4CaaSt

Building the PaaS Cloud of the Future

Use Case Marketplace for Mass market: Report on Experimentation

D8.2.5

Version 1.0

WP8 - Experimentation

Dissemination Level: Public

Lead Editor: Stéphane Carrié / FT

10/10/2012

Status: Final

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 258862

SEVENTH FRAMEWORK PROGRAMME

Information Society

Seventh Framework Programme

FP7-ICT-2009-5

Service and Software Architectures, Infrastructures and Engineering

Contributors:

Miguel Jimenez, UPM

Nico Kruber, ZIB

Boris Moltchanov, TI

Charles Souillard, Bonita

Michel Dao, FT

Pablo Arozarena, TI

Internal Reviewer(s):

Anna Gatzioura, NTUA

Charles Souillard, Bonita

Version History

Version	Date	Authors	Sections Affected
0.1	27/06/2012	Michel Dao (FT)	Initial version with Table of Contents
0.2	06/07/2012	Michel Dao (FT)	Guidelines sections removed
0.3	5/09/2012	Stéphane Carrié(FT)	
0.4	7/09/2012	Miguel Jimenez (UPM), Boris Moltchanov (TI), Nico Kruber (ZIB), Charles Souillard (Bonita)	Consolidation of report on use case sections
0.5	14/09/2012	Miguel Jimenez (UPM), Boris Moltchanov (TI), Nico Kruber (ZIB), (Bonita)	Consolidation of evaluation as developer of the prototype
0.6	26/09/2012	Stéphane Carrié (FT)	Deliverable alignment with D8.1.7 and D8.3.7.
0.7	9/10/2012	Anna Gatzioura (NTUA)	Review corrections
1.0	10/10/2012	Stéphane Carrié (Ft)	Final version

Table of Contents

List of Tables

Abbreviations

4CaaSt	Building the PaaS Cloud of the future
AC	Application Component
ARB	Abstract Resolved Blueprint
API	Application Programming Interface
BPEL	Business Process Execution Language
CIF	Context Integration Framework
CMF	Context Management Framework
COTS	Component Of The Shelf
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
KPI	Key Performance Indicator
OVF	Open Virtualization Format
PaaS	Platform as a Service
PIC	Product Instance Component
REC	Runtime Execution Container
REST	Representational State Transfer
RP1 / RP2 / RP3	Reporting Period 1/2/3 of 4CaaSt project
SaaS	Software as a Service
SMS	Short Message Service
SOAP	Simple Object Access Protocol
VM	Virtual Machine
WAR	Web application Archive
WP	Work Package
WS	Web Service

1. Executive Summary

This deliverable is the first out of three deliverables reporting experimentation results of the 4CaaSt project and one out of three reports for each of the 4CaaSt use cases, each with a specific focus in terms of scenario and scenario stakeholder, as well as the 4CaaSt features evaluated. This report focuses on the deployment of the application with 4CaaSt components, a resource efficient operation and scalability. This first iteration aims to provide feedback to the technical work packages and further evolution of the use case to better use 4CaaSt features. Most of the results are experiences of working with the 4CaaSt components during the use cases implementation as well as initial external feedback. More detailed grey-box analysis is planned for the next iterations of the deliverables.

The development of custom software benefiting from 4CaaSt reusable services (context), 4CaaSt reusable components (Scalaris key/value datastore), and 4CaaSt “application” server (JSP with Jonas, process with Bonita, Mashup with Wireshark) has revealed no 4CaaSt specific overhead and is considered to be optimal.

Making an application 4CaaSt compliant does require providing specific descriptors (blueprint, Marketplace definition, price definition...) and specific glue code (metering, monitoring probe, SaaS customer management API) but complexity and training cost are low, while the amount of work is marginal.

Current use cases’ fulfilment is not always of 100% but the whole chain needs to be available for the scenario to be operational and being in RP2, there is often a piece of the chain missing. This is hopefully changing quickly as all components are being delivered integrated and in a usable state.

Current use cases do not cover the work needed to make a standard component 4CaaSt aware. Yet this is one promise of 4Caast to allow easy addition of a new application server. In order to reuse effort, we may consider taking as an example one of WP6 components (Scalaris or Wirecloud).

For the next iteration, as 4CaaSt features become available, we will finalize the whole process including component configuration, customer purchase and automated SaaS service provisioning, as well as automated scalability feature all very important for a mass market scenario.

Present deliverable follows this structure:

- Section 2 gathers results collected while running the 7 use cases on the integration platform,
- Section 3 formalizes and aggregates results from section 2 by providing evaluation criteria for every 4CaaSt usage phase
- Section 4 aggregates section 3 results further at a business goal level.

2. Evaluation overview (per use case)

This section provides an overview of the evaluation results with respect to the mass market use cases features introduced in the scenario description [2]. For each use case, the relevant innovations of the use case are evaluated according to the frameworks specified in the evaluation guidelines [1].

Note, that the scope of this evaluation is on what has been reported in the integration report [3]. Other 4CaaSt services (Database, Application Servers and the Network APIs) are constantly being integrated with the use case when they become available and will be reported in next releases of this document.

Many of the 4CaaSt features used by the mass market scenario are not evaluated individually, but they are the result of a combined evaluation, expert interview, questionnaire or experiment, the following tables map the use case features to a section within the document to ease the mapping of use cases, features and evaluation.

For each use case (2.1 to 2.7), we provide

- A short reminder on the use cases themselves, (2 .[1 to 7] .1)
- A mapping from features to phase in chapter 3 (2 .[1 to 7] .2)
- A list of evaluation questions with answers (2 .[1 to 7] .[3 to 7])
- A connection to 4CaaSt feature (2 .[1 to 7] .8)

2.1. UC.8-2-001 Develop cloud enabled software component

2.1.1. Use case reminder

➢ **At the beginning of this use case**, a software developer needs to develop a software component that implements a desired feature

➢ **At the end of this use case**, a software developer has a software component providing the desired feature

2.1.2. Feature mapping

Innovation / Feature Id & Name	Evaluation
F#WP6.02 Mashups catalogue	Develop phase – technology integration
F#WP6.03 Mashup platform	Develop phase – technology integration
F#WP6.04 Mashups as building blocks	Develop phase – technology integration
F#WP6.05 Location Context Provider	Develop phase – technology integration

F#WP6.06 Context Manager	Develop phase – technology integration
F#WP6.07 Context Consumer API	Develop phase – technology integration
F#WP6.09 Simple API for Data Store as a Service	Develop phase – technology integration
F#WP6.10 Standardised API for Data Store as a Service	Develop phase – technology integration
F#WP6.11 Integrated telco services	Develop phase – technology integration
F#WP7.08 Support for BPMN2 Correlation	Develop phase – technology integration
F#WP7.11 Application Server Deployment Features	Develop phase – technology integration
F#WP7.16 Built on standards	Develop phase – technology integration

Table 1. Feature report for UC8.2.001 – Develop cloud enabled …

2.1.3. What parts of the use case are fulfilled? Qualify the degree of fulfilment.

Original use case description from deliverable D8.2.1	Comment
(A software developer wants to develop an application such as “TOURISM demo application”. He would like to save on training, on coding, on debugging and as a result have a shorter time to market.) (He is able to reuse the following 4CaaSt features or services:)	-

Write “context query” in “context query language” to get access to smart context information based on available context sources (location, weather forecast, social network)	100% Context as a Service Enabler has been integrated into the 4CaaSt cloud by means of a dedicated blueprint and run for this demo. The necessary information including the tourist customers, their location and social relationships are provisioned and ready to be provided on request from the tourism demo application. Moreover also the simple security has been integrated such as authentication and authorisation of an application to perform context queries regarding the tourism customer’s context.
Write new web GUI components using Mashup platform	90% Every Mashup planned for the prototype has been successfully implemented. New workspace has been created updating previous widgets and adding newer Context aaS features and Pub/Sub exploitation.
Integrate Mashup components	80% Integration of external components as Mashup has finished in the case of the Wikipedia Renderer, but is being finished in the case of the ticket shop.
Write an html page generator in Java for computing Wikipedia pages from scalable key/value storage.	90% We were able to write the page generator using standard java technology and the Java-API of Scalaris. Automatic configuration of connection parameters (to a Scalaris server) needs to be added. Similarly for the monitoring and accounting configuration. Different data models for the wiki pages are currently under review. They try to overcome certain restrictions of a simple key/value store compared to a RDBMS while keeping the inherent scalability of the data store. Finally, original use case text should be reformulated: *Write an html page generator in Java for rendering Wikipedia pages stored in a scalable key/value storage.*
Store data in scalable key/value storage using a standard java storage mechanism developer are already accustomed.	0% - RP3 The JPA connector for SCALARIS is under development at time of writing and is not in a testable state. Delivery is planned for RP3

Table 2. Use case fulfilment for UC8.2.001 – Develop cloud enabled …

2.1.4. What is needed from 4CaaSt to increase the fulfilment?

Issues regarding development environment and testing would be interesting but are not in the scope of 4CaaSt. So given current project scope, no additional feature is required from 4CaaSt to increase fulfilment.

2.1.5. What new requirements to 4CaaSt can be specified following this evaluation?

It is hard to reach both high flexibility and little to no customisation work for customer. Yet, when a customer requires a specificity he would like to specify and code only what is specific. 4CaaSt should clarify how a customer can add something specific while benefiting from existing presets.

2.1.6. What are the limitations to this use case with respect to evaluating 4CaaSt capabilities?

The challenge of deploying 4CaaSt cloud platform lies much in the complexity of the coherent configuration of multiple components. Present use cases are quite vague relative to application configuration requirements. Real world applications generally have a lot of configuration (http configuration, reverse proxy, server tweaking ...). Next iteration may consider introducing more complex configuration scenario.

The scenario shall be extended to include a staging phase in the creation of the component. A customer might want to deploy and test his component and blueprints in a staging area before going into production.

The scenario shall be also extended to include the creation of custom server component. This is listed as a feature of 4CaaSt and it is not evaluated. This could be performed by including the work done in work package 6 to turn standard components into 4CaaSt compliant components.

The scenario shall be extended to include more detailed application configuration requirements.

2.1.7. What improvements to the application prototype are needed to fully exercise this use case?

In order to evaluate the work of turning a “Component off the Shelf” (COTS) into a 4CaaSt compliant component, we have to include such a component in the scope of the prototype. This could be simply done by considering that Scalaris, for example, is not part of the 4CaaSt platform, but part of the mass market prototype. The work performed in WP6 to turn this “COTS” into a 4CaaSt compliant component could then be evaluated in this document from a business perspective.

2.2. UC.8-2.002 Deploy Software in the marketplace for commercialization

2.2.1. Use case reminder

➢ **At the beginning of this use case**, a software developer has a developed application. (Development is not fully in the scope of 4CaaSt).

➢ **At the end of this use case**, the software is published in the marketplace and is ready to be commercialised, meaning that no subsequent development action is needed.

2.2.2. Feature mapping

Innovation / Feature Id & Name	Evaluation
F#WP2.01 Support the design of a cloud enabled solution	Develop Phase – Blueprinting
F#WP2.02 Empower cloud Developers	Develop Phase – Blueprinting

Table 3. Feature report for UC8.2.002 – Deploy service …

2.2.3. What parts of the use case are fulfilled? Qualify the degree of fulfilment.

Original use case description from deliverable D8.2.1	Description of fulfilment
(A software developer has developed an application such as “TOURISM demo application”.) (The logic is developed in his favourite language, chosen among those supported by 4CaaSt.)	-

He specifies what his requirements are regarding runtime platform using 4CaaSt supported descriptor (specific versions or range, specific editor, etc.).	80% For front-end and wiki page renders, dependencies could be defined in the blueprint descriptor. Providing the value of certain field was not easy, such as the minimum and maximum number of instances. Wirecloud can be defined as a requirement of the application, and it can be provided by Mashup-as-a-Service component in two flavours, multi-tenant service or deployable component, although only multi-tenant version is fully available. It is unclear whether the artifact section simply serves documentation purposes or whether it is integrated with deployment (and therefore needs some more scripting magic).
He provides the numerous binaries or source of his application in the 4CaaSt supported format.	100% 4CaaSt provides a way to declare URL of artefact in the blueprint descriptor. The wiki page renderer was bundled in a .war file and is thus available to 4CaaSt.
He provides 4CaaSt descriptor specifying the desired architecture of his application and how his components are interconnected.	Front end : 80% We were able to provide descriptor, but since the whole process is not available, it may contain unforeseen bugs or flaws. Wiki page renderer : 80% We were able to provide descriptor, but since the whole process is not available, it may contain unforeseen issues. The endpoint location field is useless at the moment, a real URI cannot be given. Context: 80% We were able to provide descriptor but it is not easy to understand how to treat multi-tenancy for a service. Also, it is problematic to understand how to integrate the offering section (pricing model, pricing, etc.) and consequently charging for an always on and running Service Enablers;. Bonita: 80% We were able to provide descriptor but there is nothing to describe which technology is used to make components be able to communicate (e.g. database protocol). All: Blueprint is missing fields for documentation purposes. We had to insert XML comment in manually edited blueprint to account for this requirement.
He provides information about dependencies toward external services.	50% The prototype does require external services (reverse Wikipedia lookup). It is used in code but not defined in blueprint.

He provides dependencies towards 4CaaSt built in services such as key/value data store, context or any other service.	80% The dependency of the wiki page render toward the Scalaris data store has been defined in the blueprint.
He provides meta-data about application specific meters that 4CaaSt needs to be made aware of.	This section should be been moved to 8-2.007 Enforce metering use case. It is not a required step for this use case.
He provides meta-data about application specific KPIs, that 4CaaSt needs to be made aware of. He provides meta-data information about how his application nodes can be scaled up or not.	0% (This element should be moved to UC8.2-006 Enforce SLA use case)
If necessary he provides 4CaaSt optional information such as :	-
(This was Line added during evaluation. We may need to extend existing scenario) • Required specific scripts for PaaS automation.	We had not delivered chef script at the time of evaluation. Yet, learning Chef is not trivial and is not a matter of minutes. Training costs must be taken into account.
• Self diagnostic and monitoring logic or meta-data.	This is not needed in this base use case and is anyway covered in use cases 6 and 7. Deletion of this item may be considered for next iteration of the use case.
• Upgrade or downgrade logic (data schema modifications, etc.),	This feature is out of the scope of 4CaaSt although it would be useful in a real life platform.
• Specific initialization logic (initial data set, etc.),	30% 4CaaSt may support such a feature by declaring an artefact in a blueprint and have it processed by a specific initialization script. It will be implemented in RP3 for this scenario.

Table 4. Use case fulfilment for UC8.2.002 – Deploy service …

2.2.4. What is needed from 4CaaSt to increase the fulfilment?

Although blueprint structure is not very complex and can be understood quite easily, editing a blueprint is a tedious task. A Blueprint contains many long identifiers that are used for cross referencing sections of the blueprint. Manually editing this information, even with a XML editor, is a painful process as blueprints are used for application representative of real world.

4CaaSt provides a model for defining dependencies between components (blueprint).

Manual blueprint edition is a painful and error prone process as many configuration elements are identified by strings which need to be copied throughout the document. Also, the logical dependency which is a graph would be easier to understand if displayed graphically.

4CaaSt does not provide built-in mechanisms for supporting application initialization logic, upgrade or downgrade logic.

2.2.5. What new requirements to 4CaaSt can be specified following this evaluation?

Provide a blueprint editor with built-in support for extensions.

Some partners identified the need for inserting comments in the blueprint. Merely stating that a component requires another component is not always sufficient for human understanding.

Blueprinting allows creation of dependencies between nodes. Yet, the semantic of the dependency is not modelled. For example if a component would need two identical databases for different purposes, we would not be able to model which is which.

2.2.6. What are the limitations to this use case with respect to evaluating 4CaaSt capabilities?

4CaaSt consortium decided that component deployment would be demonstrated in 8.1 and 8.3 and that 8.2 would concentrate on Software as a service deployment. Thus our use case does not demonstrate these aspects. It is a limitation by choice.

The scope of the current use case is currently sufficiently ambitious and challenging for 4CaaSt platform.

2.2.7. What improvements to the application prototype are needed to fully exercise this use case?

None.

2.3. UC.8-2.003 Commercialize Service provided by software

2.3.1. Use case reminder

- **At the beginning of this use case,** software implementing services have been published by a software provider in the marketplace but are not available for purchase.
- **At the end of this use case,** the service is available for purchase.

2.3.2. Feature mapping

Innovation / Feature Id & Name	Evaluation
F#WP3.01 Product Definition	Market Phase -- Marketplace integration process
F#WP3.10 Product Specification	Market Phase -- Marketplace integration process
F#WP3.22 Business Management	Market Phase -- Marketplace integration process

Table 5. Feature report for UC8.2.003 – Commercialize …

2.3.3. What parts of the use case are fulfilled? Qualify the degree of fulfilment.

Original use case description from deliverable D8.2.1	Description of fulfilment
Service provider selects software in the marketplace that has been put there by a software provider.	100% Blueprint of TOURISM service was published in the blueprint repository and marketplace was able to access and display it for selection.
Service provider proceeds to the definition of commercial information about the service and definition of commercial conditions under which the software may be purchased. In the case of the “TOURISM demo application”, a flat monthly subscription fee is defined (Please refer to enforce metering for more advanced price consideration).	100% This price model could be specified. (It should be noted that it is a trivial price model without hard requirement from the scenario because 8.2-007 use case is dedicated to more complex real world price model.)

Service provider marks service as available for purchase so the application becomes visible to service user.	100% Feature is available.

Table 6. Use case fulfilment for UC8.2.003 – Commercialize …

2.3.4. What is needed from 4CaaSt to increase the fulfilment?

Current 4CaaSt fulfils this use case.

2.3.5. What new requirements to 4CaaSt can be specified following this evaluation?

There is no new requirement within the scope of this use case.

2.3.6. What are the limitations to this use case with respect to evaluating 4CaaSt capabilities?

This scenario was designed to cover simple nominal case and is not meant to be extended to cover advanced cases.

2.3.7. What improvements to the application prototype are needed to fully exercise this use case?

Current prototype is sufficient for this use case.

2.4. UC.8-2-004 Choose service in the marketplace

2.4.1. Use case reminder

➢ **At the beginning of this use case,** a customer is looking for a service matching his needs and expectations.

➢ **At the end of this use case,** a customer has found a service that fulfils his needs and expectations.

2.4.2. Feature mapping

Innovation / Feature Id & Name	Evaluation
F#WP3.02 Product Search	Market Phase -- Marketplace integration process
F#WP3.04 Related Products	Market Phase -- Marketplace integration process
F#WP3.05 Recommendation	Market Phase -- Marketplace integration process
F#WP3.06 Advertising	Market Phase -- Marketplace integration process
F#WP3.07 Community Rating & Comments	Market Phase -- Marketplace integration process
F#WP3.08 Social Graph Analysis	Market Phase -- Marketplace integration process
F#WP3.21 User Management	Market Phase -- Marketplace integration process

Table 7. Feature report for UC8.2.004 – Buy service from the marketplace

2.4.3. What parts of the use case are fulfilled? Qualify the degree of fulfilment.

Original use case description from deliverable D8.2.1	Description of fulfilment
A customer connects to the marketplace website.	100% Marketplace web application is online in Flexiant hosting facility.
Marketplace contains so many services that it is not practical to browse through all of them in order to find the right one.	10% Marketplace contains only a few services so we are not exactly in the situation of the scenario wheer it becomes a problem just to find a service.

He enters a set of criteria and lets the marketplace find matching services.	100% Marketplace provides a free text search which could be used successfully to find our TOURISM offering.
There are still a few services remaining. He looks at the first one. The marketplace displays additional information about the chosen service. There are also links to similar services. He decides to navigate to another service.	Feature not planned to be implemented by 4CaaSt but it is in state of the art so it does not need to be demonstrated.
This time he has found a service that could match his expectations. But he has never heard of this service and needs advices and reassurance. He decides to look at comments left on the site by previous customers. Some customer seems to have had a good experience with this service.	0% Feature is planned for RP3.
He is now reassured and decides that he will subscribe to the service.	/

Table 8. Use case fulfilment for UC8.2.004 – Buy service from the marketplace

2.4.4. What is needed from 4CaaSt to increase the fulfilment?

Nothing besides implementing features planned for RP3.

2.4.5. What new requirements to 4CaaSt can be specified following this evaluation?

There is no new requirement identified for the core features presented at RP2.

2.4.6. What are the limitations to this use case with respect to evaluating 4CaaSt capabilities?

Use cases could be more specific about what is expected from social web in the decision making. Renaming the use case “Support customer purchase decision making” would help focus the attention on important benefits.

2.4.7. What improvements to the application prototype are needed to fully exercise this use case?

We have not identified need for improving prototype.

2.5. UC.8-2-005 Buy service from the marketplace

2.5.1. Use case reminder

➢ **At the beginning of this use case,** software implementing services have been published by a software provider in the marketplace but are not available for purchase.

➢ **At the end of this use case,** service provided by the software can be purchased from the marketplace and users can use it.

2.5.2. Feature mapping

Innovation / Feature Id & Name	Evaluation
F#WP3.09 Product Resolution	Contract phase – Service contracting process
F#WP3.11 Product Customization	Contract phase – Service contracting process
F#WP3.13 Basket Management	Contract phase – Service contracting process
F#WP3.14 Contract Management	Contract phase – Service contracting process
F#WP3.15 Delivery Support	Contract phase – Service contracting process
F#WP3.16 Payment Support	Contract phase – Service contracting process
F#WP3.17 Pricing & Charging	Contract phase – Service contracting process
F#WP3.21 User Management	Contract phase – Service contracting process
F#WP5.01 PIC Administration	Deploy Phase – Deployment & provisioning of a service
F#WP5.02 Monitoring infrastructure: Set up and configure probes	Deploy Phase – Deployment & provisioning of a service
F#WP5.03 Monitoring infrastructure: product and platform monitoring	Deploy Phase – Deployment & provisioning of a service
F#WP7.18 Monitoring capabilities (solution specific)	Deploy Phase – Deployment & provisioning of a service

Table 9. Feature report for UC8.2.005 – Buy service from the marketplace

2.5.3. What parts of the use case are fulfilled? Qualify the degree of fulfilment.

Original use case description from deliverable D8.2.1	Description of fulfilment
Customer is on the marketplace and has decided to buy a service. He is on a page displaying the service.	100% A button to purchase the service is available.
He has selected a service but there are still configurable items such as sub-services used or price plan.	0% Current scenario does not require configuration of sub-services at time of purchase nor does that platform provides integrated support for this feature.
Marketplace provides hints about what would be the best priced combination for him.	This feature is demonstrated in 8.1 for service provider. We have not identified need to demonstrate it for end users as well.
Customer decides based on this information and proceeds to subscription.	/
Marketplace proceeds to service provisioning.	70% Unlike scenarios 8.1 and 8.3, this scenario does not deploy a software instance per customer but grants user access to a SaaS. Provisioning of the service is done via the SaaS provisioning API, and the user is granted (or removed) access to the service, using the 4CaaSt ID of that contract for charging individual concepts to the user. “Service provisioning” should be renamed “SaaS provisioning” in next iteration of the scenario.
Customer is informed of delivery evolution.	0%
Customer gets access to the service.	50% Service is manually deployed and can be used.
Customer receives a bill and gets charged.	0%

Table 10. Use case fulfilment for UC8.2.005 – Buy service from the marketplace

2.5.4. What is needed from 4CaaSt to increase the fulfilment?

Delivery of features planned for RP3 will increase fulfilment.

2.5.5. What new requirements to 4CaaSt can be specified following this evaluation?

No requirement.

2.5.6. What are the limitations to this use case with respect to evaluating 4CaaSt capabilities?

Billing of sub-services is not very well defined in the scenario. It shall be possible to reuse sub-contracts based on

2.5.7. What improvements to the application prototype are needed to fully exercise this use case?

Configuration

2.6. UC.8-2-006 Enforce SLA

2.6.1. Use case reminder

- **At the beginning of this use case,** a service is available in the marketplace but KPIs and SLAs have not been defined and are not enforced.
- **At the end of this use case,** KPIs are defined and measured and the SLA is defined and enforced.

2.6.2. How do features contribute to the use case fulfilment

Innovation / Feature Id & Name	Evaluation
F#WP3.01 Product Definition	Market Phase -- Marketplace integration process
F#WP3.10 Product Specification	Market Phase -- Marketplace integration process
F#WP3.11 Product Customization	Contract phase – Service contracting process
F#WP4.02 Automated Application Elasticity	
F#WP4.03 PaaS API	
F#WP5.02 Monitoring infrastructure: Set up and configure probes	Deploy Phase – Deployment & provisioning of a service
F#WP5.03 Monitoring infrastructure: product and platform monitoring	Deploy Phase – Deployment & provisioning of a service
F#WP6.08 Scalability of Data Store as a Service	
F#WP7.10 Right Sized and Incremental Application Server Platform	
F#WP7.11 Application Server Deployment Features	
F#WP7.18 Monitoring capabilities (solution specific)	Deploy Phase – Deployment & provisioning of a service
F#WP7.19 Horizontal scalability support	

Table 11. Feature report for UC8.2.006 – Enforce SLA

2.6.3. What parts of the use case are fulfilled? Qualify the degree of fulfilment.

Original use case description from deliverable D8.2.1	Description of fulfilment
Software developer is developing an application such as “TOURISM demo application”. It is decided what are the meaningful key performance indicators of the application. These indicators are business related and need some level of support from the application. As an example, it could be that a special back office process should run in less than 15 minutes, or that a page rendering should be less than so many millisecond or whatever.	100% 4CaaSt project has acknowledged that predefined probes are not sufficient and that custom metering must be supported by the whole 4CaaSt stack.
Software developer develops the KPI probe as he sees fits in the application. He develops the needed glue code so that the 4CaaSt platform can get informed of the value of the KPI.	80%, We were able to implement a custom KPI probe in the java wiki page renderer using JMX technology. It remains to be tested in a fully integrated process.
Software developer extends the application descriptor so that 4CaaSt knows that a KPI probe is provided by the application.	0%, At time of writing, 4CaaSt specification of how to define a KPI probe in a blueprint is not fully available. The current wiki renderer blueprint does not have a KPI defined yet.
The marketplace is automatically aware of the existence of the KPI.	0%, Planned RP3.
In the marketplace, service provider can specify SLA based on KPI constraints associated with a price plan. Some customer may choose stronger SLA constraints than others and thus be charged different prices.	0%, Planned RP3

Original use case description from deliverable D8.2.1	Description of fulfilment
When application is deployed in the marketplace, 4CaaSt is aware of the KPI and setups the environment so the KPI value will be picked up by the monitoring software.	0%, Planned RP3
4CaaSt is aware of the desired SLA, is able to monitor the KPI values and can detect if a SLA breach is happening or may happen in a near future.	0%, Planned RP3
4CaaSt takes appropriate actions to ensure SLA respect. 4CaaSt optimizes resource usage in order to reach SLA at a minimal cost. Minimal cost shall be proportional to load. In other word, if load doubles, cost is expected to double and if load is divided by a factor of 2, cost is expected to be divided by 2 as well.	0%, Planned RP3
4CaaSt scalability will be evaluated using the following scenario. First, We need to develop some additional code: Implement a basic load injector to simulate many parallel page loads on the “TOURISM demo application” web component, in order to simulate load on context, on J2EE server on SCALARIS key/value store.	0%, Planned RP3
Add a dummy load simulator in the page generator in java to ease load simulation on a J2EE server. It could be used to arbitrarily consume variable quantities of processing power or memory.	- (This feature is scenario specific and is not related to 4CaaSt.)

Original use case description from deliverable D8.2.1	Description of fulfilment
Extend Flexiant platform with a new fine grained resource usage report. This report would provide per virtual machine report on a given time interval and allow fine grain analysis of 4CaaSt resource consumption.	50%, This feature has been introduced in the roadmap of Flexiscale platform. A high end request interface is provided allowing this kind of feature. In the meantime, custom queries can be performed on request by Flexiant support team.
Then, a 3-steps test is performed. First a load factor of 100 will be applied to the platform and resource consumption will be measured (RC100). Then load factor will be lowered to 50 and resource consumption will be measured again (RC50) Finally, load factor will be raised back to 100 and resource consumption will be measured one last time (RC100back) Analysis of 4CaaSt platform resource consumption will provide insight of 4CaaSt scalability efficiency.	0%, Planned RP3

Table 12. Use case fulfilment for UC8.2.006 – Enforce SLA

2.6.4. What is needed from 4CaaSt to increase the fulfilment?

Delivery of feature planned for RP3 will increase fulfilment. Also, blueprint descriptor shall finalize work on a standard way to declare KPI shared by every 4CaaSt component.

2.6.5. What new requirements to 4CaaSt can be specified following this evaluation?

Interoperability of 4CaaSt cloud platform relies on the blueprint descriptor as well on required extensions such as KPI definition. Extensions shall be part of the standard 4CaaSt platform as well as the blueprint schema. These extensions shall also be supported by the blueprint editor to be provided.

While probes are defined in the use case, it is not clear yet how these probes shall be combined into KPI and lead to decisions on starting new server nodes. Also, automatic configuration of application node is not clearly defined.

2.6.6. What are the limitations to this use case with respect to evaluating 4CaaSt capabilities?

Current use case already challenges 4CaaSt platform and does not need to be extended.

2.6.7. What improvements to the application prototype are needed to fully exercise this use case?

We identified no required improvement for the application prototype.

2.7. UC.8-2.007 Enforce metering

Please note that due to the 4CaaSt platform being in early stage, this use case is very partially implemented and any form of conclusion would be preliminary.

2.7.1. Use case reminder

➢ **At the beginning of this use case,** a service is available in the marketplace without metering capabilities.

➢ **At the end of this use case,** a service is available in the marketplace and its price model includes business and/or technical metering data.

2.7.2. 4CaaSt feature mapping

Innovation / Feature Id & Name	Evaluation
F#WP3.01 Product Definition	Market Phase -- Marketplace integration process
F#WP3.10 Product Specification	Market Phase -- Marketplace integration process
F#WP3.11 Product Customization	
F#WP3.17 Pricing & Charging	Contract phase – Service contracting process
F#WP5.02 Monitoring infrastructure: Set up and configure probes	
F#WP5.04 Metering Capabilities	
F#WP7.18 Monitoring capabilities (solution specific)	

Table 13. Feature report for UC8.2.007 – Enforce metering

2.7.3. What parts of the use case are fulfilled? Qualify the degree of fulfilment.

Original use case description from deliverable D8.2.1	Description of fulfilment

Original use case description from deliverable D8.2.1	Description of fulfilment
Software developer is developing an application such as “TOURISM demo application”. 4CaaST marketplace is providing built-in meters. Although useful, these meters are not sufficient for charging a SaaS service.	100% 4CaaSt project has understood and acknowledged the need to support custom accounting meters.
It is decided that it is meaningful to charge customer based on the number of pages viewed but only those that contain more than a certain number of characters.	/ (introductory sentence)
Pages that are one-click away from a page charged in the last 48 hours are not charged. A page that has been displayed in the last 48 hour is not charged again.	4CaaSt does not provide any mechanism for supporting such custom complex scheme. Thus it would have to be implemented at application level. We will not implement such logic inside application as it does not demonstrate any feature of 4CaaSt.
This example meter, like many real life meters, requires applicative knowledge from the application, and thus must be implemented by the application providing the service. Software developer develops the meter probe as he sees fits in the application. He develops the needed glue code so that the 4CaaSt platform can get informed of the value of the meter.	90% An accounting plug-in for the Wikipedia renderer has been fully implemented and pushes an SDR to the accounting server for each page view. During deployment, the URL of the accounting server must be passed to the Wiki Servlet (untested); a tenant ID is provided in each call from the Mashup. Note: Replace “meter probe” by “accounting probe” in the description.
Software developer extends the application descriptor so that 4CaaSt knows that a meter is provided by the application.	0% Specification of how to describe an accounting probe was not available at time of evaluation, although it is available today.
The marketplace is automatically aware of the existence of the meter.	30% At time of writing, marketplace does not implement this feature but mechanisms are well understood and planned to be realized.

Original use case description from deliverable D8.2.1	Description of fulfilment
In the marketplace, service provider can specify price plan taking into account the value of the meter. Some customers may have a small or big plan and get different pricing scheme for usage of the application.	0% Marketplace currently uses a predefined list of parameter for price definition. Also, marketplace only supports one price model per blueprint.
When application is deployed in the marketplace, 4CaaSt is aware of the meter and setups the environment so the meter value will be picked up by the metering infrastructure.	0% By project choice, automatic application deployment is validated for RP2 in scenarios 8.1 and 8.3. Mass market scenario concentrates on SaaS deployment.

Table 14. Use case fulfilment for UC8.2.007 – Enforce metering

2.7.4. What is needed from 4CaaSt to increase the fulfilment?

There is no identified improvement on feature covered during RP2.

2.7.5. What new requirements to 4CaaSt can be specified following this evaluation?

There is no identified improvement on feature covered during RP2.

2.7.6. What are the limitations to this use case with respect to evaluating 4CaaSt capabilities?

There is no identified improvement on feature covered during RP2.

2.7.7. What improvements to the application prototype are needed to fully exercise this use case?

3. Evaluation results (per phase)

In this section, we provide an aggregated and formalized as criterion view of the results gathered during the evaluation of the use cases in the previous section.

Criteria and phase definitions are provided by the [1] Experimentation Guideline document.

3.1. Develop Phase – Blueprinting

3.1.1.1. Scalaris& J2EE based components

Criterion	**Questions**	**Overall evaluation --, -, 0, +, ++**	**Answers**
Learnability	How easy it was to grasp the blueprint model with respect to your needs?	Easiness of the blueprint model understanding.	++, very easy
	What aspects did you fail to understand?		It is unclear whether the artifact section simply serves documentation purposes or whether it is integrated with deployment (and therefore needs some more magic) – two philosophies exist: since chef recipes determine how to deploy/install PICs, there it seems only documentary, but the deployment of ACs uses generic scripts and thus needs the artifacts, e.g. a war file and some configuration. This is still unclear.
Effectiveness	Did you achieve to describe all the relevant elements of the application or technology?	Accuracy and completeness of the blueprint that you designed with respect to your application or technology.	Mostly yes, the endpoint location of an offering is still a bit unclear though, as it cannot be described without runtime information (see below)
	What aspects of blueprint should be added or improved to allow a complete description of the application or technology?		The endpoint location is useless at the moment, a real URI cannot be given, maybe a pseudo-URI like “http://<host>:8080”, additionally, in the case of a Servlet, its endpoint location depends on the Servlet Container, so it will be even more generic: “http://<container_location>/scalaris-wiki/” or similar

Efficiency of use	How straightforward was the mapping from the relevant elements of the application or technology to the blueprint model artifacts?	Effort put in designing the BP.	++
	What aspects of blueprint should be added or improved to make this mapping more straightforward?		None

3.1.1.2. *Mashup based components*

Criterion	**Questions**	**Overall evaluation --, -, 0, +, ++**	**Answers**
Learnability	How easy it was to grasp the blueprint model with respect to your needs?	Easiness of the blueprint model understanding.	+ Not much problem
	For what aspects did you need support to understand them fully?		Requirements of architecture and components
	What aspects did you fail to understand?		None
Effectiveness	Did you achieve to describe all the relevant elements of the application or technology?	Accuracy and completeness of the blueprint that you designed with respect to your application or technology.	Yes, as far as I know every requirement of our technologies could be described
Efficiency of use	How straightforward was the mapping from the relevant elements of the application or technology to the blueprint model artifacts?	Effort put in designing the BP.	Quite easy, once understood all the sections, translating it to our technologies was not a problem

Criterion	Questions	Overall evaluation **--, -, 0, +, ++**	Answers
	What aspects of blueprint should be added or improved to make this mapping more straightforward?		

3.1.1.3. Context related components

Criterion	Questions	Overall evaluation **--, -, 0, +, ++**	Answers
Learnability	How easy it was to grasp the blueprint model with respect to your needs?	Easiness of the blueprint model understanding.	++ It was very easy to prepare the service blueprint due to the CaaS Enabler simplicity and its ability to work as always run for any service or application. Nevertheless some additional features such as monitoring and charging are still to be handled and evaluated in further version of the blueprint and its integration into the experimentation.
	For what aspects did you need support to understand them fully?		+ It is not easy to understand how to treat multi-tenancy for a service, which is not multitenant itself but requires some multitenant-like provisioning per service/application or per customer.

Criterion	**Questions**	**Overall evaluation** **--, -, 0, +, ++**	**Answers**
	What aspects did you fail to understand?		- It is impossible to understand how to integrate the offering section (pricing model, pricing, etc.) and consequently charging for an always on and running Service Enablers, which are used by many different applications and customers in a pull (on request) and push (subscription) modes;
Effectiveness	Did you achieve to describe all the relevant elements of the application or technology?	Accuracy and completeness of the blueprint that you designed with respect to your application or technology.	0 All required functionality has been achieved based on the designed blueprint and deployed as the service.
	What aspects of blueprint should be added or improved to allow a complete description of the application or technology?		0 None for the moment.
Efficiency of use	How straightforward was the mapping from the relevant elements of the application or technology to the blueprint model artifacts?	Effort put in designing the BP.	+ Very short time, couple of hours for understanding what is where and what to leave and customize in the blueprint template.
	What aspects of blueprint should be added or improved to make this mapping more straightforward?		0 Being already provided with a blueprint compositor and validator, none for the moment.

3.1.1.4. *Bonita related components*

Criterion	Questions	Overall evaluation --, -, 0, +, ++	Answers
Learnability	How easy it was to grasp the blueprint model with respect to your needs?	Easiness of the blueprint model understanding.	+ It was understandable with some already done examples
	For what aspects did you need support to understand them fully?		None
	What aspects did you fail to understand?		Perimeter of components: If we provide a Tomcat in our bundle does it have to be described in the blueprint?
Effectiveness	Did you achieve to describe all the relevant elements of the application or technology?	Accuracy and completeness of the blueprint that you designed with respect to your application or technology.	0 There are missing concepts: What kind of service is provided by a component: e.g. user interaction using browser or URL to start a process
	What aspects of blueprint should be added or improved to allow a complete description of the application or technology?		There is nothing to describe which technology is used to make components be able to communicate. e.g. interact with database using database protocol
Efficiency of use	How straightforward was the mapping from the relevant elements of the application or technology to the blueprint model artefacts?	Effort put in designing the BP.	+

3.2. Develop Phase –Technology integration

3.2.1.1. Scalaris& J2EE based components

Criterion	Questions	Overall evaluation --, -, 0, +, ++	Answers
Learnability	How easy it was to grasp the Chef cookbooks and recipes model with respect to your needs?	Easiness of the Chef cookbooks and recipes specification process.	n/a (deploy/un-deploy AC is provided by application server)
		Easiness of the custom meter specification process.	n/a
Effectiveness	Did you achieve to specify all the needed deployment aspects with the Chef cookbooks and recipes?	Accuracy and completeness of the Chef cookbooks and recipes that you specified with respect to the target technology.	Passing connection details and other configuration data of the AC to the (generic) deploy/un-deploy AC recipe is undefined at the moment.
	If relevant: what aspects of Chef cookbooks and recipes should be added or improved to allow a complete description of deployment of the target technology?		n/a
	Accuracy?	Accuracy and completeness of the custom meters and KPI that you defined with respect to the target technology.	n/a

Efficiency of use	How straightforward was the specification of the deployment of the technology using a Chef cookbook and recipes?	Effort put in specifying Chef cookbooks and recipes.	n/a (deploy/un-deploy AC provided by application server)
	Effort?	Effort put in defining custom meters and KPI.	n/a
	How much effort did you put in the adaptation of the technology?	Effort put in adapting the technology in order to make it available in 4CaaSt (besides blueprint and Chef cookbooks and recipes specifications and custom meters and KPI definition).	not much – only needed to add support for as-a-service monitoring/accounting data to be gathered

3.2.1.2. *Mashup based components*

Criterion	**Questions**	**Overall evaluation** **--, -, 0, +, ++**	**Answers**
Learnability	How easy it was to grasp the Chef cookbooks and recipes model with respect to your needs?	Easiness of the Chef cookbooks and recipes specification process.	Yet learning. Technology is complicated, but it might be not so hard once fully understood
Effectiveness	Did you achieve to specify all the needed deployment aspects with the Chef cookbooks and recipes?	Accuracy and completeness of the Chef cookbooks and recipes that you specified with respect to the target technology.	On process the time of writing.

Efficiency of use	How straightforward was the specification of the deployment of the technology using a Chef cookbook and recipes?	Effort put in specifying Chef cookbooks and recipes.	Not measurable until finished, most of effort concerns learning.

3.2.1.3. Context related components

Criterion	**Questions**	**Overall evaluation --, -, 0, +, ++**	**Answers**
Learnability	How easy it was to grasp the Chef cookbooks and recipes model with respect to your needs?	Easiness of the Chef cookbooks and recipes specification process.	Not available as well as CaaS is an always on and running Service Enabler not requiring Chef for the integration.
		Easiness of the custom meter specification process.	Not available as far as these features are not yet implemented in CaaS Enabler.
Effectiveness	Did you achieve to specify all the needed deployment aspects with the Chef cookbooks and recipes?	Accuracy and completeness of the Chef cookbooks and recipes that you specified with respect to the target technology.	Not available as well as CaaS is an always on and running Service Enabler not requiring Chef for the integration.
		Accuracy and completeness of the custom meters and KPI that you defined with respect to the target technology.	Not available as far as these features are not yet implemented in CaaS Enabler.

	How straightforward was the specification of the deployment of the technology using a Chef cookbook and recipes?	Effort put in specifying Chef cookbooks and recipes.	Not available as well as CaaS is an always on and running Service Enabler not requiring Chef for the integration.
Efficiency of use		Effort put in defining custom meters and KPIs.	Not available as far as these features are not yet implemented in CaaS Enabler.
	How much effort did you put in the adaptation of the technology?	Effort put in adapting the technology in order to make it available in 4CaaSt (besides blueprint and Chef cookbooks and recipes specifications and custom meters and KPI definition).	Not available as well as CaaS is an always on and running Service Enabler not requiring Chef for the integration.

4. Evaluation summary (per Business goal)

The following questions are defined in the Experimentation Guideline [1] to be answered by each use case scenario and serve for evaluating the use case specific 4CaaSt objectives/goals presented in the Scenario Definition report D9.3.1 [2] . The business goals are also used to structure the section and are linked with their names and ID in *italic*.

Deliverable [2] D8.2.1 defines 3 mass market business goal mapped to one or more use cases where fulfilment can be analysed. This section consolidates the analysis of each scenario shown in previous section by business goals.

A business goal is fulfilled if every use case it depends on is fulfilled, so this section is a consolidation of results provided in part 1.

4.1. Business goal WP82_001: Support trading of service ecosystem

1	2	3	4	5	6	7
Develop cloud enabled software component	Deploy Software in the marketplace for commercialization	Commercialize Service provided by software	Choose service in the marketplace	Buy service from the marketplace	Enforce SLA	Enforce metering
/	/	Mostly Fulfilled	Partial * Some features planned RP3	Partial * Specific support for service provisioning	/	Very Partial * Some features Planned RP3

Table 15. Status of business goal WP82_001 : Support trading

4.1.1. How well do the currently available 4CaaSt platform capabilities fulfil the business goals from the experiments of this report?

This business goal is already quite well covered by RP2 version of the 4CaaSt platform with the exception of metering use case which will allow custom price model to be created by 4CaaSt on custom meters. This is an important use case and without it this business goal cannot be achieved.

Yet, coverage of the use cases is coherent with the features planned for RP2 and current status is satisfactory.

4.1.2. What evolutions of 4CaaSt are needed to accurately and completely fulfil those goals?

This business goal deeply depends on features planned for RP3 by 4CaaSt. 4CaaSt is so far on the right track and no fundamental flaw was discovered. Delivery of RP3 features should fulfil completely this business goal.

4.1.3. What evolutions of the current business goals and use cases are needed in order to be a better support to 4CaaSt evaluation?

The business goal itself does not need to be modified as it is labelled as a high level business goal and does not include assumptions on the technical solutions. Some modifications are however needed for use cases and are reported in the first section of this document.

4.2. Business goal WP82_002: Lower marginal hosting cost

1	2	3	4	5	6	7
Develop cloud enabled software component	Deploy Software in the marketplace for commercialization	Commercialize Service provided by software	Choose service in the marketplace	Buy service from the marketplace	Enforce SLA	Enforce metering
/	/	/	/	/	Optimal * Very partially implemented * Many important features planned RP3	/

Table 16. Status of business goal WP82_002 : lower marginal …

4.2.1. How well do the currently available 4CaaSt platform capabilities fulfil all or some of the business goals from the experiments of this report?

Without 4CaaSt features, a customer deploys his architecture and as it is done with manual deployment, creates more machines than what is needed “most” of the time. Customer has to choose, whether to get closer to 100% of the time and spend more resources, or lower its quality of service, his customer satisfaction and ultimately his incomes.

Lowering hosting cost is linked to enforce SLA because savings are obtained in 4CaaSt by using as little dedicated resources as possible for a given SLA at any point in time.

4CaaSt project roadmap has defined that scalability feature would be available for RP3 so evaluation of this business goal is not possible in present deliverable.

We however already have performed some of the required steps:

- Definition of custom probes in application
- Declaration of custom probes in blueprint

These steps are indeed costs for service providers and they should be minimized. We have found that these costs are optimal and could not be minimized further given current state of the art and the need to perform custom actions specific to each application.

As a conclusion, we would say that what the steps currently performed form the optimal way from a cost analysis point of view.

Regarding future steps, we envision advance automatic scalability feature will allow savings but we will have to wait RP3 to validate this claim.

4.2.2. What evolutions of 4CaaSt are needed to accurately and completely fulfil those goals?

Given that use case could be fulfilled optimally by 4CaaSt currently available process, there is no evolution request to submit at this stage.

4.2.3. What evolutions of the current business goals and use cases are needed in order to be a better support to 4CaaSt evaluation?

This use case is currently fulfilled very partially so it would be premature to extend it now.

4.3. Business goal WP82_003: No limit scalability and reduced development cost

No limit scalability means in short that an application developed with 4CaaSt shall scale horizontally beyond a few servers without major development cost.

1	2	3	4	5	6	7
Develop cloud enabled software component	Deploy Software in the marketplace for commercialization	Commercialize Service provided by software	Choose service in the marketplace	Buy service from the marketplace	Enforce SLA	Enforce metering
Mostly Fulfilled * 4CaaSt Application developed	Mostly Fulfilled * Blueprint	/	/	/	Partial * Some features planned RP3	/

Table 17. Status of business goal WP82_003 : No limit …

4.3.1. How well do the currently available 4CaaSt platform capabilities fulfil all or some of the business goals from the experiments of this report?

In this business goal, we were expecting to bring no limit scalability (horizontal scalability beyond a few servers) for a reduced development, which means to most qualified internet developer working in standard companies.

We were able in RP2 to

- Develop the application (UC 1)
- Describe the application and deploy it in the marketplace (UC 2)

We will be able in RP3 to

- Enforce SLA to control horizontal scalability (UC6)

We plan to validate scalability on the Wikipedia page render which uses Scalaris key/value backend. Applications that are 4CaaSt compliant will be able to benefit from future 4CaaSt scalability features. Yet, we were not able to test 4CaaSt scalability features as they are planned for next reporting period.

However, fulfilling most of UC1 and UC2 is already an achievement of the utmost importance for the future of 4CaaSt.

4.3.2. What evolutions of 4CaaSt are needed to accurately and completely fulfil those goals?

Use cases required evolution of the blueprint relative to documentation and more precise definition of the intended use of some of the field.

4.3.3. What evolutions of the current business goals and use cases are needed in order to be a better support to 4CaaSt evaluation?

This use case is currently fulfilled very partially so it would be premature to extend it now.

5. Conclusion

The evaluation results corresponding to the first release of the 4CaaSt scenario 8.2 [2] have been described in this document. Based on these results, a set of improvements to the 4CaaSt platform have been identified and will be addressed in next releases of the 4CaaSt platform. The most relevant conclusions reached can be summarised as follows:

4CaaSt evaluated components fulfil business goals defined for the mass market.

Qualitative evaluation:

> Current evaluation of 4CaaSt platform has not revealed design flaw that would be problematic for the future of the platform.
>
> Graph of dependencies between application elements could be successfully described as XML descriptors, and a level of indirection to link to dependencies through blueprint requirement is more than enough to account for variability due to multiple choices and multiple instantiations.
>
> Links between the logical description of the application and action performed by automation is not always perfectly understood and some work needs to be done on clarifying these aspects for all partners.

Quantitative evaluation is planned for next reporting period and is already well detailed in the scenario. This evaluation will provide the percentage of benefit relative to a standard un-optimized platform.

Improvement points are mainly focused on the blueprint schema which has been used thoroughly.

- 4CaaSt blueprint schema is easy to understand and is well understood as description formalism.
- 4CaaSt blueprint schema is used for every stage of an application lifecycles, yet not all fields are meaningful at all time and no guidance is provided. The lifecycle state of a blueprint is also not clear.
- 4CaaSt blueprint descriptor schema does not provide enough description fields. Blueprint users have spontaneously added XML comments in manually edited files to detail the intended use.
- 4CaaSt blueprint describes services with web services in mind. This does not always fit technical services which uses optimized protocols.
- 4CaaSt blueprint schema delegates description of accounting probe and monitoring probe to extension schema which need to be provided.

Future work related to 8.2 should then encompass the following activities:

- Update the scenario description 8.2 [2] to cover the improvements identified in this document.
- Implement the corresponding changes in the scenario prototype.
- Integrate the scenario prototype with the new release of the 4CaaSt platform.
- Conduct a new evaluation round of the scenario.

6. References

[1] Reports on Experimentation, Experimentation Guidelines, Lead Editor: Michel Dao, FT, July 2012

[2] 4CaaSt D8.2.1 “Use Case Application eMarketplace for Mass market: Scenario Definition”, Deliverable D8.2.1, 2011.

[3] 4CaaSt D8.2.3 “Use Case Application eMarketplace for Mass market: Integration report”, Deliverable D8.3.3, 2012.~~.~~