Brokerage for Quality Assurance and Optimisation of Cloud Services: An Analysis of Key Requirements

Dimitrios Kourtesis[1,2], Konstantinos Bratanis[1,2], Andreas Friesen[3], Yiannis Verginadis[4], Anthony J.H. Simons[2], Alessandro Rossini[5], Antonia Schwichtenberg[6], and Panagiotis Gouvas[7]

[1] South-East European Research Centre, International Faculty, The University of Sheffield, 24 Proxenou Koromila Street, Thessaloniki, 54622, Greece
{dkourtesis,kobratanis}@seerc.org
[2] Department of Computer Science, The University of Sheffield, Regent Court 211 Portobello Street, Sheffield, S1 4DP, United Kingdom
{d.kourtesis,k.bratanis,a.simons}@dcs.shef.ac.uk
[3] SAP AG, Vincenz-Priessnitz-Strasse 1, Karlsruhe, 76131, Germany
andreas.friesen@sap.com
[4] Institute of Communications and Computer Systems, National Technical University of Athens, Zografou, Athens, 15780, Greece
jverg@mail.ntua.gr
[5] SINTEF, P.O. Box 124 Blindern, 0314 Oslo, Norway
alessandro.rossini@sintef.no
[6] CAS Software AG, Wilhelm-Schickard-Str. 10-12, 76131 Karlsruhe, Germany
Antonia.Schwichtenberg@cas.de
[7] Singular Logic S.A., A. Panagouli & Siniosoglou Str., Nea Ionia, 14234 Athens, Greece
pgouvas@gmail.com

Abstract. As the number of cloud service providers grows and the requirements of cloud service consumers become more complex, the latter will come to depend more and more on the intermediation services of cloud service brokers. Continuous quality assurance and optimisation of services is becoming a mission-critical objective that many consumers will find difficult to address without help from cloud service intermediaries. The Broker@Cloud project envisages a software framework that will make it easier for cloud service intermediaries to address this need, and this paper provides an analysis of key requirements for this framework. We discuss the methodology that we followed to capture these requirements, which involved defining a conceptual service lifecycle model, carrying out a series of Design Thinking workshops, and formalising requirements based on an agile requirements information model. Then, we present the key requirements identified through this process in the form of summarised results.

Keywords: Cloud Service Brokerage, Cloud Service Broker, Requirements Analysis Methodology, Quality Assurance, Optimisation, Cloud Services.

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 150–162, 2014.

1 Introduction

As the number of cloud service providers grows and the requirements of cloud service consumers become more complex, the need for third party entities to intermediate between consumers and providers of cloud services is becoming stronger. A number of cloud service intermediaries have already appeared on the market, helping enterprises to find and to compare cloud services (e.g. service marketplaces), to develop and to customise services (e.g. application Platform as a Service offerings), to integrate services (e.g. integration Platform as a Service offerings), and more [1]. What all these intermediation services have in common is that they offer a form of brokerage for cloud services. Cloud Service Brokerage (CSB)[1] is becoming increasingly recognised as a key component of the cloud computing value chain [2] with market analysts predicting that it will soon be the fastest growing segment of the cloud computing market [3].

Consumers of cloud services will come to depend more and more on the intermediation services of cloud service brokers, and as the needs of consumers evolve, so will the intermediation services offered by the brokers. A type of intermediation service with high added value to consumers, especially to those who rely on multiple external cloud service providers for their daily operations, will be brokerage for continuous quality assurance and optimisation of cloud services.

Broker@Cloud [4] is an EU-sponsored collaborative research project that was set up to investigate the challenges associated with introducing such capabilities into cloud service brokers. The project will deliver an extensible software framework allowing cloud service intermediaries to equip their platforms with advanced means for continuous quality assurance and optimisation of cloud services. The framework will comprise methods and mechanisms for platform-neutral description of enterprise cloud services; cloud service governance and quality control; cloud service failure prevention and recovery; and continuous optimisation of cloud services.

This paper reports on the methodology employed in the scope of Broker@Cloud to capture the high-level requirements for the envisaged framework, and presents the results obtained from this analysis. In Section 2 we set the context for this work by motivating the need for continuous quality assurance and optimisation brokerage for cloud services. In Section 3 we discuss the methodology that we followed to derive key requirements for the software framework. The methodology section comprises three parts: the cloud service lifecycle model that we used as conceptual framework to guide our thinking about cloud service brokerage requirements, the Design Thinking process that we followed to collect requirements, and the specification methodology that we followed to formalise the requirements. To the best of our knowledge there are not any similar requirements analysis efforts from the state-of-the-art that are focusing specifically to brokerage for quality assurance and optimisation of cloud

[1] There is an on-going debate on the definition of Cloud Service Brokerage, with disagreement over the characteristics that an intermediary should have in order to qualify as a Cloud Service Broker. The authors understand Cloud Service Brokerage as a business model, and we use the term Cloud Service Broker to denote an (IT) role of a business entity that creates value for consumers and providers of cloud services by acting as an intermediary.

services. In Section 4 we provide the actual requirements in the form of summarised results. For a full description of the results we refer the reader to [5], which covers the requirements analysis in full extent.

2 The Need for Cloud Service Brokers with Continuous Quality Assurance and Optimisation Capabilities

We are already witnessing a growing number of cloud service intermediaries that allow consumers to integrate, customise or aggregate cloud services [6]. In the future, however, service consumers will require much more sophisticated brokerage services, going far beyond the capabilities of today's cloud service brokers. One such type of brokerage services will be continuous quality assurance and optimisation [7].

As users come to depend on more and more cloud services, it will become increasingly more difficult to keep track of how these services evolve over time — through changes to their terms of provision, to their APIs, or variations in service performance and availability. Moreover, it will become increasingly more difficult to stay on top of all the implications that a change to a service can have, such as whether or not there is continuing compliance to different policies and regulations, continuing conformance to normative technical specifications or Service Level Agreements, and generally, continuous fulfilment of all the different kinds of functional and non-functional requirements surrounding a particular service's usage. The proliferation of increasing numbers of cloud services with similar functionality and comparable terms of provision will contribute to complexity, forcing users to invest more and more effort in identifying alternatives to the cloud services they are using.

For all these reasons, continuous quality assurance and optimisation of cloud services will become increasingly difficult for individual consumers to cope with by themselves, creating opportunities for a market of cloud service intermediaries addressing these needs. Brokerage services will step up to help consumers make sure that the cloud services they rely on meet quality standards on a continuous basis, and that they represent the optimal set of services to be using at any given time [1].

Much of the enabling technology that is needed to support continuous quality assurance and optimisation brokerage is certainly not new. Recent years have seen a proliferation of many relevant proprietary and open source tools that could provide building blocks for the implementation of such capabilities in brokers. Examples include tools for monitoring and managing applications, services and virtualised infrastructures, or tools for integrating heterogeneous data, processes and applications [1]. However, there exists no consolidated software design theory or set of best practices on how to engineer brokerage capabilities of this kind, and there is lack of dedicated software tools to build on [8].

Broker@Cloud aims to bridge this gap by delivering an extensible software framework which will allow cloud service intermediaries to equip their platforms with core capabilities for continuous quality assurance and optimisation of cloud services.

The framework will comprise methods and mechanisms for governance and quality control of cloud services, prevention and recovery of failures, as well as continuous

optimisation, building on common means for platform-neutral description of cloud services.

3 The Requirements Derivation Process

In this section we describe the process that was followed in the scope of Broker@Cloud to derive the key requirements for the envisaged continuous quality assurance and optimisation brokerage framework. In Section 3.1 we present an abstract model of the cloud service lifecycle, the role of which was to frame our thinking about cloud service brokerage requirements. Then, in Section 3.2 we outline the Design Thinking process that was followed to organise the requirements analysis effort. Finally, in Section 3.3 we present the requirements information model that we adopted to formalise the requirements.

3.1 Service Lifecycle Model

To guide our requirements derivation process we started with defining a generic cloud service lifecycle model. The motivation behind defining this model as the first step in the requirements analysis process was to ensure that we have a consistent conceptualisation of the context in which the sought software brokerage framework is meant to operate. The model is generic as it covers phases and processes that are relevant in a variety of settings, with no grounding to a specific type of cloud service delivery platform or cloud service intermediary.

Our abstract lifecycle model comprises three plus one phases. The first three are *Service Engineering, Service Onboarding,* and *Service Operation*. The fourth, crosscutting phase is *Service Evolution*. The phases and processes under each phase are illustrated in Figure 1.

By analogy with software engineering, the service lifecycle starts with the *Service Engineering* phase. The *Service Engineering* phase consists of *Design, Development* and *Testing* processes, carried out by the cloud service provider.

Once a cloud service has been successfully developed and tested, and a "go to market" decision has been taken by the cloud service provider, the service enters the *Service Onboarding* phase. Processes under this phase include *Registration, Certification/Assessment*, and, once the service is successfully qualified, *Enrolment*, to make the service visible to potential consumers and make it available for subscription.

A service enters the *Service Operation* phase with the first Cloud Service Consumer deciding to use the service. The tasks performed during this phase can vary significantly from one setting to another, depending on the nature of the cloud service (e.g. if integration is required) and the conditions of its usage as agreed between the parties involved. Typical processes under this phase include *Service Management, Support* and *Assurance,* to manage relationships and meet agreed usage conditions.

Finally, there is a fourth, *Service Evolution* phase which cuts across the whole lifespan of a service. The prominent process here is *Change Management*. Ultimately, the service lifespan ends with the process of *Deprovisioning* the service.

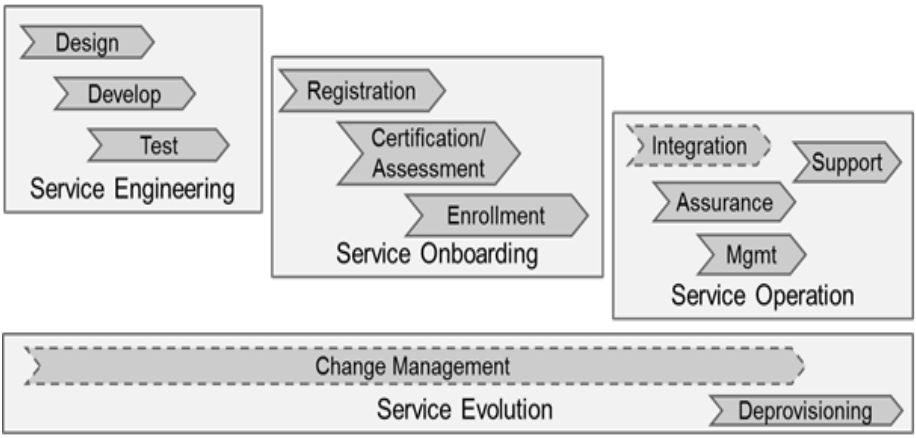

Fig. 1. Service Lifecycle Model

3.2 The Design Thinking Process for Deriving Requirements

To capture key requirements with respect to the framework developed by Broker@Cloud we carried out a series of Design Thinking workshops [9] with two companies that are active in the cloud computing market as cloud service providers and cloud service intermediaries. Both companies see potential in introducing capabilities for continuous quality assurance and optimisation into their cloud platforms and are presently considering a technology roadmap towards this direction.

We note that the Design Thinking is a methodology for collaborative analysis of the problem and solution space within a predefined timeframe. It takes into account requirements from different users and guides the design thinking team through the identification and prioritization of requirements profiles and corresponding solutions associated to different identified user types (personas). The scope and the approach of the Design Thinking methodology is very well fitting the challenge we are facing and is proved to be very helpful for derivation of requirements in our case, since our requirements analysis is based upon general state-of-the-art analysis and in-depth analysis of two industrial cloud platforms in the PaaS/SaaS area. Furthermore, it takes into consideration views of different stakeholders of the platform ecosystems.

Through Design Thinking workshops we gathered and analysed the requirements for the Broker@Cloud framework by mapping the existing and planned activities of the two pilot cloud platforms onto the phases and processes of our generic Service Lifecycle Model.

A Design Thinking process could have up to seven stages: *define, research, ideate, prototype, choose, implement,* and *learn*. Within these seven steps, problems can be framed, the right questions can be asked, more ideas can be created, and the best

answers can be chosen. The steps are not linear; they can occur in parallel and can be repeated. For our requirements analysis we chose to apply a four stage Design Thinking process consisting of *research, synthesis, ideation,* and *prototyping*. The additional *synthesis* step was introduced to combine the results of separate investigations. In the *research* and *synthesis* steps we identified requirements. In the *ideation* and *prototyping* phases we focused on identification and prototyping of methods and mechanisms providing solutions to the chosen requirements.

For the *research phase* we relied on customer interviews. We developed a questionnaire guiding interviewers and interviewees from each company through different aspects of current and future usage of the cloud platform of each company, asking which processes they could imagine handing off to intermediaries, what kinds of optimisation they consider to be relevant, etc. The interviews were conducted with a number of employees from each cloud platform company who work in different positions and therefore have different perspectives on the theme of cloud service brokerage. The interviews were collated and analysed to extract information relevant to continuous quality assurance and optimisation. The information was classified and clustered by topic, and the interviewees were asked to prioritise the requirements for their usage scenarios. In the *ideation phase* we selected some requirements with high priority to develop solution ideas. This was performed through subsequent steps of brainstorming, clustering and selection. The selected solution ideas were taken into the *prototyping phase* to develop conceptual paper-based prototypes, in order to investigate the technical feasibility of the identified solutions and obtain feedback.

3.3 Requirements Specification Methodology

We used the results from the Design Thinking workshops as starting point for identifying, clustering and analysing requirements for cloud service brokerage, focusing on requirements for the continuous quality assurance and optimisation capabilities outlined earlier.

To formalise these requirements, we followed a methodology inspired by the agile requirements information model of Leffingwell and Aalto [10], who propose to think of requirements in terms of *Themes, Epics, Features* and *User Stories*. According to Leffingwell and Aalto, these four concepts represent different forms of expressing user need and implied benefit, but at different levels of abstraction [10]. Variants of this requirements analysis model have become very popular in agile software development, especially in connection with agile methodologies such as Scrum and Kanban [11]. Building on this information model, we organised requirements into *Themes, Epics, Capabilities* and *User Stories*. The four concepts are explained below and the logical relationships between them are illustrated in Figure 2.

Themes and Epics. A *Theme* is a strategic level objective of a software product. For instance, one of the strategic Themes for our proposed brokerage framework is 'Governance and Quality Control'. An *Epic*, on the other hand, is a high level expression of a customer need. Derived from the portfolio of strategic product *Themes*, *Epics* are units of software development work that are intended to deliver the

value of a *Theme* and need to be prioritised, estimated and planned as part of the software development process [10]. In our methodology, every *Epic* is associated with exactly one *Theme*, whilst a *Theme* is associated with many *Epics*. For instance, one of the *Epics* for our software framework is 'Service Certification', and it maps to the *Theme* of 'Governance and Quality Control'. The *Theme* of 'Governance and Quality Control' is mapped to four *Epics* in total: 'Service Certification', 'SLA Enforcement', 'Policy Enforcement' and 'Service Lifecycle Management'.

Capabilities. A *Capability* is analogous to a *Feature* in the requirements information model of Leffingwell and Aalto. *Capabilities* can be understood as high level, complex (and possibly composite) services to be provided by a software system to fulfil a user need. As Leffingwell and Aalto put it, the purpose of this concept is to "bridge the gap from the problem domain (understanding user needs) to the solution domain (specific requirements intended to address the user needs)" [10]. In our methodology, a *Capability* may be mapped to more than one *Epic*. For example, 'Policy Evaluation' represents a *Capability* associated with two *Epics*: 'Service Certification' and 'Service Lifecycle Management'.

User Stories. A *User Story* is a brief statement of intent describing something the system needs to do for the user. A *User Story* often takes the following canonical form: "As a <role>, I want <goal/desire> so that <benefit>". *User Stories* should comply with "INVEST" properties, which means that they should be "Independent, Negotiable, Valuable, Estimatable, Small and Testable". In our methodology, each *User Story* maps to exactly one *Capability* and to exactly one *Epic*. For example, one *User Story* is the following: 'As a <broker>, I want to <check service descriptions against (broker's or consumers') policies> so that <I can recommend them with confidence>'. This *User Story* is associated with the 'Service Certification' *Epic*, and at the same time with the 'Policy Evaluation' *Capability*. The mapping of *User Stories* to *Epics* helps to capture the context in which a certain *Capability* is put into use, as exemplified by a *User Story*.

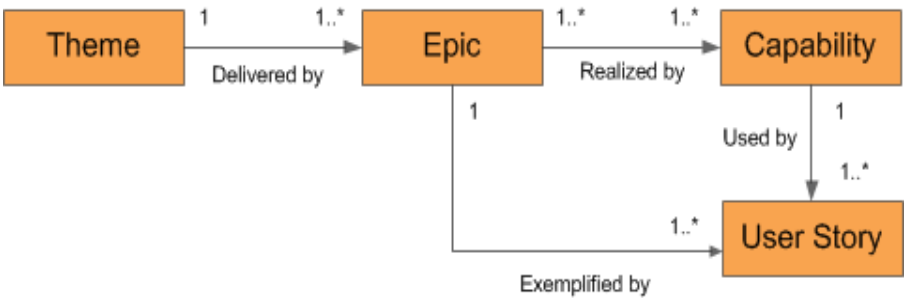

Fig. 2. Requirements information model adopted in Broker@Cloud

4 Key Requirements for a Software Framework Enabling Continuous Quality Assurance and Optimisation

In this section we summarise our requirements formalisation, by presenting the *Themes, Epics* and *Capabilities* that we identified. The results of our requirements analysis process include 4 *Themes*, 9 *Epics*, 15 *Capabilities* and 38 *User Stories*. Due to space limitations *User Stories* are not presented in this paper. For the complete list of *User Stories* that exemplify the *Epics* presented here we refer the reader to [5], which describes the requirements analysis results in full extent.

4.1 Themes and Epics

Governance and Quality Control. This *Theme* is concerned with managing the lifecycle of cloud services as they evolve; creating policies with respect to technical, business and legal aspects of service delivery and checking services for policy compliance; continuously monitoring services for conformance to Service Level Agreements; repetitively testing services to certify conformance to specifications or regulations and compatibility with expected behaviour. We have identified four *Epics* for the *Governance and Quality Control Theme*. The Epics are introduced in the table below (Table 1):

Table 1. Epics associated with the Governance and Quality Control Theme

No	Name	Description	Service Lifecycle
E1	Service certification	Service certification is a process that occurs during the onboarding and evolution of a cloud service. The process aims at certifying that a cloud service conforms to various requirements of the broker (e.g. pricing, fault-tolerance, correctness, etc.).	Onboarding, Evolution
E2	SLA enforcement	SLA enforcement is a process that aims at guaranteeing the expected service levels with respect to the agreements in place between a cloud service provider and a consumer.	Operation
E3	Policy enforcement	Policy enforcement is a process aiming at guaranteeing the conformance of the brokered cloud services to a variety of policies [12] – where policies may originate from different stakeholders.	Onboarding, Evolution
E4	Service lifecycle management	Service lifecycle management is a process that aims at controlling the evolution of different governed entities (e.g. providers, consumers, services, etc.) within the ecosystem of the broker.	Onboarding, Operation, Evolution

Failure Prevention and Recovery. This *Theme* is concerned with the reactive and proactive detection of cloud service failures; selection of suitable adaptation strategies to prevent or to recover from problematic situations as they surface; recommendation or (where possible) automated enactment of appropriate adaptation actions such as service

substitution or renegotiation of service terms. We have identified two Epics for the Failure Prevention and Recovery *Theme*. They are introduced below (Table 2).

Table 2. Epics associated with the Failure Prevention and Recovery Theme

No	Name	Description	Service Lifecycle
E5	Failure identification	Failure identification is a process that aims at the detection of failures that have either occurred or are likely to happen in the near future, by monitoring and analysing runtime data, through a combination of different monitoring approaches [14].	Operation, Evolution
E6	Failure prevention & recovery decision making	Failure prevention & recovery decision making is a process that aims at the suggestion of actions to recover from a failure, or to prevent an impending failure, by analysing an identified failure in order to decide a corrective action.	Operation, Evolution

Service Optimisation. This *Theme* is concerned with continuously identifying opportunities to optimise the set of services consumed by an enterprise with respect to different goals such as cost, quality, or functionality; ranking of optimisation alternatives through multi-criteria decision making, based on precise and imprecise characteristics of services and their providers thus exploiting a large number of QoS attributes, such as accountability, agility, assurance of service, cost, performance, usability. We have identified three *Epics* for the Service Optimisation *Theme*. The Epics are summarised below (Table 3).

Table 3. Epics associated with the Service Optimisation Theme

No	Name	Description	Service Lifecycle
E7	Consumer preferences analysis	Consumer preferences analysis is a process that aims at the aggregation and processing of user preferences (e.g. regarding functionality, precise and imprecise criteria [13]) in a unified way. It involves the management of criteria values expressed as crisp numbers or linguistic terms, in order to enhance the optimisation mechanism.	Operation, Evolution
E8	Optimisation opportunity identification	Optimisation opportunity identification is a process that aims at identifying appropriate situations during which optimisation can be performed.	Onboarding, Operation, Evolution
E9	Optimisation decision making	Optimisation decision making is a process that aims at deciding the appropriate optimisation action and recommending that to relevant stakeholders.	Onboarding, Operation, Evolution

Platform-Neutral Cloud Service Description. The first three *Themes* described above are concerned with processes executed in different phases of the Service Lifecycle to achieve certain quality assurance and optimisation characteristics. This *Theme* is concerned with declarative descriptions of inputs/outputs consumed/ produced by the above processes. Hence, it is a cross-cutting concern that appears in

the majority of the *Epics* presented so far. Platform-neutrality of descriptions is a precondition for addressing the above themes in the frame of an interoperable software framework. Many of the functional capabilities rely on the availability of certain kinds of suitable declarative descriptions defining the format of their inputs and outputs. The most of those descriptions can be specified as an integral part of a service or policy description. Therefore we define requirements on platform-neutral cloud service description by considering declarative descriptions such as service description and policy description to be capabilities as well.

4.2 Capabilities

To bridge the gap from the problem domain (understanding user needs) to the solution domain (specific requirements intended to address the user needs) we have identified 15 *Capabilities* as key requirements for our envisaged brokerage framework. The *Capabilities* are summarised in Table 4. For each *Capability* we provide a short description and the identifier of the *Epics* that it helps to realise.

Table 4. Capabilities and their association with Epics

No	Name	Description	Epics
C1	Functional testing (blackbox)	Functional testing is a capability that aims at validating the conformance of a cloud service to its behavioural specification, which is provided as part of the service description.	E1
C2	Policy evaluation (e.g. pricing model, security characteristics)	Policy evaluation is a capability that aims at checking if a process or an artefact complies with various policies established by different stakeholders (consumers, providers or broker).	E1, E4
C3	Code auditing (whitebox)	Code auditing is a capability that refers to the manual or automated inspection of the implementation of a cloud service with the intention to uncover faults, inconsistencies, security vulnerabilities and other issues.	E1
C4	Service description	Service description is a capability that aims at representing information about a cloud service in a form suitable to allow other capabilities in the same software framework to fulfil their goal.	E1
C5	Policy description	Policy description is a capability that aims at representing the policies of the various stakeholders (consumers, providers or broker), in order to enable policy evaluation.	E1
C6	Consumer optimisation preference description	Consumer optimisation preference is a capability that aims at representing the consumer preferences to be considered for the purposes of optimisation.	E7
C7	Consumer optimisation preference analysis	Consumer optimisation preference analysis is a capability that aims at handling and exploiting preferences expressed as crisp numbers or as linguistic terms in a unified way, in order to enhance optimisation.	E7

Table 4. (*Continued.*)

C8	Monitoring	Monitoring is a capability that aims at collecting, aggregating and correlating runtime and marketplace data, in order to facilitate several capabilities of the broker.	E2, E3, E5, E8
C9	Optimisation analysis	Optimisation analysis is a capability that aims at analysing optimisation opportunities, in order to identify optimisation actions.	E8, E9
C10	Optimisation recommendation	Optimisation recommendation is a capability that aims at reasoning about alternative optimisation actions, in order to recommend the best alternatives to the relevant stakeholders.	E9
C11	Optimisation validation	Optimisation validation is a capability that aims at collecting feedback about the recommended optimisation actions, in order to improve the optimisation process.	E9
C12	Failure recovery & prevention rules description	Failure recovery & prevention rules description is a capability that aims at representing the rules required for reasoning about potential failure recovery and prevention actions.	E6
C13	Failure analysis	Failure analysis is a capability that aims at identifying the cause of a failure which has already occurred or is impending, and to reason about the appropriate recovery or prevention actions.	E5, E6
C14	Failure recovery & prevention recommendation	Failure recovery & prevention recommendation is a capability that aims at recommending the best alternative recovery or prevention actions to the relevant stakeholders.	E6
C15	Failure prevention and recovery validation	Failure recovery & prevention validation is a capability that aims at collecting feedback about the recommended recovery or prevention actions, to improve the failure recovery and prevention process.	E6

5 Conclusions

As the number of cloud service providers grows and the requirements of cloud service consumers become more complex, the latter will come to depend more and more on the intermediation services of cloud service brokers. For many cloud service consumers, continuous quality assurance and optimisation of cloud services will become a mission-critical objective that they will find difficult to cope with by themselves, thus creating room for intermediaries to offer their services.

Broker@Cloud is a research project aiming to make it easier for cloud service intermediaries to address this emerging need. This is to be achieved by creating an extensible brokerage framework that allows cloud service intermediaries to equip their platforms with core capabilities for continuous quality assurance and optimisation of cloud services. The framework will comprise methods and mechanisms for governance and quality control of cloud services, prevention and recovery of failures, as well as

continuous optimisation of cloud service usage, building on common means for platform-neutral description of cloud services.

In this paper we reported on the methodology followed to capture high-level requirements for the envisaged framework, and presented the results obtained from this first-level analysis. We presented the abstract cloud service lifecycle model which helped us to frame our requirements thinking, presented the Design Thinking process that was followed to derive initial requirements, and discussed our adopted information model for the formalisation of requirements. We then presented the key requirements identified through this process in the form of summarised results.

The Design Thinking process that was followed was rather effective in helping us to kick-start the requirements analysis process and to derive initial requirements from two companies that are already offering a number of cloud services on the market and are presently considering enhancing their platforms with capabilities for continuous quality assurance and optimisation of cloud services. This process served as groundwork for further internal discussion and reflection, and shed light on critical aspects to consider. The agile requirements capturing methodology that we followed was effective in helping us to ground these insights and to move forward, from analysis to specification. The resulting identified requirements are organised around 4 *Themes*, 9 *Epics*, 15 *Capabilities* and 38 *User Stories*. Next steps of this work include early prototypes to cover the core requirements discussed here. This will be the first step towards defining and implementing the architecture of a framework bringing capabilities for continuous quality assurance and optimisation brokerage closer to the reach of cloud service intermediaries.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°328392, the Broker@Cloud project (www.broker-cloud.eu).

References

1. Verginadis, Y., Patiniotakis, I., Mentzas, G., Kourtesis, D., Bratanis, K., Friesen, A., Simons, A.J.H., Kiran, M., Horn, G., Rossini, A., Schwichtenberg, A., Gouvas, P.: D2.1 State of the art and research baseline. Broker@Cloud Project deliverable (2013)
2. Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D.: Cloud Computing Reference Architecture, pp. 292–500. National Institute of Standards and Technology, USA (2011)
3. Plummer, D., Lheureux, B., Karamouzis, F.: Defining Cloud Service Brokerage: Taking Intermediation to the Next Level. Gartner (2010)
4. Broker@Cloud project website, `http://www.broker-cloud.eu/`
5. Kourtesis, D., Bratanis, K., Friesen, A., Simons, A., Kiran, M., Verginadis, Y., Rossini, A., Schwichtenberg, A., Gouvas, P.: D2.3 Requirements Analysis Report. Broker@Cloud Project deliverable (2013)
6. Cloud Services Brokerage Is Dominated by Three Primary Roles. Gartner (2011)
7. Bratanis, K., Kourtesis, D., Paraskakis, I., Verginadis, Y., Mentzas, G., Simons, A., Friesen, A., Braun, S.: A Research Roadmap for Bringing Continuous Quality Assurance and Optimization to Enterprise Cloud Service Brokers. eChallenges (2013)

8. Kourtesis, D., Bratanis, K.: Towards Continuous Quality Assurance in Future Enterprise Cloud Service Brokers. In: Proceedings of the 8th South East European Doctoral Student Conference, SEERC (2013)
9. Cross, N.: Design Thinking: Understanding How Designers Think and Work. Berg, Oxford UK and New York (2011)
10. Leffingwell, D., Aalto, J.: A Lean and Scalable Requirements Information Model for the Agile Enterprise. Leffingwell LLC(2009)
11. Kniberg, H., Skarin, M.: Kanban and Scrum - Making the Most of Both. LULU (2010)
12. Kourtesis, D.: Towards an Ontology-driven Governance Framework for Cloud Application Platforms. Tech. Rep. CS-11-11. Department of Computer Science, The University of Sheffield, Sheffield (2011)
13. Patiniotakis, I., Rizou, S., Verginadis, Y., Mentzas, G.: Managing Imprecise Criteria in Cloud Service Ranking with a Fuzzy Multi-criteria Decision Making Method. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOCC 2013. LNCS, vol. 8135, pp. 34–48. Springer, Heidelberg (2013)
14. Bratanis, K.: Towards Engineering Multi-layer Monitoring and Adaptation of Service-based Applications. Tech. Rep. CS-12-04. Department of Computer Science, The University of Sheffield, Sheffield (2012)