture. Following the main objective of the project to "*facilitate real-time interactivity in SOIs*", the advantages of SOIs and the required real-time attributes are explained along with initial approaches on how these principles will be applied on IRMOS.

For the purposes of the analysis and design of the platform architecture, WP3 decided to follow a specific methodology and language. This methodology (*Unified Process*) and UML are generally used to describe in a formal way the basic actions for software analysis and design. Chapter 4 provides an overview of the Unified Process and definitions of the UML diagrams that are used later in this report.

Chapter 5 starts with an analysis, from the architectural point of view, of the user requirements and use cases produced in WP2, as a result of the application scenarios and of the survey to application users. The initial list of the user requirements and use cases was quite long and in most cases application specific. This means that further analysis was necessary to associate them with the SOI and real-time principles and finally identify the platform capabilities. The key IRMOS platform capabilities are presented while UML diagrams are also included, whenever this is needed. An initial security analysis is presented as part of this section, which will be though further analyzed in the next versions of the platform architecture report. Additionally, the conceptual models for the main processes of the project are produced. In these models an overview of the involved components is described and their interactions and the domains that they belong to are illustrated.

Finally, chapter 6 presents an outline of the IRMOS overall architecture as well as the boundaries and the interfaces between the subsystems (Framework Services and ISONI). For the analysis and initial design of the subsystems, WP3 worked closely with the component "owners" of the development WPs, providing valuable feedback to them. It should be noted that even if the Execution Environment and Network are considered in the IRMOS proposal two separate subsystems, in practice they work together as part of the ISONI framework. Therefore these two concepts are presented in the same section within this document. Finally, part of this section is the data storage subsystem that interacts with the Execution Environment and Network services.

3. IRMOS Platform Design Principles

The main objective of IRMOS project to develop a Service Oriented Infrastructure (SOI) with real-time capabilities is a great challenge for WP3, which will design the platform and provide the implementation guidelines to the development WPs. The platform, following the service oriented design principles, should be able to allow the adaptation and operation of interactive applications with high real-time requirements, such as the multimedia applications for example.

Service Oriented Infrastructures are currently inextricable part of ICT and of Grids in specific, facilitating a new trend for the next generation of ICT systems. Moreover, since IRMOS will pursue to manage and make use of several types of resources (network, storage, computational) as well as to communicate with legacy systems, a service oriented architectural approach seems to be the only option for the platform. On the other hand, SOI infrastructures have several advantages -as described in the next section- that IRMOS will take advantage of and exploit in order to design and develop a number of services that interact dynamically and continuously spanning between different domains, and ranging from the application level and down to the level of network resources management and the execution environment. Therefore the challenge is to carefully design and synchronize this rich set of services so as to efficiently operate, manage and reconfigure, all the resources under real-time conditions, providing to the end users and the associated applications the appropriate and required Quality of Service, agreed in the SLAs.

The overall management and control of the infrastructure and of the individual services will be designed, developed and deployed to support the required real-time interaction between them also including the interaction between the applications and the resources. IRMOS SOI is expected to solve several problems in reference to the guaranteed QoS, the scheduling of the service execution, the mapping of the application workflows and requirements to low level resource parameters and individual services. The following figure depicts an overview of the IRMOS SOI:

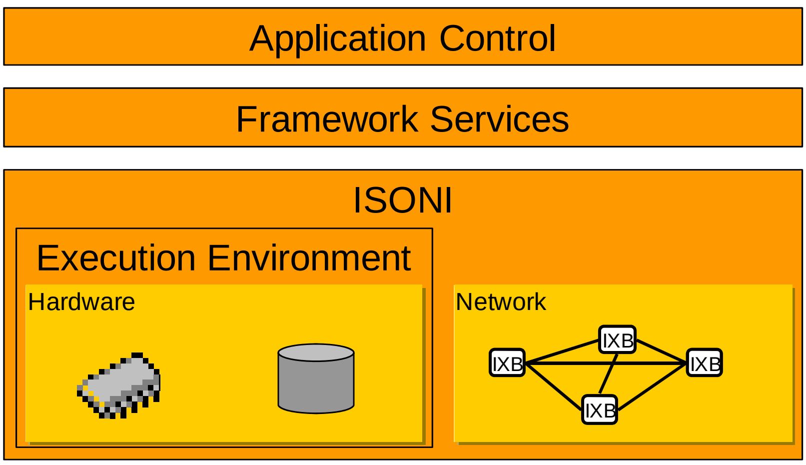

Figure 1: The IRMOS SOI

3.1. Service Oriented Infrastructure

Service Oriented Architecture is an architectural paradigm based on reforming application functions and pieces of information into a “service” that can be accessed through a common interface regardless of the location of the function or of the piece of data [3].

The IRMOS SOI has to exhibit the following characteristics [4]:

- **Loosely Coupled Services**: Loose coupling is an approach to the design of distributed applications that emphasizes on agility (that is to adapt to changes). Loose coupling intentionally sacrifices interface optimization to achieve flexible interoperability among systems that are disparate in technology, location, performance, and availability. A loosely coupled application is isolated from internal changes in others by using abstraction, indirection, and delayed binding in the interfaces between the applications. As compared to traditional, tightly coupled applications, loosely coupled applications aim to be more reusable and adaptable to the unexpected.
- **Synchronous:** Supporting the synchronous invocation and execution of services, in the sense that when an end user requests information or invokes a function, a connection between the two end-systems must be maintained until a response is received.
- **Asynchronous**: Supporting also asynchronous interactions in which information is sent without the expectation of getting back an immediate response. This characteristic is very important in the cases where there is no requirement to maintain a connection between the two end-systems while waiting for a response.

The most important aspect in the proposed architecture is the fact that it will be designed on the basis of real-time application requirements. Thus, services and their orchestration need to be built and developed in a way so as to preserve the real-time attributes throughout the whole infrastructure including the resources, the virtual execution environments as well as networks to the applications and to the end user.

3.2. Real-time Attributes

A real-time system is a computing system the absolute correctness of which depends not only on the correctness of the output, but also on the time that this is produced. In a hard real-time system, the slightest violation of even a single timing constraint may be detrimental for the overall system functionality. On the other hand, in a soft real-time system, some violations of the timing constraints are acceptable, as long as these are not too frequent and they remain within acceptable boundaries [5]. IRMOS will need to be such a soft real-time system and in that way it needs to incorporate real-time functionalities in all of its layers. As a general aspect, interactions between the various components in IRMOS will have to be kept as minimum as possible so as to minimize latencies without of course missing in functionality.

An important part of the platform is the Execution Environment which will have to provide rich real-time capabilities. This includes multitasking capabilities, threads with priorities and an appropriate number of interrupt levels.

Because faults are inevitably going to occur, strong fault detection and recovery mechanisms need to be developed. This can have a great impact on the real-time capabilities of the platform, since intelligent fault recovery mechanisms will allow timing constraints to still be met in case of a failure[6].

In this preliminary version of IRMOS platform architecture, the parts of the platform and the processes in which real-time functionality is required have already been identified. In general, real-time is required for the execution and monitoring processes[7], while for the SLA management[8], the workflow management[9] (initialization/pre-execution phases) "off the shelf" approaches will be used.

Nevertheless, there are several open issues at the moment of preparation of this report, mainly related with ways to achieve the real-time functionality of the platform. First of all we expect the subsystem to be self-managed and reconfigurable in order to avoid the escalation of a problem (such as an SLA violation or Network link failure) or information in general that will probably cause unnecessary delays. That will probably require additional functionality for some subsystems or components such as the storage system and ISONI. On the other hand the notifications and the techniques, processed when more than one component are involved, should be defined in a real-time-aware manner. Finally the hardware platforms and the software frameworks on which the real-time services will be deployed and operate have to be carefully specified. Many of them are already described in D2.3.1 [2] and the discussions about the available options are under way within the development WPs. We expect that in the next versions of the architecture report all the open issues for the real-time functionality of the platform will be clarified further.

4. IRMOS Platform Design Process

WP3 followed the Unified Process [10], [11] principles and UML for the analysis and design of the IRMOS initial architecture and it is anticipated that this process will be also used by the development WPs during the implementation and validation of their components.

4.1. Unified Process

This Process is a framework which guides the tasks, people and products of the software design process. It is a framework because it provides the inputs and outputs of each activity, but does not restrict how each activity must be performed. The Unified Process is:

- **Iterative and Incremental:** The Unified Process has an iterative and incremental model. That is, the design process is based on iterations which either address different aspects of the design process or move the design process. In essence the end result is incrementally produced.
- **Use case driven:** In the Unified Process use cases are used to ensure that the evolving design is always relevant to what is required by the end user. In fact, the use cases act as the one consistent thread throughout the whole of the development process.
- **Architecture centric:** The Unified Process explicitly acknowledges the importance of the architecture for the successful completion of the project. It prescribes the successive refinement of the executable architecture thereby attempting to ensure that the architecture remains relevant.

4.2. Unified Modelling Language

Unified Modelling Language has been used in the analysis and design of the IRMOS platform in WP3. The platform functionalities, models and processes presented in this document are illustrated using UML diagrams, at the level this is possible for an initial architecture, and these diagrams will be further detailed in the upcoming WP3 reports. Using UML, the WP3 work is well organized and the results are efficiently capitalized from the development WPs.

Specifically, in paragraph 5.2 - IRMOS Platform Capabilities, use case diagrams have been used to describe the key functionalities of the platform that have been identified in the platform analysis process. Use case diagrams depict an overview of the usage requirements of the system. The actors can be easily identified as well as their association with any interactions described by the use cases. Use case diagrams do not provide in depth analysis of the use cases, but rather a visualization that can help in the preliminary phases of the requirements analysis process.

In paragraph 5.4 - IRMOS Conceptual Models UML, sequence diagrams are used. These diagrams model the flow of logic within a system in a visual manner, enabling both to document and validate the logic behind the system's development, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artefact for dynamic modelling, which focuses on identifying the behaviour within a system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are the most important design-level models for modern business application development.

Also in this paragraph component diagrams are used. They are especially helpful in the initial architectural modelling efforts which focus on identifying the architectural landscape for the IRMOS system, as they enable modelling of the high-level software components and more importantly the interfaces to those components. Components may both provide and require interfaces. An interface is the definition of a collection of one or more methods, and zero or more attributes. The component diagram shows the interrelationships between different components that may be developed by different partners and in that way establishes and formal communication channel between them that is necessary during the development process.

5. IRMOS Platform Analysis

In this section an analysis of the IRMOS platform is carried out. Initially the application scenarios are analyzed so as to identify key technical requisites for the platform and thereafter to extract the capabilities that IRMOS must implement in order to meet the goals set. Additionally, in collaboration with WP2, platform requirements are extracted from the user requirements and finally associated with specific capabilities and building blocks of the platform. These key capabilities are presented in detail as the first step for defining the functionality of the platform and of its subsystems. An analysis on the security features is also presented and the platform's conceptual models along with sequence and use case diagrams are illustrated.

Input to this chapter was provided from the problem statement that IRMOS addresses, the use case models as these are described in WP2 (Market and Technical Requirements analysis) and WP4 (Application analysis and adaptation) and the requirements analysis process derived from WP2. Domain experts from WP5 (Framework services), WP6 (Execution environment) and WP7(Intelligent Networking) were also involved in order to define the separate working modules and general processes along with the state of the art solutions as these are identified from the real world in D2.3.1 [2].

5.1. Use Case and Requirements Analysis

5.1.1. Application Scenarios and Use Cases

The application scenarios are briefly described in the following section highlighting the technical challenges for the IRMOS platform and thereafter the results of the use case analysis are presented.

5.1.1.1. Collaborative Digital Film Postproduction

In this scenario, a group of highly talented artists, colourists, editors, VFX and sound operators, located in different countries are jointly working during the hot phase of post-production of an international Sci-Fi movie co-production. They *log into* the IRMOS platform and the system *automatically configures their GUI* according to their personal capabilities and expectations, in order to create a *collaborative environment* in which they are also able to use the multimedia capabilities it offers in order to *communicate with each other*. They start the visualisation of the already edited film in a *synchronised way*, i.e. the application using the IRMOS platform provides multiple streams of the same content to the group in order to discuss the different effects or changes to be applied. When needed, *the streaming can be stopped*, the necessary changes discussed with the others and, after performing the corresponding adjustments, *replay the edited sequence immediately*, so everybody can agree to the final impression. While the group did review last days work new material from the set has been *ingested into the application*, *automatically annotated*, and is ready to be previewed by the group. *The system processes any changes required in real-time* using the processing power in the IRMOS enabled computing networked computer cluster, so that at the end of the process, the group gets a good impression of the futures result.

Technical challenges:

- In advance of the work beginning, all the parties involved (artists, editors, post-houses with specific facilities, network operators, software application providers, data centres etc.) need to come together and rapidly agree upon, and then *assemble, all the services and applications involved in the post-production workflow.*
- *The workflow, temporal constraints, real-time guarantees and resource needs* all need to be analysed so the necessary services can be put in place, including *SLAs and QoS guarantees*, in a way that optimises cost, the need for over-provisioning, variations expected, and probability of degraded service.
- *Security and trust* need to be applied to protect the business relationships in the value chain, the services provided under them, and the IPR of the movie content. It is essential that this *does not obstruct the real-time interactivity*.
- The services available to each participant (applications, user interfaces, content) is *configured according his capabilities and role in the process* – e. g. the director only needs the film pre-visualizer and annotation service, but experts need more powerful interfaces to interact with the post-production system).
- *Intensive real-time processing for performing the content manipulation* – usage of distributed nodes, parallel processing – and for guaranteeing the perfect synchronisation of the situation all participants can see.
- *Integration of real-time multimedia applications and collaborative tools* – some application to share the results of a production and also for the communication between the different actors in the post-production activity.
- *Real-time integration of real and animated images.* Allocation of requested nodes for high consuming processing for the integration of real images into the virtual environment, which requires a tight synchronisation of the pictures streamed into the virtual environment.

5.1.1.2. Virtual and Augmented Reality

At one side, a development team with virtual and hybrid prototypes wants to discuss the outcome of a simulation and the experiments on a car within a wind tunnel with a remote expert. The local team itself is distributed. The wind tunnel contains a car body; the experiment will involve the production of stream lines with a smoke probe. Some of the engineers, using a head mounted display device, will do a local analysis of the experiment, *overlaying it with the visualisation of the simulated data*. The image, together with a set of other images captured by fixed cameras arranged around the physical prototype, will be transferred to a local CAVE and to the remote expert, situated in another location. An *audio/video conference* is used in the discussion by all partners. The connection will be routed via the IRMOS network. An initial bandwidth is reserved that allows the smooth transmission of all simulation data, video and audio streams. The simulation data itself will be generated by a *continuously running simulation* that is *directly connected to the visualisation*. As post-processing of the simulation data can be quite complex and resource intensive, resources have to be found that are able to cope with this task. IRMOS platform will be used to *locate the required resources, deploy* a COVISE installation there and *connect* it to the existing COVISE session, running the CPU-intensive modules on the new nodes. All *partners can influence the visualisation of the simulation results*. Adding a *real-time distribution of data, streams and feedback* would give distributed teams a perfect workspace for carrying out collaborative sessions as if they were sitting together in the same room performing the same task, without the usual lag and noticeable inconsistencies in transmission. As mentioned earlier, the quality of every visualisation session is highly dependent on the speed and also reliability as when new data is available. IRMOS *flow control and path monitoring tools* will be exploited for ensuring a smooth and reliable frame rate.

Technical challenges:

- *Intensive real-time processing for performing the simulations* – usage of distributed nodes, parallel processing – for guaranteeing the perfect synchronisation of the experiment with the simulation.
- *Integration of real-time multimedia applications and collaborative tools* – some application to share the results of a simulation/experiment and also for the communication of the different actors.
- *Real-time integration of real and virtual images.* Allocation of requested nodes for CPU-intensive processing for the integration of real images and simulation results into the virtual environment, which requires a tight synchronisation of the pictures streamed into the virtual environment.

5.1.1.3. Interactive real-time eLearning application

Anna is one of a large group of Spanish students visiting Athens as part of a School trip. They are learning about classical Greece and before their visit they have already been enjoying an interactive, real-time virtual course on the subject of ancient Greek architecture. Whilst visiting the Acropolis, she accesses the eLearning application again and indicates the topics she is interested in. The application running on an IRMOS enabled platform *considers her device capabilities* and the kind of services she needs and, *constructs a virtual reality learning environment that she can move around in with her fellow students*. This allows her to *interact with her colleagues* and other virtual visitors as she moves around the venue. She is be able to interact with the learning material by *accessing real-time multimedia content*, using video streaming and seeing a visualisation of how the Acropolis used to look. The virtual environment allows *access to information from other locations*, for example she can access the National Museum of Athens to gain further information on artefacts found at the Acropolis and to interact with other members of her class who are visiting the museum at the same time. After the visit, all the students gather together in a nearby hotel where they are staying and request to *video conference* with an expert on classic Greek architecture in order to have a deeper explanation on the history of the Parthenon and to discuss what they have collectively learnt during the day. IRMOS allows them to video conference with a curator on duty at the site who also delivers some multimedia material for supporting his explanation.

Technical challenges:

- *A large number of people interacting with one another through a virtual environment* with a need for synchronisation of experience.
- *Real-time delivery of multi-media material to users* with adaptation of the service rates based on the users' ability to consume the delivered data rate and on the number of simultaneous users of the system.
- *Automated selection of the best source of material* for delivery to the user and automated construction of the service over the network.

5.1.1.4. Use Cases

Following the scenarios and the initial technical analysis of them, use cases have been produced that describe these scenarios in a more consistent and formal way. The use cases have been presented extensively in Deliverable 2.1.1 [1] where more information concerning them can be found. These use cases are representative of the demands application will have on the IRMOS platform to provide real-time functionalities.

By analyzing the use cases various key capabilities have been identified. These are *Workflow Management, SLA Management, Data Management, Streaming (including both audio and video), Discovery of services, Monitoring of tasks and Identification and Security*. More details about these key capabilities for the platform can be found in section 5.2. In the following table the aforementioned capabilities are associated with the use cases they have been derived from. The numbering of the use cases follows that of D2.1.1 report [1].

Table 1: IRMOS Platform Capabilities & Use Cases

Key Capability	Use Case
Workflow Management	**Film Post Production**:FPUC01, FPUC04, FPUC12, FPUC13, FPUC14, FPUC15, FPUC16, FPUC17, FPUC016
	E-Learning: FELUC007, FELUC011
	Virtual and Augmented Reality: VAR003, VAR005, VAR007, VAR009, VAR010, VAR012
SLA Management	**Film Post Production:** FPUC04, FPUC06, FPUC13, FPUC14, FPUC15, FPUC16, FPUC17, FPUC016, FPUC017
	E-Learning: FELUC006
	Virtual and Augmented Reality: VAR002, VAR003, VAR008, VAR011
Data Management	**Film Post Production:** FPUC05, FPUC07, FPUC08, FPUC11, FPUC12, FPUC13, FPUC14, FPUC15, FPUC16, FPUC17, FPUC016, FPUC017
	E-Learning: FELUC001, FELUC003, FELUC004, FELUC005, FELUC011
	Virtual and Augmented Reality: VAR007, VAR009, VAR010, VAR012, VAR013
Streaming	**Film Post Production:** FPUC09, FPUC10, FPUC16, FPUC17, FPUC016, FPUC017

	E-Learning: FELUC007, FELUC011,
	Virtual and Augmented Reality: VAR008
Discovery	**Film Post Production:**
	E-Learning:
	Virtual and Augmented Reality: VAR003
Monitoring	**Film Post Production:**
	E-Learning:
	Virtual and Augmented Reality: VAR004
Identification and Security	**Film Post Production:** FPUC02, FPUC10, FPUC018, FPUC019
	E-Learning: FELUC006
	Virtual and Augmented Reality: VAR001, VAR008

5.1.2. Platform Requirements Analysis

Concurrently the analysis of the platform requirements took place. This analysis was carried out in collaboration with the WP2 and is expected to continue until PM8 that ID2.1.2 [12] will be released. In this process the user requirements (a complete list is available in D2.1.1), which have been extracted from the application scenarios, the use cases and the survey to potential users, examined so as to elaborate a list of functional requirements. This is an ongoing work and we anticipate that this list will be extended with additional security requirements and requirements coming from development WPs as the project advances and dependences between various components are identified.

The platform requirements have been grouped in the following categories (ID) as in WP2:

- VS: Video and Audio Streaming/Multi-videoconference functionality
- DM: Data Management
- NW: Network
- SLA: Service Level Agreement
- EE: Execution Environment
- SEC: Security and Users profiles

For each one of the following platform requirements its importance is illustrated as well as the user requirements and use cases (included in D2.1.1) that emerge from.

VS-1 Video quality

Description: Video quality of video streams send between collaborative partners in a collaborative VR session is of importance for keeping information contained in a video stream to remote partners. This quality is of course dependent on the video camera recording the video stream as well as encoders used for encoding the video stream and used parameters. Based on the used parameter settings in relation with a negotiated SLA these parameters provide a trade-off between quality and affordability.
Priority: High
Reference: EUGFR029

VS-2 Secure video and audio exchange
Description: Besides the mentioned secure data exchange in DM-1 there should also be an optional method that provides security for video and audio transmissions as part of video conferencing in a collaborative session.
Priority: Medium
Reference: EUGFR031

VS-3 Visualization of multimedia content
Description: The IRMOS platform should provide with a mechanism to visualize multimedia content in a synchronized manner for several simultaneous users.
Priority: High
Reference: FPUC06, FPUC07, FPUC08, FPUC09, FPUC10, FPUC11, FPUC12, FPUC13, FPUC14, FPUC15, FPUC17, ELUC06, ELUC07, ELUC08 and ELUC11

VS-4 Multi-videoconference service
Description: The IRMOS platform should provide the users with a multi-video conference service. It would be desirable that this tool provides other collaborative functionalities as chat, common desktop and so on, for giving the users the ability of establishing collaborative sessions.
Priority: High
Reference: Use-cases FPUC04 and ELUC09

VS-5 Multi-video Conference
Description: The users should have the ability to use both Audio and Video conference. Initially, this seems to be an application specific requirement. However this may require *authentication*, *authorization* and *workflows* from WP5. This is also tied with *video streaming* capabilities of IRMOS. The audio/video quality should be customizable by the user.
Priority: High
Reference: EUGFR01, EUGNFR06

DM-1 Secure Data exchange
Description: This actually relates to two different cases. First is the situation where video data is streamed across a network. Furthermore there will be situations where data has to be transferred by a potential user of an application utilising the IRMOS framework, from his workspace to IRMOS data storage for processing. This data has to be secured so it is not leaked during the transfer or altered in anyway.
Priority: Medium
Reference: EUGFR030

DM-2 File Transfer
Description: The IRMOS platform should provide the users with a mechanism to upload audiovisual material (or whatever type of files) in a central repository and to transfer files between users.
Priority: High
Reference: FPUC05, ELUC01, ELUC05, VAR07 and VAR13

DM-3 Data Storage
Description: The IRMOS platform should provide the users with the data storage means needed to upload audiovisual material (or whatever type of files) and for accessing and modifying further these contents.
Priority: High
Reference: FPUC05, ELUC01, ELUC05 and VAR07

DM-4 Scanning/Ingest service
Description: The Data Storage service should provide the capability of *digitizing/ingest* of A/V content including *metadata generation*. Work is also required on the part of WP4 for the implementation of the respective application services.
Priority: High
Reference: EUGFR29

DM-5 High storage size capability
High storage size capability is required.
Priority: Low
Reference: EUGFR39

DM-6 Data/Information Integrity
Description: The platform should be able to provide *data/information integrity* meaning that the data does not suffer any change in the transmission process and if a change occurs the receiver has to be notified.
Priority: Medium
Reference: EUGFR02

NET-1 Network provision reservation
Description: Each end user should be able to access the IRMOS platform at any time with a specific agreed bandwidth. It should be a target for SLA.
Priority: Very High
Reference: All Use-cases D2.1.1

NET-2 Various users connected to the same application
Description: The IRMOS platform should provide with the needed mechanisms for various end users in different locations to access the same application at the same time.
Priority: Very High
Reference: FPUC03, FPUC04 and ELUC09

NET-3 Possibility of defining a Computer Cluster Network
Description: The IRMOS platform should provide the means for defining a networked computer cluster and presenting it as a single resource.
Priority: High
Reference: Film Postproduction Storyboard (Step 2 - Dailies Review) D2.1.1 [1]

NET-4 Network Resource Monitoring and Reporting

Description: The usage of agreed resources must be monitored to permit the representation of statistics to the end user for tracking. The network layer must provide means for monitoring and reporting measures.
Priority: Medium
Reference: VAR04

NET-5 Provide edge-to-edge QoS for soft real-time services
Description: The IRMOS platform should provide QoS in the transport network, which connects the service components of a virtual service network located at potentially different locations within the borders of the physical network(s) under control. QoS is defined by parameters like bandwidth, delay, jitter and others.
Priority: High
Reference: FPUC01, FPUC07, FPUC09, FPUC10, FPUC12, FPUC13, FPUC14, FPUC15, FPUC17, FPUC18, VAR02, VAR03, VAR04, VAR10, VAR11, VAR12, VAR13 and general requirement EUGFR06

NET-6 Collaborative Work
Description: The platform should provide the appropriate *API* and *interfaces* to support *collaborative work* to be executed through IRMOS. This includes *collaborative tools* shared by all the participants (e.g. shared whiteboard) and must be *real-time* enabled.
Priority: Medium
Reference: EUGFR02, EUGFR03

NET-7 Virtual Reality and Real World in the same scenario
The platform should provide the *means*, especially concerning *workflow capabilities*, to combine *Virtual Reality* and *Real World (Augmented Reality)* in the same scenario. The teacher and the students are real, but they have got their avatars present in the virtual world.
Priority: High
Reference: EUGFR08

SLA-1 SLA Management service
Description: The IRMOS platform should provide an SLA Management service to negotiate and handle QoS guarantees.
Priority: Very High
Reference: FPUC01, FPUC13, FPUC14, FPUC18, VAR02, VAR03, VAR04, VAR10, VAR11, VAR12 and VAR13

SLA-2 Provide tools to perform SLA Negotiation (between WP5 and ISONI)
Description: The IRMOS platform should provide means for negotiating SLAs on the technical level, that describe and fix the required resources and QoS requirements requested. These need to be directly linked to monitoring mechanisms to proof that the SLAs are not being violated or to report that they are.
Priority: High
Reference: FPUC01, FPUC13, FPUC14, FPUC18, VAR02, VAR03, VAR04, VAR10, VAR11, VAR12 and VAR13

EE-1 Real-time deployment
IRMOS needs not only to ensure that the services deployed are running under real-time constrains, but also that the mechanism that deploys these services is real-time aware and therefore the deployment, instantiation and availability of the required underlying resources complies with the real-time constrains.
Priority: High
Reference: Potentially all use cases D2.1.1 [1]

EE-2 Control of the application
The application user must be able to *start*, *stop, pause* and *resume* the application. This involves work done both by the *framework services* as well as the *execution environment*.
Priority: High
Reference: EUGFR04

SEC-1 Provide Secure user Authentication to access the IRMOS platform
The IRMOS platform should provide the end users with a means for accessing it through an identification mechanism.
Priority: Very High
Reference: FPUC02, ELUC06 and VAR01

SEC-2 Access Denied to the IRMOS platform
The IRMOS platform should be able of denying the access to unidentified users and reporting this identity violation attempt.
Priority: Very High
Reference: FPUC02, ELUC06 and VAR01

SEC-3 Authorization mechanism
Once a user has been successfully accessed and identified, the IRMOS platform should be able of associating this user with a predefined profile.
Priority: Very High
Reference: FPUC03 and ELUC02, ELUC06

SEC-4 Logout from the IRMOS platform
The IRMOS platform should provide the users with a means to logout from the platform.
Priority: Very High
Reference: FPUC20

SEC-5 Secure data exchange
Secure data exchange allows secure transmission of data, e.g. simulation data, video and audio, to be transferred between resources in IRMOS or from outside IRMOS into the IRMOS framework in advance or during a session. This should be *customizable* by the end user, meaning that he can have a choice between *secure/non-secure transmissions*.
Priority: Medium
Reference: EUGFR26, EUGFR27

SEC-6 Data/Information Confidentiality

By *confidentiality* we mean that the data stored within the platform can be *accessed* only by *authorized* users. It is closely related to secure data exchange.
Priority: Medium
Reference: EUGFR01

The following table summarizes the identified platform requirements and associates them with the platform capabilities. As already mentioned, the requirements list will be extended as the application scenarios are detailed in depth and the development WPs are in position to provide more information about the implementation of their components. Several requirements, such as the EE-1, EE-2, are already associated with building blocks or components (the execution environment in this case) and it is expected that in the next versions of the IRMOS Overall Architecture report, WP3 will provide a complete list of requirements and the specific components of the platform that address them.

Table 2: IRMOS Platform Capabilities & Requirements

Key Capability	Requirements
Workflow Management	VS-4, VS-5, NET-1, NET-2, NET-3, NET-5, NET-6, NET-7,
SLA Management	VS-1, NET-1, NET-2, NET-5, SLA-1, SLA-2
Data Management	DM-1, DM-2, DM-3, DM-4, DM-5, DM-6,
Streaming	VS-1, VS-3, VS-5,
Discovery	NET-1, NET-3, SLA-2,
Monitoring	NET-4, SLA-2
Identification and Security	VS-2, DM-1, DM-6, SEC-1, SEC-2, SEC-3, SEC-4, SEC-5, SEC-6

5.2. IRMOS Platform Capabilities

The following capabilities for the IRMOS platform emerged from the application requirements and use cases as they are described in the previous sections and also from the general functionality that a real-time aware SOI must provide. For each requirement, a general description is given at the beginning, about the capability and the sub-sections involved. These sub-sections may include additional steps necessary for the implementation of this capability. In chapter 6, the IRMOS subsystems are presented, the capabilities and functionalities are associated with particular blocks or components and we expect in the next versions of the architecture report to provide additional details for the capabilities and their implementations in IRMOS.

5.2.1. Workflow Management

Workflow modelling and definition tools are used to generate a workflow description. This workflow description is then propagated to a service called the *Workflow Enactment Engine*, which is a run-time service responsible for the execution of the described workflow. Workflow enactment services are concerned with the tasks of scheduling, reservation and fault management.

5.2.1.1. Workflow Enactment & Scheduling

The *scheduling capability* of IRMOS should be a centralized one meaning that all scheduling decisions for all tasks in the workflow should be taken by one central scheduler. This is the best architecture concerning the performance of the platform. However, the resources that are used for the workflow execution may have their own mechanisms for monitoring and management of the workflows execution process to prevent failures and therefore SLA violations.

There should be also a *performance estimation scheme*. This is very important as it can be used to predict the overall performance of a workflow and the best decision can be made on which resources will be used. Performance estimation can easily be done through historical data, though this is not the best approach. In this case the capability for *logging* must be provided. Simulation-based performance estimation should also be considered.

Finally, the *selection service,* that is responsible to select the most suitable service from a list of candidates, should be extensible with any given selection algorithm, performance or market driven.

5.2.1.2. Workflow-Aware Advance Reservation

Advance Reservation allows the service user to request resources from systems for a specific time interval, i.e. start time and end time, and obtain a sufficient number of resources during this time interval to support the execution of the specific job for which the resources were reserved. It is widely considered as a mechanism to provide end-to-end QoS guarantees. Therefore IRMOS should provide the following capabilities:

- The user should be able to *reserve resources* for a *specific time period*.
- Advance reservation should be done at *the workflow level*.
- The user should be able to *impose QoS constraints* both on the workflow level and/or at task level. This is because the user may be interested either on the performance/cost of the workflow or on specific parts of his workflow.

5.2.1.3. Fault Detection & Recovery

Fault Recovery must be an integral part of IRMOS in order for the platform to be robust. To this end the following capabilities should be provided:

- <u>On task level</u>:
    - o If an error occurs, the system should be able to execute the same task on the same resource. This should be provided by IRMOS as it is the simplest and fastest way to recover from minor errors.
    - o The system should have the ability to execute a failed task on a different resource. This should be investigated further in conjunction with the *redundancy* that ISONI provides.
    - o The workflow state and URL references to intermediate data should be saved so as to be able to be recovered in case of a failure from the same task running on a different resource.

- On the workflow level:
    - The workflow should be able to continue executing even if a task has failed, if this is possible. This can be the fastest way to recover from a minor failure.
    - The system should provide the user with the ability to define corrective actions in certain cases of failure. This can make the system much more tailored to the needs of the users.
    - Heavy weight checkpointing should be provided. In this case backup copies of intermediate data should be stored on a database in order for the execution of the workflow to be able to continue at any given time from any available resource. This can make the system heavily robust as the entire workflow can be rescued as the intermediate data are not lost and resources can be compensated by the redundancy that ISONI provides. The *delays* that this may pose should be investigated.

5.2.1.4. Interactions & Dependences

All parts of the workflow enactment engine rely heavily on the *Monitoring Services* provided by the platform, in order for it to make decisions during scheduling or take corrective actions in case of a fault or performance degradation. Workflow enactment also relies on *SLA Management*.

5.2.2. SLA Management

The management of the SLAs represents different capabilities all along the different phases of the service lifecycle.

5.2.2.1. Publication

SLAs templates should be provided by Service and infrastructure providers as the initial point of SLA establishment and stored somewhere in the IRMOS platform. Tentative components that apply to this step: SLA Template Repository, SLA Repository, SLARepositoryAccessor (which is a component that performs select/insert/update in both repositories).

5.2.2.2. Negotiation

Before a certain service is selected to be executed, the SLA has to be negotiated between the two different parties (user or agent on behalf of it and Service provider) taking as initial input the template provided in the publication step. The output of the negotiation will be a well-established and agreed SLA that should be stored in a SLA Repository. Components that are foreseen to participate in this process are: SLANegotiator, MappingComponent, SLARepositoryAccessor, SLATemplateRepository, and SLARepository.

5.2.2.3. Execution

During the service execution two different processes regarding SLA Management need to be considered:

- **Monitoring**: The SLA Management should also include/incorporate mechanisms to monitor if the execution of a certain service is always within the limits specified in the SLA or if some service violations occur. Components present in this process could be SLAMonitoring that will take as input relevant metering information and SLAEvaluator that will be in charge of analysing if the metering information is within the limits of the previously signed SLA.
- **Enforcement**: In case a SLA violation is detected or foreseen, corrected or recovery actions have to be performed. Relevant components of this process are: SLAViolationManager that will be in charge of defining actions that should be performed for the particular case of violation and also will communicate to the appropriate components the actions to be performed in order to re-establish the service execution under the SLA conditions.

5.2.2.4. Dissolution

Once the execution of the services finished, a process should take place in order to free resources used within the execution phase. SLAs should be then deactivated or deleted from the SLARepository. Components: SLARepository, SLARepositoryAccesor.

The presence of an SLA Manager component is necessary in order to coordinate and drive all the SLA actions identified in the different phase we have just mentioned.

The nature of the SLA Management adds two more important capabilities:

- **Dynamic**: The SLA negotiation should be a dynamic process at least in the negotiation between the platform and the Infrastructure Provider so as to allow all the high level requirements of the application to be mapped in low level (resource) parameters, which will be finally included in the SLAs.
- **Automatic**: The SLA negotiation and monitoring should include automated mechanisms since this is one of the main inventions that should be addressed by IRMOS.

It is foreseen that there will be two different kinds of SLAs to be handled in the IRMOS platform:

- **Application SLAs**: These contracts are signed between the service/application provider and the user of the service/application.
- **Technical SLAs**: Related to the application SLA, this technical SLA is signed between the IRMOS middleware (layer responsible of WP5) and the ISONI layer (Infrastructure of network of resources like CPU, storage and network managed and controlled by a middleware).

5.2.3. Mapping

When an application is going to be executed in the IRMOS platform, it has some requirements that must be met so that the application will run efficiently and within the

QoS that is required. However, the application users usually define high level requirements such as frames per second or execution time, so there must be a mechanism that maps these requirements to low level parameters which can be understood by the framework services and the execution environment.

The mapping services of IRMOS should have the following capabilities:

- *Analyse* the application and *estimate* the required resource parameters. The resources parameters can be CPU cycles, RAM memory, data storage, etc.
- Take into account the users that will potentially use the application and calculate the expected bandwidth.
- *Estimate* license fees if the application is commercial.
- *Calculate* the pricing of the service according to the SP's business model, the costs, the time constraints posed and the market policy that is going to be applied.
- The mapping component should be as *generic* as possible, to cover a wide range of applications.
- Use a well structured descriptive language. For all resources which have to be described, there should be a uniform syntactic construct. Different kinds of resources should be specified by means of attributes and their values. In addition, the description language should distinguish between concrete resource instances and groups of similar resources. By this way, the SLA negotiator could use this information so as to search for more generic resources and increase the possibility of fulfilling the SLAs.

5.2.3.1. Interfaces

The mapping component should probably interact with:

- the workflow modelling service to acquire the high level application parameters,
- the simulation/planning components to acquire the parameter and workflow mapping hints,
- a mapping policies repository from which it retrieves mapping policies,
- and the SLA Manager/negotiator to which it sends the low level parameters.

5.2.4. Discovery

Service Discovery (SD) is the process that enables the locating of either a specific application service or a service that nearly matches the user criteria among a pool of services. In a large scale, heterogeneous network the provided resources increase rapidly and it is hard to locate the desired service, so it is very important to develop mechanisms which will be able to do the matchmaking efficiently and in time. SD process may involve several components that span between different layers of the platform and interact each other and with other platform services.

The IRMOS platform has certain requirements so service discovery mechanisms must have the corresponding capabilities so that these requirements will be fulfilled. Based on the user requirements we expect that the applications and the end users will require access to legacy systems such as post production dedicated devices and location detection systems. These systems may communicate in different ways with the platform but in all cases the discovery mechanisms should be able to trace them and inform the application and the end users for their existence and availability. The scope of discovery process in IRMOS is extended in order to take advantage of mechanisms for Virtual Service Network (VSN) Discovery, Mobile Resources Discovery using SIP implementations and Legacy Systems Discovery as presented in the following UML diagram. Part of the discovery process is the Advertisement/Registration of all type of resources so as to be discovered using specific criteria about their type and QoS characteristics.

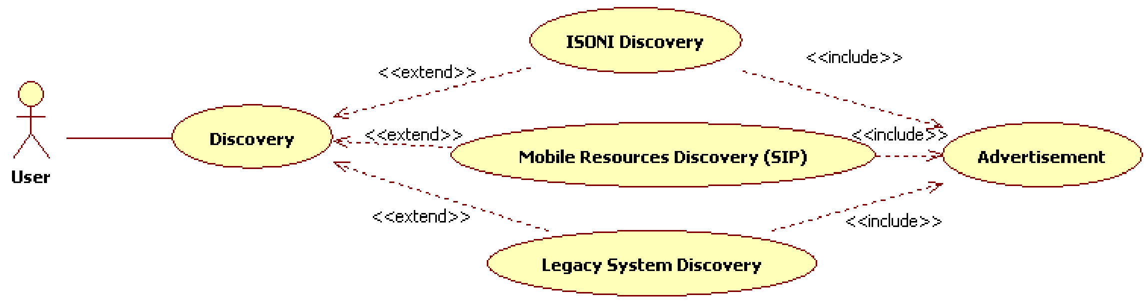

Figure 2: IRMOS Platform Discovery

Additionally, we expect the discovery capability of the IRMOS platform to enclose the following features:

- *Event Notification:* The status of the provided services and systems should be reported and updated periodically so as for the discovery and matchmaking mechanisms to be able to find functional resources that meet the advertised parameters. This functionality will improve the provisioning system of the IRMOS platform and will constitute the basis for accomplishing the real-time requirements.
- *Syntax Expression:* The description of the provided service is very important because it affects directly the matchmaking. The description syntax has to be simultaneously flexible and precise. Xml-based schemata are very powerful in describing the service and resources and at the same time allow the use of OWL [13] for semantics.
- *Semantics:* Sometimes when the SLA negotiator asks for a service, and the discovery service cannot find one in the services pool that fulfils the SLA parameters, it must be capable of returning a list of services that are close to satisfy the SLA parameters. Semantics and ontologies can solve this problem and grant additional functionalities to the discovery service. It is possible to make use of the Class/Subclass hierarchy of OWL to group services based on their functionality and the QoS class that they belong to. For example potential resources can be found that are not known in advance but are able to satisfy user requirements in a more practical and automated manner.
- *Multiple Matchmaking Mechanisms:* The discovery service must be able to use multiple searching and matchmaking techniques which can increase the adaptivity and effectiveness of the IRMOS platform in terms of variety of the devices and service types that will be used and their priority.

- *SIP Architecture Capabilities:* It is expected that SIP [14] architecture will be used for the discovery. SIP architecture provides a set of capabilities that certainly apply to the IRMOS platform. SIP SD provides subscription-notification mechanisms which make the architecture very flexible. For instance, besides the conventional SD functionality, it is possible to subscribe to a certain event of a certain service (even if the service is not there yet). Low bandwidth consumption and reduced message interchange are also key requirements for the real-time application constraints that are satisfied by the SIP SD.

5.2.5. Monitoring

Monitoring is necessary for analysing the complex behaviour of a distributed system, for fault detection and for the discovery of performance issues. To achieve this, data about the system must be gathered and processed to reveal important information. If a performance problem arises, notifications are sent to the appropriate components for taking corrective actions. Monitoring data represents an operational snapshot of the system behaviour along the time axis. Such information is fundamental in determining the origin of the problems or for tuning different system components. Especially in IRMOS where we care about real-time performance and fault tolerance the monitoring system must be able to:

- monitor the Quality of Service,
- monitor service level agreements,
- check resources status,
- and provide monitoring information used for controlling tasks.

In order to accomplish the above, the monitoring system should have the following characteristics:

- **Short lifetime:** If a particular server has a deficient performance over a period of time, such information is mainly important at that time in order to correct such behaviour.
- **Dynamic data monitoring:** Since the monitoring data reflects the online system performance, the information updates are very frequent. At the same time monitoring should not interfere with the system being monitored to ensure accuracy of measurements and balance must be kept between the accuracy and the freshness of the data.
- **Statistical aspect of monitoring data**: A single value of a certain parameter such as CPU load is not as important as the collected values over a certain period of time.
- **Generic Data representation:** In a heterogeneous system like a grid, a consistent set of possible measurements must be agreed upon and provided by all parts of the distributed system. These grid metrics must be standardized so that the data performance can be comparable.
- **Minimal measurement overhead and quick transmission:** Monitoring data should not be very thorough because that would reduce the performance of the measured resources, and it would occupy the network with not so useful information. Moreover the monitoring information should be transmitted as quickly as possible so as to notify other components in time for resource failures.

- **High data rate generation and scalability:** In a distributed environment we have a very high rate of monitoring data so the monitoring system should be able to handle them efficiently. Furthermore it should be scalable enough so as to handle a possible extension of the platform.
- **Secure:** Information generated by the monitoring system is sometimes very sensitive. In that sense, the monitoring system should ensure the data integrity and access permissions.

5.2.5.1. Interfaces

The monitoring & metering services should interact with the ISONI interface so as to gather information about the availability and the status of the resources and the SLA enforcement components because they should take notifications for services failures.

5.2.6. Data Management

From the initial requirements analysis report D2.1.1 [1] as analysed in paragraph 5.1 of this document the main functions of any storage within the IRMOS framework must be:

- Large capacity
- Resiliency
- Secure access
- Known access latencies and jitter
- Association of metadata and data

Addressing the capacity of the storage on its own is insufficient at this point, initial work within the development of the execution environment of IRMOS and the ISONI network overlay will look at the transmission and so the data access bandwidths that are both required by the applications and possible for the full end to end system. Looking at the capacities alone as they are reported today the development of enclosures that are able to present 100TByte as a single resource are required by the IRMOS use cases. Some of the improvement will come from the increase in capacity that is foreseen in the disk market, in addition there will be work undertaken within IRMOS to increase the number of devices that can be contained in a given space to enable both increases in capacity and increases in the aggregate media access bandwidth.

Object based storage is currently under discussion within standards bodies and the early indications are that adoption of the object rather than the block as the atomic unit of data access has the potential to deliver a number of the other requirements for storage within the IRMOS framework. In particular the object framework includes the concept of access control restricting access to those who have the appropriate credentials. This is an intrinsic part of the data access process as illustrated below and will provide secure access to the storage system. The use of the object storage device approach also enables custom meta-data to be stored along with the data as a single object. This results in a close association of these items.

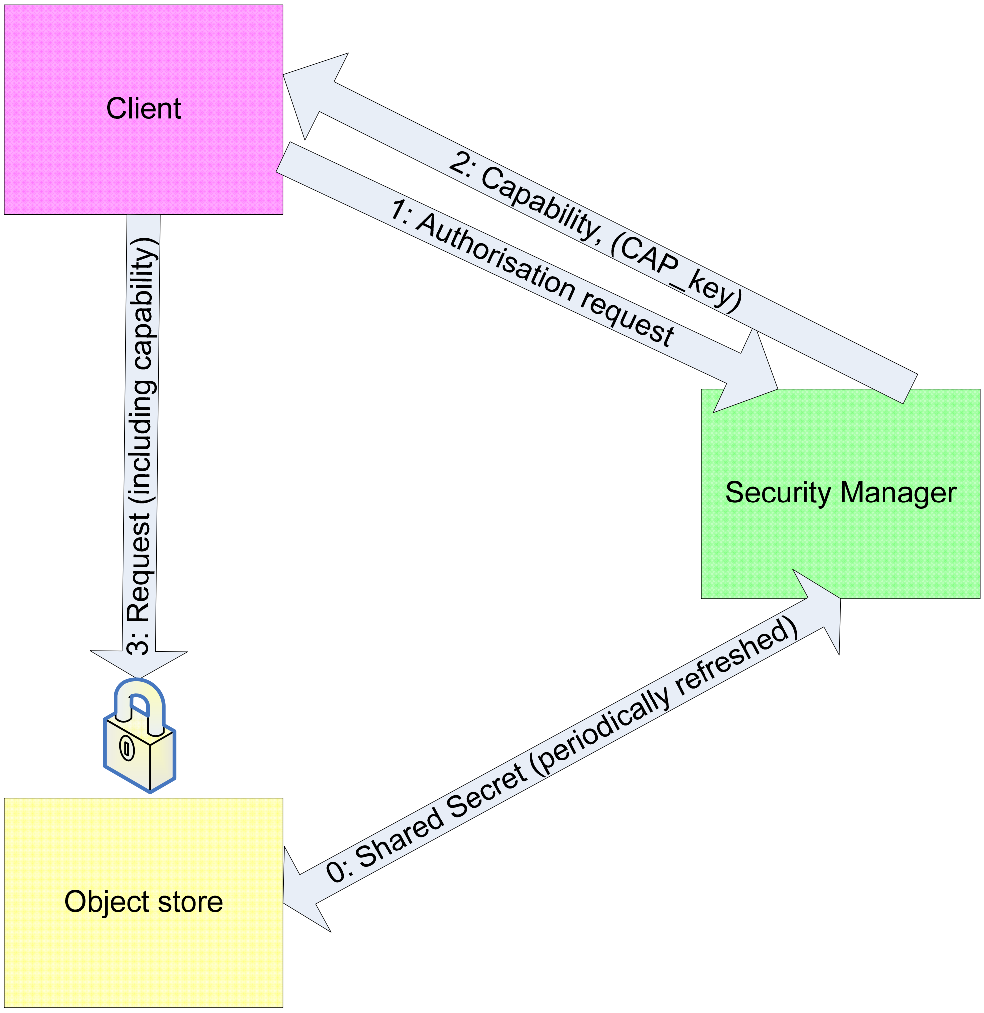

Figure 3: Object Storage Security model

Quality of service for storage systems is not something that has received much attention to date, there are a small number of trials at the present time (see [2] for details) however none of these involves the installation of the device in a QoS aware environment. Determining how QoS can be described, measured and negotiated is one of the key items of research for the early phases of development within work-package 6 of the IRMOS project.

5.2.7. Streaming

Data streaming is an important factor and widely used for data transmission of different types, e.g. video or audio streams. This makes streaming also valuable within IRMOS especially when offered as a service to applications. The idea behind this is to have a streaming service that can be used by arbitrary applications through an entry and an exit point in the IRMOS framework. For best performance these points would be in the same ISONI domain so as the QoS between these points to be guaranteed. Conceptually to perform generic streaming mechanisms a separation of the service in two layers would be advisable.

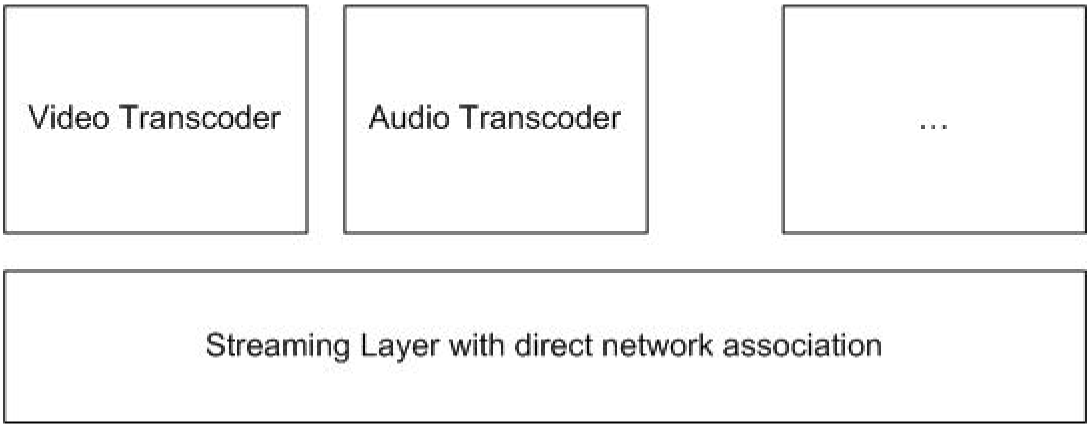

Figure 4: Basic conceptual approach to a streaming service

The basic layer performs all necessary network communication expecting data to be passed to him by the above layers, e.g. a video transcoder. It has to be pointed out that transcoding is only exemplary and could also be only a component that simply passes an already encoded stream to the streaming layer.

Another question is on how the service should be placed in the overall context of IRMOS. As the service itself could be time-consuming when assuming usage of video transcoding and with regards to the requirement of close physical proximity to an application making use of this streaming service, it is most advisable to design the whole service as a service that actually can be deployed in an execution environment of the ISONI layer and in that way, ISONI will be able to guarantee the required QoS level for the service. The main focus is on video transcoding that is required by most application scenarios of IRMOS. Considering security of streams it should be decided whether encryption of a stream should take place on the streaming layer or whether this might rather be a task for below located layers, e.g. virtualised network layers as provided by the ISONI.

5.2.8. Execution Environment Management

Essentially, the Execution Environment offered by WP6 is a virtual machine with added capabilities. These added capabilities are what make our Execution Environment unique, innovative, advanced and perfectly suitable to the goals of the whole IRMOS project.

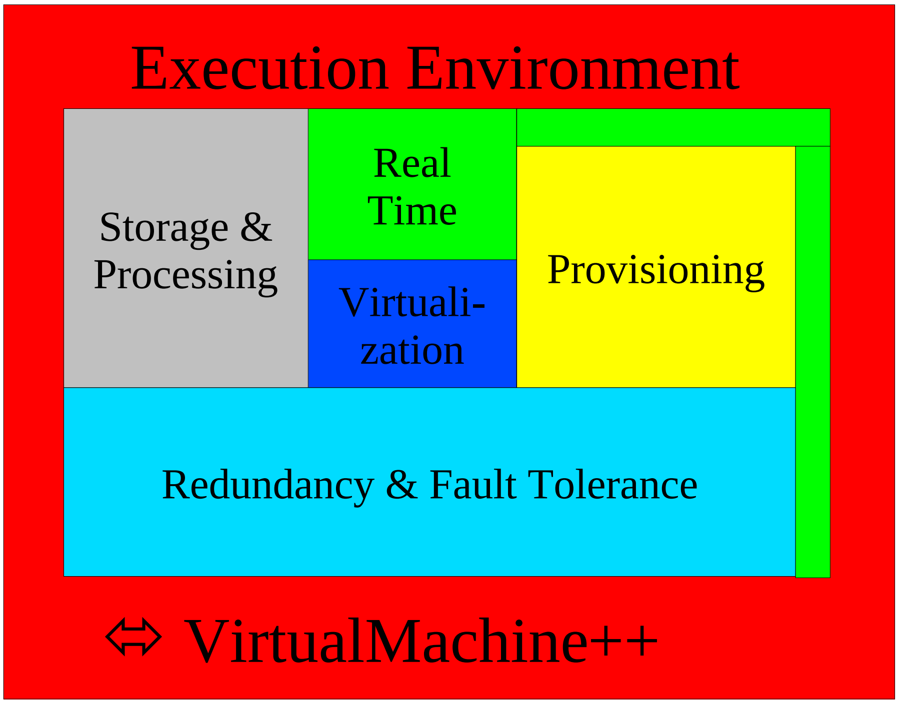

Figure 5: Execution Environment

One of the fundamental capabilities of an EE is its isolation capabilities. We prohibit services from interaction with other services that are not explicitly allowed by policy. Services are also isolated from the failure or "misbehaving" (like a program bug that makes the software run in an infinite loop requesting all CPU power) of other services.

Real-time is a distinctive capability of our EE and the IRMOS platform will be able to detect and exploit this real-time capability. We mention that to achieve this innovative feature, we should be developing a global resource management and allocation policy. Also we use appropriate schedulability methodologies.

Our execution environment will have redundancy and fault tolerance capabilities that, combined with the migration process, will assure that the SLAs engaged with the services running on top of our execution environment are kept even in case of failures. Particularly challenging is the capability of the EE and IRMOS platform to let real-time applications to suffer as little as possible, if at all, during a fault.

Very good fault tolerance can be achieved by providing physical backup capacities for each single physical instance, but this is obviously expensive. Another capability of our Execution Environment is to calculate when to start the booting process so that the service becomes ready just in time, not after -that would violate the SLA- but no much before -that would allocate resources during a non necessary time interval-.

Our EE has the capability of providing appropriate mechanisms and interfaces for resource metering, monitoring and reporting of the software running in it.

5.2.9. Network Management

Network management will provide a failsafe network overlay by connecting components through an intelligent network infrastructure, enabling automated SLA negotiations and monitoring & reporting to enable delivery of QoS assurances as required by real-time interactive applications.

Specifically the following capabilities are required for the Network Management:

- One of the fundamental capabilities of the ISONI (Intelligent Service Network Infrastructure) research topic is to offer a framework for automated provisioning of multi-component services (described by VSNs) across different locations under consideration of "edge-to-edge" service guarantees. One of the key components required to fulfil this ambitious task is to provide QoS guarantees in the transport network, which connects the service defining components in potentially different locations.
- The important concepts with respect to network virtualization in the context of IRMOS are segmentation, isolation, and encapsulation. Segmentation allows several different services to share a physical link with given QoS properties. Means for isolation and encapsulation are needed to mimic a dedicated virtual network for every individual service and to suppress any unwanted crosstalk between services sharing the same physical link.
- Network resource management focuses on ensuring the efficient and reliable utilization of available network resources. This includes topics like ensuring Quality of Services, ensuring reliable connectivity, reducing costs and so on. In a framework oriented to support Real-Time applications and services, it is a crucial aspect the global path supervision and link control tasks, for checking both the status of the network and links.
- When focusing on soft real-time systems, the research efforts shift completely on the problem of finding appropriate trade-offs between an efficient/optimum resource usage and the guarantees provided to the real-time activities. In this context, QoS optimization and adaptive QoS control for soft real-time tasks are particularly relevant research areas.
- In order to support redundancy and life migration as performed on Execution Environment layer to meet real world threats like: components failure, connection failures, etc… the intelligent network layer will support those EE counteractions on network layer.

5.2.10. Modelling, Analysis and Planning (MAP) Environment for interactive real-time services

IRMOS services can provide strict guarantees on responsiveness and performance. However, ISONI services are soft real-time services. The `soft' aspect of such services means that in practice response times and performance levels are uncertain and that exceptions have to be allowed for and handled without adversely affecting service behaviour. IRMOS guarantees are therefore guarantees that response and performance values will occur within a given probability distribution. The guarantee expresses that there is an extremely high level of confidence that this distribution will match real world observed behaviour.

Managing the uncertainty in order to optimise services is critical for all parties in the value chain. The uncertainties arise because

- ISONI Network and EE management will not have absolute guarantees of QoS resource performance
- Service providers will break SLAs on occasion, possibly intentionally because of external business factors
- Workflows are complex and may not be completely defined before run time
- When users interact with services they may decide to alter the current workflow "on the fly"

The purpose of the modelling analysis and planning (MAP) environment is to give actors in the value chain support in managing these uncertainties. Hence the MAP environment will require capabilities that help actors in the value chain to:

- estimate and optimise resource provisioning needs necessary to enact a workflow according to SLA guarantees
- design workflows and services that have measurable and predictable behaviour
- estimate service performance and responsiveness within normative workflows
- discover exceptional behaviour that is likely to break QoS guarantees at both the service provider level and the ISONI level
- identify critical interdependencies and resources that can lead to bottlenecks in order to improve service performance
- refine high level application and workflow performance requirements into ISONI QoS parameters

The MAP environment will also have the potential to help application developers understand how best to take their applications and break them down into a set of services underpinned by ISONI in a way that is efficient and results in behaviour that is predictable.

5.2.10.1. IRMOS Architectural Dependencies

The MAP environment will be dependent on inputs from many of the other IRMOS architectural components. Workflow management, SLA management, performance monitoring, parameter mapping, all have the potential to influence the MAP environment.

- input from Workflow management
    - interdependencies between activities
    - timing constraints on start/stop/duration of activities
    - attributes of timing constraint, e.g. hard/soft/priority
- input from SLA management
    - QoS performance and response requirements for each activity class in a workflow
    - policy for timing and performance constraints and dynamic modification
    - policy for workflow optimisation with regard to QoS parameters

- input from Mapping (note these may actually come from the service provider who supplies these to the Mapping component via some suitable interface)
    - translation between service resource requirements and ISONI requirements
    - translation between service level and ISONI level responsiveness constraints
- input from EE and Network management
    - timing and performance constraints on ISONI parameters, e.g. probability density functions defining performance metrics
- input from Monitoring (note these may be provided by EE and Network management, but be transmitted to the Monitoring component on request)
    - historical ISONI performance data
    - historical resource reservation timing data, e.g. time required to reserve particular resources for particular duration
- outputs
    - ISONI resource estimate for workflow enactment, including timing constraints on resource reservations
    - estimate of workflow compliance with service provider SLA, e.g. given QoS failure profiles for certain activities estimate overall failure profile for workflow
    - sensitivity analysis of service performance to workflow variation
    - scenarios describing potential exceptions to QoS criteria
    - enhanced workflows compensating for particular exceptions occurring from activity failures during runtime

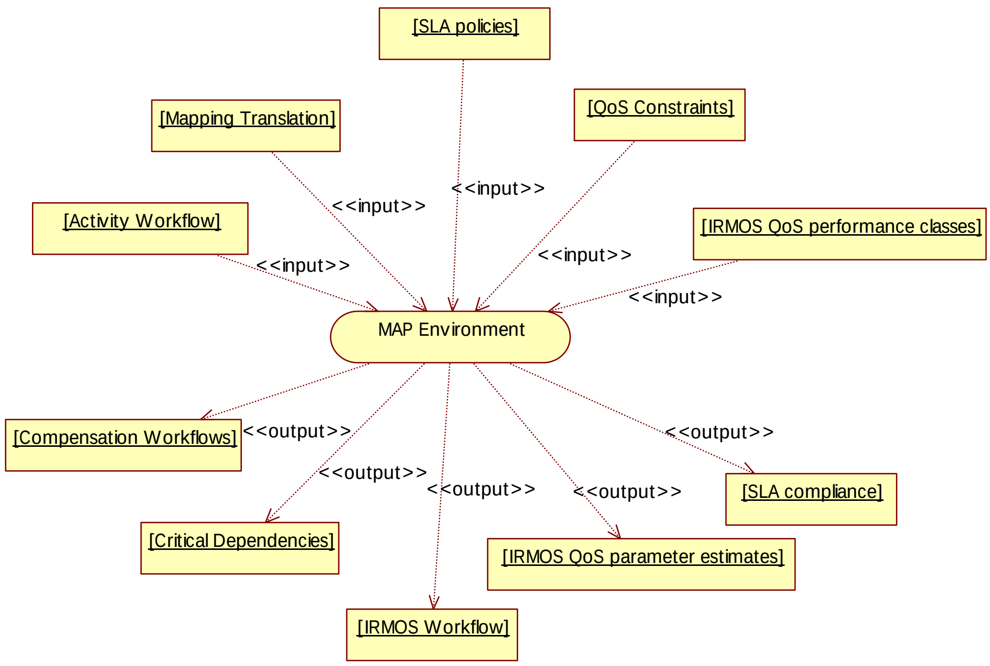

Figure 6: Modelling Analysis and Planning Environment Input Output

Figure 6 describes possible outputs that the MAP environment can provide once all of the inputs are available.

5.2.10.2. Implementation references

Section 4.5 in deliverable D2.3.1 State of the Art on IRMOS technologies [2] details various tool sets that have been developed in relation to MAP capabilities. In particular Section 4.5.2 includes details of queuing theory, stochastic process algebra, and continuous time Markov chain (CTMC) analytical technologies.

Several of the above concerns are not unique to IRMOS and have been considered in the context of embedded real-time systems. Aspects of performance modelling for real-time systems have been considered by OMG in the UML profile standards [15] and [16].

These profiles do not provide an implementation as such, but define a framework for describing real-time modelling activities. Section 4.5 also describes three FP7 research projects, SENSORIA, TAPAS and EUQoS that have applied some of these technologies in domains that are complementary to IRMOS.

5.3. IRMOS Platform Security

Security is a crucial and complicated task in IRMOS platform. In such systems security cannot be considered as a plug-in to be added once the product is finished. Security analysis and design should be considered from early stages in an integrated fashion with the design of IRMOS components and their interactions. IRMOS and particularly its security architecture need to address the following issues.

5.3.1. Trust

A trusted system or actor is one we assume will operate in the (beneficial) way we expect. We depend on trusted systems or actors to behave as expected. Thus a good working definition of trust is "dependence on the actions of another party". The trusting party can gain advantage by assuming the trusted party will behave as expected. However, if the trusted party does not behave properly, the trusting party will at best not gain anything, and may be damaged.

The security architecture should make clear which systems or actors are trusted, and show either that they are very unlikely to misbehave, or else that the damage this would cause is relatively minor.
Trust terms are expressed in an SLA. The SLA defines and specifies the actions underlying the trust relationship. It constitutes a collaboration security policy in the generic sense of term. It may also include "overdraft" facilities as well as penalty clauses.

Trust is required for flexible, distributed security architecture. User identification, authentication and authorization for accessing resources requires defining trust relations amongst the different value chain actors mainly the client organisation, service provider and ISONI provider.

5.3.2. SLA and Security

Security concerns different aspects in IRMOS infrastructure and affects the application lifecycle starting from SLA negotiation until collaboration dissolution passing by workflow specification, environment set up and SLA enforcement.

Secure SLA negotiation and agreement is the first issue to be considered. To have a legally binding SLA, there should be a way for the partners to guarantee integrity (and may be confidentiality) of the exchanged messages.
The client would need a way to prove that he has an active SLA approved by the Service Provider (SP). The SP would need a way to prove that a client has indeed asked for an SLA and accordingly some resources were allocated.

There may be different scenarios to achieve this. SLA negotiation protocols that guarantee non-repudiation may be used. This requires clear specification of SLA terms and multiple negotiation rounds to agree on the terms before signing the final document. The signed SLA may be stored for later use even after its expiry date.

As alternative we may stay independent of SLA structure or content but require logging the secure message exchange as a proof.
The key(s) used to sign the SLA (or exchanged messages) should be kept private and should be legally linked to the owner identity.

SLA itself may contain terms about security requirements to be guaranteed during the collaboration lifetime. This issue requires further treatment within the IRMOS SLA specification process as well as service/Service Provider/ISONI provider discovery process. In the following sections we deal with security during the SLA lifetime.

5.3.3. Identification and authentication

User identification process is needed and authentication is required as a basis of the other security services. According to the value chain, authentication may be influenced by the need for identity federation, single sign-on and require management interfaces.

Different authentication mechanisms may be employed [17]. Note that the participants would not change their existing authentication mechanisms to adapt to specific collaborations. If a domain is using Kerberos where another is using X509 certificates, credential translation is then necessary to authenticate users.

Identity federation is a process where a partner trusts identity credentials issued by the customer organisation. Two scenarios may be considered:

- Using the same technology e.g. x509 certificate. The SP may trust the issuing authority at the customer side.
- Using different technologies: user is authenticated at the home organisation and the result is sent back to the SP. The SP should trust the authentication

mechanism being used. This is part of the federation policy. The SP may ask for other attributes concerning user attributes.

The identity federation concept may be extended to attribute federation which may then include any user attribute.

Authentication system may also benefit of single sign on service that helps the client authenticate to different resources without the need to go through authentication process every time (e.g. enter the password each time).

5.3.4. Communication confidentiality and integrity

In a distributed architecture security should be supplied along the different paths of data. For instance, it is pointless to secure the connection between the client and Service Provider to transfer data, without requiring equivalent security on the communications amongst the framework services and ISONI service processing these data.

The SP should be able to deploy secure services on an open network i.e. services with communications confidentiality and integrity.

The end user may need to know if the SP services communicate in a secure way. This information may be a criteria for choosing this specific SP and not another. The SP on his side needs to ensure that security requirements can be met at the ISONI providers too. VSN naturally guarantee confidentiality and integrity being built on a VPN. However, access control and its enforcement still needs to be investigated (see below).

5.3.5. Access control management

Access control should be enforced on any point where data can be accessed. IRMOS framework services as well as ISONI services and resources need to be protected.

Regarding the potentially huge data volume and QoS requirements, it is highly probable that IRMOS architecture proposes that the user interacts directly with ISONI resources to upload/download data.

There are different issues to be addressed:

- Access control policy management- IRMOS should propose a process for managing access control policy at different components. Framework services, ISONI services as well as user information asset have to be protected against unauthorised access. Obviously, these services may fall within and interact across different administrative boundaries. For instance, an application user needs to access the framework services to design, simulate and start a workflow, framework services require access to local and ISONI resources for that, finally the user may need access privileges to access ISONI resources directly. A lot of questions arise here: Who manages these policies? How to enforce security policy through the different IRMOS layers? What are the possible policy translation and distribution steps needed starting from the user/application requirements till the execution? How to make the security policy available for the ISONI frameworks to enforce it whenever a client connects to it directly or via framework services?

- Delegation- It would be necessary in some cases for the framework services to do actions or access data on behalf of a certain user. These data may be at the Service Provider or at ISONI provider. User may need to delegate a service some of his privileges if they are required to access any resource.
- Access control granularity: in Web Services context, access control can be done on service, method, or resource levels. In the ISONI framework, a Virtual Service Network (VSN) can be accessed via a proxy in/out. It is not clear how fine we can go to control access to EE on ISONI nodes since ISONI framework is not application aware and therefore does not provide end user management / authorization. An application service may be only authorised to access certain nodes within the VSN and the responsibility of the end user access may be handled by the application itself.
- Until now, we have been considering dynamic users. However, in IRMOS we should consider the case of dynamic resources as well. How to deal with EE node migration? What is the impact on security and specially access control? If data has to be transported from an ISONI storage node to another, confidentiality and integrity may be offered within an ISONI domain. But it is not clear how to proceed if the data need to be stored in another ISONI domain or a non-ISONI domain.

5.4. IRMOS Conceptual Models

In this section the IRMOS Conceptual Models is produced. The models consist of the different use cases expected to be implemented within the platform, mainly describing through diagrams the interactions between the system and the actors. Entities (such as SLA Manager, Repositories etc) that will eventually form the overall IRMOS platform are identified in the form of modules and the relative sequence of necessary actions needed to fulfil the functionality are presented in the according sequence diagrams. At the time of finalizing this document, the IRMOS Value Chain definition has been still in progress, therefore, the names of the actors/entities here may differ from the ones in the final version of the Value Chain.

5.4.1. Application Modelling

In the following diagrams we abbreviate MAP Environment to MAPE when convenient. In Figure 7 the actors in the use case represent the IRMOS entities and the external actors that are associated with the use cases. The remaining figures provide a high level view of the protocols that specify conceptual scenarios for the use cases.

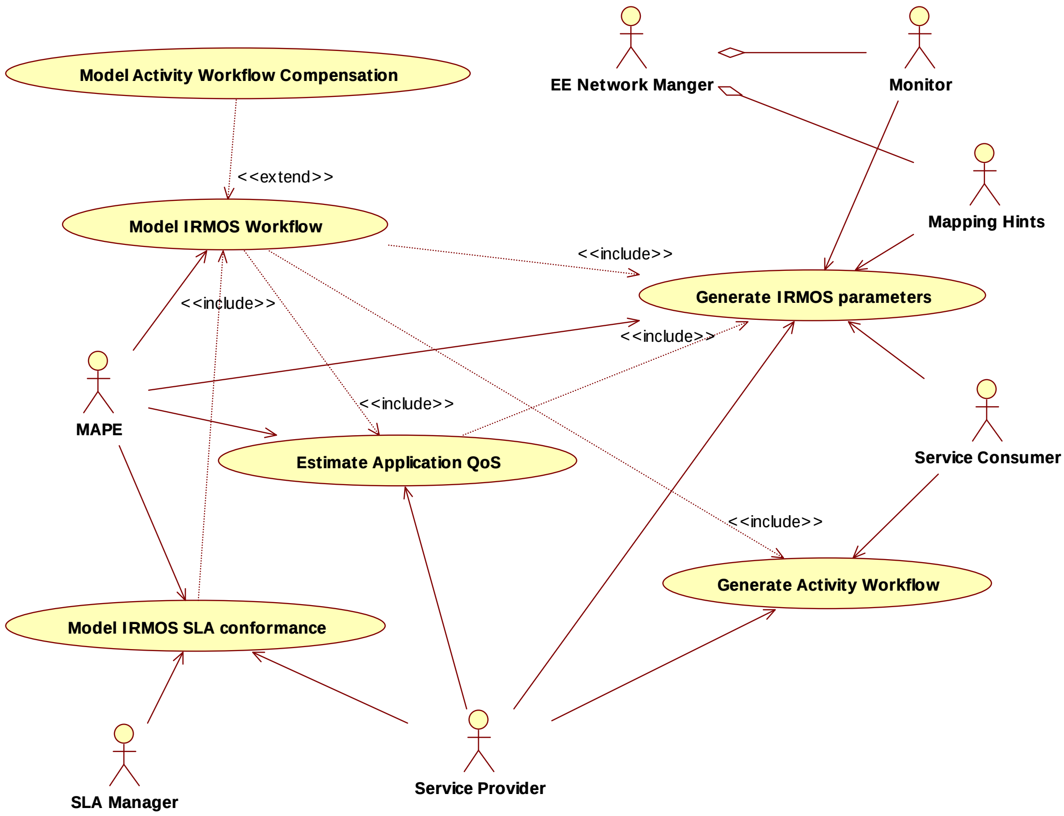

Figure 7: Main use cases for MAP

The following sequence diagrams provide a high level conceptual view of the use cases in Figure 7. The service consumer can be the end user or another service (workflow encapsulation). Also it is assumed that the ISONI Mapping actor has a repository of QoS data that is available to other IRMOS entities via a suitable interface. Similarly where there is some obvious output in a scenario we assume it can be copied to a repository for later access. Such a repository may be just a temporary object that is created for the lifetime of the service instance or it may be a permanent artefact.

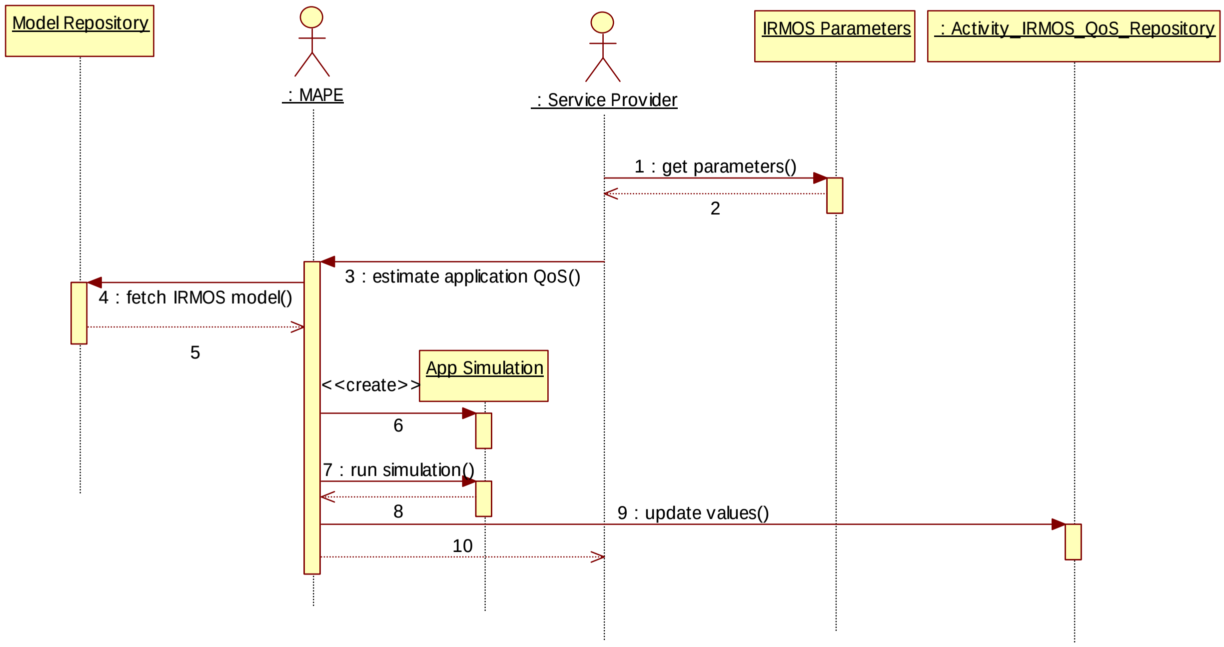

Figure 8: Estimate Application QoS Sequence Diagram

Figure 8 starts from the point where service activity performance constraints have been mapped into ISONI performance constraints. In the scenario the service provider is using the mapping to estimate ISONI QoS parameters that will be required to execute an application activity. For example, the service provider may be estimating the resources and responsiveness required to run a single interactive Vfx editing session with a single user for some period.

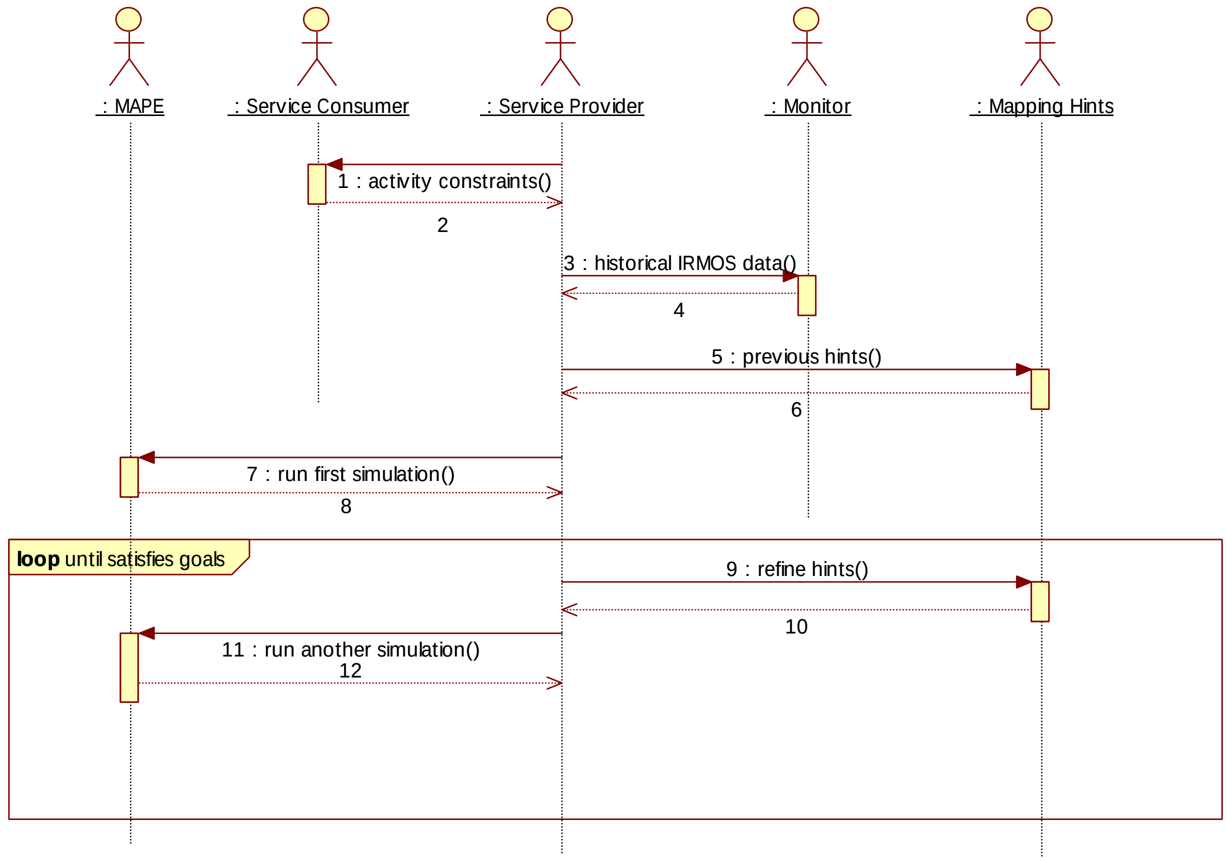

Figure 9: Generate IRMOS QoS parameters

Figure 9 describes the scenario where the service provider is using historical ISONI QoS data, hints from the Mapping entity and data from the service consumer to define a translation between activity performance constraints and ISONI performance constraints. Note the service consumer data will only define constraints at the service level. It is assumed the service consumer has no ISONI domain knowledge in this scenario.

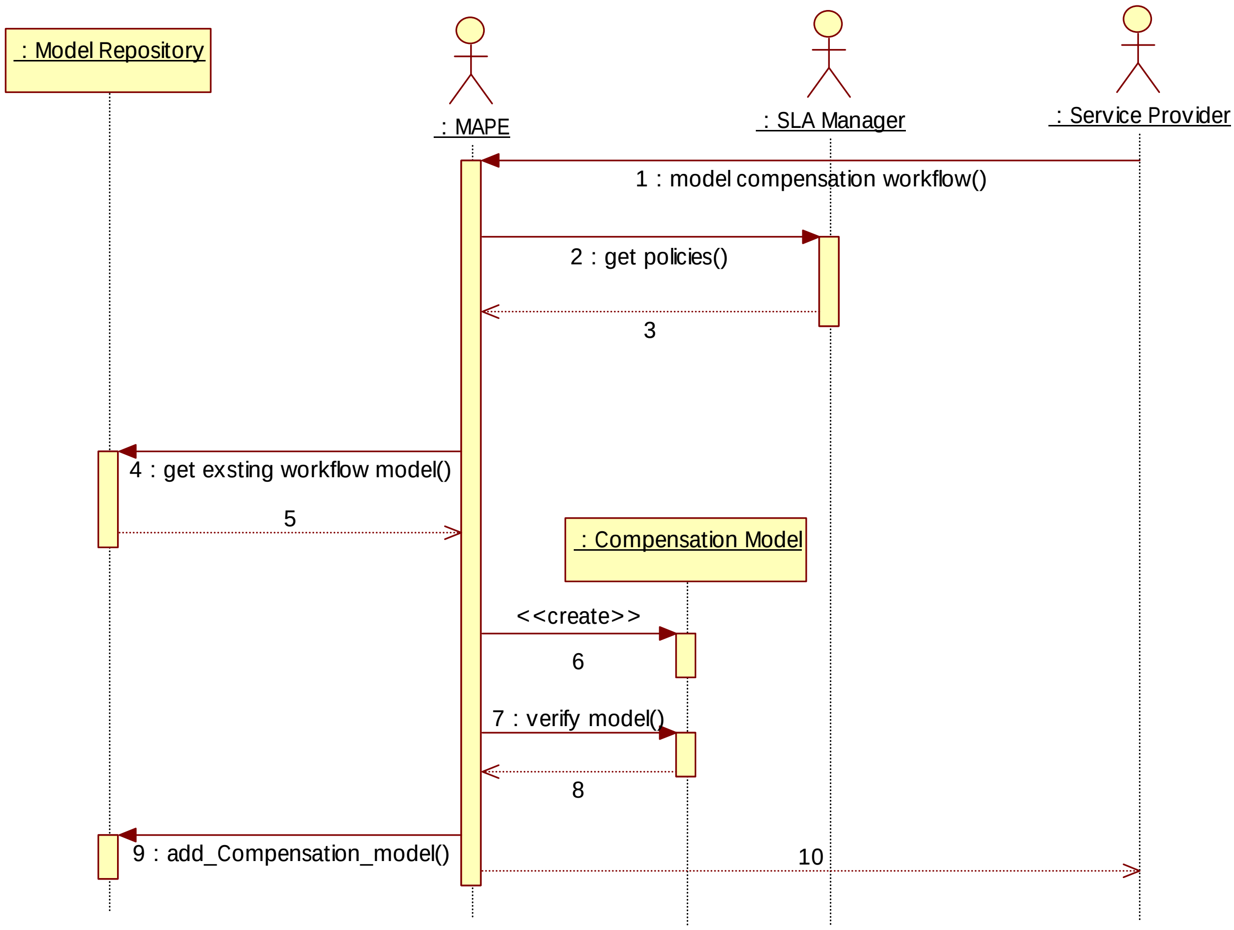

Figure 10: Workflow Conformance Sequence Diagram

Figure 10 describes the scenario where the MAP environment is creating compensation activity workflows that handle exceptional performance.

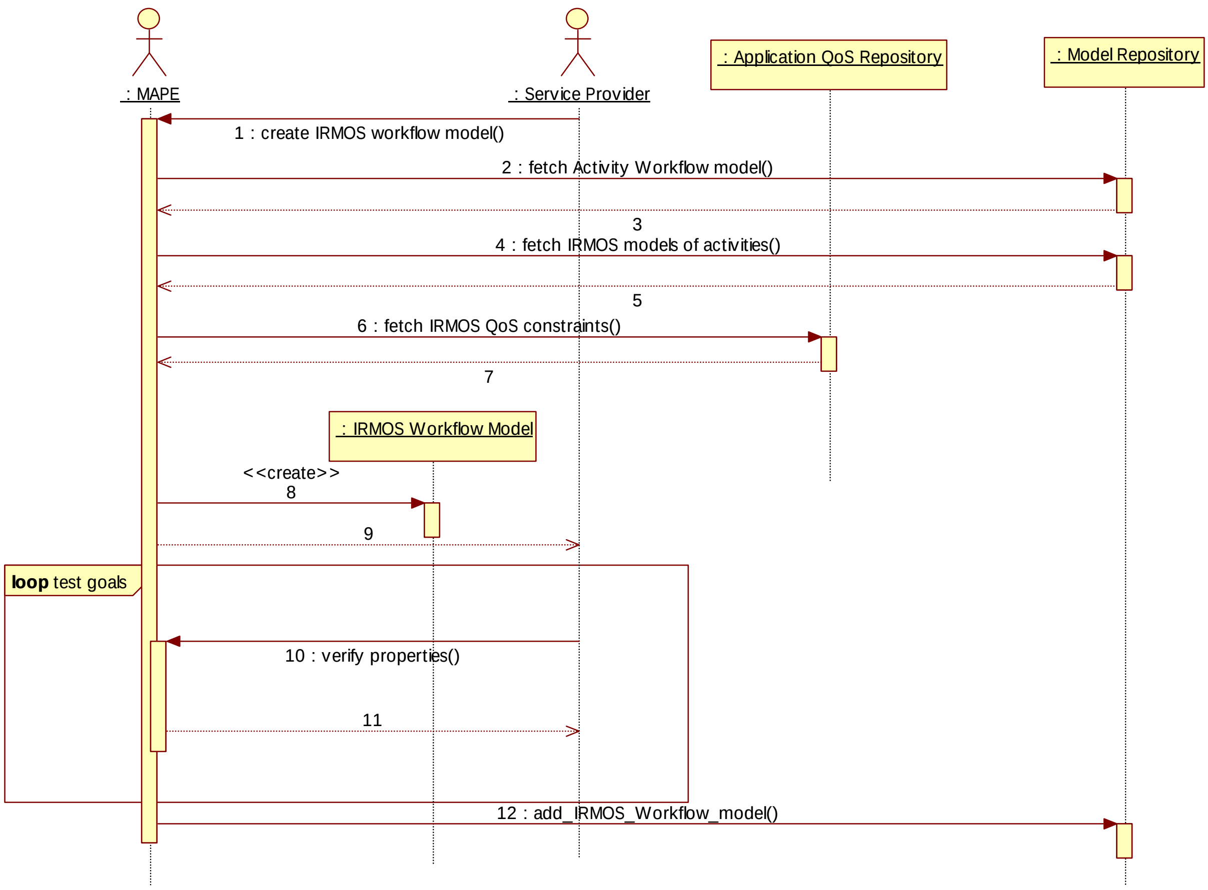

Figure 11: Synthesise IRMOS Workflow Model Sequence Diagram

Figure 11 describes the scenario where the MAP environment is synthesising an ISONI workflow from a service activity workflow model. This requires ISONI level models of each activity and a high level model of the activity workflow.

5.4.2. Workflow and Parameters Mapping

In the following paragraphs and diagrams we present how we expect the application workflow and its high level requirements to be automatically mapped into an IRMOS specific workflow and low level parameters.

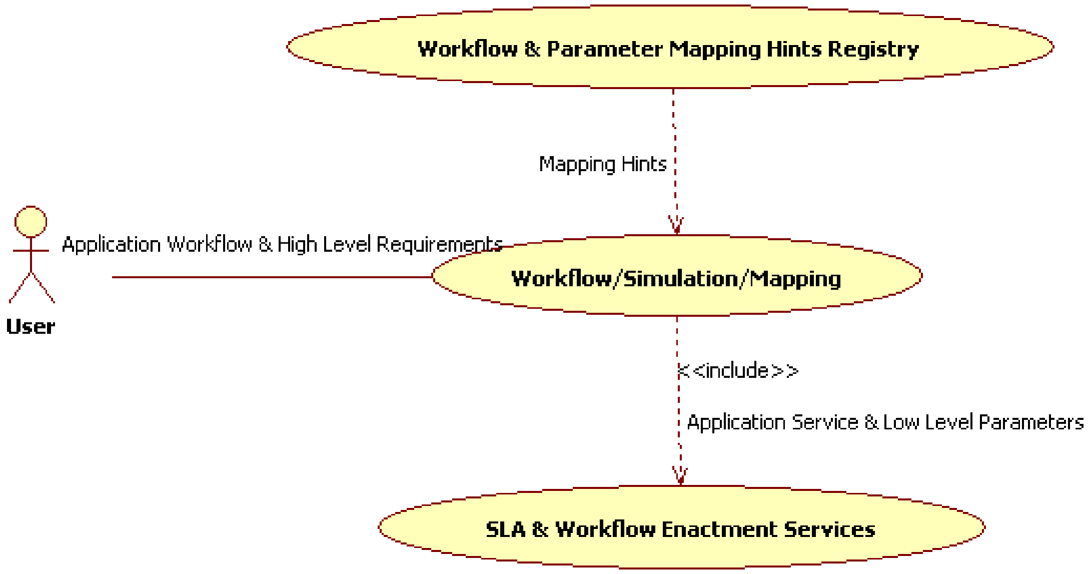

Figure 12: Workflow and Parameters Mapping Process

In this process the following entities are identified:

- Application User
- Workflow and parameter mapping hints registry
- SLA and Workflow Enactment services

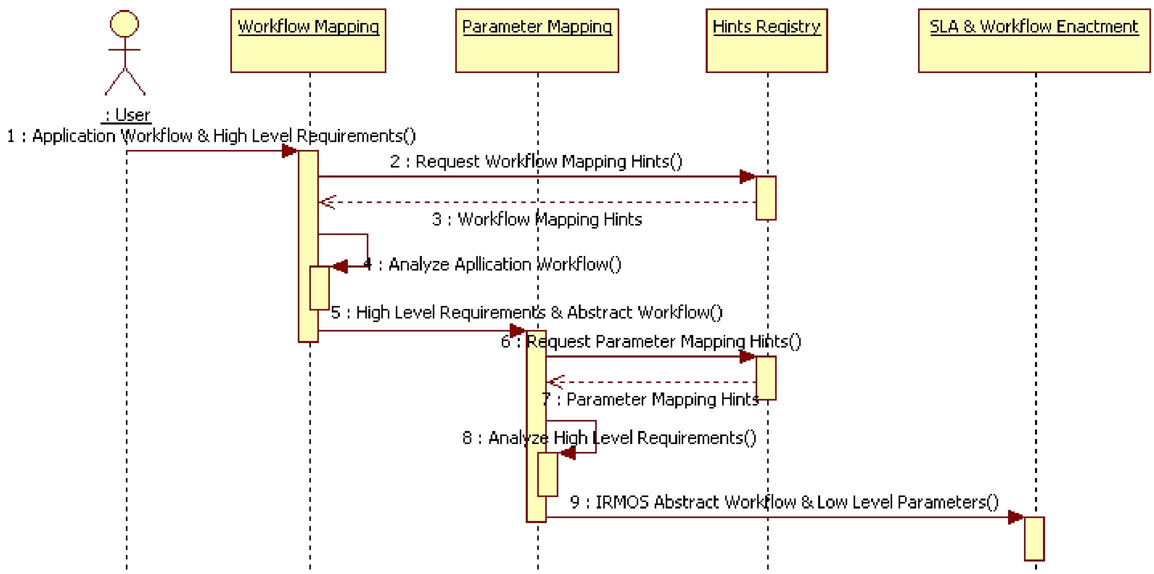

Figure 13: Workflow and Parameters Mapping Process (Sequence)

In this process the user provides as input to the platform, the Application Workflow along with its High Level (Application) Requirements. This is processed by the Workflow Mapping components, which gives as output the Application Workflow (divided into subtasks that can be mapped to specific resources). In order for this to be accomplished, Application Specific Hints from a Registry are used. These Hints have been produced through simulation or analytical modelling during a previous phase.

The High Level Requirements are processed by the Parameter Mapping component to produce the low level requirements per subtask. In this process the mapping components makes use of the newly created IRMOS specific workflow and also the mapping hints from the registry. In sequel, the IRMOS specific workflow and the low level requirements per task are propagated to the SLA management component that will initialize the reservation phase.

5.4.3. Resource Advertisement

The resource advertisement is very important in the service oriented infrastructures and grids.

For the resource advertisement in IRMOS it is expected that the following entities will be involved:

- **ISONI providers:** many different ISONI providers may be available and offer resources at various classes of QoS and a number of special services such as legacy systems that are incorporated inside the ISONI network.
- **Legacy Systems:** these systems may also be located outside ISONI infrastructures, communicating directly with the framework services.
- **Information Service:** This component is contacted by the resource providers and collects all the information regarding the resources.
- **Registry (Index Service):** this component stores information about the resources and of course their status and availability.

The only control in this process is:

- The Information Service that evaluates the information from the resource provides and relays the information to the Registry in order to be stored.

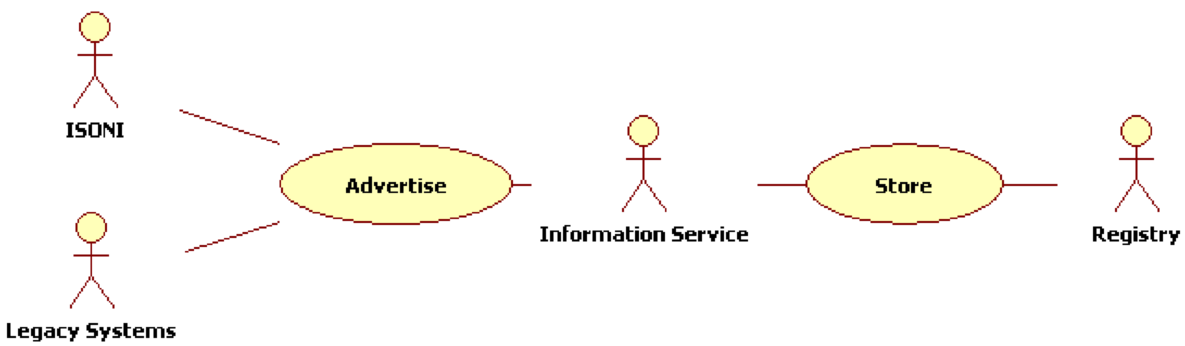

Figure 14: Resource Advertisement Process

5.4.4. SLA Management

This section presents the process on how the SLA management will be performed in IRMOS. From the initial analysis of the process and the discussions with the involved

WPs, there are three sup-processes identified, the publication of SLA templates, the negotiation of Application SLAs and the negotiation of Technical SLAs.
The entities involved in these processes are: Service provider, user of the service, SLA Management ISONI component, SLA Management component, SLA Template, Technical SLAs, Application SLAs, SLA Template Repositories and SLA Repositories.

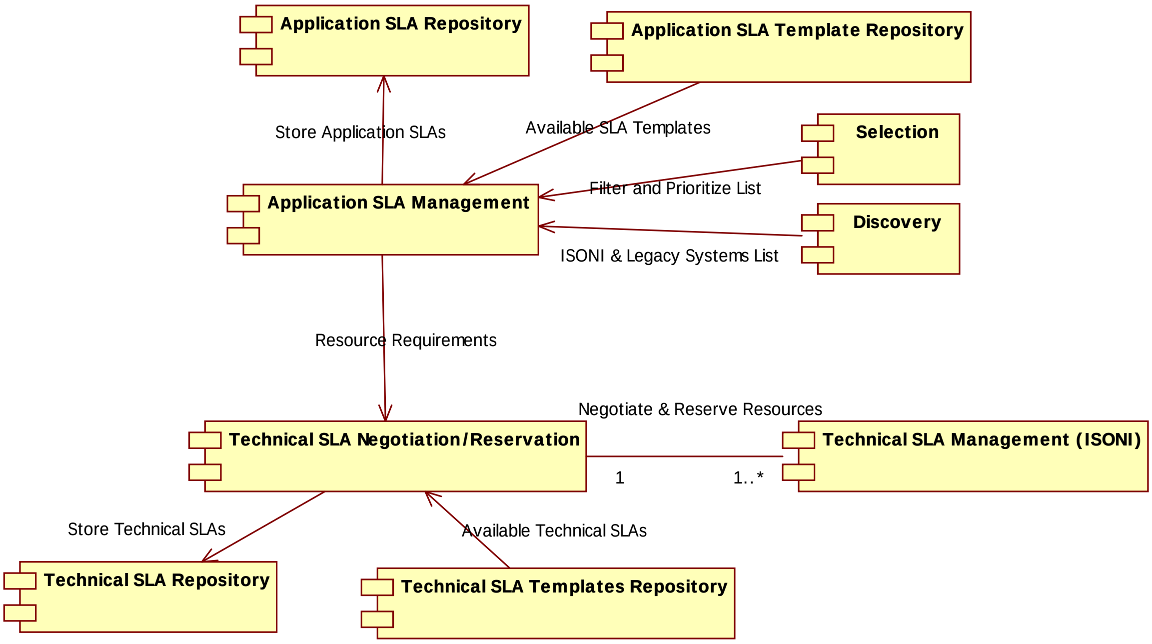

Figure 15: SLA Management Process

5.4.4.1. *Publication of SLA Templates*

The **Publication of SLA Templates** is associated to both types of SLAs: Application and Technical SLAs. Conceptually we will consider two different repositories and middleware to query the repositories.

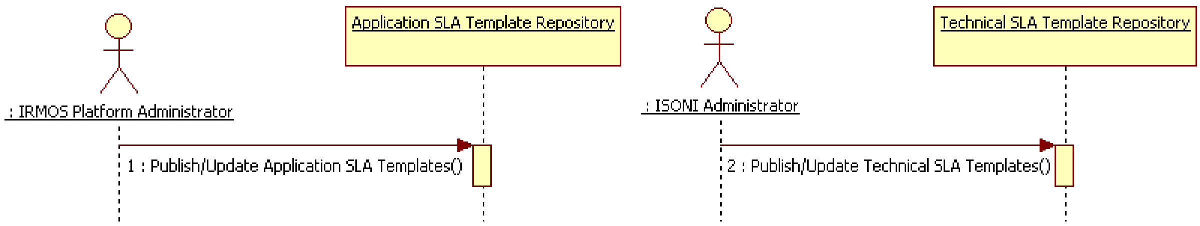

Figure 16: Publication of SLA Templates

5.4.4.2. *Application SLA Negotiation*

This SLA is negotiated between the end user of the service and the service provider. Among conditions of the SLA cost will be included and also some high level parameters not related to technical details (present in the Technical SLA). The steps for the application SLA negotiation process are the following and are also illustrated in Figure 17:

- Request of the user of the service to use a certain service under certain restrictions.

- Analysis of the conditions imposed by the user of the service.
- Search of ISONI domains able to fulfil with user restrictions.
- Selection of the ISONI domains to deploy the services requested.
- Check availability of the selected ISONI domains.
- Reservation of the ISONI domains to execute the service.
- Search of the corresponding SLA Template.
- Signature of Application SLA between the two parties. The input of this phase is the SLA Template and the output is the signed Application SLA Template.
- Storage of the Application SLA into the Application SLA Repository.

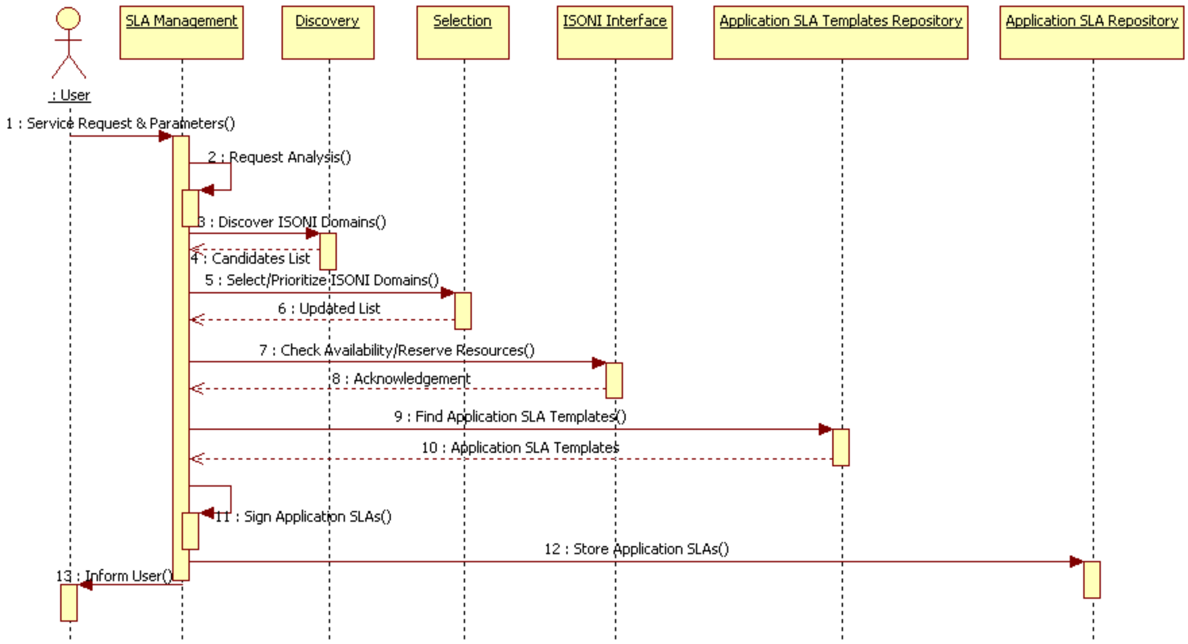

Figure 17: Application SLA Negotiation

5.4.4.3. Technical SLA Negotiation

It is expected that the metrics included in this kind of SLAs will be more related to resources characteristics like CPU, storage capacity, bandwidth, jitter, delay and other time constraints. The parties that will negotiate technical SLAs will be: Technical SLA Negotiator (WP5 side) and Technical SLA Management (ISONI side). The technical SLA negotiation process consists of the following steps that are also presented in Figure 18:

- Confirmation of parameters previously exchanged in the discovery of the appropriate ISONI domain.
- Search the corresponding SLA Technical Template.
- Negotiation of Technical SLA.
- Signature of the Technical SLA.
- Storage of the Technical SLA into the Technical SLA Repository.

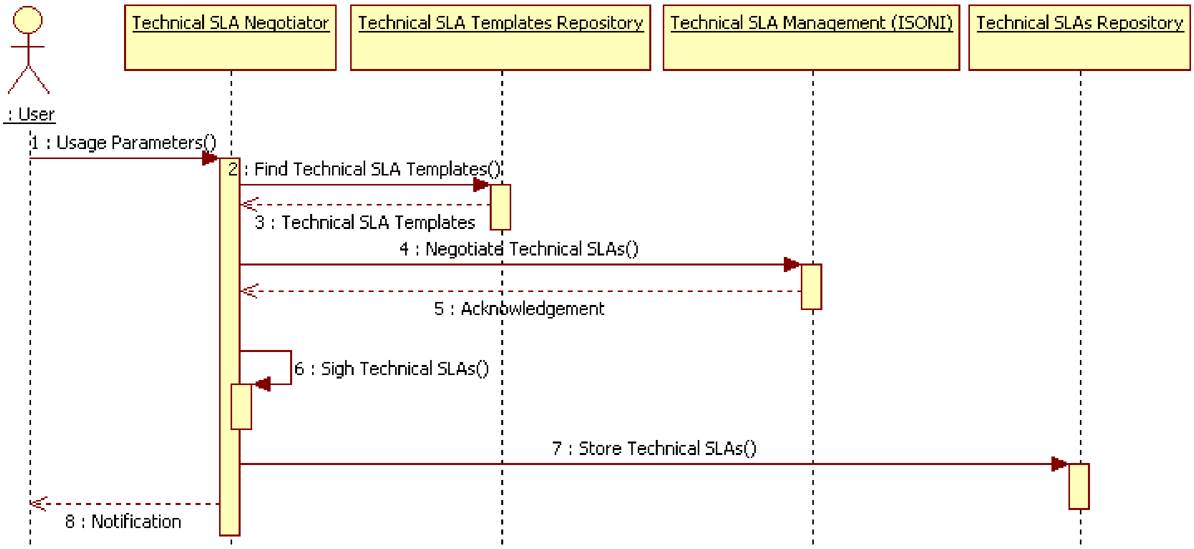

Figure 18: Technical SLA Negotiation

5.4.5. Workflow Execution

The entities involved in the workflow execution process are:

- Application User
- SLA's
- Parameters
- Policies
- VSN description

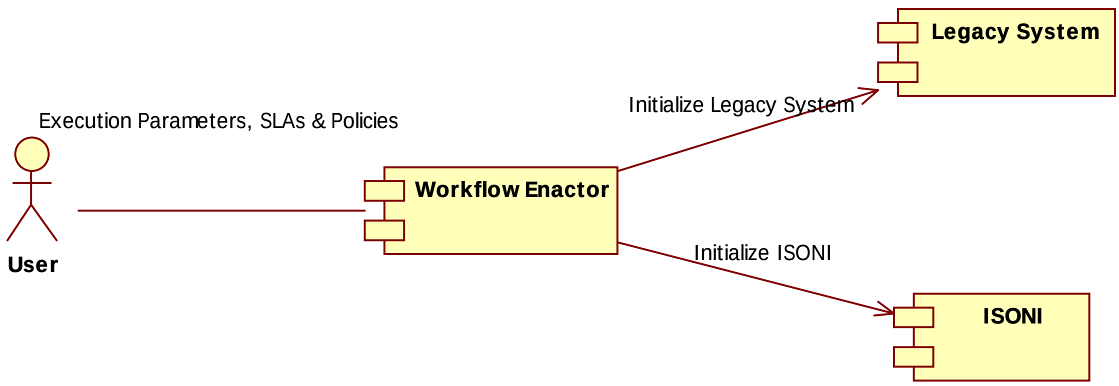

Figure 19: Workflow Execution Process

The following figure presents the sequence diagram for the process:

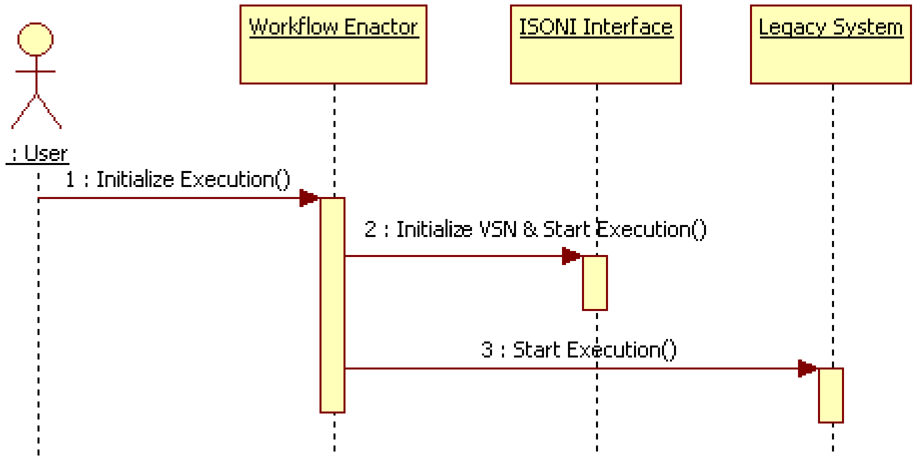

Figure 20: Workflow Execution Process (Sequence)

Here we will present the workflow execution phase. We take for granted that the user has already reserved the needed resources and all SLA's have been signed. In this phase the application user initiates the execution of the workflow by sending a message to the Workflow enactor containing the execution parameters, the SLA's and the policies. The Workflow enactor either initializes the legacy system or initializes the ISONI provider by requesting a VSN via the ISONI interface. In case an error occurs, the workflow enactor automatically contacts the SLA negotiator in order for the renegotiation phase to start.

5.4.6. Monitoring

The monitoring process for the IRMOS platform spans between different domains and involves several components:

- **ISONI Framework**
- **Legacy System**: A legacy system might be a device that communicates directly with the framework services or a processing system inside an ISONI domain.
- **Monitoring**: This entity aggregates metering information data.
- **Evaluator**: The evaluator is a very important component because it communicates with a policy repository and evaluates the monitoring data. Then it decides which component will send notifications to, and what type these notifications will they be of. This component can be also part of the monitoring component.
- **Mapping component**: Transforms high level parameters to low level and vice versa.
- **SLA Management**: Manages SLAs and compares them with the monitoring data.
- **Workflow Enactor**: Coordinates the actions that should be taken.

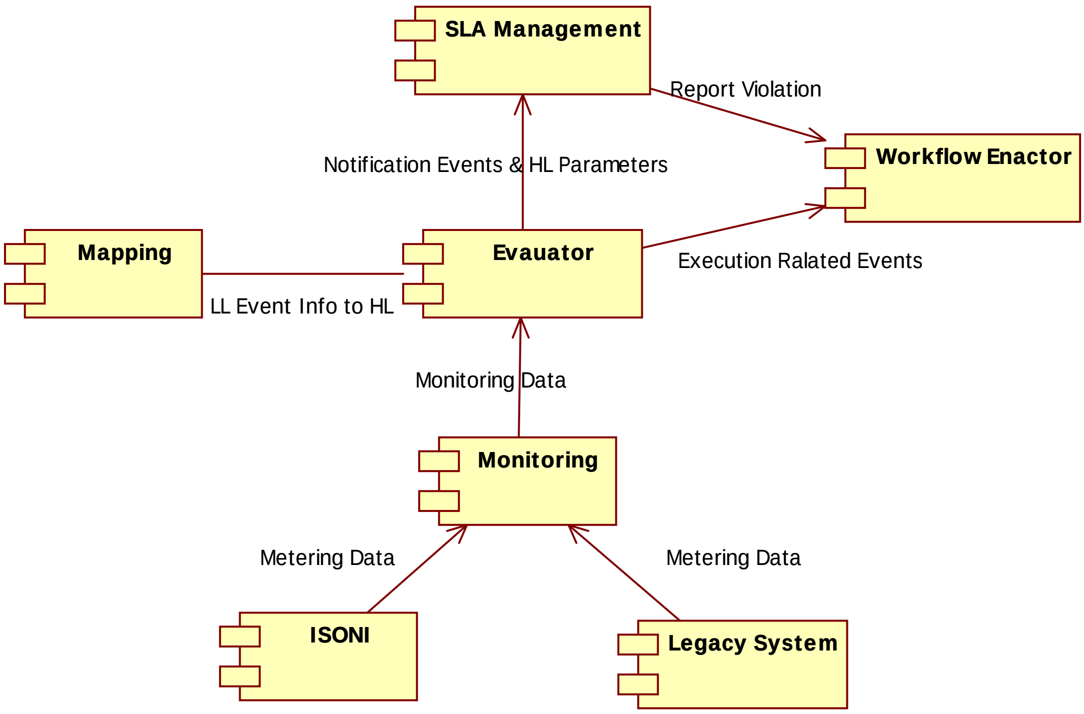

Figure 21: Monitoring Process

The monitoring process in IRMOS is expected to have two variations, as depicted in Figure 21. The difference in these variations is based on the type of the event that is produced from the monitoring component. In the first variation the event requires the mapping of the low level monitoring information to high level so as to be checked from the SLA Management. In the second variation, in which the event has to do with the workflow execution, the event is forwarded directly to the workflow enactor.

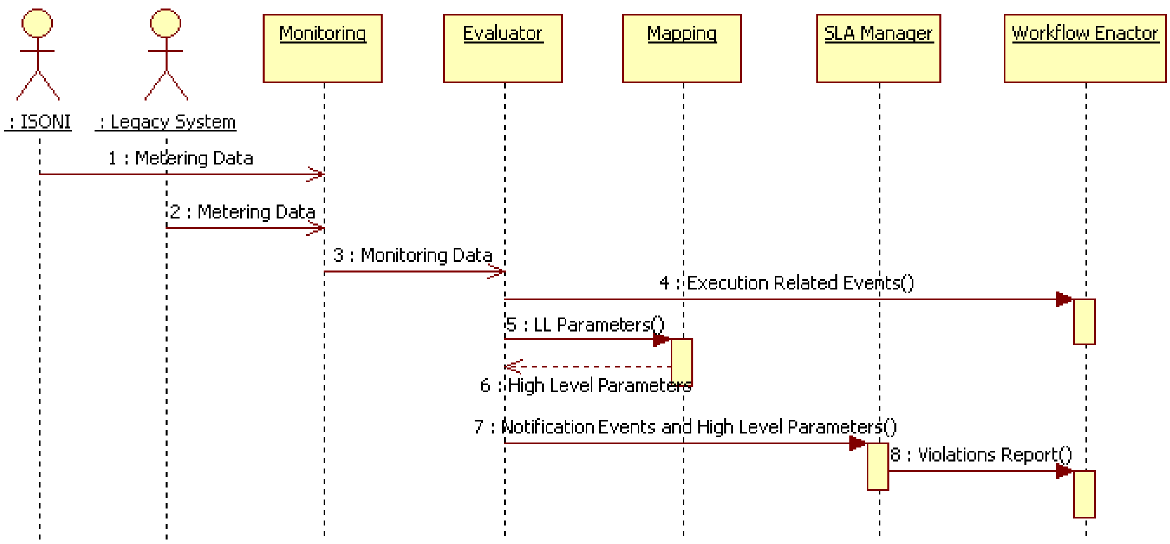

Figure 22: Monitoring Process (Sequence)

The sequence of the process is the following:

- The metering service of either the ISONI component or the Legacy system sends events to the Monitoring component.
- The Monitoring Component aggregates the metering information from the ISONI interfaces and propagates it to the Evaluator.
- At this point two variations are identified based on the type of the event that is produced:
    - If the event is associated with the workflow execution e.g. a job is finished, the information is relayed to the enactor so as to take the actions required.
    - On the other hand, if the event is associated with some kind of failure or SLA violation, the evaluator component sends an event to the Mapping component.
- The Mapping component transforms the data to high level terms which are comparable to the high level parameters of the service SLA. Mapping component returns the transformed data to the evaluator.
- The Evaluator sends the high level parameters of the monitoring data to the SLA Manager.
- The SLA Manager compares the monitoring data with the SLA of the service. If a violation occurs, it communicates with a policy repository to take the appropriate actions. A notification with the selected policy is sent to the Workflow Enactor.
- The Workflow Enactor coordinates the next actions.

6. IRMOS Platform Design

6.1. IRMOS Platform Architecture Overview

The main objective of IRMOS to design a real-time aware Service Oriented Infrastructure for interactive multimedia applications and also identify its functional and non-functional capabilities is carried out by WP3. In the initial version of the architecture, WP3 focuses on producing an overview of the IRMOS overall architecture since it is not possible from the beginning of the project to address all the requirements and describe in detail all the components and their interactions. However, this section clearly outlines the main IRMOS platform subsystems, layers and boundaries, something that is very important at this moment for the whole project and especially for the development WPs so as to have clear starting point for their work and a common understanding for the key interactions.

Based on the description of work and the preliminary user requirements, it was clear that the IRMOS platform consisted of main subsystems: the framework services, the execution environment and the network. The role of framework services subsystem is to support the lifecycle of applications in the IRMOS SOI. The components of this subsystem will be exposed as services through well defined interfaces to interact each other and also with the Intelligent Networking and the Execution Environment services. Almost all the identified IRMOS capabilities (section 5.2) are related with this layer of the platform and are analyzed in detail in the next section. Furthermore, it should be highlighted that the framework services will support the integration of legacy applications.

On the other hand, the Execution Environment and the Intelligent Networking subsystems are architecturally close and are expected to communicate continuously during all the IRMOS processes. Therefore both subsystems become one under the ISONI framework -for the structure of this document we followed the same approach- that is able to manage the computational, hardware, storage and network resources and offer them as VSNs to the framework services layer and then to the application users.

ISONI, analyzed in section 6.2.2, has as main objectives the virtualization of the resources, the deployment of the application services and the resource monitoring without any knowledge for the application itself. The Execution Environment subsystem, considered as an enhanced virtualization platform, includes the storage systems that are implemented so as to address the QoS and especially the real-time requirements of the application services. The network resources, provided in VPN like approach, are classified and advertised to the framework services in QoS classes while the framework services will have to setup the VSN for the execution of the application workflows.

The following figure depicts an overview of the IRMOS platform:

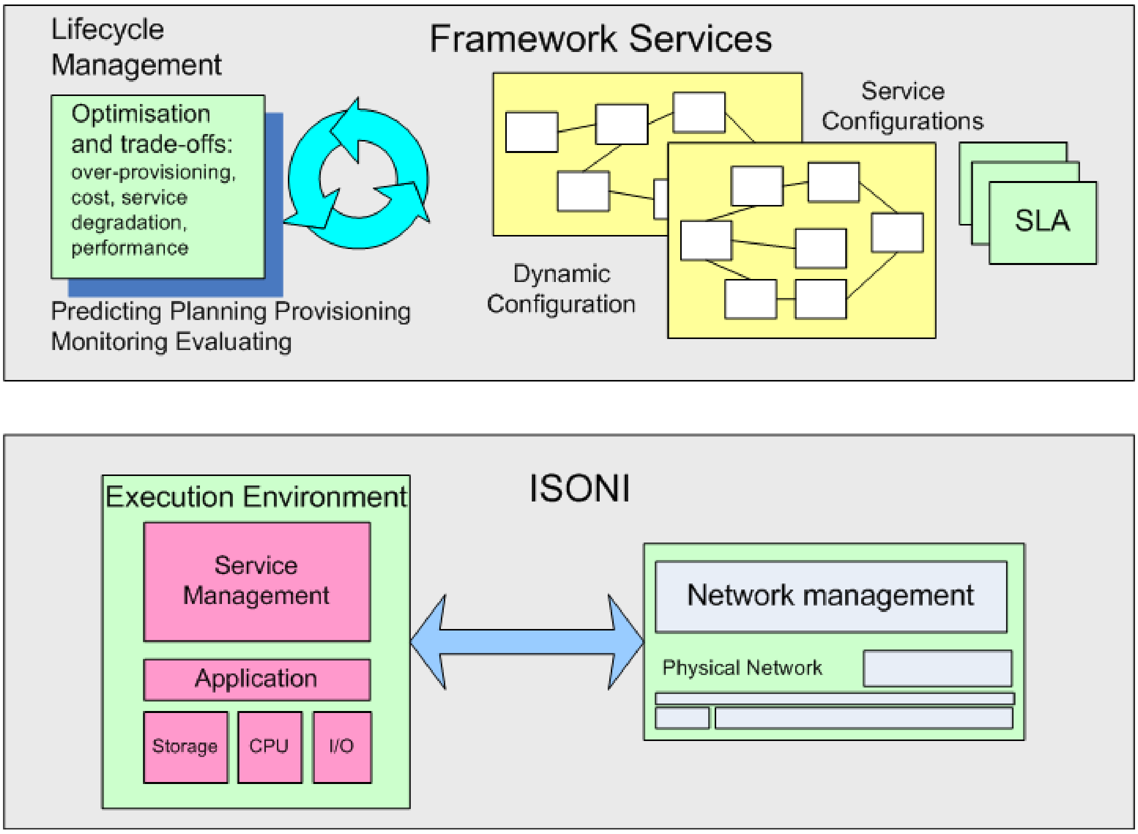

Figure 23: IRMOS Platform Overview

6.2. IRMOS Subsystems

6.2.1. Framework Services

6.2.1.1. Overview

SOA is a design discipline conceived to achieve the goals of increased interoperability, other benefits such as portability, re-usability and simplified design. The Framework Services realize these benefits by ensuring that their design is consistent with the service-oriented architectural vision and adheres to principles of service-orientation.

Thus, instead of designing a small number of complicated services, the Framework Services are a set of services, with each of them addressing a specific task. In more detail, the Framework Services are considered as a composition of the services listed below:

- **Index Service**, which collects and publishes, aggregated information about ISONI resources. It collects information from potentially many ISONIs and publishes it to one place.
- **Information Service**, which acts as a gateway service in front of the Index service, supporting operations such as creation, update, deletion of registered advertisements.
- **Advertisement Service**, which is used by the ISONI providers to advertise their ISONIs in the index service used by the Framework services in the discovery process. The ISONI providers run an Advertise client application on their local

machines, providing all information necessary to form the advertisement of the ISONI resource classes.

- **Simulation and Planning Services**, which are the service-oriented versions of the application modelling and services planning tools of Tasks T5.1 (Real time application modelling) and T5.2 (Planning service networks). These services will be used as services by other Framework services both at negotiation and execution time, to provide information needed to dynamically adjust service networks in response to execution-time monitoring and reporting as well as assist the application user in creating an abstract workflow and specifying the QoS related parameters.
- **SLA Negotiation Service**, which is responsible for interpreting requests for the negotiation of the terms of the application's user SLA contracts and orchestrating the interactions among the rest services that are involved in the negotiation and reservation process.
- **SLA Manager Service**, which creates, queries, publishes, and updates SLA and SLA templates and to this end, it interacts with two repositories, one for concrete SLAs (SLA repository) and SLA templates (SLA template repositories).
- **Reservation Service**, which is in charge of performing advance reservation on the service and network resources needed for the execution and monitoring of the application service.
- **Workflow Enactor Service**, which is in charge of the coordination of the execution of the reserved application resources as well as being responsible to take corrective actions when needed.
- **Discovery Service**, which is responsible for finding candidate locations/services, registered in the underlying ISONIs that meet the QoS parameters defined in the application user's SLA. It is designed so as to follow rules based on low level performance parameters related to the execution of the service, such as CPU, memory, network bandwidth and disk capacity as well as the availability of the service and the price that the client is willing to pay.
- **Mapping Service**, which is responsible for mapping application level requirements to resource level specifications and vice versa. To this end, the Mapping service interacts with the MAP services to identify the risks (probability and impact) inherent if there are errors or uncertainties when mapping the application parameters onto resource needs and vice versa.
- **Selection Service**, which is responsible for sorting the list of candidate resources/services (produced by the Discovery service) with the assistance of the Simulation and Planning services.
- **Monitoring Service**, mechanism for monitoring the consumption of resources during the execution of the services. This monitoring mechanism will interact with Metering services that reside in the ISONI domain and will communicate this information to other Framework services.

The following components appear in the diagrams but are not part of the Framework services:

- **Metering Service**, which is responsible for the measurement of the resources that are being consumed. It measures low level performance parameters and communicates them to the Framework services through the ISONI interface.

- **ISONI Interface**, which acts as a gateway service, offering a single-point of access to the ISONI services (WP6/WP7) whilst “hiding” its details and complexity from WP5 services. It is therefore responsible for interpreting requests from the Framework services, delegating them to the appropriate sub-services, executing the requested task and returning the results to the Framework services.

6.2.1.2. Functionality

The functionality of the Framework Services is divided into two phases: 1) Negotiation and Discovery phase, and 2) Execution and Monitoring phase.

Negotiation and Discovery phase

An overview of the framework services and their role in the negotiation and discovery phase is presented in the following figure:

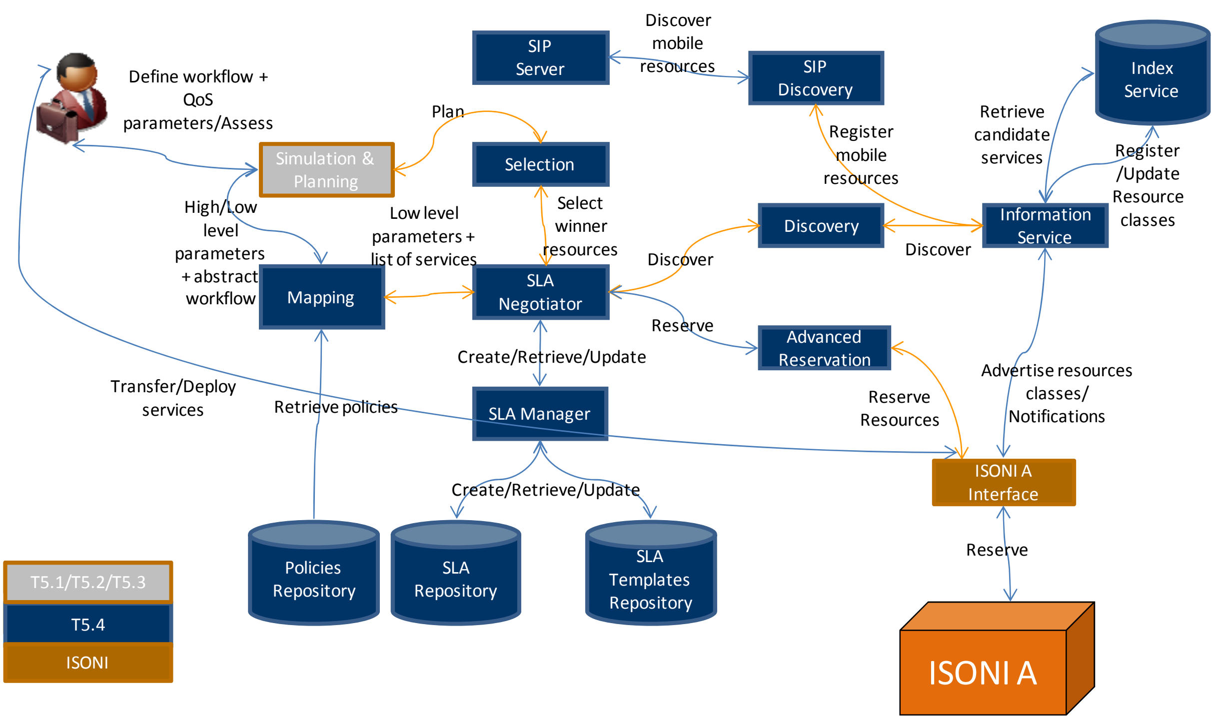

Figure 24: Component Model of Negotiation and Reservation Phase.

During this phase the Framework Services are in charge of negotiating and performing advance service and network reservation on the resources that were discovered during this phase and are needed for the execution of the application services. It should be noted that the Framework services treat the ISONIs as black boxes and are aware only of the information included in their corresponding advertisements that are registered with the WP5 Information service. In addition the Framework services communicate with the underlying ISONIs through a WS-based interface (one interface per ISONI).

Figure 25, shows the sequence diagram for this phase:

1. The application user invokes the Simulation and Planning services providing all information needed including the desired range of values for the high level

parameters (on workflow and/or service level) that the user wants. The end user will also be able to define low level information if this is desired.

2. The Simulation and Planning services assist the end user to create an abstract workflow description and a list of QoS parameters and contacts the Mapping service.
3. The Mapping service checks the format of the input parameters. If their format is valid (i.e. the Mapping service is able to recognize the input parameters), the high level parameters that are included in the list are mapped to low level ones. Otherwise, it returns an error message to the Simulation and Planning services. Furthermore, the Mapping service is able to parse the abstract workflow description and extract the list of required services.
4. The Mapping service parameters invokes the SLA Negotiation service passing the obtained list of services and the ranges of the values for their corresponding low level QoS parameters.
5. At first step, the SLA Negotiation service iterates through the list of required services (5a) and for each of them queries the Discovery service trying to find resources that meet the end user's requirements (= low level QoS parameters, 5b). In doing so, the Discovery service works with the Information service that acts as a gateway service in front of the WP5 Index service (= the service where all advertisements are registered, 5c) The Index service aggregates information about mobile resources through the SIP Discovery service. The following steps up to step 9 are repeated for each service in the abstract workflow description.
6. At the end of a successful discovery process, a list of candidate resources is returned to the SLA Negotiation service.
7. The SLA Negotiation service contacts the Selection service providing the list of candidate resources.
8. The Selection service selects a winner resource with the assistance of the Simulation and planning services. The Selection service will also be able to sort the candidate service according to the user's needs.
9. This sorted list of the candidate resources is passed at next step to the Advance Reservation service. The Advance Reservation service contacts the corresponding ISONI interface to reserve the winning resource. If the ISONI is not able to reserve the resources needed, the Advance Reservation service iterates through the sorted candidate resources until it succeeds. Otherwise it returns a failure message to the SLA Negotiation service that is propagated back to the end user.
10. At the end of a successful reservation, the SLA Negotiation service initiates the creation of the SLA through the SLA Manager service. In doing so, the SLA Manager uses pre-defined SLA templates. Once the SLA is created, the SLA Manager stores it into a repository where SLAs are kept.
11. Once the SLA has been established successfully, the SLA Negotiation Service Agreement resource (AR) is created. All information related to the negotiation request and the discovered resources is stored into this WS-resource.
12. The SLA Negotiation service returns the End Point Reference (EPR) to the AR to the Simulation and Planning service.
13. Finally, the EPR to the AR is returned through the Simulation and Planning services to the end user.

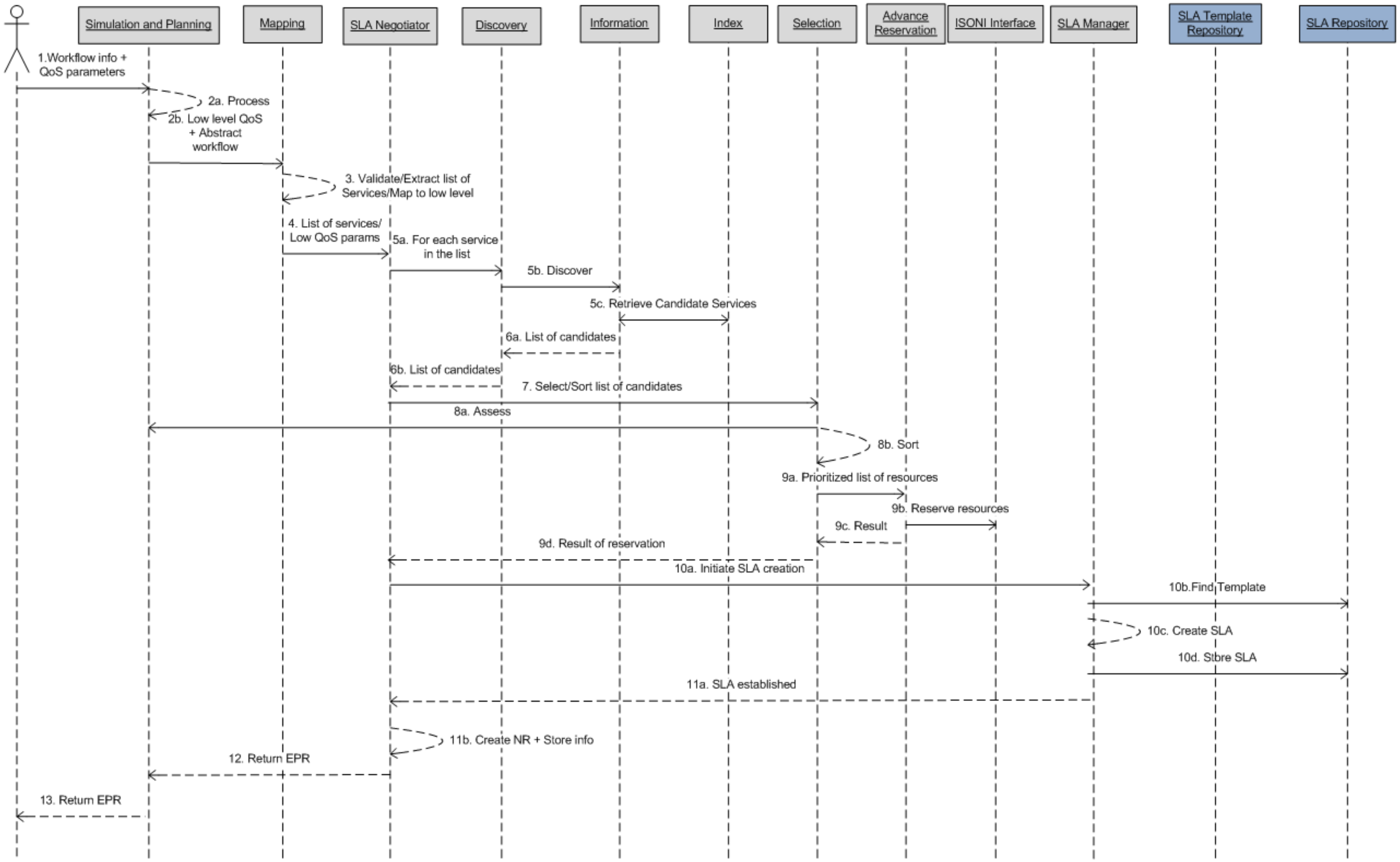

Figure 25: Sequence diagram for the Negotiation and Discovery phase.

It should be stressed that the ISONIs can advertise their resources with QoS capabilities or update existing advertisements at any time, using the Information Service. In more detail, the ISONI must specify the URI to its interface. These advertisements should also give more insight into their capabilities using resource classes.

Figure 26, shows the sequence diagram demonstrating how an ISONI can be advertised in the WP5 Information service:

1. The SP runs an Advertise client application, specifying a URI and values for resource classes.
2. The Advertise client filters given values to check if they have the appropriate format and then invokes the Advertisement service, which runs in the same machine as the given ISONI interface.
3. This operation will trigger the creation of a WS-Resource called Advertisement.
4. Upon creation, the Advertisement resource registers with the Index service running the same machine as the ISONI interface, which gathers information on the advertisements within the specific ISONI.
5. After some time, the local registrations will propagate to the WP5 Information service which will also cache the data stored in the Advertisement Index services across all ISONIs.

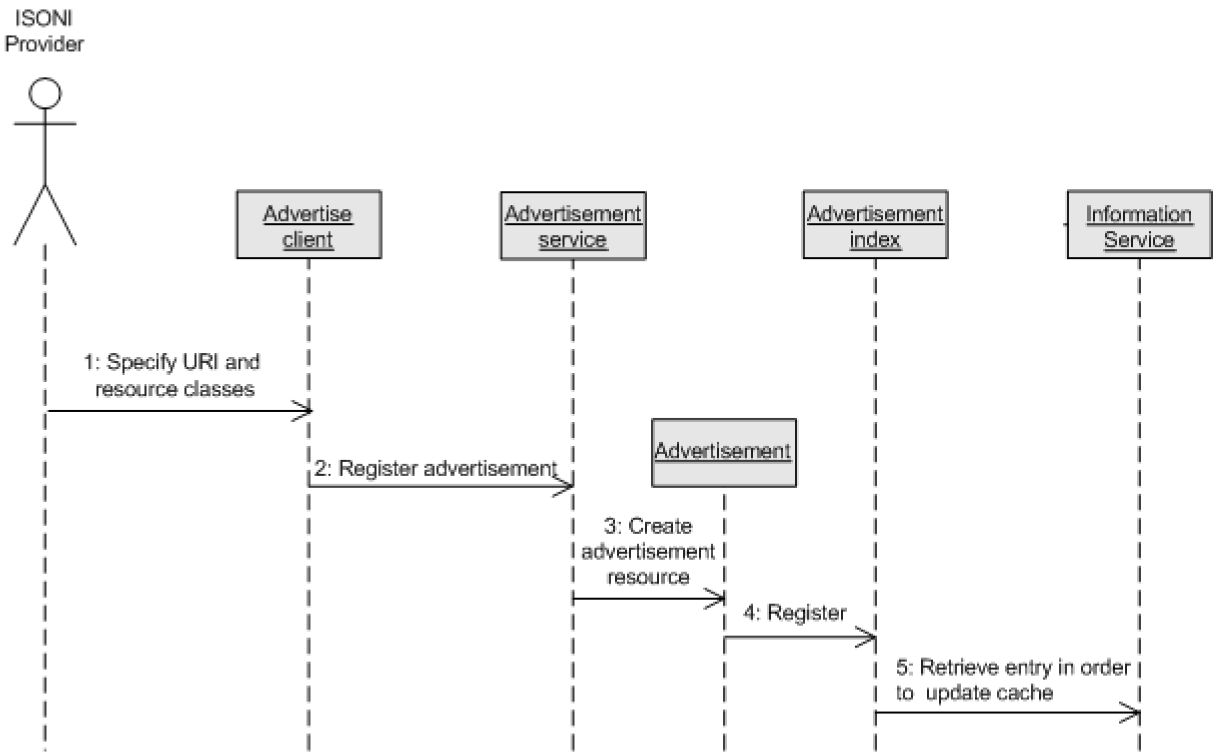

Figure 26: Sequence diagram for the registering ISONI advertisements

Execution and Monitoring phase

In this phase, the Framework Services are responsible for the initialization and management, up to completion, of the workflow execution. A high-level view of the interactions that take place during this phase is illustrated in the Figure 27.

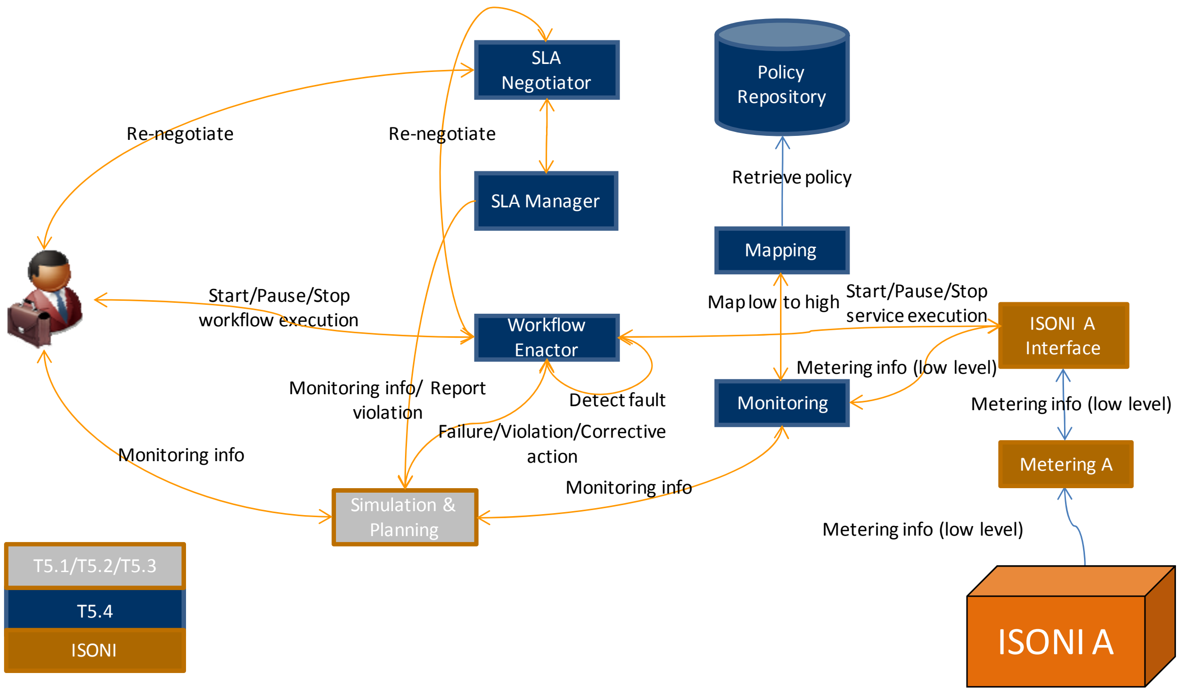

Figure 27: Component model of Execution and Monitoring Phase

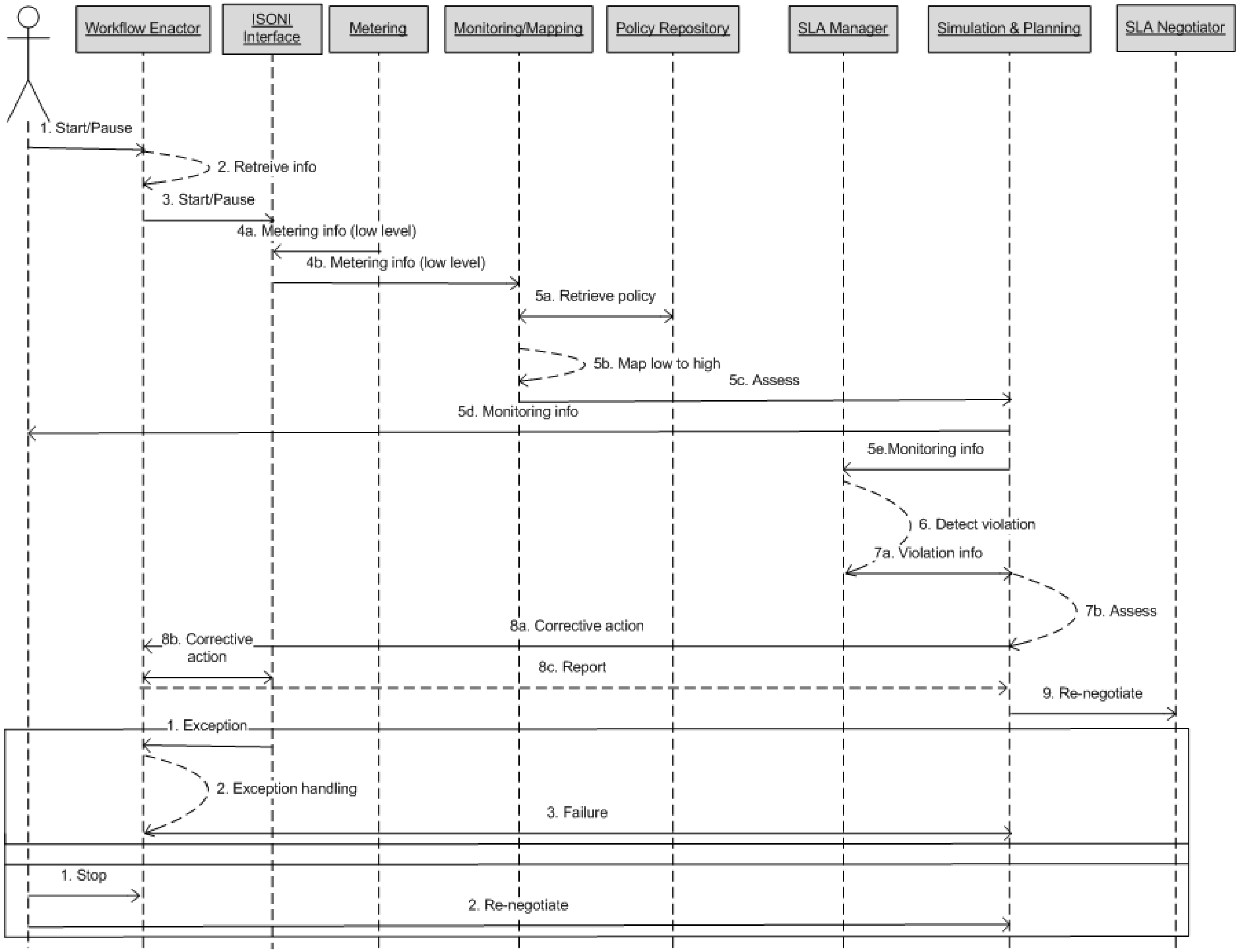

Figure 28: Sequence diagram for the Execution phase.

Figure 28 shows the sequence diagram for this phase:

1. The sequence begins with the application user invoking the Workflow Enactor service, providing the EPR to the AR that was obtained at the end of the Negotiation and Reservation phase.
2. The Workflow Enactor retrieves the information stored inside the NR during the Negotiation phase needed for the execution while updating certain Reservation properties for auditing purposes. By using the end time defined in the SLA, the Workflow Enactor service is able to verify the validity of the end user's SLA.
3. After setting up the resources that perform the monitoring and SLA management, the Workflow Enactor service initiates the execution of the workflow through the ISONI interface.
4. During the execution, the ISONI Metering service sends notification messages to the Monitoring service through the corresponding ISONI interface.
5. The Monitoring service translates the low level parameters into high level ones through the Mapping service, which uses a repository where relevant policies are stored. It also contacts the Simulation and Planning services in order to make a decision about high level utilization parameters based on low level values as well as an assessment. The Simulation and Planning service continuously communicates monitoring info back to the application user and forwards it to the SLA Manager service.
6. When a violation is detected based on the end user's SLA, the SLA Manager propagates the notification to the Simulation and Planning services.
7. The Simulation and Planning services inform the Workflow Enactor of the corrective action that should be undertaken. The latter contacts the ISONI interface to apply this action and sends a report to the Simulation and Planning services. Based on this report the Simulation and Planning services may decide to launch a re-negotiation process that will result in the setup of a new VSN with as less interruption to the application user as possible.

At any time during the workflow execution, the application user is able to pause and restart the workflow execution and also stop and re-negotiate its SLA by contacting the SLA Negotiator. The Workflow Enactor will also be able to trigger a re-negotiation process through the Simulation and Planning services in case of severe exceptions that should remain as transparent to the application user as possible.

6.2.2. Intelligent Service Oriented Network Infrastructure (ISONI)

6.2.2.1. Introduction and relation to WP6 and WP7

The ISONI (Intelligent Service Oriented Network Infrastructure) is an infrastructure, consisting of a network of resources (hardware like CPU, storage,…, and software) managed and controlled by an ISONI middleware, that allows resource sharing among multiple services. The general idea is to provide a SOI (Service Oriented Infrastructure) for SOA components and services. A service is usually composed out of several smaller and simpler services, in the following called Service Components (SC). ISONI is agnostic to services, thus the decomposition of services into SC is not its responsibility, and this is done by the Framework Services layer, in IRMOS developed by WP5. ISONI will provide these SCs the best resources (Execution Environments and network links).

The work related to the Execution environment and its related resources like CPU and Storage will be carried out in WP6, see chapter 6.2.2.3. All issues regarding the intelligent networking layer will be handled by WP7, see chapter 6.2.2.4 for more details. Figure 29 gives some first view on the rough structure of the ISONI.

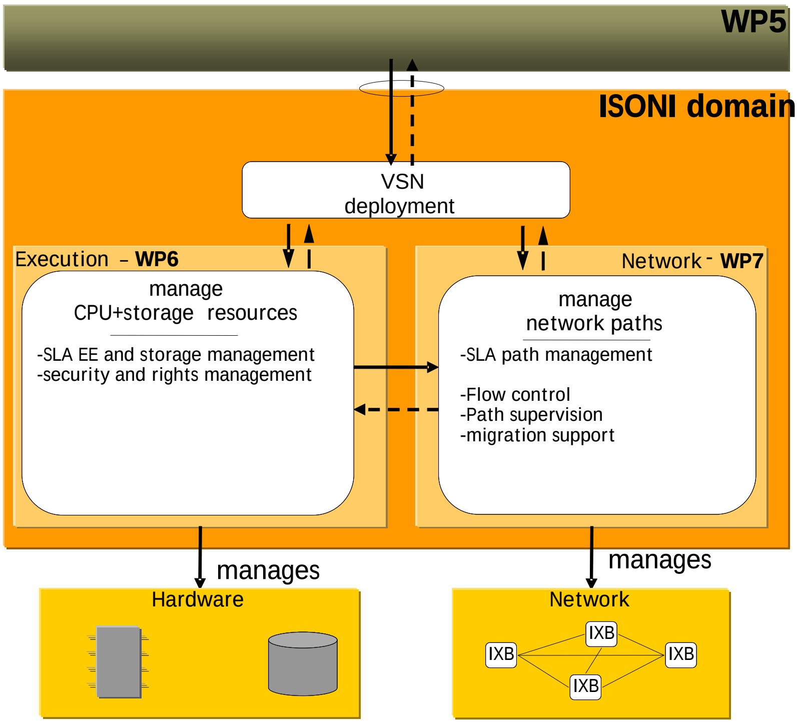

Figure 29: ISONI high level structure

6.2.2.2. General description

The basic purpose of the ISONI is to considerably reduce the complexity for service providers/developers to roll out new network based services as it takes care of the automatic deployment of the services on best fitting resources distributed in a network. The solution strives to reduce global costs introducing a resource provider in the value chain that optimizes costs by means of virtualization techniques, whereby tailored resources can be provided for the deployment of services. Additionally, the ISONI will provide means to isolate different deployed services from each other in order to prevent unwanted crosstalk between them. Future expansion stages shall allow necessary resource re-arrangements and (re-)allocations for running services that adapt better to changing resource availability. To live up to that purpose the ISONI has to carry out several tasks.

The first major task of the ISONI is to completely separate the management of all kind of hardware resources distributed in a network from that of deployed services and their associated service components. By that, the actual status and distribution of resources are hidden from the service developer's view. The infrastructure provides him with fully virtualized resources, including the network resource. That enables a service developer to deal with a complicated network of resources in a simplified way at a level of high abstraction. This full virtualization of a network of distributed resources is the essential prerequisite to get the freedom of resource and service management we need to serve the purpose of the ISONI as described above.

The second major task of the ISONI middleware is to deploy and instantiate the service developers' service on the ISONI. The ISONI will be able to accomplish this task automatically and autonomously, which is the main goal of the ISONI development. For that the ISONI needs an abstract description of all the requirements of the service on the execution environment including the description of the interconnections and their individual QoS demands. The level of abstraction of this description should be as high as possible to ease its creation, while still allowing for automatic matchmaking. In particular, the creation of the description shall not require special knowledge about the network infrastructure. This description has to be delivered by the service developer in form of a Virtual Service Network (VSN) description, which will be presented in more detail in chapter 6.2.2.5.

The third task of the ISONI will be the monitoring of running services concerning resource usage. Those monitoring data will be available to the service developer via, e.g., web service interfaces. It is part of the support functions the ISONI will offer in the future during service lifetime.

6.2.2.3. Execution Environment & Storage Systems

Execution Environment description

Execution Environment can be seen as a virtual machine with enhanced capabilities (see section 5.2.7) where IRMOS applications and services will be executed. Storage Systems are a part of this execution environment. The Execution Environment is part of ISONI, a generic platform offering execution and networking resources to the IRMOS applications.

Execution environment is not application aware, it just provides resources to binaries, with the advanced features described in section 5.2.8. What these binaries are, is absolutely transparent to the execution environment.

Execution environment will use KVM [18] (XEN [19] is also being considered) for virtualising resources.

Storage Systems description

Storage services will run within an execution environment. The processing resources required will come from virtualisation and / or abstraction of the physical processing that are available at the node selected for service execution. In the case of storage, there is no conceptual need for additional virtualisation over that which is currently available. Current devices already offer multiple simultaneous users to interact with a device with the application being responsible for locking etc.

The major task for the storage is to move from a model of access where commands arrive (through whatever protocol) and are serviced in a manner determined by arrival time, previous activity in the system and current outstanding activity in the system to a model where access of one service is made pseudo-independent from access by another service. Full independence will not be possible as the same physical storage elements will be used by all services and it is not possible to change the firmware running on these elements.

Instead of full isolation, a logical isolation based on the service quality requested will be attempted. This isolation will be performed in the processing that will be associated (and probably co-located) with the storage; low level architecture definition is still under way and details will be provided in the next versions of the architecture reports.

6.2.2.4. Intelligent Networking

Network resources

The ISONI makes use of VPN like transport resources between ISONI nodes with predefined QoS properties (provided by Network Provider(s)), augments them with own shaping and scheduling mechanisms, and assigns bandwidth of individual links to VSNs according to the QoS requirements provided in the VSN description. The ISONI manages an own set of QoS classes that are appropriately mapped to the QoS classes and types of the individual links. The ISONI is able to act in this way because it owns dedicated transport resources in differently managed IP networks.

Network

The networking layer is the ISONI part responsible for granting QoS on this ISONI level. It has to trigger path reservation (taking QoS demands into account) between VMUs. The critical part of path discovery and reservation is performed at ISONI domain level. The network at the domain level is WAN-like. Thus, efficient methods of path building on domain level are essential for providing QoS by ISONI whereas local networks at node level should pose no problems here.

QoS

ISONI provides link resources that are classified in several ISONI QoS classes. The definition and number of QoS classes may change in the future, but that does not change the principle in which those QoS classes are implemented by the ISONI. The Virtual Link within the VSN description the required properties of the links are described by parameter sets comprising bandwidth, delay, jitter, max loss rate. The ISONI will map those parameters into an ISONI internal QoS class that provides the required links with at least the QoS properties as requested by the VSN description. If the parameter values cannot be mapped on to ISONI QoS classes it could infer that the affected SCs have to be collocated on one physical host or even in one VMU.

The four example ISONI QoS classes for framework services and applications are the following:

- Best effort data (be-data) for data traffic without any QoS guarantee.
- Best effort data with minimum guaranteed bandwidth (be-prio) for data traffic, which needs minimum guaranteed connectivity, e.g. SIP signalling traffic among IMS applications

- Real-time voice (rt-voice) and
- Real-time video (rt-video) dealing with loss-tolerant media traffic. Rt-voice is treated by ISONI with higher priority than rt-video due to its lower bandwidth requirement and its smoother traffic characteristic.

The real-time QoS classes may be split up into streaming and interactive classes due to the different strength of QoS requirements. Loss-intolerant real-time traffic is not regarded for the time being, but the ISONI QoS concept can be extended by adding additional ISONI QoS classes.

6.2.2.5. Interactions and interfaces with other subsystems

ISONI is not application aware, it just provides resources to binaries. On its side, the Framework Services layer (developed by WP5) deals with applications and their logic; it is, thus, application aware. A gap between both then exists and proper interfaces need to be designed to bridge it.

The VSN Concept (Virtual Service Network)

In order to deploy a service the service developer has to attach an abstract description of the service's (run-time) requirements and its classification when transferring his service to the ISONI. To generate the description he

- designs his service by selecting and parameterizing appropriate Service Components that are either already present within the ISONI or have to be transferred to it. Each of those SCs is annotated by its programmer with a document we call *Metadata* (see also Figure 30 for illustration), a description of all the requirements an execution environment has to comply with in order to enable the respective components' proper execution. That includes the description of the interfaces or sockets and their respective properties, provided by the SC.
- defines the interconnections of those SCs in form of a Virtual Link Description (see also Figure 30 for illustration), resembling the actual communication structure of the SCs in the VSN.

All the descriptions and parameters are merged in a so-called Virtual Service Network (VSN) description. The VSN can be seen as a graph whose vertexes are the SCs and whose edges are the Virtual Links, see Figure 30. The VSN description is transferred to the ISONI with the request to instantiate the service. Then, the ISONI has to automatically and autonomously map the highly abstracted resource request in form of the VSN description onto the network of real resources, to deploy the components in tailored execution environments on suitable resources, and to interlink them while observing QoS requirements. This instantiated VSN builds an independent layer 3 overlay network, i.e., there is no limitation on the L3 protocol stack used by the SCs.

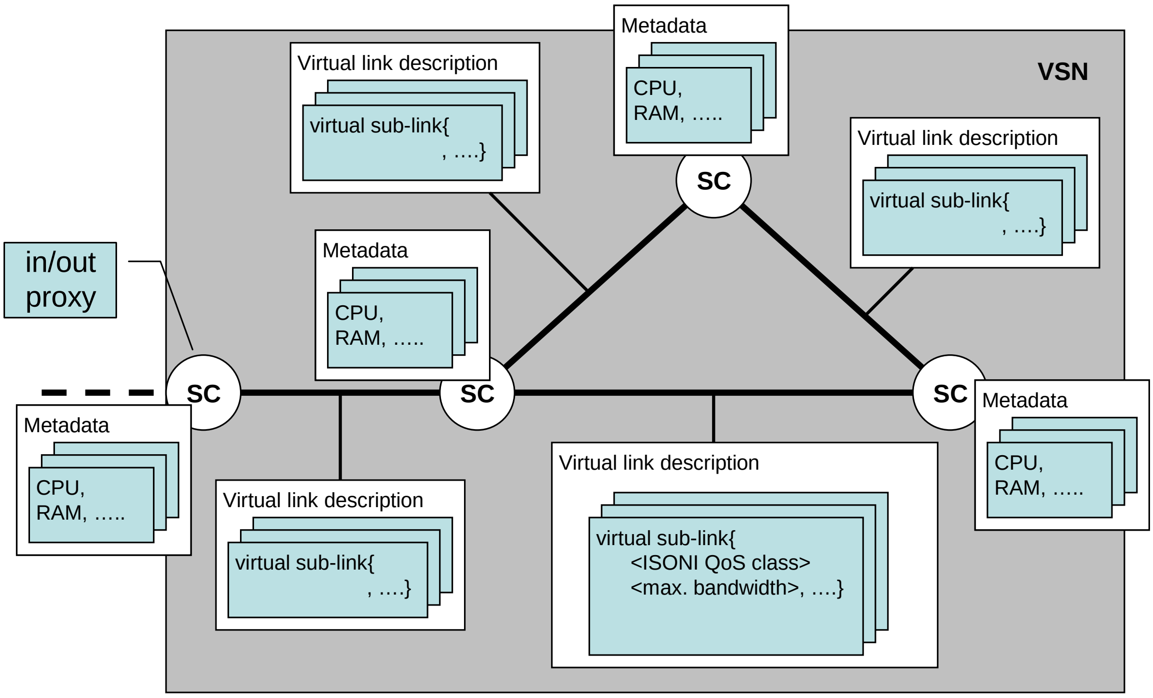

Figure 30: VSN description

Usage of the VSN

The ISONI is exposed to the Framework Services layer as a Virtual Service Network. The VSN will be described by a set of vertexes (Execution Environments) connected by a set of arcs (the network virtual links). The WP5 services will decompose its Services and Applications into Service Components (SCs). The Framework Services will build the VSN, with a vertex for each of its SCs and the needed links between the vertexes. It is Service Framework responsibility to choose the parameters for the vertexes and links.

The semantics of a request from the Service Framework towards ISONI could be:

- I need a VSN with 4 vertexes (1,2,3,4) and 5 arcs (A,B,C,D,E)
- vertexes 1,2,3 have CPU: 2GHZ and RAM: 4Gb
- vertex 4 has CPU: 1GHZ, RAM: 2Gb and Storage: 4TByte
- Arc A links 1 and 2, B 2 and 3, C 3 and 4, D 4 and 1, E 4 and 2.
- Arcs A,B,C,D have BW: 1Mbps and delay: 10 ms
- Arc E has BW: 10Mbps and delay: 15ms

This VSN will be mapped to ISONI. Note that the actual topology used in ISONI to implement it is not exposed.

To be able to perform its request, the Framework Services need to know what ISONI offers. ISONI providers will not publish what actual resources they have e.g. total number of CPUs. But ISONI providers will publish the class of resource they offer, e.g. whether they have AMD or Intel CPUs, along with the terms for using the resource, e.g. SLA terms including general network capabilities. ISONI will provide basic negotiation capabilities when allocating a VSN, allowing finding the most suitable VSN for the Service Framework and that is able to be allocated by ISONI.

In this VSN the software will run, enjoying the capabilities described in section 5.2. ISONI can give feedback to the Framework Services stating the usage of the Execution Environments and the networks links. We state again that ISONI is not application aware, so reporting will not be available on a 'per application' or per user basis. In other words, if an agreement is made for X processing, Y storage and Z bandwidth then ISONI will report how much of X, Y, Z was actually used but it will not distinguish between the applications or users that make use of this resource. Continuing with our previous example, the report could be of the type:

- The CPU usage of vertex 1 was 60% and the RAM usage 110% (in relation with the SLA/reservation)
- The BW usage of arc A was 50%
- The BW usage of arc B was 105%

The general communication flow between ISONI and the Service Framework in IRMOS is depicted in Figure 31.

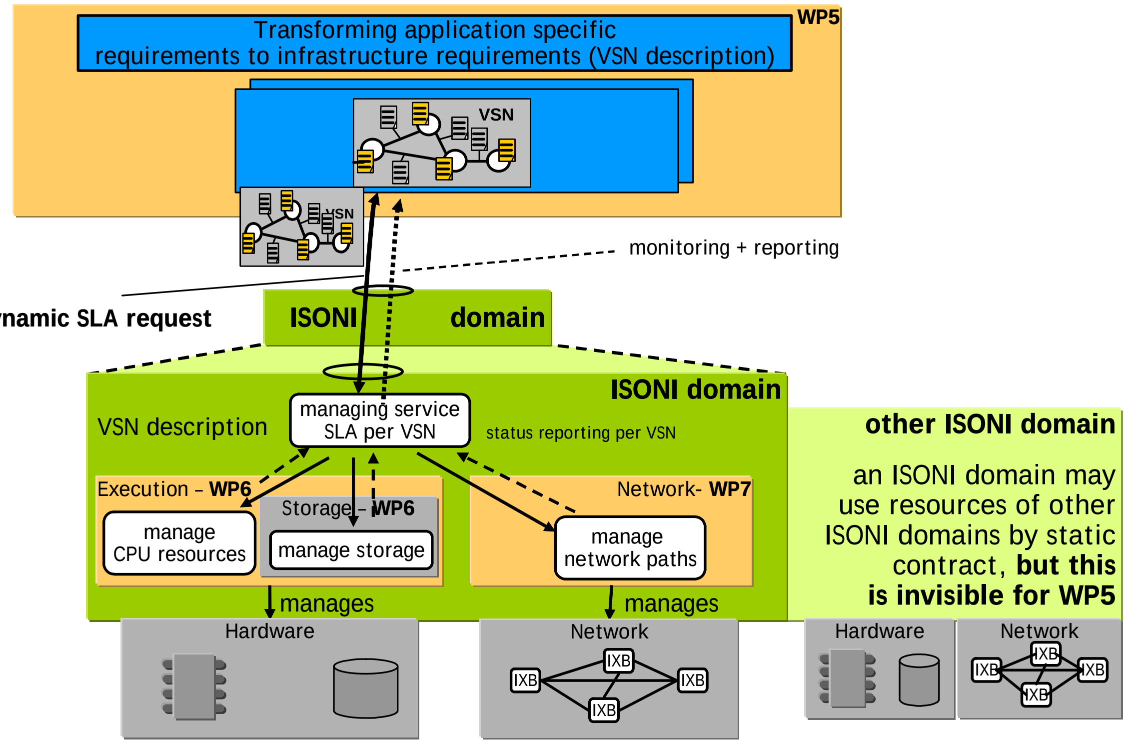

Figure 31: Communication flow between ISONI and the Service Framework

The Service Framework relies on one main technology: WebServices; WebServices are described in XML and so will be the VSN, to ease the interface development between the EE and the Framework Services. More precisely, XML-based ontologies will be used. These ontologies will be detailed in the IRMOS deliverable D6.1.1 [20].

7. Conclusion

This report *"Preliminary Version of the IRMOS Overall Architecture"* presents the results of the work carried out in the scope of WP3 *"IRMOS Platform Specification"* of IRMOS project during the first 7 project months.

Initially, this preliminary version of the IRMOS architecture highlighted the key architectural principles for the design of the platform, namely a service oriented architectural approach with real-time functionality. In addition, the methodology *(Unified Process and UML)* that has been followed for the analysis and design, in order to improve the efficiency of the WP3 work and quality of the results, was briefly described.

Furthermore an analysis of the input from WP2, *Market and Technical Requirements analysis,* from the architectural point of view, took place. Specifically the User Requirements and Use Cases were examined and the result was their prioritisation and association with specific parts of the platform.

The required capabilities for the platform and main processes -as part of the conceptual model- have been identified. Also an overview of the architecture and the main subsystems were presented, having as the centre of interest the main objective of IRMOS for a SOI with real-time attributes.

It is expected that this report will be useful input for the whole project and especially for the development WPs (WP5, WP6 and WP7) that are the main consumers of the work carried out in WP3. In the initial description of the subsystems/components and their interactions, the role of each one in the overall architecture was clarified providing guidelines for their implementation and how these are going to be integrated from WP8.

As the project evolves, we anticipate in providing more insight for the overall platform and the individual components in the updated versions of the report. Specifically, all the processes of the infrastructure and the interfaces between the components will be described in depth while detailed specifications for each one will be produced. WP3 will further examine the application scenarios upon which will be based to create guidelines for the platform.

WP3 is in continuous collaboration with the other WPs of the project in order to acquire more requirements from WP2 and WP4 to "fine-grain" the architecture design. As soon as the D2.4.1 *Security Requirements Analysis* is available, WP3 will work on the security issues of each building block and the security coherence across the security layers. On the other hand, WP3 will get feedback from the development workpackages and more technical details about their components and their implementations in order to avoid any inconsistencies between the subsystems developed by different partners.

8. References

[1] IRMOS Project D2.1.1 Initial version of Requirements Analysis Report, TID and other partners, May 2008.

[2] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, ICCS/NTUA and other partners, July 2008.

[3] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, Section 3.1 - State of the Art in SOIs, ICCS/NTUA and other partners, July 2008.

[4] OASIS Service Oriented Architecture Reference Model http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[5] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, Section 4.8.1 Execution & Resource Management-Scheduling, ICCS/NTUA and other partners, July 2008.

[6] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, Section 4.6.4 Fault Detection & Recovery, ICCS/NTUA and other partners, July 2008.

[7] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, Sections 4.3 Service Discovery & Information Services, 4.4 Monitoring & Metering, ICCS/NTUA and other partners, July 2008.

[8] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, Section 4.7.4 SLA Management & Enforcement, ICCS/NTUA and other partners, July 2008.

[9] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, Section 4.6 Workflow Management, ICCS/NTUA and other partners, July 2008.

[10] I. Jacobson, G. Booch, J. Rumbaugh. The Unified Software Development Process. Addison-Wesley, 1999.

[11] J. Hunt. Guide to the Unified Process featuring UML, Java and Design Patterns. Springer Professional Computing. Springer Verlag, Sept. 2003, page 33.

[12] IRMOS Project ID2.1.2 Updated version of Requirements Analysis Report, TID and other partners.

[13] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWLWeb Ontology Language Reference. Technical Report REC-owl-ref-20040210, The WorldwideWeb Consortium, February 2004.

[14] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, SIP: Session Initiation Protocol RFC 3261, June 2002.

[15] UML Profile for Schedulability, Performance, and Time Specification, 2005.

[16] A UML Profile for MARTE, Beta, 2007.

[17] IRMOS Project D2.3.1 State of the art report on IRMOS technologies, Section 4.11 Security & Trust, ICCS/NTUA and other partners, July 2008.

[18] KVM - Kernel-based Virtual Machine: http://kvm.qumranet.com

[19] XEN: http://www.xen.org

[20] IRMOS Project D6.1.1 Formal description language for application requirements of Execution Environment.