# Desired Quality Characteristics in Cloud Application Development

Leah Riungu-Kalliosaari, Ossi Taipale and Kari Smolander
*Department of Software Engineering and Information Management, Lappeenranta University of Technology, Lappeenranta, Finland*

Keywords: Cloud Computing, Cloud Applications, Quality Characteristics, Software Development.

Abstract: This qualitative case study describes how software development organizations reach for their own context-dependent quality in cloud application development. The study collected the data from selected organizations through interviews and applied the grounded theory method in the analysis. The study concludes that the desired quality varies among the organizations. However, usability was found to be an important quality characteristic in all the organizations. The organizations involved a set of three similar activities to attain the desired quality characteristics. These activities are summarized as (1) Selecting a suitable life-cycle model, during which (2) the customer is engaged and (3) the most suitable tools are used. The organizations incorporated these activities so as to establish supportive working practices for acquiring the desired quality.



## 1 INTRODUCTION

Every software organization aims at developing products and services that meet their specified requirements in order to achieve specific quality goals. Software quality has for a long time remained an elusive target in software organizations (Blaine and Cleland-Huang, 2008). Different factors within an organization, such as communication and outsourcing (Kasurinen, et. al, 2011), can affect software quality. When developing software applications, organizations seek to meet both the functional requirements, e.g. functional correctness and non-functional requirements, e.g. reliability. This is sometimes a task that can be challenging to balance (Blaine and Cleland-Huang, 2008).

Cloud computing avails a suitable environment for developing and hosting different applications (Yau and An, 2011). Hence, cloud applications are applications that are built, deployed and hosted in cloud environments which are spread across any of the service delivery models – Infrastructure as a service (IaaS), platform as a service (PaaS) or software as a service (SaaS) (Hobfield et al., 2012). Developing cloud applications entails that the developer uses a number of pre-defined architectural structures, resources, interfaces or service access and discovery mechanisms. This may emphasize certain quality requirements, e.g., the need for strict conformance and high interoperability. Therefore, it is important to know the desired quality and to understand how it reflects on the way cloudapplications are developed and vice versa. In other words, the development of cloudapplications needs to be aligned with the quality expectations and cloud environment may affect reaching of certain quality characteristics, for instance scalability.

According to the ISO/IEC 25010 standard (ISO/IEC, 2010), software product quality is the "degree to which the software product satisfies stated and implied needs when used under specified conditions." The ISO/IEC 25010 quality standard contains two parts - product quality model and quality in use model. The product quality model is made up of eight characteristics; functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability, and portability (ISO/IEC, 2010). The system quality in use model is made up of effectiveness, efficiency, satisfaction, freedom from risk and context coverage together with their associated subcharacteristics (ISO/IEC, 2010). The ISO/IEC 25000 series standards (ISO/IEC, 2005) are used in this study as a starting point in exploring the desired quality characteristics.

The aim of this paper is to evaluate the important product quality characteristics as perceived by software organizations developing cloud-based

Riungu-Kalliosaari L., Taipale O. and Smolander K..
Desired Quality Characteristics in Cloud Application Development.
DOI: 10.5220/0004489303030312
In *Proceedings of the 8th International Joint Conference on Software Technologies* (ICSOFT-EA-2013), pages 303-312
ISBN: 978-989-8565-68-6

applications or hosting applications in the cloud. We look at the development and testing activities that the studied organizations incorporate in order to achieve the desired quality. To that effect, our research questions are: (1) What quality characteristics do developers consider to be important for cloud-based applications? (2) What activities support developers in achieving these quality characteristics? To answer these questions, we decided to perform a case study of organizations that are either developing cloud-based applications, or whose applications are hosted in the cloud. We used the grounded theory analysis steps (Strauss and Corbin, 1990) to generate the observations related to the research questions.

Due to the different nature of the software applications developed by the participating organizations, the important quality characteristics vary between the organizations. Thus, our aim is to outline a set of quality characteristics that might be essential while developing cloud-based software applications. These quality characteristics may be used towards the creation of specific cloud-based development and testing techniques and tools that can help to achieve the quality desired by cloud applications developers. Therefore, this paper contributes towards understanding the activities that support development of cloud applications. In addition, the paper identifies the quality characteristics that can be seen as essential for cloud-based applications and can be used as building blocks for evaluating the quality of a cloud-based software application.

The rest of the paper is organized as follows: Section 2 describes related work and Section 3 presents our research approach in the study. Section 4 describes the findings, followed by a discussion about the results in Section 5. Section 6 concludes the paper.

## 2 RELATED WORK

Developing and delivering (or hosting) software applications in the cloud is becoming more common-place and so are research efforts in improving the quality of cloud-based software applications. Lee et al. (2009) propose a quality model for evaluating SaaS cloud services. They first define the main features of SaaS and then map them onto corresponding quality characteristics derived from the ISO/IEC 9126 standard (ISO/IEC 2001). For example, they map SaaS features reusability and customizability to reusability, which is one of the ISO/IEC 9126 quality characteristics, along with justification for mapping and defining the quality attribute. Consequently, they describe the appropriate metrics for each quality attribute. Their quality model may be used by both service providers and consumers to evaluate the quality of SaaS services.

Zheng et al. (2009) developed a ranking framework for evaluating the quality of different components that make up a cloud application. The framework enables a user to select the best-performing components for the application. By doing so, this is expected to improve the application's quality of service (QoS).

Every service provider wishes that the users would have an enjoyable experience when using the provider's service(s). The quality of experience (QoE) associated with cloud management has been evaluated as a critical goal for cloud service providers (Costa et al., 2012); (Jarschel et al., 2011); (Kafetzakis et al., 2012); (Qian et al., 2011). One way to evaluate the users' QoE is by an analytical approach that enables the service provider to accommodate the changing needs of the users without compromising the users' QoE (Qian et al., 2011). Another is by a systematic cloud-based service delivery framework which enhances the QoE through personalized services to the users (Costa et al., 2012).

Effective management of tools used across distributed teams in global software development (GSD) projects has been cited to be a challenge for which cloud computing provides a solution. Chauhan and Babar (2012) outline a set of quality characteristics that they estimate to be essential for cloud-based infrastructure that can be used to provide tools as a service (TaaS) to globally distributed teams. The quality characteristics are mostly functional in nature, spanning aspects such as multi-tenancy, supporting multiple devices as well as compatibility with commercially available tools.

Application development and hosting in cloud computing environments and platforms holds promise for various business operations. Along with that, and as suggested by the literature mentioned in this section, there are different ways to dissect quality constructs related to cloud-based software applications. The literature we have presented describes the quality of cloud applications based on their features, functionality and expected quality of experience (QoE). However, it does not tell about the real life practices that can be applied to generate the quality in the cloud applications. Thus, we take an approach that evaluates how the different

organizational activities contribute to the quality of the developed cloud applications. In particular, our study focuses on the quality of cloud applications from the developer's perspective.

## 3 RESEARCH APPROACH

We conducted a qualitative multiple case study composed of five organizations. The cases were selected because their applications were connected to cloud computing. Table 1 contains details about the interviewed participants and their respective organizations. We selected the five organizations from eleven interviewed organizations during October – December 2011. We applied theoretical sampling (Pare and Elam, 1997) and chose cases to provide examples of polar types (Eisenhardt, 1989). Theoretical sampling is used to select cases that can be compared (Glaser and Strauss, 1967), aiming at a deeper understanding of the studied cases along with the identified concepts and their relationships. According to polar type sampling, the organizations were different in size and varied in the type of applications that they produced.

We used semi-structured, theme-based interviews to collect the data. The main themes included software development, testing, quality and change management. The themes and interview questions are available at http://www2.it.lut.fi/project/STX/. All except one interview were face-to-face, and were conducted at the interviewees' work locations. The exception was answered by use of Google Docs – because the interviewer and interviewees were located in different countries. Google Docs was chosen so as to allow two persons from the organization to have access to the same document and answer the questions. Later on, the first author also accessed the document and downloaded it for archiving and analysis. All the face-to-face interviews were tape-recorded and transcribed for analysis. The interviews generated a sum of 73 standard A4 pages (Times New Romans Font, Size 12).

We applied the grounded theory approach (Strauss and Corbin, 1990) in the analysis. Grounded theory includes three coding steps namely open, axial and selective coding. In open coding, the data is classified into groups, themes or families, which are more specifically referred to as categories (Strauss and Corbin, 1990). The interview questions and themes were used as seed categories for conceptualizing the data into different categories. We used the ATLAS.ti software in coding the data.

Axial coding follows after a sensible set of categories has been developed. During this step, the categories are analysed deeper in order to identify connections and relationships between them. In the final selective coding, the core category is established, and this depicts the central phenomenon or concept that covers most if not all of the categories. In this study, we focused on the theme related to quality as outlined within the interview questions. Therefore, we concluded the core category for this study to be "Activities that aid in attaining desired quality characteristics of cloud-based applications".

Table 1: Interviewees and organizations.

| | Description | Role of interviewee(s) | Company profile and relation to cloud | Company size (EC, 2005) |
|---|---|---|---|---|
| Case A | IT company | Two testing managers | Provides IT, R&D, and consulting services. Runs a cloud-based service for test management. | Large |
| Case B | Software company | R & D manager<br>Quality assurance manager | Provides a web-based system for managing IT assets. One of this company's customer is running the system in their own private cloud. | Medium |
| Case C | Cloud computing start-up | Owner | One-man cloud computing startup that provides consulting and educational services and acts as a cloud service broker, providing a system that lets customers buy server instances from large cloud providers, targeting the developer community within small- and medium-sized enterprises | Micro |
| Case D | Cloud computing start-up | CEO<br>Tester | A cloud-computing start-up that is building an international social web site integrated with the world's most popular social networks | Micro |
| Case E | Software company | CTO | The organization uses Amazon EC2 environment, for example, in running windows servers and software applications | Micro |

# 4 FINDINGS

## 4.1 Categories

In this study, we focused on how the organizations targeted and sought to achieve the quality characteristics that they deemed to be most important for their respective applications. The analysis started with reading through the transcribed interviews for each organization. This was accompanied by making open codes, and writing memos corresponding to the codes. On a separate document, high level notes were taken highlighting the impressions about each organization. The impressions were noted either as text, or in reference to codes. As an example, we made this reference to a code for Case A: "Check the code: organizational change/perceiving future changes – about user experience being important for the end user, and therefore organizations need to bring that value to the user."

We wrote the following text as part of the impressions about Case D: "They are developing a cloud application. The team is using their previous experiences in deciding the development approach and tools. The development was taking place at the time of the interview. Changes [in requirements] would be addressed along the way. Quality requirements are adjusted according to the performance of the end product/service and the user preferences. Quality section in the interview has interesting views."

The impressions noted about the organizations were then used to identify the potential categories. For example, from the impressions mentioned above, some of the identified potential categories were "Type of developed application", "Development approach and tools", "Choice of development approach and tools" and "Perceived future changes". After identifying the potential categories, we went through the text again - during which we refined the categories based on the similarities, correlations and differences among them. The process of deriving the final categories was iterative, mostly focusing on finding the common thing that the organizations were talking about, and establishing how it related to other categories. As the end result, we derived the categories that affected how the important quality characteristics were targeted and achieved. Table 2 summarizes the categories together with their associated observations and information.

The category *life-cycle model and tools* describes the software development life-cycle model and the tools used to produce the software. Waterfall, incremental and agile development featured among the organizations, with agile development being the most popular. The development tools used were mainly chosen due to the developers' familiarity with them, which helped the organizations to focus on the development rather than using valued time in learning to use new tools.

The category *software application* describes the type of software that the organization is developing. Case A provided a cloud-based test management solution and Case B provided a software product, which was installed in different devices and controlled via a web console. One of Case B's customers was running the software product in their (customer's) private cloud. Case C provided infrastructure-as-a-service (IaaS) in the cloud and Case D was developing a software-as-a-service (SaaS) application. Case E provided web-based services, which were not necessarily running in the cloud, but the organization made use of cloud platforms to host some of their application services.

The category *important quality characteristics* focused on those aspects of quality that the organizations found to be must-haves, or at least most desirable in their respective software applications. There are different definitions of quality in literature. Gavin (1984) gives five definitions for quality based on transcendent, product-based, user-based, manufacturer-based and value-based views. In this study, we concentrate on quality from the perspective of the software application provider, and specifically, the developer point-of-view. These quality characteristics were likely those that were considered to "make or break" the application with obvious economic impacts on the organization's business. The important quality characteristics mostly included functional suitability, reliability, and security. Overall, usability featured in all organizations as an important quality characteristic.

In the category *practices for handling requirements,* we observed that the organizations had different mechanisms for handling the requirements. Even though different, these mechanisms were geared at facilitating a smooth development process that would aid in achieving the targeted quality characteristics. There was a difference in the origin of the requirements. The initial set of requirements for Cases B, C, and D came from the developers themselves, stemming from their previous experiences, needs and ideas for the software. Once the software had been rolled out to customers, the user feedback was used for

Table 2: Categories and observations.

| Category/ Case | Case A | Case B | Case C | Case D | Case E |
|---|---|---|---|---|---|
| Life-cycle model and tools | Commercial testing tools for test management | Agile development (scrum) | Agile development; Drupal, Amazon web services, Google App Engine Python, PHP, C | Incremental development; Codeignite, FIT and Selenium; JIRA, wiki, Google docs, spreadsheets | Agile development ; Drupal Amazon EC2 |
| Software application | Cloud based test management service (SaaS) | Web-based system for managing IT resources | Amazon web services-based value adding service for Infrastructure as a Service (IaaS) | Cloud-based application for an international social website integrated with the world's most popular networks (SaaS) | Web-based applications |
| Important quality attributes | Availability, functional suitability, reliability, usability, security | Functional suitability, security, usability, performance efficiency | Security, usability | Functional suitability, performance efficiency, reliability, security, operability, usability | Functional suitability, reliability, security, operability, usability |
| Practices for handling requirements | -Varies depending on customer; -Aim at measurable requirements; Internal guidelines are used throughout development; -Unclear requirements passed down from the business people -Lack of good developers; Customers not knowing exactly what they want | -User feedback the most critical input to requirement changes. -Documentation becoming more important as organization grows. -Quality (non-functional) requirements are not clearly defined. -More work than the internal capcity can handle -Timing constraints. | -User feedback, back-end analysis of the usage of the system and market evolution provide input to requirement changes. -Documentation is in the form of notes that are relevant only to the writer. -No problems in handling requirements at the time of the interview. | -Requirements can be refined at any stage. -Reinforced consistency most important approach. -Documentation is critical for communicating changes. -Problems with unpredictability of user preferences. | -Small team, everything is discussed and there is not much documentation.- Customers are involved in prioritizing the requirements. -Customers not knowing exactly what they want. -Customers not able to communicate their needs clearly. |
| Customer involvement | External customers determine the processes, and approaches to use. | Customers actively engaged in giving feedback about the system. | Customers actively engaged in giving feedback about the system. | None at the time of the interview. Organization was developing the service. | None at the time of the interview. Organization was developing the service. |

improving the software. In Cases A and E, the requirements came from the customer, and these varied depending on the customer's needs and preferences.

The organizations also experienced different problems while handling the requirements. We observed a connection between these problems and the important quality characteristics for the

organizations. For example, Case D had a set of requirements that it wished to incorporate in the first version of the software, and deemed functional suitability as important quality characteristic. The organization wanted to develop a functional system, but it anticipated problems when the requirements (and subsequently, the functionality) change due to user preferences after the application was ready for use.

The category *customer involvement* deals with the effect the customer has on the development process throughout the software development life cycle. Cases A and E often dealt with different customers and this meant that a substantial level of flexibility was required in order to address the unique demands of each customer. In Cases B and C, the customer was not involved during the development of the software. However, after releasing the software, these organizations relied on customer feedback to improve the software applications. In Case D, the customer was not involved, mainly because the organization was developing the first version of its software.

## 4.2 Observations

After deriving the categories, we continued with the analysis by evaluating the commonalities and differences between the categories. We wanted to produce general observations that would explain these commonalities and differences. Below we summarize these findings.

***Observation 1:*** *The most important quality characteristics vary among the organizations, but usability was important in all the organizations.*

When developing software, it is often that the application domain defines the most important quality characteristics, but common important characteristics also exist. The interviewees were asked to evaluate the ISO/IEC 25010 software product quality characteristics in order of importance. The quality characteristics were evaluated differently within each organization. Three of the organizations stated functional suitability to be one of the most important characteristics. Case A found it most important to provide excellent quality of service by ensuring that the systems were readily available and that the customers could get support services whenever needed. Case B mentioned usability as the most important characteristic and security became more important when running the software in the customer's private cloud. For Case C, security was most important. Case D noted functional suitability to be most important, but also emphasized performance efficiency because a wide customer base was expected to have simultaneous access to the system.

Security was generally mentioned and was motivated by the organizations' intentions to satisfy the customers' needs for data integrity and confidentiality. In connection with security, developers using cloud platforms for developing their applications find themselves with the need for strict conformance to the cloud platform provider's guidelines. This also calls for adhering to strict rules to ensure interoperability between the developed application and the cloud provider's development platform.

*"Before releasing a product, the most important quality for me has been [the] security issue. Because I'm using the [cloud] APIs … if the security could be breached, then that would first of all break the whole service, and second of all compromise my relationship with the platform [provider]. And that's the key quality issue that I'm worried about before I release the product"* – Owner, Case C

Overall, although not on the same ranking, all the organizations mentioned usability to be an important quality characteristic.

*"Functional suitability is obviously important since product is targeted to suit a wide range public, to satisfy [user] needs and be easy and friendly enough to use for clients to stay with the service."* – Tester, Case D

*"…customers are expecting an easy-to-use tool, which has all the possible functionalities, and it shouldn't cost anything, so all this in one picture, easy to take into use, I would like to pay only when I'm using something, and it should be so that I can handle as much as I can on my own."* - Testing manager 2, Case A

The emphasis on usability was interesting and unique because it was mentioned by all the organizations as being among the important characteristics for the respective software applications.

***Observation 2:*** *Agile development methods are preferred when developing cloud-based applications.*

Cloud-based applications are subject to changes due to factors such as changes in the cloud platforms and user preferences. Agile methods were considered to be most suitable because they allowed the organizations to react to changes and therefore, keep up with the quality expectations. The development iterations enabled Cases B and C to ensure that the security requirements were intact.

Case B used agile methods so as to implement new features and improve the software product periodically. Agile development methods helped Case C to keep up with the new features that are released by the cloud platform providers and thereby creating valuable services for the market. Doing so also facilitates quick development and publishing of new features, which might be slower when using life-cycle models such as the waterfall approach. Agile methods enabled Case E to adjust to different customer needs and requirements.

Agile methods also supported the development of most important features first, with room for improvement during subsequent development sprints. Customers' were reported to be more accommodating of incomplete features, as long as there were no risks to the customers' businesses.

*"There are no fool proof systems I guess, but when it comes to cloud I guess the tolerance levels [to errors] are much higher. Users don't expect the services to be perfect. When using cloud services, there might be some small broken things or [some] quirks, but they will eventually be fixed."* – Owner, Case C

The interviewees in Case A dealt mainly with testing activities, and were mainly involved in testing of software that has already been at the hand of developers – following a waterfall life-cycle model that incorporated a feedback system. The feedback system was relevant for clarifying the customer needs so as to improve the quality of service. However, the interviewees reported that some of their customers were taking agile development approaches into use.

***Observation 3:*** *Customer input is valued at different stages of the development.*

The organizations interacted with the customers at different stages of the development, depending on whether the requirements came from the customer or the developer. In situations where the customer presents a need and requirements for the software (as in Cases A and E), the organizations work together with the customers in developing a unified understanding of the requirements. The customers' feedback was a valued input to the development, but the biggest barrier was often times the customers' inability to translate their needs into clear requirements for the developers.

*"They [customers] all think that they know [what they want]. Seldom do they actually know what they are after…"* – CTO, Case E.

*"I feel that there are too many customers who really don't understand what testing means."* – Testing manager 2, Case A

Cases B and C developed the first version of their software based on their previous experiences leading to the identification of possible software solutions. Therefore, the initial requirements were drawn solely by the development team which had a clear picture of what the software was required to do. After releasing the software, cases B and C incorporated a steady interaction with the customer, with the aim of improving the usability aspects of the system. The customer feedback was also regarded as a vital source of information on how to improve the system in general.

*"We get a lot of feedback directly from [customers], in different ways - by email and by customer visits and so on. But then we also have this kind of development workshops with the main customers … coming up with ideas … and then they present and prioritize them."* – R & D manager, Case B

*"We also realize that there might be some deficits that need to be fixed, but it's really hard to pinpoint them before you have the interaction with the end users and end customers."* – Owner, Case C

Case D was developing the first version of their software following the same approach as Cases B and C, i.e. the organization was initially in full control of the requirements. The software was aimed at a wide variety of customers, and this posed a potential for future changes in the software to fit with different customer needs, and hence changes in the functionality of the system. The organization anticipated the changes to occur according to user preferences.

*"Obviously there may be problems with requirement changes, since it is almost impossible to predict what functionality is going to be more preferred or demanded by users. Statistics are going to be used to track [the changes] and monitor how they affect user activity on web."* – Tester, Case D.

***Observation 4:*** *The developers choose development tools according to their knowledge, skills and familiarity with the tools.*

Each organization was most familiar with some development tools, systems and programming languages. In most cases, these were used in developing the software, and any additional tools required were chosen based on their learnability, usability and appropriateness for the intended need. Case B was looking to improve the development activities and this motivated the choice for a suitable test management tool. On the basis of offering cloud brokerage services, Case C used the APIs provided by the cloud platform provider and this was appropriate for adhering to the security guidelines

provided by the cloud platform provider. Cases D and E chose their respective development frameworks because they were appropriate for building web-based applications.

*"…because some of our programmers had experience with it [the tool used] and we chose it as the best option for our needs."* – CEO, Case D

*"I decided to have it [the test management tool]. I had to have some tool for managing the test plans, and we didn't have it earlier."* – Quality assurance manager, Case B

For organizations whose software was heavily based on cloud platforms, as in Case C, the developer was constantly on the look-out for new tools or application programing interfaces (APIs) that would support further development of the software.

*"Whenever Amazon or Google for example release a new API or new feature, if you want to benefit from it and create something new that has market value, you have to move fast."* – Owner, Case C.

### 4.3 Summary of the Observations

We observed that the desired quality varied among the organizations. However, usability was found to be important in all the organizations. Despite the differences in desired quality, we observed that the organizations involved three activities geared towards attaining the desired quality characteristics. These activities are represented in the second to fourth observations. We summarize them as (1) Selecting a suitable life-cycle model, during which (2) the customer is engaged and (3) the most suitable tools are used. The organizations incorporated these activities so as to establish supportive working practices for acquiring the desired quality.

The life-cycle models were such that they allowed the developers to interact with the customers and cloud platform providers. Interacting with the customers helped in improving requirements, and consequently, the functionality of the software. This was especially useful for enhancing the usability of the end-products. On the other hand, Interacting with the cloud platform provider helped the developers to align the functionality of their (developers) applications according to the specific platforms within which the applications are developed. The developers used development tools that were deemed to be most relevant for the software being developed. For example, by selecting development frameworks suitable for building web-based applications, Cases D and E were able to focus on developing the important application features.

The observations we have described above relate to the activities that the organizations incorporated to develop cloud applications. Building cloud applications is no doubt a complex process, which requires multidisciplinary techniques for providing traceable links between development activities and the desired quality. The use of cloud computing platforms for developing and hosting applications necessitate the need for close interaction between the cloud platform providers and the application developers.

## 5 DISCUSSION

The purpose of this study was to observe how organizations incorporate various activities in their development and testing processes in order to achieve the desired quality of cloud applications. The studied organizations had software applications that were either developed and/or hosted in the cloud. The activities that supported the development of the cloud applications involved the use of appropriate life-cycle models and tools. Furthermore, the customer view was incorporated in order to enhance the value that the applications give to the customer.

It seems that cloud applications need to have a "good first impression" and this is likely why all the organizations mentioned usability to be important. It is also an indication that using the cloud for developing and delivering software might draw more attention to usability and user experience aspects. Cloud-based software applications may be expected to be intuitive and easy to use. Providing good user experience may be particularly important for small organizations, motivated by their intention to attract and retain customers. Cloud-based services and products are essentially Web 2.0 applications, whose success is claimed to "depend on loyal rather than casual users" (Orehovacki, 2011). Therefore, it is important for the developers to evaluate their application's quality in order to meet the customers' needs and generate profits.

Notwithstanding the common cloud aspect and usability's value, there was a variation in other important quality characteristics for the software applications developed in each organization. This confirms Gavin's (1984) argument that even for one product, there would be different definitions of the product's quality by different stakeholders. The quality characteristics functional suitability, security,

performance efficiency and reliability were each, to a certain extent, regarded to be important. In addition, developers using cloud platforms have to ensure conformance to the cloud provider's guidelines and interoperability with the platforms.

The studied organizations generally preferred agile development methods. When using cloud platforms to develop cloud applications, the application developers are not in full control of some components. Cloud platform providers frequently release new features, and using agile methods would allow an organization the opportunity to take advantage of the new features to enhance the existing services or build new ones. Agile development methods also enable an organization to involve the customer as early as deemed appropriate during the development activities. Customer feedback was also appreciated in providing suggestions for refining the requirements and improving the software in subsequent sprints. The observed alignment to agile practices in the studied organizations correlates to the extended agile process model proposed by Patidar et al. (2011). The model accommodates interaction between the developers and cloud providers with a focus on enhancing the effectiveness of cloud application development.

Releasing the software to the market as soon as it is able to deliver some value to the customer, even if not fully ready, was seen to be an acceptable approach in finding those bottlenecks that may not be seen prior to the customer using the software. The view that end users are more tolerant to some errors might be more subjective from a cloud start-up's perspective, and it would be interesting to compare this view with that from other cloud start-ups and larger organizations. It seems that the organizations selected the tools, systems and programing languages that were most familiar to them in order to avoid learning curves that come with using new tools, systems and programing languages. Sodhi and Prabhakar (2011) suggest a method for evaluating different cloud oriented platforms for application development. They divide the platforms into three groups - traditional non-cloud, virtualized and cloud aware platforms – and suggest a selection criteria based on the targeted non-functional quality attributes (NFQAs). For example, their selection criteria showed that cloud aware platforms are best suited "for achieving high scalability and availability" (Sodhi and Prabhakar, 2011).

From our original sample of eleven organizations, we selected five "polar point" organizations for the study. External validity deals with the extent to which the results can be generalized (Runeson and Höst, 2009). Case study settings impose some limitations to the possibility of generalization. The results can only be directly generalized when discussing comparable organizations. However, we believe that our findings may be relevant to other organizations developing cloud applications. Our study confirmed that the application domain strongly affects the most important quality characteristics, but also common important quality characteristics exist. Specifically, we observed an emphasis on usability as a much highly desired quality characteristic for cloud-based software applications. We used grounded theory for analysis, as described by Strauss and Corbin (1990). The development of a theory is a dynamic process, whereby the theory can be extended by observing additional cases. Therefore, it is logical to expect that our findings can be extended to provide more insight on cloud application development practices.



## 6 CONCLUSIONS

This focused on the desired quality characteristics of cloud applications and the activities that developers undertake to achieve the desired quality. The results are based on experiences of software organizations whose software applications were either developed and/or hosted in the cloud.

To achieve the desired quality goals, the organizations incorporated activities that encourage the use of appropriate life-cycle models and tools while keeping the customer engaged during the development. We found that even for cloud applications, quality is context-dependent, and varies across organizations. We also observed that usability was of general importance. Industrial practitioners can take the experiences discussed in this study and use them to enhance their development practices focused on producing cloud applications that bring value to the customer.

The research community can take the findings in this study and conduct in-depth studies related to the development of cloud applications. In the future, we intend to use the elicited quality characteristics to widen the scope and impact of the study in understanding the desired quality characteristics of cloud applications along with testing techniques that can be used to test corresponding quality characteristics.

## ACKNOWLEDGEMENTS

The STX project (http://www2.it.lut.fi/project/STX/), the Finnish Funding Agency for Technology and Innovations (TEKES), the companies involved in the project and the Graduate School on Software Systems and Engineering (SoSE) supported this study.

## REFERENCES

Blaine, J. D., Cleland-Huang, J., 2008. Software quality requirements: How to balance competing priorities. *IEEE Software,* Issue 2, Volume 25, pp. 22-24.

Chauhan, M. A., Babar, M. A., 2012. Cloud infrastructure for providing tools as a service: Quality attributes and potential solutions. *Proc. IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture,* pp. 5-13. .

Costa, P. M., Pitt, J., Cunha J. F., Galvao, T., 2012. Cloud2Bubble: enhancing quality of experience in mobile cloud computing settings. *Proc. 3rd ACM workshop on mobile cloud computing and services,* pp. 45-52, doi: 10.1145/2307849.2307860

Eisenhardt, K. M., 1989. Building Theories from Case Study Research. *Academy of Management Review,* vol. 14, no. 4, pp. 532-550.

European Commission, 2005. The New SME Definition: User Guide and Model Declaration, Enterprise and Industry Publications. http://ec.europa.eu/enterprise/policies/sme/files/sme_definition/sme_user_guide_en.pdf

Gavin, D. A., 1984. What does "product quality" really mean? *Sloan Management Review,* Issue 4, pp. 25-43.

Glaser B., Strauss, A.L., 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research,* Aldine Publishing Company, Chicago.

Hobfield, T., Schatz, R., Varela, M., Timmerer, C., 2012. Challenges of QoE Management for Cloud Applications, *IEEE Communications Magazine,* pp. 28-36.

ISO/IEC, 2001. ISO/IEC 9126-1 Software Engineering – Product Quality – Part 1: Quality Model.

ISO/IEC, 2005. ISO/IEC 25000 Systems and software engineering - Systems and software quality requirements and evaluation (SQuaRE) - System and software quality models.

ISO/IEC, 2010. ISO/IEC 25010 Systems and software engineering - Software product quality requirements and evaluation (SQuaRE) - Quality models for software product quality and system quality in use.

Jarschel, M., Schlosser, D., Scheuring, S., Hibfeld, T., 2011. Gaming in the Clouds: QoE and the Users' Perspective. *Mathematical and Computer Modelling.*

Kafetzakis, E., Koumaras, H., Kourtis, M. A., Koumaras, V., 2012. QoE4CLOUD: A QoE-driven Multidimensional Framework for Cloud Environments. *Proc. International Conference on Telecommunications and Multimedia,* pp. 77-82.

Kasurinen, J., Taipale, O., Vanhanen, J., Smolander, K., 2011. Exploring perceived quality in software organizations. *Proc.5th IEEE International Conference on Research Challenges in Information Science,* pp. 1-12.

Lee, J. Y., Lee, J.W., Cheun, D. W., Kim, S.D., 2009. A quality model for evaluating software-as-a-service in cloud computing. *Proc. 7th ACIS International Conference on Software Engineering Research, Management and Applications*, pp. 261-266.

Orehovacki, T., 2011. Perceived quality of cloud based application for collaborative writing. *Book chapter in Business Systems and Services: Modeling and Development, Information Systems Development,* pp. 575-586, doi: 10.1007/978-1-4419-9790-6_46

Pare, G., Elam, J. J., 1997. Using Case Study Research to Build Theories of IT Implementation. *Proc. International Conference on Information Systems and Qualitative Research (IFIP TC8 WG),* pp. 542 – 568.

Patidar, S., Rane, D., Jain, P., 2011. Challenges of software development on cloud platform. *Proc. World Congress on Information and Communication Technologies,* pp. 1009-1013.

Qian, H., Medhi, D., Trivedi, K., 2011. A hierarchical model to evaluate quality of experience of online services hosted by cloud computing. *Proc. 12th IFIP/IEEE International Symposium on Integrated Network Management,* pp. 105-112.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering,* vol. 14, Issue 2, pp. 131-164, doi: 10.1007/s10664-008-9102-8

Sodhi, B., Prabhakar, T. V., 2011. Assessing suitability of cloud oriented platforms for application development. *Proc. 9th Working IEEE/IFIP Conference on Software Architecture,* pp. 328-335.

Strauss, A., Corbin, J., 1990. *Basics of qualitative research: Grounded theory procedures and techniques,* SAGE Publications, Newbury Park, CA, USA.

Yau, S. S., An, G. H., 2011. Software engineering meets services and cloud computing. *IEEE Computer*, 44, 10 pp. 47-53, doi:10.1109/MC.2011.267.

Zheng, Z., Shang, Y., Lyu, M. R., 2010. CloudRank: A QoS-driven component ranking framework for cloud computing. *Proc. 29th IEEE Symposium on Reliable Distributed Systems,* pp. 184-193.