D5.3.1 Assessment metrics, test cases and guidelines

Version 1.0

WP 5.3 Architecture Evaluation and Assessment

Dissemination Level: Public

Lead Editor: Giuseppe Laria, CRMPA

28/09/2005

Status: Final

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2002-2.3.1.18

Grid for complex problem solving

Proposal/Contract no.: 004293

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.
d. **"Original Author"** means the individual or entity who created the Work.
e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.
f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

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

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

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(d) and 4(e).

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

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested.
b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.
c. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.
d. For the avoidance of doubt, where the Work is a musical composition:
    i. **Performance Royalties Under Blanket Licenses**. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.
    ii. **Mechanical Rights and Statutory Royalties**. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created

by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation.

e. **Webcasting Rights and Statutory Royalties.** For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

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

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

7. Termination

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

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Context

Activity 5	**Integration and Application Case Studies**
WP 5.3	**Architecture Evaluation and Assessment**
Task 5.3.1	**Definition of Testing Criteria**
Dependencies	**This document will use inputs from the D2.3.2, D3.1.1, D4.1.1, D4.2.1, D4.3.1 and D4.4.1**

Contributors: CRMPA, TID, USTUTT, CRMPA

Contributors:	**Reviewers:**
CRMPA: sections 1, 2, 3; collected contributions; section 4: interoperability, scalability and reliability subsections; section 6 scenarios definition related to the above attributes **TID**: section 4: security attributes; section 6 scenarios related to the previous attributes; section 7 **USTUTT**: section 5 **UPM**: section 4: availability and performance attributes; section 6: scenarios related to the previous attributes	Per-Oddvar Osland (TN) Brian Ritchie (CLCRC)

Reviewers: Per-Oddvar Osland (TN), Brian Ritchie (CCLRC)

Approved by: Stefan Wesner as IP manager

Version	Date	Authors	Sections Affected
0.1	22/06/2005	CRMPA	ToC + some additional comments
0.2	01/07/2005	CRMPA	Introduction
0.3	15/07/2005	CRMPA	Section 3
0.4	22/07/2005	CRMPA	Added interoperability sub-attributes in section 4 and related scenarios in section 6
0.5	27/07/2005	Robert Piotter	First version of section 5
0.6	24/08/2005	Giuseppe Laria Isabel Alonso Mediavilla	Collected different contributions related to: section 1, section 4 and section 6. Some comments have been included. They affect section 2 and 3
0.7	26/08/2005	Robert Piotter	Updated version of section 5
0.8	05/09/2005	Mario del Campo Melgar Dirk Haage	Updated UPM and TID sections, in particular: 4.4, 4.5, 4.6, 6.1, 6.2.6, 7
0.9	15/09/2005	All	Merged partners contributions
0.91	20/09/2005	J.I.Moreno	Updated sections 4.4, 4.5, 6.2.4 and 6.2.5
0.92	21/09/2005	G.Laria	Final version to be reviewed, included some minor changes in section 5 provided by USTUTT
1.0	28/09/2005	G.Laria	Final version

Table of Contents

List of Figures

List of Tables

Abbreviations

A table with used abbreviations is strongly recommended

Akogrimo	Access To Knowledge through the Grid in a Mobile World
A4C	Authentication, Authorization, Accounting, Auditing and Charging
BP	Business Process
BVO	Base Virtual Organization
COTS	Commercial Off The Shelf
DoS	Denial of Service
EMS	Execution Management System
MTBF	Mean Time Between Failures
OpVO	Operative Virtual Organization
QoS	Quality of Service
SIP	Session Initiation Protocol
SLA	Service Level Agreement
TBD	To Be Done
TBU	To Be Updated
VO	Virtual Organization
WP	Work Package
WSDL	Web Services Description Language

Executive Summary

This deliverable explains which kind of evaluation approach we are going to adopt in Akogrimo that in any case will be based on what was written in the DoW [2].

Due to its nature the deliverable needs inputs from other artefacts of the Akogrimo project and, in particular, deliverables related to:

1. Test bed definition (in order to understand which aspects we want to evaluate)
2. The architecture description (in order to evaluate if it respects the selected requirements)

With these assumptions, we are going to identify:

1. People involved in the evaluation process.
2. The evaluation process itself.
3. The relevant requirements that we are going to evaluate.
4. The criteria to be taken in account for evaluation.

Following the WP description[2], in the introduction we explain the expertise/role of people that are interested in reading this document and we underline what is the general meaning of "evaluation" in Akogrimo, explaining the relevance of software architecture in a computing system. Finally we provide a brief list of methods focusing on software evaluation that can be found in related literature.

In Section 2, we will start to identify the expertise/role of project partners that will be involved in the different phases of the evaluation and also will be in charge of the evaluation process. We then describe the evaluation process itself and how it is based on the existing methods introduced in Section 1. Furthermore we describe the whole evaluation process and clarify which parts of this process are in the scope of this document and which steps are in the scope of following deliverable of this WP (D5.3.2).

In Section 3 and 5, we apply the process defined in Section 2, implying the definition of some basilar attributes (requirements to be compliant with) that are identified on the basis of a requirements analysis of our reference applications carried out in the frame of validation scenarios [6]. Moreover, for each attribute we provide some scenarios of use (evaluation criteria) that should be taken into account by the evaluation team in order to understand if the architecture is suitable for each attribute.

In Section 4, we identify the components of our architecture that can affect the suitability of each different attribute. It is clear that this section is not completed and it will in the future experience some updates according with changes introduced in the architecture of each layer. These updates will be needed as the detailed architecture definition is an on going work as the document is produced and then therefore not completely available, yet.

Finally the specified evaluation process will mainly serve as input to the D5.3.2 ("architecture and prototype evaluation report"). We would like to underline that an updated version of this deliverable will be provided by the end of month 19; as in the current version we are mainly focusing on qualitative evaluation, we intend the updated version to include some quantitative criteria in order to evaluate the prototype. In this way it will be possible to compare the likely limitations coming from implementation problems related to architecture inadequacies; of course the latter needs to be handled carefully in order to avoid having a final prototype that is difficult to turn into a real product at the end of the project.

1. Introduction

This document describes how we are going to perform the evaluation process of the Akogrimo project architecture. Expected readers include:

- Architecture designers who will evaluate whether the selected attributes and identified scenario (see following sections) are correctly related to the architecture components [1].
- The staff involved in the evaluation process that will provide as output a final report (D5.3.2 – Architecture and prototype evaluation report) planned to be delivered by the end of month 18 [2]. In any case, we are planning a shift of this deadline because the final implementation of the first prototype is scheduled at the same time and, instead, it seems feasible to produce this report one or two months after the release of this prototype.
- The participants of the Activity Committee [2] that will approve the evaluation methodology proposed in this document.

Of course this document will be the starting point for D5.3.2 that will summarize the results of the evaluation process performed following the guidelines and by applying the criteria we are going to define in the following sections.

The evaluation will focus on the software architecture, because architectural decisions affect the quality of software systems and it is important that potential risks should be detected as early as possible during the software development process.

Today, there is a general agreement about the importance of identifying architecture problems during the software life cycle and, of course, the architecture definition is in the early phase of this life cycle.

This kind of evaluation assumes more relevance if we are developing a large software system that will have a long lifetime (as is the case of the Akogrimo software system); furthermore a good understanding of the architecture design quality will simplify the solution of problems that, otherwise, could be difficult to handle during the development phase.

Having in mind this viewpoint, it is clear that the evaluation is strictly related to the architecture and our aim is to understand if our design is valuable. This means that in case we had poor results from quantitative testing of the Akogrimo prototype we should evaluate if the architecture design provides solutions for these poor results and, in this case, we will have a positive feedback concerning the value of our design and the reason of this poor result could be just a shortcut taken during the implementation.

In general, we can identify two different approaches to the software evaluation:

- *Predictive*[1]: select the goals of the software system and then evaluate how the architecture meets these goals (e.g. performance, scalability, reliability…). This approach is predictive because it tries to anticipate what the behaviour of the implemented system will be on the basis of its design.
- *Retrospective*: based on the analysis of different releases of the system, evaluate changes which are introduced at the architecture level in order to assess the stability of the designed architecture.

[1] In this context, the term predictive is not strictly related to the Predictive and Retrospective methods for evaluating architecture stability and evolution.

We anticipate that we are going to follow a scenario based approach that aims to understand in which way the architecture addresses some identified goals without performing real measurements on the prototype that implements the architecture itself. This evaluation is performed before the implementation phase and in this sense we can consider this method as a predictive method.

1.1. The role of software architecture

Before introducing the role of a software architecture we must define what we mean by this term: "The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them." [3]

In fact there is no common agreed definition of software architecture; but there are different definitions that introduce similar concepts, such as: components, relationships, connections, configuration, and externally visible properties.

As explained in [4], an architecture should be described using a language that is a formal notation allowing the description of a software system as a composition of connected yet independent components. Of course for each component the externally visible behaviour should be described.

We can then gather that architectural decisions are very important for the individual components because on the basis of architecture value they can be implemented relatively independently. Furthermore it is clear they affect the overall behaviour of the system and, in particular, they can determine the quality of the main properties of the software system. From this viewpoint the architecture should be designed so that particular kinds of changes (introduced in order to improve system properties) should be easy to accomplish in order to guarantee a straightforward evolution of the system.

The following Figure 1 shows the relationship between Software Architecture and System Quality Requirements ([5]).

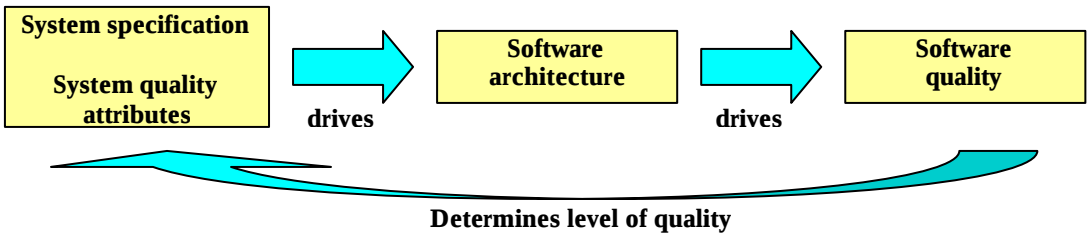

Figure 1 Role of software architecture

The system specification is the artefact ([6]) that describes the functional and non-functional requirements of the system. The non-functional requirements can be considered as quality attributes (e.g. scalability, reliability, availability, performance) and they will be taken into account during the software architecture design. Of course the software architecture will drive the software development and, then, its quality.

The relevance of these points explains how the architectural decisions can significantly affect the results related to the accomplishment of prefixed quality attributes. For example, if you need a high performance system you have to take into consideration the communication between the components and their efficiency; or if you are designing a high-availability system then you should foresee many redundant components. These are typical examples of quality attributes affecting the architecture design.

We have to underline how most design decisions should be made very early in the software development process and in any case before the implementation phase; by then they are very hard to get right and to change, considering that they can have a big impact on the final system.

Due to this relevance, architecture evaluation is a very important activity that helps to reduce risks related to the software development process. Further, this evaluation process isn't a simple task and many studies have been conducted in order to introduce valid approaches to the creation of a cost effective architecture evaluation process. In the following section, we'll introduce a brief summary of possible architecture evaluation approaches.

1.2. Architecture evaluation and assessment

Due to the relevance of the architecture's role with respect to the life of a software system, its evaluation and assessment has great importance.

First of all, it is necessary to identify which quality elements we are going to evaluate and what the expected results of the evaluation are. On the basis of identified elements an evaluation procedure that typically is based on qualitative considerations will be defined.

We can have different kinds of evaluation approaches and we can provide a sort of categorization of them:

- **Scenario based methods** [7]: maybe the most mature evaluation approaches are scenario based. They are founded on the definition of some scenarios that are a meant to describe the behaviour of the system with respect to a particular quality attribute and in a particular context. For example, scenarios can be defined to understand how a software architecture responds with respect to attributes such as maintainability, reliability, usability, performance, flexibility…
  Typical examples of scenario based methods are: Software Architecture Analysis Method (SAAM), Architecture Trade-off Analysis Method (ATAM) and Architecture Level Modifiability Analysis (ALMA).
- **Questionnaire-based methods** [9]: if the goals of the software system are easy to be identified and characterized, it is possible to define a list of questions that can be applied to the overall architecture of the software system. These questions constitute the questionnaire to evaluate the architecture and they can deal with different aspects of architecture definition (e.g. generation, documentation, description…);
- **Check-list based methods**: this method is similar to the questionnaire based but, generally, it focuses on specific qualities to be addressed by the architecture. The check-list based method requires a more mature evaluation practice with respect to the previous one;
- **Metric based methods**: these are quantitative analysis based on measurement of the architecture components. The aim of this measurement is to find the places where there are problems in the overall architecture in order to introduce changes to improve the design.
- **Proof-of-concept-based methods**: these are based on prototypes used for experiments and simulation. These are artefacts that are usually results of the development process. They test implementations that represent a model of the architecture. These prototypes can be used to answer questions arisen in the early phase of design and, for example, defined in the evaluation methods described above.

The first three methods can be considered predictive evaluation methods; they are applied in the early phases of system development process rather than during the architecture design. They try to anticipate how well the architecture will address the requested requirements by examining the

architectural decisions made during the design cycle. The result of this kind of evaluation provides qualitative conclusions on the appraised goals.

The last methods are more retrospective approaches and they are based on quantitative measurements that are performed during the implementation phase. This way, it is possible to evaluate different releases of software that will provide information about the changes to be introduced in the architecture. Of course on the basis of the changes added in each following release is possible to understand the value of the initial designed architecture.

We anticipate here that in our evaluation approach we'll mainly adopt a scenario based methodology.

This choice is motivated by the need of having an iterative evaluation process that fits in the general development process of the Akogrimo project. Following this approach will have available a first evaluation result by the end of first project cycle. Of course this evaluation will focus just on the architecture and it will be based on qualitative approach because at that time it will not be available a prototype, yet. Otherwise, at the beginning of the second cycle, we can start a quantitative testing of the first prototype. The results of the qualitative and quantitative evaluations will be compared to understand if we have to impute identified problems to the architecture design or to the implementation. In this way, we will improve the part that is the origin of the inadequate results with respect to the final goals of the Akogrimo project.

The following figure describes the highlighted evaluation process and the different steps to achieve it: the yellow blocks stand for a task and the green circle is the result of the related task. Furthermore the arrows represent the temporal dependencies between tasks and results. Of course, dependency exists between same kind of arrows (dotted or continuous)

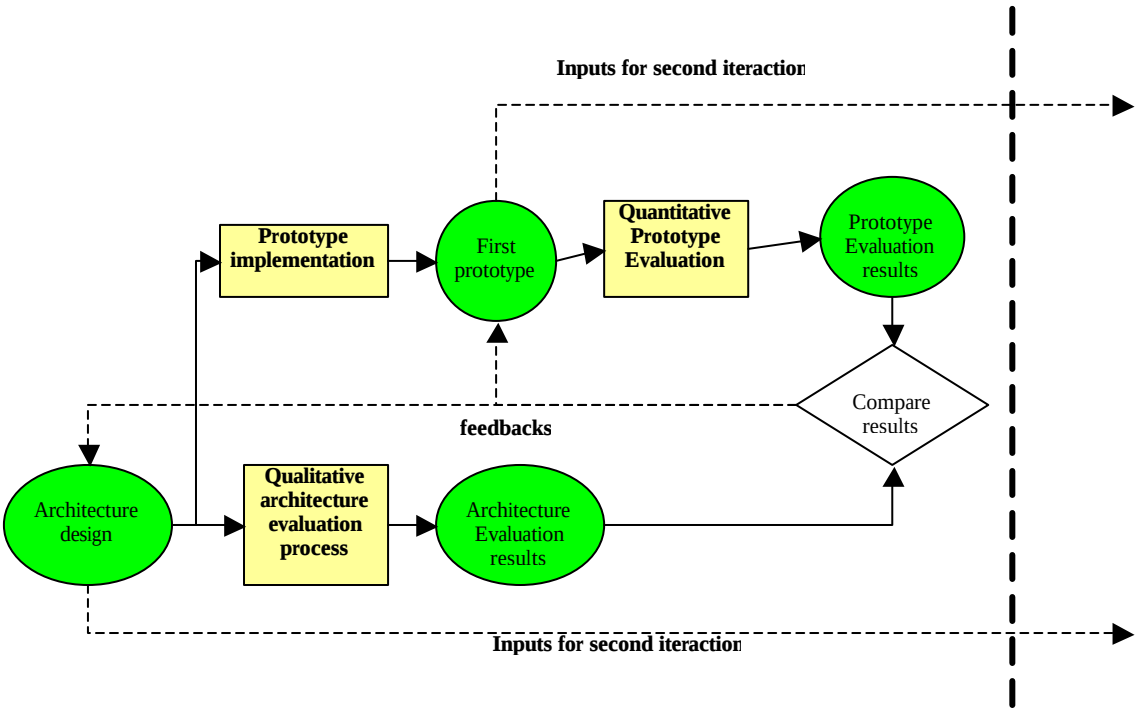

Figure 2 - Evaluation process with respect to Akogrimo project iteractions

2. Evaluation methodology

In spite of the definition of many methods to allow architecture evaluation, we'll mainly refer to scenario based approaches and even if we leverage on existing works on this topic, we'll personalize these approaches following in some cases more pragmatic considerations. Our aim is mostly to identify during our analysis any risk, sensitive point, and trade-off that can be associated to the architecture.

First of all, we have to define a list of stakeholders that have a different role in this process and they can have different special interest in the architecture along with the infrastructure that will be developed from it:

- *Evaluation team*: They will conduct the evaluation, from this evaluation they will provide results to be further evaluated by the project decision makers
- *Project decision maker*:
  - Designers
  - Project management
- *Development team*:
  - Developers
  - Testers
- *Application providers*: In the frame of Akogrimo project, they are the scenarios providers. They have defined the requirements to be addressed by the infrastructure, as they have sketched the specific applications that are supposed to be executed on top of the layered infrastructure.

In general, we have identified some stages to be executed during the evaluation process and in the following we are going to introduce what will be performed in each of them, furthermore we have to underline that each stakeholder will be involved in a different phase each stage based on their specific interests.

Step 1. *Definition of attributes to be evaluated*: to perform this step, it is necessary to have an interaction with the stakeholder "Application Providers" that have provided the requirements to be addressed. Actually, the "quality attributes" will be extracted from the requirements that have been defined in the validation scenario document [6]. Because of the mentioned deliverable is not public (while this document is public), we'll add an annex containing the sections of [6] which we are interested in. Mainly, we will focus on non-functional requirements; in addition in some cases we will take in account some main functional requirements. In particular, we will investigate functionalities requested by the specific application domain in order to understand if someone can be considered relevant with respect to architectural choices.
A typical quality attribute definition should have the following table format:

Table 1 Example of table to define an attribute

Quality Attribute	Scalability: scalability is a qualitative measure of how easy it is to expand the infrastructure
Sub-Attribute 1	Scalability in the AAA subsystem
Sub-Attribute 2	Scalability in the SLA subsystem
Sub-Attribute 3	…

In the above table, the first row contains the definition of the high level quality attribute, in fact scalability can be considered with respect to different subsystems of our architecture. In the other rows it is specified with a list of sub-attributes in respect to which aspects of scalability are really relevant and that we want to evaluate.

Step 2. *Key components of the architecture*: the stakeholder involved in this step is the designer of the Akogrimo architecture. Actually, it is possible to refer to [1] that provides all the information related to first version of Akogrimo architecture. In this step, we identify all components of the architecture that contribute to address a specific sub-attribute. Also in this case, we can adopt a table format:

Table 2 Example of table to list key architecture components related to a quality attribute

Sub-Attribute	Scalability in the SLA subsystem
Component 1	Brief component description
Component 2	Brief component description
Component 3	…

In relation to any explanation related to the components interactions, we will refer to [1] or to the description of each layer of Akogrimo architecture [8].

Step 3. *Identification of several scenarios for each sub-attribute*: the stakeholders involved in this step will be the application providers and the development team. The aim is to define several scenarios of use explaining the meaning of each sub-attribute in different cases.

The need of different scenarios for the same sub-attribute comes from the evidence that, in some cases, the quality attribute is quite complex and does not exist in isolation, thus making it difficult to evaluate them on a single scale, as they can have a relevant meaning just in a specific context. In fact, a system can be considered: secure with respect to specific threats, usable with respect to specific classes of user, efficient with respect to the resources involved in its utilization.

Summarizing a scenario is a specified set of steps (related to specific sub-attributes) that involves different components of the architecture to be achieved. On the basis of the demands that the scenario imposes to the architecture, it is possible to evaluate the changes to be introduced, then, the value of the architecture.

The following table shows an example of scenarios related to a specific sub- attribute:

Table 3 Examples of scenarios related to a specific sub-attribute

Sub-Attribute	Scalability in the AAA subsystem
Scenario 1	During the life time of the virtual organization the number of users of the AAA subsystem increases severely. In this case, the number of concurrent (or in a time slot) accesses can increase as well. Does the architecture manage this case? To what extent?
Scenario 2	For different reasons, the number and/or the kind of data to be provided by the AAA subsystem changes during the lifetime of the system.
Scenario 3	…

Step 4. *Architecture evaluation:* this step is performed by the Evaluation team. In general, it should be composed by persons that were not involved in the architecture design process. This evaluation should have as result a report describing which parts of the architecture don't guarantee compliance with the quality attributes. Actually, in this step it will be identified where potential architectural problems are and the results will be provided as input to the next step to aid decision making.

Step 5. *Risks identification*: the involved stakeholders are the "project decision makers". On the basis of the evaluation phase results they have to individuate:

- The risks: that is a set of problematic architectural decisions to be taken. The level of risk will be related to the relevance of quality attributes that they address and it will be evaluated in terms of the costs and changes that these decisions need in order to be achieved.
- A rationale for the positive or negative effects that the individuated decisions have on addressing the related quality attribute
- A list of issues that are currently not solved and which need a decision to be taken in relation to them.

The following Figure 3 shows the flow between the different steps:

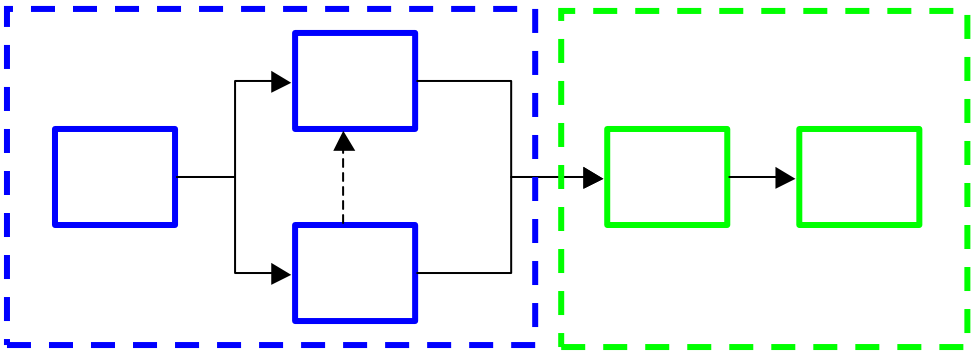

Figure 3 Sequence flow of the different steps

In Figure 3, we have two different blocks (blue and green). The steps included in the blue block will be dealt in this document, the others will be a topic of the following deliverable (D5.3.2) which deals with the evaluation process and its results.

3. Attributes to be evaluated

As starting point of our attribute evaluation we are going to take in account the table summarizing the Akogrimo non-functional requirements defined in [6], we have to notice that this deliverable is not public, therefore we have added the table we refer to in Annex A.

3.1. Interoperability

The first attribute we are going to define is the interoperability that is a non-functional requirement of Akogrimo.

Interoperability is the "*ability to work with other systems*", in Akogrimo it means that we have to be able to guarantee component integration with external legacy applications, middleware or COTS components.

We have to notice that in this context the role of standards is primary and then we should consider if:

- we are using standardized (open) protocols
- we are proposing extensions, which conform with the protocol
- we are trying to standardize them

In order to better lead the evaluation phase, we are going to identify the specific aspects of interoperability that are relevant with respect to Akogrimo requirements. In general we can refer to two different critical points:

- Internal interoperability: related to the communication between internal components/service of the Akogrimo infrastructure (e.g. because will be used different grid middleware).
- External interoperability: related to the communication with other systems that want to exploit Akogrimo capabilities

The next table lists some critical cases that we consider important to be evaluated from the architecture viewpoint.

Table 4 Interoperability sub-attributes

Quality Attribute	Interoperability
Sub-Attribute I.1[2][3]	Interoperability between network and grid layers: the interoperability should be guaranteed between the modules of network layers accessible by the external layers and vice versa
Sub-Attribute I.2	Interoperability between modules of Grid layers built on different Grid middleware
Sub-Attribute I.3	Interoperability between mobile devices and the modules of infrastructure that communicate with them
Sub-Attribute I.4	Interoperability of the front-end infrastructure modules. Each module that has to communicate with an external entity needs to guarantee interoperability in the sense that it should be easy to integrate an external application or component with Akogrimo infrastructure.

3.2. Scalability

The scalability is another non-functional Akogrimo requirement that provides information about the suitability of an infrastructure with respect to problem of its expansion. Indeed we can affirm that the scalability should be foreseen as system requirement in order to guarantee the minimization of efforts and expense when the infrastructure system (hardware and software) is enriched (or better overloaded) with additional data/users/resources[4].

In general, to achieve the scalability requirements there are several mechanisms that mainly focus on the increasing of hardware components and their capabilities, like:

- CPU number
- CPU power
- Storage devices
- Storage capacities
- RAM components
- RAM component dimension
- …

[2] I.1 :

- I is the Quality attribute symbol (first type of the quality attribute name, the second type should be added if the matching is not unambiguous).
- 1 the sub-attribute number

[3] The scenario definition will describe the specific scenarios of each sub-attribute that we want to address with our architecture, in this way it will be better detailed what we are going to evaluate.

[4] With these terms we want to collect all parameters, context change and infrastructure updates that can arise during the evolution of hardware/software infrastructure, in order to apply on them that the scalability function is valid

These mechanisms don't have to be confused with redundant systems that are usually introduced to achieve fault tolerance requirements. Below we summarize some scalability sub-attributes that can be applied in Akogrimo environment.

Table 5 Scalability sub-attributes

Quality Attribute	Scalability
Sub-Attribute S.1	Hardware scalability: changes into the number of machines into Akogrimo environment shouldn't affect the overall system behaviour[5].
Sub-Attribute S.2	Software scalability: it should be possible to modify the services distribution inside the Akogrimo environment in order to face different kinds of requests and load situations.
Sub-Attribute S.3	Network scalability: changes into the number of simultaneous accesses to services shouldn't affect performance of the service itself.

3.3. Reliability

The reliability non-functional requirement represents the ability of our infrastructure to consistently perform according to its specifications.

In theory, a reliable component of the Akogrimo infrastructure should be totally free of technical errors; in practice, however, we will consider some classic parameters that will express the reliability quotient as a percentage. In particular, we are referring to the MTBF (Mean Time Between Failures) that is the a measure of the average time that the implemented module works without failures following the defined conditions.

The MTBF figure can be developed as the result of intensive testing and on actual modules experience, it can be then used as a quantifiable objective.

On the basis of these premises, with respect to the reliability attribute, we are going to perform a quantitative evaluation and the metric, used for this test, will be the MTBF. Otherwise, if necessary, we'll introduce some additional metrics before the end of the prototype implementation phase.

We are not going to define here sub-attribute and related scenarios in section 6, but what we expect from the architecture evaluation process is:

- A list of architecture's modules to be tested from the reliability viewpoint
- For each identified module, a look-out for architectural items that could compromise reliability (e.g. adoption of technology that has known stability problems, use of existing software release that are not in a final version)

When we will test the first prototype with respect to the reliability, in case of poor results, we have to report if they can be imputed to architecture and/or implementation issues.

[5] The system should be able to transfer the execution performed on one machine on other machine and in a transparent way with respect to the user.

3.4. Availability

Availability of a system is the probability that it accomplishes its tasks at or within a given time. For computer systems, availability also reflects the ratio of the time the system is actually usable to overall time. Mean Time To Repair (MTTR) along with MTBF gives a measure for availability: MTBF/(MTBF+MTTR)

System components can be unavailable due to:

- Hardware problems (no network connection, broken hardware, etc.)
- Software problems (hight number of requests, DoS attacks, mainteinance service, etc.)

As the architecture consists of several dependent components, not only the availability of each component is relevant, but also the impact of one failing component to the system (e.g. what happens if the A4C server is down). In order to do so, each component must be checked whether:

- It may be a single point of failure, if so, how fundamental is this component to the system (i.e. how big is the impact of this failure, how many other components won't be available during the period it is down?)
- It is laid out for the number of proposed request/time, even during rush hour
- It is vulnerable to DoS attacks (-> Security)

Depending on the results of the analysis the Akogrimo component should be replicated or distributed to maximize Akogrimo availability service. However, replication/distribution is not always possible, and should in general be handled with care. Risk of data inconsistency may in many cases reduce the ability to distribute a component.

Summarized in the next table are some quality availability sub attributes that have to be guaranteed by the Akogrimo architecture.

Table 6 Availability sub-attributes

Quality Attribute	**Availability**
Sub-Attribute A.1	Key service availability: Fault/unavailability of a key service[6] shouldn't affect the availability of related Akogrimo functionality
Sub-Attribute A.2	Maintainability: the scheduled downtime (e.g. due to configuration activities) should be low enough to guarantee an acceptable percentage of availability for each single service.
Sub-Attribute A.3	Redundancy: Hardware/Software components should be replicated to guarantee that any single point of failure will not affect other the Akogrimo services.
Sub-Attribute A.4	Error Monitoring: Monitoring tools and procedures to detect errors in the Akogrimo system as well as guide the solution (software/hardware/switch to mirror systems) to minimize the MTTF (mean time to fix)
Sub-Attribute A.5	Component Upgrade: Architectural implications of software/hardware upgrades affecting Akogrimo services.

[6] The architecture evaluator will decide which are the key services of Akogrimo infrastructure

3.5. Performance

Performance of a system indicates the efficiency of the system while performing tasks. It includes total throughput of an operation as well as memory and disk space efficiency. In Akogrimo we should consider performance at Grid Subsystem and Network Subsystem and it should be necessary to:

- Establish Requirements of the hardware components (CPU, Memory, Disk) under usage conditions.
- Maximize throughput of request/responses perform over the system
- Analyse behaviour of the Akogrimo architecture if number of request/response increase decrease. Identify bottleneck for Akogrimo services and how they could be fitted
- Identify wasted time by Akogrimo modules while waiting response from other modules.
- Identify performance dependency of the network and grid subsystems to support Akogrimo services.
- Feedback support from Akogrimo architecture to keep the user in touch of the status of the request.
- Analysis of the maximum time delay and average time delay to be supported by Akogrimo services.
- Use of soft requirements (e.g. 90% of requests with response time less than 1 sec) vs hard requirements (no requests have response time over 1 sec).
- The various services will have different requirements and different number of users. There is need for a metric that accounts for several aspects, including
    - Different service levels/classes with ordering, e.g. vital, priority, best effort
    - Number of users per class
    - Performance requirements for each class
    - Aggregated view of requested performance vs. capacity

Performance should be guaranteed by the Akogrimo architecture to avoid degradation periods of overloads. So depending on the service provided, the maximum number of users/requests should be controlled avoiding degradation of the service to the running users.

In next table are summarized some quality availability sub attributes that have to be guaranteed by Akogrimo architecture.

Table 7 Performance sub-attributes

Quality Attribute	Performance
Sub-Attribute P.1	System Performance: Identify performance in terms of CPU, Memory and Disk required by each hardware component in Akogrimo architecture according to the usage conditions of each service
Sub-Attribute P.2	Bottleneck: Identify bottlenecks of Akogrimo architecture according to each particular service.
Sub-Attribute P.3	Performance Guarantee: Identification of police blocks in Akogrimo architecture to avoid or limit performance degradation of the existing users by denying new requests in case of overload.
Sub-Attribute P.4	Latency: waiting times should be minimized by increasing networking and hardware capabilities.
Sub-Attribute P.5	User Feedback: User should be able to enquire about the status of his/her request.
Sub-Attribute P.6	Akogrimo characterization of performance (latency, response delay and throughput) per service supported.

3.6. Security

In any software system and, consequently, in Akogrimo, security is a crucial aspect for the better performance and commercial success of the services offered in such a software platform. In general, security deals with secure access of the players (users, programs or services) in such a way that they can only execute actions that have been authorized. Security threats compromises and data protection must also be considered, and it has to be guaranteed that any intruder entity can't access, modify, destroy or disclose sensitive data and communications. At architecture level it is possible to anticipate the degree of security that the system is going to offer and, within Akogrimo, the security analysis at architecture level is going to be performed in the following subsections:

- Identification
- Authentication
- Authorization
- Integrity
- Intrusion detection

3.6.1. Security Identification

This is the security requirement concerning the ways in which an application or component identifies players (e.g., users and client applications) before granting them access.

Under this point of view, the system should detect and prevent the use of forged identification data and the reuse of identification data. Here, it is highlighted that the mobility concept is intrinsically bound to the Akogrimo framework thus allowing the same user identification to use different access devices.

Identification requirements are often specified in terms of the following measurements:

- Minimum percentage of valid users (by role) identified.
- Maximum percentage of invalid users (by role) identified (false positive).

The next table lists the quality attributes concerning Security Identification that are considered of special importance to be evaluated from the architecture viewpoint.

Table 8 Security Identification sub-attributes

Quality Attribute	Security Identification
Sub-Attribute SI.1	Secure Identification in the A4C Server
Sub-Attribute SI.2	Secure Identification in the BVO Manager

3.6.2. Security Authentication

The Authentication requirements are part too of the Security requirements and, as in the case of identification, they are not enough by themselves, but they are necessary prerequisites for authorization requirements. Authentication could be defined as the process of verifying the identity of a computer or computer user. In the case of the users, it generally involves a user name and password. For the case of computers, they usually require a pass a code for proving identity.

In Akogrimo, the Authentication process consists of asking the user for a Username and Password. Once the username and password are validated, the user can perform actions on the platform, such as to start making requests and using Services depending on his or her profile.

Authentication requirements are often specified in terms of the following measurements:

- Minimum percentage of valid identities (by role, group) authenticated.
- Maximum percentage of invalid identities (by role, group) authenticated (false positive).
- Mean time for an attacker to become authenticated (manually, using a computer of given processing power).

The next table lists the quality attributes concerning Security Authentication that are considered of special importance to be evaluated from the architectural viewpoint.

Table 9 Security Authentication sub-attributes

Quality Attribute	Security Authentication
Sub-Attribute SA.1	Secure Authentication in the A4C Server
Sub-Attribute SA.2	Secure Authentication in the Base VO Manager

3.6.3. Security Authorization

Authorization is the process in which a valid authenticated user or application can gain access to specific services relating to the function of their profile. Therefore, unauthorized outsiders have to be prevented from obtaining access to restricted data and services.

Authorization requirements are typically specified in terms of the following measurements:

- Minimum percentage of authenticated externals authorized [by role, by task].
- Maximum percentage of non-authenticated externals authorized [by role, by task] (false positive).
- Mean time for an attacker to gain unauthorized access [manually, using a computer of given processing power]

The next table lists the quality attributes concerning Security Authorization that are considered of special importance to be evaluated from the architecture viewpoint.

Table 10 Security Authorization sub-attributes

Quality Attribute	Security Authorization
Sub-Attribute SZ.1	Secure Authorization in the A4C Server
Sub-Attribute SZ.2	Secure Authorization by the VO Authorization Service

3.6.4. Security Integrity

Integrity is the capacity of a system to protect its data and communications from intentional corruption. These attacks can come from different sources, like unauthorized individuals and programs.

Within the Akogrimo platform, a special mention has to be done for the Data Management subsystem since it is in charge of handling all transmitted, received and stored data, and in all of these topics, the platform shall protect, detect corruption and act to prevent corruption.

Some measurements for integrity requirement can be:

- Maximum percentage of data files/records corrupted per unit time.
- Maximum percentage of messages corrupted.
- Maximum percentage of programs corrupted per unit time.

The next table lists the quality attributes concerning Security Integrity that to be evaluated from the architecture viewpoint and are considered of special importance.

Table 11 Security Integrity sub-attributes

Quality Attribute	Security Integrity
Sub-Attribute SY.1	Integrity in the Data Management System
Sub-Attribute SY.2	Integrity in the Messaging System
Sub-Attribute SY.3	Integrity in the Programs

3.6.5. Security Intrusion Detection

Intrusion Detection can be defined as the process in which an unauthorized access or modification by intruders (individuals or programs) is detected, recorded and notified. Then, Intrusion Detection is strongly related to the Identification, Authentication and Authorization processes, so its requirements will depend on them.

Inside Akogrimo, an intrusion attack could be, for example, a repeated authentication failure and the system should be able to detect and notify it. Moreover, the system should notify periodically all attempted and succeed intrusions in a given period of time. Regarding this question, a proper measurement for this attribute would be the minimum percentage of the attempted intrusions detected and notified to the security log.

The next table lists the quality attributes concerning Security Intrusion Detection that are considered of special importance to be evaluated from the architecture viewpoint.

Table 12 Security Intrusion Detection sub-attributes

Quality Attribute	**Security Intrusion Detection**
Sub-Attribute SD.1	Security Intrusion Detection in the A4C Server

4. Gross architecture description

Table 13 gives an overview of the architectural components in relation to the work packages. Their function, structure and interactions will be detailed in the following sections with respect given to the architectural qualities that are to be evaluated.

Table 13 Components Overview

WP4.4 – Grid Application Support Services Layer
▪ VO Management ▪ BP Enactor ▪ SLA High Level Services ▪ VO Security Services
WP4.3 – Grid Infrastructure Services Layer
▪ Execution Manager (EMS) ▪ Data Manager ▪ Policy Manager ▪ Security Services ▪ Metering ▪ Monitoring ▪ SLA Enforcement
WP4.2 – Mobile Network Middleware Architecture, Design and Implementation
▪ Context Manager ▪ Service Discovery Server ▪ A4C ▪ SIP User Agent ▪ SIP Presence Agent
WP4.1 – Mobile Network Architecture, Design and Implementation
▪ Mobile Terminal ▪ Home Agent ▪ Access Router ▪ SIP Server ▪ QoS Broker ▪ Network Policy Manager

Architectural qualities can be attributed to different phases in the overall lifetime of the system. The following list gives an overview of the phases and the related activities:

- Preparation
  - Metadata definition (SLAs, Policies, WSDL, ontology descriptions, device profiles, network QoS bundles, others)
  - Business process definition
  - Definition of Base VO roles
- Setup
  - Network Setup
    - Deployment of network related metadata
  - Hosting Environment Setup
    - Deployment of service related metadata
    - Deployment of workflow templates
  - Service Setup
    - Deployment
    - Configuration
    - Publication
  - Membership Management
    - BVO Membership
    - A4C Registration: User identity attributes
    - SIP Registration
- Runtime
  - Normal Flow (including OpVO creation and termination)
  - Failure and exception handling
- Maintenance
  - Setup steps have to be repeated in case of changes to the metadata or in case of changes to the network topology or hosting environment configuration, e.g. addition or removal of hardware.
  - System behaviour inspection
  - Auditing of accounting data
- Charging & Billing

Different components are involved in different phases of the overall lifetime of the system. The normal flow during the runtime of the system is most important for satisfying the business goals. Runtime characteristics are:

- Realisation of a business process
  - This is done by the BP Enactor component
- User context collection, processing and distribution
  - This is done by the Context Manager component
- Adaptation of the runtime behaviour to context changes

- This is realized by a combination of application specific services, the BP Enactment component and the Context Manager

- Support for mobile and nomadic users
  - This is enabled by the underlying network infrastructure and influences especially the Context Manager and the BP Enactor
- Support for integration and operation of a large variety of different resources
  - This is achieved by the grid infrastructure layer
- Support for Charging and Billing
  - This is based on the integrated A4C infrastructure

Each of these main functionalities is supported by and depends on several components of the architecture. The dependency for achieving a certain architectural quality will be described in the following.

4.1. Architecture components and attributes to be evaluated

Following the methodology outlined in section **¡Error! No se encuentra el origen de la referencia.**, we describe the components that are involved in achieving the architectural qualities.

4.1.1. Interoperability

Table 14 Sub-Attribute I.1

Sub-Attribute I.1	Interoperability between network and grid layers
QoS Broker	The QoS Broker provides information about network resource availability to the EMS. Network resources can be allocated by the EMS (see D4.3.1 [8]).
Network Policy Manager	The network policy manager interfaces with a VO level policy manager to exchange information about policies. It might be possible to align network level policies with VO policies.

Table 15 Sub-Attribute I.2

Sub-Attribute I.2	Interoperability between modules of Grid layers built on different Grid middleware
Service Discovery Server	The service discovery server is a central component of service oriented architecture. Entities like the Workflow Manager (which is part of the BP Enactor component) can perform a search for services based on a meta data description of the required services. Also application services, which may be implemented using different toolkits, have to interoperate with the Service Discovery Server when they register their service description. (See D4.4.1 and D4.2.1 [8]).
Execution Manager	The Execution Manager (EMS) has to be able to search services in the Service Discovery Server. It also interfaces with the QoS Broker to allocate required network resources.
Context Manager	The Context Manager allows the receipt of notifications relating to context changes. The receiver might be the BP Enactor component to allow context dependent workflow execution.

Table 16 Sub-Attribute I.3

Sub-Attribute I.3	Interoperability between mobile device and modules of infrastructure that communicate with them
Context Manager	The mobile node has a SIP Presence User Agent installed to communicate with the context manger. By use of a local service discovery agent the mobile node also interacts with the local service discovery server that is part of the context manager.
Access Router	When the mobile terminal moves the access router is involved in the hand over between two networks.
Home Agent	The home agent of the mobile terminal ensures that the terminal can be reached by use of its home address. When the mobile node is in a foreign network, route optimization occurs and the care-of address is used instead of the home address of the mobile terminal.
BP Enactor	In case the mobile node provides a service it will also interact with the BP Enactor component.

Table 17 Sub-Attribute I.4

Sub-Attribute I.4	Interoperability of the front-end infrastructure modules
Service Discovery Server	An application specific service has to register with the Service Discovery Server to make itself available for use.
SLA High Level Services	An application specific service can negotiate the terms and conditions of use with the SLA High Level Services.

4.1.2. Scalability

4.1.2.1. Hardware Scalability

Sub-Attribute S.1 – Hardware Scalability is concerned with situations when additional hardware could potentially have a visible impact for service consumers.

Table 18 – Sub-Attribute S.1

Sub-Attribute S.1	Hardware scalability
Execution Management Service	The EMS is able to instantiate services on different machines. Adding more machines should be possible and should not have negative effects on scalability and performance.
Data Management Service	The Data Management Service should be able to handle additional storage resources.
Access Router	As routing packages is not a time consuming task, Access Routers seldom have a scalability problem.

4.1.2.2. Software Scalability

Sub-Attribute S.2 – Software Scalability is influenced by the average processing time of a service. Performance of individual services can affect the overall scalability if shared resources like databases or RAM on a machine produce performance bottle neck. Also centralized services can become a scalability problem, if it is not possible to distribute them transparently.

Table 19 – Sub-Attribute S.2

Sub-Attribute S.2	Software scalability
Execution Manager	The Execution Manager (EMS) performs load balancing to ensure the requested QoS.
Context Manager	The Context Manager has to be able to send and receive a large number of context updates.
SIP Server	The SIP Server has to handle session related signalling information and presence updates from a potentially large number of terminals.
Monitoring	Monitoring is used to provide information about the utilisation of resources, thereby providing the basis to counter scalability problems. On the other hand, the generated information flow could be a scalability problem.
OpVO Broker / SLA High Level Services	The scalability of components involved in SLA negotiation clearly depends on the number of involved services.
Participant Registry	With an increase of VO members the Participant Registry has to handle more requests.
VO Security Services	To be able to access VO services users have to be authorized by the VO Security services.

4.1.2.3. Network Scalability

Sub-Attribute S.2 – Network scalability is basically related to the amount of traffic produced by the users or the services that the users launch and is obviously directly proportional to the number of users to whom services need to be granted.

Table 20 – Sub-Attribute S.3

Sub-Attribute S.3	Network scalability
A4C	The diameter protocol used for A4C is designed to be scalable.
Access Network-Access Router	A high traffic load is supposed to be palliated by the accommodating traffic within the QoS Broker. Other than that it is just a hardware scalability problem.
QoS Broker	The QoS Broker is scalable as long as Access networks are balanced or split if they grow too much.
Network Policy Manager (PBNM)	The Network Policy Manager is designed precisely to solve the drawbacks of growth, therefore making the network more scalable. If the growth is too big, then it becomes a matter of hardware and splitting networks.
Core Network	The Core Network is considered to have a surplus of resources, especially compared to the Access Network and therefore no scalability issues are envisaged.

4.1.3. Security

In the following we will provide tables that relate to the components of several architectural building blocks linked to security attributes.

4.1.3.1. Identification

Table 21 Sub-Attribute SI.1

Sub-Attribute SI.1	Secure Identification in the A4C Server
Mobile Terminal	Users have to identify with the A4C Server when they connect to the network.
VO Security Services	The users have to identify with the VO Authentication Service (which is part of the VO Security Services) by providing the tokens issued by the A4C Server.

4.1.3.2. Authentication

Table 22 Sub-Attribute SA.1

Sub-Attribute SA.1	Secure Authentication in the A4C Server
Mobile Terminal	After the user has been identified, A4C has to verify the identification of the user.

Sub-Attribute SA.1	Secure Authentication in the A4C Server
VO Security Services	The VO Authentication Service can check the user identity by contacting the A4C and verifying the tokens provided by the user.

4.1.3.3. Authorization

Table 23 Sub-Attribute SZ.1

Sub-Attribute SZ.1	Secure Authorization in the A4C Server
Mobile Terminal	The A4C Server authorises the terminal to use the network after being authenticated.
Mobile Terminal	In order to authorize the user access to another network a foreign A4C server can contact the user's A4C server for authorization purposes.

Table 24 Sub-Attribute SZ.2

Sub-Attribute SZ.2	Secure Authorization by the VO Authorization Service
Mobile Terminal	After the mobile terminal is authorised to use the network, the user contacts the VO Authorisation Service in order to access VO services. The VO authorisation service can query identity attributes from the A4C server.

4.1.3.4. Integrity

Table 25 Sub-Attribute SY.1

Sub-Attribute SY.1	Integrity in the Data Management System
Reliable File Transfer	Given two URLs the Reliable File Transfer service can move data between two locations.
Data Replication	Data replication is concerned with location and replication of data and meta data.

Table 26 Sub-Attribute SI.1

Sub-Attribute SY.2	Integrity in the Messaging System
Security Services	Integrity of messaging is guaranteed by signing the messages and using SSL communication channels.

Table 27 Sub-Attribute SI.1

Sub-Attribute SY.3	Integrity in the Programs
Security Services	Integrity in programs consists of ensuring that programs work correctly with unexpected actions, not enabling hacker and cracker attacks.

4.1.3.5. Intrusion Detection

Table 28 Sub-Attribute SI.1

Sub-Attribute SD.1	Security Intrusion Detection in the A4C Server
A4C	Intrusion detection is avoided by the verification of identity in the authentication process, by logging the failure authentication requests.
A4C	When the user receives the token, this token can only be used by the mobile terminal. If an intruder steals the token and then uses it, the A4C detects the intrusion by checking the token.

5. Evaluation scenarios

5.1. Introduction

In this section, some scenarios are described for each sub-attribute identified within section 3 of this document. In section 4 all attributes of interest were divided into sub-attributes for a more concrete and defined architecture evaluation process. Here, for each identified sub-attribute, some real scenarios are being identified with the main purpose of having use cases. Doing this, a qualitative evaluation of the architecture will be performed, in such a way that it will be possible to understand if the architecture is designed to provide the requested functionalities with the identified requirements without having real tests. For each sub-attribute, one table with two columns and various rows is built. Each row represents a scenario (use case), the column to the left contains the code for it and the column to the right contains its description.

5.2. Identified scenarios

5.2.1. Interoperability scenarios

Table 29 Scenarios related to sub-attribute I.1

Sub-Attribute	I.1
I.1.1[7]	The network layer transfers data to the grid layer that will receive them, it is able to understand and elaborate them to provide specific grid functionalities
I.1.2	As I.1.1 but from grid layer to network layer.
I.1.3	The network layer invokes the grid layer to start actions available here (e.g. a simulation service)
I.1.4	As I.1.3 but from grid to network layer (e.g. start a SIP session)

[7] Notation X.x.y: X.x refers to the sub-attribute (I.1 first sub-attribute of interoperability); y refers to the scenario (I.1.1 is the first scenario of sub attribute I.1)

Table 30 Scenarios related to sub-attribute I.2

Sub-Attribute	I.2
I.2.1	Service "A"[8] invokes a method on Service "B"[9]. It receives the results of invocation and uses the return value for further elaboration
I.2.2	As I.2.1 but from service "B" to service "A".
I.2.3	Service A is managed by management services of middleware B (e.g. manageability information is understandable by middleware B)
I.2.4	As I.2.3 but service B is the managed service

Table 31 Scenarios related to sub-attribute I.3

Sub-Attribute	I.3[10]
I.3.1	Modules of grid layer invoke the mobile device to receive data that will be matched with recorded data about the owner of mobile device
I.3.2	Mobile user authenticates against the VO and it is able to contact administration functions (e.g. to create a new VO member)
I.3.3	Mobile user is able to instantiate services inside the Grid, hopefully downloading some software that allows to perform some elaboration locally

Table 32 Scenarios related to sub-attribute I.4

Sub-Attribute	I.4
I.4.1	Components of the individuated test beds have to communicate with the Akogrimo infrastructure to achieve this communication, we need to understand if it is necessary to: • Modify existing interfaces • Modify existing software components • Modify test bed application logic

[8] Built on middleware A

[9] Built on middleware B

[10] Scenarios about mobile device have been taken from our validation scenarios (D2.3.2)

5.2.2. Scalability scenarios

With respect to the S.1 sub-attribute we will focus on scenarios that have the aim of evaluating the capability of the EMS module (D4.3.1 [8]). These scenarios are listed below in Table 33

Table 33 Scenarios related to sub-attribute S.1

Sub-Attribute	S.1
S.1.1	Add a new server machine and distribute on it some services already deployed on the available machines.
S.1.2	Remove a machine with specific services and distribute them on the machine available after removing phase. Change references to reach moved service. In this case with a minor number of machines will be coved same functionalities. This evaluation can be tested in local and distributed environment.
S.1.3	On the base of previous scenario (S.1.1) the scenario aims to add machines with different hardware features (CPU, memory, disk space, etc).
S.1.4	On the base of previous scenario (S.1.2), to remove machines with different hardware features (CPU, memory, disk space) and that host particular services.

Table 34 Scenarios related to sub-attribute S.2

Sub-Attribute	S.2
S.2.1	The EMS adds a specific service (business or core) that is already deployed in the start up configuration
S.2.2	The EMS removes a specific service (business or core) that is already deployed in the start up configuration
S.2.3	The EMS moves a specific service (business or core) that is already deployed in the start up configuration on a different machine
S.2.4	The Registry module allows the storage of increasing numbers of services inside it, and to also manage an increasing numbers of service discovery request connections
S.2.5	Growth of VO participants and their role change during their lifetime in the VO itself

Table 35 Scenarios related to sub-attribute S.3

Sub-Attribute	S.3
S.3.1	Increase number of users that are able to access to a specific service using the same access point
S.3.2	Increase the number of users that access simultaneous to the same resource

5.2.3. Reliability scenarios

This section will be written in the following update of this deliverable because reliability deals with quantitative measurement, therefore we will not define usage scenarios but here we will include metrics to be evaluated on the first version of implemented prototype

5.2.4. Availability scenarios

Table 36: Scenarios related to sub-attribute A.1

Sub-Attribute	A.1
A.1.1	The key service doesn't work properly (or it is unavailable) due to reliability problems of the service itself or due to a failure of the machine/device where the service is installed.
A.1.2	The key service is not available during its invocation due to a network problem with the original network operator the user has connected to.

Table 37: Scenarios related to sub-attribute A.2

Sub-Attribute	A.2
A.2.1	A new service is available in the VO: some services could be updated and this work could affect the availability of involved services (e.g. because they need to be restarted)
A.2.2	New policies could be introduced to access VO resources: some services could be updated and this work could affect their availability
A.2.3	A new user joins the VO: some services could be updated and this work could affect their availability
A.2.4	Maintenance of the Hardware or Software component and how it affects the user service. How affects maintenance (backup of user profiles) to the service.

Table 38: Scenarios related to sub-attribute A.3

Sub-Attribute	A.3
A.3.1	A networking link is cut. Service is supported by backup lines or a new routing path.
A.3.2	The user profile database is unavailable. Mirror database support of incoming user authentications without lost of service.
A.3.3	A call between a patient and a doctor is lost because a power problem in part of the network. System is recovered by using alternative paths/systems

Table 39: Scenarios related to sub-attribute A.4

Sub-Attribute	A.4
A.4.1	A hardware component (eg Database) is down. Networking Monitoring tool detect the error, setup an alternative hardware component, modification requests are to be attended by this block and the maintenance team would be informed in order to solve the problem.

Table 40: Scenarios related to sub-attribute A.5

Sub-Attribute	A.5
A.5.1	A new SIP server is installed in the system. The transition from the old SIP server to the new one will be done without disruption of the service. New SIP requests will be managed by new entity while previous request will be managed by the old one. Transition should be done smoothly.

5.2.5. Performance scenarios

Table 41: Scenarios related to sub-attribute P.1

Sub-Attribute	P.1
P.1.1	This scenario implies the specification and setup of Akogrimo infrastructure to support specific service and usage conditions.
P.1.2	Specification of each hardware component in terms of CPU, Memory and Disk will be provided
P.1.3	Implications of this specification in terms of maximum users/request/responses as well as delay/latency and throughput provided

Table 42: Scenarios related to sub-attribute P.2

Sub-Attribute	P.2
P.2.1	Based on previous scenario, identification of bottlenecks will be done.
P.2.2	For each bottleneck (hardware/software/network) a limitation in terms of users/request/response will be provided for each Akogrimo service to avoid degradation of the service to the current users.

Table 43: Scenarios related to sub-attribute P.3

Sub-Attribute	P.2
P.3.1	Policy Block will control the status of the system.
P.3.2	In case of a number of users/request/response upper to the threshold the policy will deny and limit new users/request/responses in the system to avoid a degradation of the service.

Table 44: Scenarios related to sub-attribute P.4

Sub-Attribute	P.2
P.4.1	If a latency of some process reaches an upper limit, alternative blocks will be setup to avoid degradations. If the delay to authenticate in the system due to the wait in server queue is up to a threshold a mirror system will collect new requests thus decreasing latency and response times.
P.4.2	In case the network suffers from overload in some links/paths, alternative routing will be provided to reduce latency.

Table 45: Scenarios related to sub-attribute P.5

Sub-Attribute	P.2
P.5.1	A patient tried to contact an Urgency Service by using Akogrimo. After initial contact with an agent, a VO is established and the system tries to put it in contact with one doctor. In case the call will be delayed by more that 15 seconds (due to unavailability of the doctor, etc) the agent will recover the call and talk with the patient.
P.5.2	A student is trying to perform a simulation over a Grid platform supported by Akogrimo. He will be informed about the expected time to deliver the output of the simulation.

Table 46: Scenarios related to sub-attribute P.6

Sub-Attribute	P.2
P.6.1	Depending on the service load, we will characterize service performance (latency, response delay and throughput) considering low load (50% of normal conditions), normal conditions and extreme conditions (next to the upper threshold) per service offer by Akogrimo platform.

5.2.6. Security scenarios

5.2.6.1. Security Identification scenarios

Table 47 Scenarios related to sub-attribute SI.1

Sub-Attribute	SI.1
SI.1.1	A valid user tries to identify with the A4C Server
SI.1.2	An invalid user tries to identify with the A4C Server

Table 48 Scenarios related to sub-attribute SI.2

Sub-Attribute	SI.2
SI.2.1	A valid participant tries to identify with the BVO Manager
SI.2.2	An invalid participant tries to identify with the BVO Manager

5.2.6.2. Security Authentication scenarios

Table 49 Scenarios related to sub-attribute SA.1

Sub-Attribute	SA.1
SA.1.1	A valid identified user tries to authenticate in the A4C Server
SA.1.2	An invalid identified user tries to authenticate in the A4C Server

Table 50 Scenarios related to sub-attribute SA.2

Sub-Attribute	SA.2
SA.2.1	The BVO Manager verifies the tokens provided by a valid identified user
SA.2.2	The BVO Manager verifies the tokens provided by an invalid identified user

5.2.6.3. Security Authorization scenarios

Table 51 Scenarios related to sub-attribute SZ.1

Sub-Attribute	SZ.1
SZ.1.1	A valid authenticated user tries to be authorized in the A4C Server
SZ.1.2	An invalid authenticated user tries to be authorized in the A4C Server

Table 52 Scenarios related to sub-attribute SZ.2

Sub-Attribute	SZ.2
SZ.2.1	An authorized user contacts the VO Authorization Service in order to access VO services
SZ.2.2	An authorized user contacts the VO Authorization Service in order to access forbidden VO services for his profile
SZ.2.3	A valid authorized user launches multiple requests for an authorized VO service (Denial of Service Attack)

5.2.6.4. Security Integrity scenarios

Table 53 Scenarios related to sub-attribute SY.1

Sub-Attribute	SY.1
SY.1.1	Given two URLs the Reliable File Transfer service moves data between two locations
SY.1.2	A copy of a sending file is stored locally until this file reach its destination
SY.1.3	A received file is duplicated and distributed immediately
SY.1.4	All data are duplicated and distributed

Table 54 Scenarios related to sub-attribute SY.2

Sub-Attribute	SY.2
SY.2.1	All messages are signed
SY.2.2	All communication channels are SSL
SY.2.3	The sent messages have acknowledge of receipt
SY.2.4	There are methods for resend messages
SY.2.5	There are methods for request the sending of the message again

Table 55 Scenarios related to sub-attribute SY.3

Sub-Attribute	SY.3
SY.3.1	A malicious program attacks some critical routines and programs
SY.3.2	All the programs are replicated or distributed
SY.3.3	There are methods for detecting a correct work execution
SY.3.4	There are methods for reassign a failed work
SY.3.5	The language chosen for coding is type-safe

5.2.6.5. Security Intrusion Detection scenarios

Table 56 - Scenarios related to sub-attribute SD.1

Sub-Attribute	SD.1
SD.1.1	An invalid identity tries to authenticate in the A4C Server and this intrusion is detected, recorded and notified
SD.1.2	An intruder steals a valid token and tries to use it. The A4C Server detects this intrusion by checking the token and notifies it to the security

6. Conclusions

Any software system, and Akogrimo in particular, needs an architecture evaluation process in the sense that the system, at architecture level - not at execution level - must comply with certain functionalities and requirements (in the case of Akogrimo, identified in other Akogrimo documents including deliverables D2.3.2 [6] and D3.1.1 [1]). After examining different methods of software architecture evaluation, the method chosen for this evaluation is predictive and scenario based. This method is capable of evaluating the non-functional goals of the system before the system is built. It has been specially noted that the main objective of this document is a non-functional or qualitative evaluation of the architecture, that is, to evaluate if the architecture itself is able to comply with the requirements for each attribute identified, developing scenarios or use cases for this purpose and before any real test case takes place.

This deliverable provides a complete set of use cases that will inform about the Interoperability, Scalability, Reliability, Availability, Performance and Security of the Akogrimo architecture. In a subsequent phase (deliverable D5.3.2), the evaluation team will perform the evaluation process, analysing the different scenarios to decide if the architecture complies with the requirements for each attribute. In this phase, perhaps, it will be necessary to add some quantitative use cases for a complete evaluation of the prototype. Once this evaluation is complete, a set of risks will be identified at architecture level (nothing to do with a poor performance of the system), and a set of corrections should be provided to improve it, with the main purpose of having an architecture that is not going to limit improvements in performance and the possible enhancement of the Akogrimo prototype.

References

[1] Akogrimo Deliverable 3.1.1 – Overall architecture definition and layer integration

[2] Akogrimo Description of Work

[3] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*. Reading, MA: Addison Wesley, 1998.

[4] M. Shaw and D. Garlan. Software Architecture. Perspectives on an Emerging Discipline. Prentice-Hall, 1996.

[5] Fisher, M. "Software Architecture Awareness and Training for Software Practitioners", CECOM Course, June 98

[6] Akogrimo Deliverable 2.3.2 – Validation Scenarios

[7] Mugurel T. Ionita, Dieter K. Hammer, Henk Obbink, "Scenario-Based Software Architecture Evaluation Methods: an overview".pdf

[8] Akogrimo deliverables - D4.1.1 "Consolidated Network Layer Architecture"; D.4.2.1 "Overall Network Middleware Requirements report"; D4.3.1 "Architecture of the infrastructure services layer"; D4.4.1 "Architecture of the Application Support Services Layer"

[9] Victor R. Basili Gianluigi Caldiera H. Dieter Rombach "The Goal Question Metric approach"

Annex A. Non functional requirements table

The following table is extracted from [6] and it is added here because that deliverable is not public. Each row of the table defines a non functional requirement that should be addressed by the Akogrimo infrastructure.

Table 57 Table from D2.3.2 section 5.1.2.15

Requirement Name	Requirement Description
Interoperability	Ability to work with other systems
Scalability	How easy is to expand the infrastructure
Security	Being free from danger
Identification	Evidence of identity
Authentication	Verification of the identity
Authorisation	Grant rights for accessing data or using resources
Integrity	Data has not been altered or destroyed without authorisation
Intrusion detection	Detecting, recording and intrusion notification
Availability	Percentage of the time one system is in productive work
Performance	Efficiency of a system while performing tasks
Latency	Now long, on the average, it takes to get a response from something
Response Time	Elapsed time between the end of an inquiry and the beginning of the response
Throughput	The amount of I/O requests that can be completed within a given amount of time
Reliability	Performance maintenance under stated conditions for a period of time