Quality of Experience Assessment Based on Analytics

Samuel A. Fricker, Farnaz Fotrousi, Markus Fiedler
Blekinge Institute of Technology
Karlskrona, Sweden
{samuel.fricker, farnaz.fotrousi, markus.fiedler}@bth.se

Philippe Cousin
Easy Global Market
France
philippe.cousin@eglobalmark.com

***Abstract*—This work, which is connected to the Future Internet Public Private Partnership (FI-PPP) Integrated Project FI-STAR, presents a validation approach for Future Internet applications based on the use of analytics. In particular, it discusses how to use and combine software use and health statistics for the assessment of user-perceived Quality of Experience, in order to monitor user satisfaction, the risk of user churn, and the status of the corresponding ecosystem.**

Keywords—QoE; QoS; response times; usage; analytics; churn; ecosystem

I. INTRODUCTION

Since more than a decade back in time, Quality of Experience (QoE) has become a key issue of concern for operators and providers, as bad QoE implies the risk of user churn [1]. Indeed, when a service or application does not meet its stakeholders expectations, economic loss is an almost unavoidable consequence. In particular, innovative applications are at risk once they do not succeed to satisfy their users.

In many sectors, the concern for quality has led to market entry barriers related to compliance, certification, and access to mission-critical data. In health and care, for example, IEC 80001 compliance, ISO 13485 certification, and access to data such as patient records are considered problematic [2]. Software product lines have been successfully used by companies to capture such domain-specific knowledge and thereby achieve systematic reuse across their product portfolio [3]. Such reuse is achieved in a software product line by engineering design specifications and components that embed commonality and variability across use cases of potential products. The impact is faster development and productization and better quality of applications and services.

The Future Internet Public Private Partnership (FI-PPP), a “European programme for Internet-enabled innovation”, builds on this idea of product lines and attempts to scale it from a single product or services company to a whole industry [4]. FI-PPP aims at establishing an evolving set of common components, called Generic Enablers (GE) that capture solutions to common problems in the building of internet-enabled applications and domain-specific problems such as interoperability with common devices and systems and those outlined above. The hope is that the resulting infrastructure advances the European markets for smart infrastructures, increases the effectiveness of business processes delivered through the Internet, and ultimately stimulates the economy.

In its first phase, a set of GEs have been developed, which aim at providing the basis for innovative applications in virtually any application domain (e.g. e-Health, logistics, energy, etc.) within development cycles that are significantly shorter than those achieved so far. The GEs are offered by potentially competing manufacturers and producers. Application and service developers acquire these GEs for building applications in question.

The GE-based approach is comparable to buying the ingredients for a delicious home-prepared meal in a supermarket. Obviously, both the quality of the ingredients and their skilful preparation determine the quality of the prepared meal. The host can judge the quality of the meal by looking at its look, smell, and taste. The ultimate judgments of that quality, however, is seen in the appraisals of the host’s guests and in the amount that people eat and are willing to return to eat upon the host’s invitation. Translation of this metaphor to the domain of the Future Internet, makes it obvious that (1) the quality of the GEs and (2) the way these GEs are composed make a difference for a developed application as well as the corresponding ecosystem [5]. The impact of these two concerns can be seen from (a) the comments of the users, and (b) the degree of usage.

How hosts, respectively product and service organizations, achieve good-enough quality throughout the whole value chain, from ingredients to the guests’ experience and attitude, is the research underlying this paper. Our approach is based on the idea that the health of applications and their ingredients (such as GEs) needs to be measured, and that its impact on usage needs to be monitored, in order to be able to assure sufficient Quality of Experience.

The FI-PPP Integrated Project FI-STAR [6] will address such validation, and develop and implement the corresponding measurement and analysis tools as follow-up of the ongoing requirement elicitation work. This paper reflects the approach to application and GE validation within FI-STAR and its seven use cases.

The remainder of the paper is structured as follows. Section II introduces an example of a FI-PPP based system and reviews existing work for quality evaluation of such system. Section III describes the analytics-based approach for QoE prediction and assessment. Section IV summarizes and concludes the paper with planned future work.

II. Background

Building a new system that meets its quality requirements is inherently difficult. Such requirements are often stated qualitatively like "the system must be fast", hence are ambiguous and thus difficult to verify [7]. When implementing such requirements the following kinds of problems may be encountered. Developers build a system that delivers less than the stakeholders expect. This results in stakeholder dissatisfaction and might render a system useless. Developers build a system that delivers more than the stakeholders need. This results in a system that is more expensive than necessary.

Quality is particularly important for heterogeneously sourced systems such as FI-PPP-based systems. When engineering such system, developers depend on components, applications, and services provided by third-parties. Developers give such trust only if solution providers keep their promises regarding the service levels that will be achieved. Analytics provide transparency for evaluating such third-party contributions, for predicting the quality of the system, and for monitoring if the running service performs as promised. Analytics also provide the basis for root-cause analysis if quality objectives have not been met.

Figure 1: Patient Data Sharing Solution shows such a heterogeneously sourced system, a simplified and anonymized version of a FI-STAR use case scenario (www.fi-star.eu). The system allows patients and clinicians to collect and exchange biometric and other patient data. The system creates value by empowering the patient with rapid feedback about his condition and by providing treatment decision-support to the clinician.

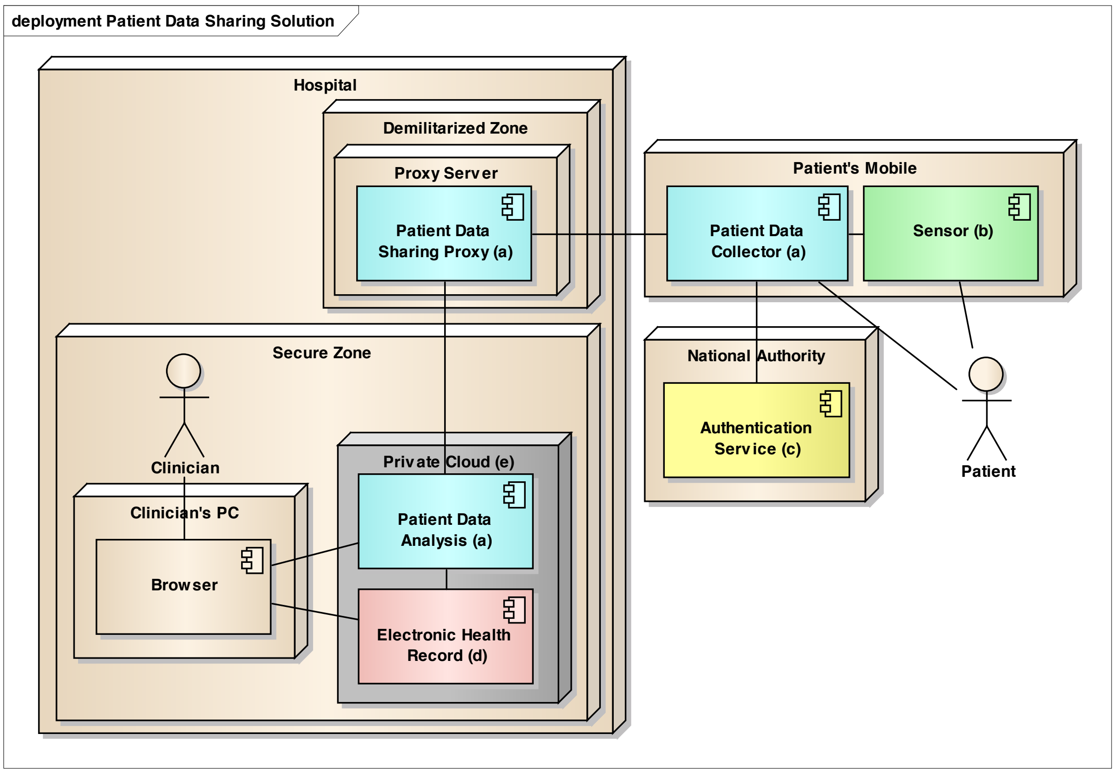
deployment Patient Data Sharing Solution
Hospital
Demilitarized Zone
Proxy Server
Patient Data Sharing Proxy (a)
Patient's Mobile
Patient Data Collector (a)
Sensor (b)
Secure Zone
National Authority
Authentication Service (c)
Patient
Clinician
Private Cloud (e)
Clinician's PC
Patient Data Analysis (a)
Browser
Electronic Health Record (d)

Figure 1: Patient Data Sharing Solution. The letters in parentheses refer to suppliers the corresponding items are sourced from.

According to the system architecture specification, the system consisted components, applications, devices, and services sourced from multiple parties. Patients would access the system with their personal mobile phone. The patient data collector, sharing proxy, and analysis applications would be developed by a software product company active in the healthcare domain. The sensors would be procured from a device manufacturer. User authentication services would be provided by the relevant national authority. The electronic health record would be managed by the hospital for which the solution was designed. The hospital-internal private cloud services, accessed by the clinician with one of the common web-browsers, would be provided by a local service provider. Components for connectivity and interoperability, finally, would be provided as GEs by FI-WARE platform providers.

A potentially wide variety of quality characteristics need to be fulfilled for given components, applications, and services to become useful. Such quality characteristics include functional suitability, performance, compatibility, usability, reliability, security, maintainability, and portability [8]. The quality levels achieved by a component or application is specified in the release requirements of that component or application. Warranties are used to guarantee that a product performs as promised in the specification. Usually, such a warranty is agreed between the supplier and the customer in a licensing contract [9]. Correspondingly, if a supplier provides a service for a customer, they agree on the quality of the service in a service level agreement (SLA). An SLA again specifies the quality levels, which the supplier gives warranty for. Norms, standards, and certificates are used to specify minimal quality levels to be achieved by products in a given industry [2].

Once developed, integrated, and deployed, the quality of the system affects the quality of the user's experience [10]. Quality can be so good that it allows the supplier to compete with alternative solutions [11]. If quality falls below the utility breakpoint, however, users will turn away and discard the solution [12].

One approach to manage quality proactively is the use of software analytics [13]. With analytics, attributes of software entities are measured, the measurements analysed and transformed into indicators that are useful for decision-making [14]. Such measurements give transparency, thus allows developers and management to decide about the course of actions for evolving the software system [15].

A wide variety of analytics are used to manage the quality of the software engineering process, the quality of the resulting software products, and software systems that are in operation. Developer dashboards improve awareness of a project's situation to support planning and coordination [16]. Such dashboards include information about the organization, plans and tasks, source code and builds, and quality assurance [17]. Prior to release, analytics allow analysing performance and reliability of software and services [18]. Similar analytics and geo-location are used to monitor and improve performance of the service in a real-world context with the intended users [19]. Voting buttons were proposed for measuring quality of experience. In comparison to laboratory testing, such late-stage analytics give diverse and representative results because they come from real use. Learning organizations use them to validate and improve testing assumptions.

Even-though analytics are effective for managing quality of software, their use is difficult to plan. In particular, it is unclear what an effective analytics approach is for managing quality when a heterogeneously sourced system such as the one outlined above is being developed. Too many variables could be measured, and trade-offs need to be made between ease of data collection and value of the analysis [20]. In addition, the composition of a system with multiple heterogeneous parts by one player and the use of the same part by different players makes standardization of a small set of broadly useful measurements important.

III. Approach

Our approach of predicting quality of experience (QoE) is based on three models: a measurement model, a composition model, and a lifecycle model.

The measurement model defines how quality attributes are measured and used to assert about properties of software or of users. It closely follows ISO/IEC 15939 for analytics measurement and ISO/IEC 25010 for quality attributes.

The software composition model defines how quality propagates as a result of composing software into real-world solutions. The approach follows the ideas of soft goal networks that allow deriving high-level global quality properties from low-level measurements [21].

The software lifecycle model determines when measurements are made and quality assessed or predicted. It follows the principles of product management [22], where a the release of a software product is prepared, made available for customers, and integrated by such customers into larger solutions.

A. Measurement Model

The measurement model describes how data is collected to make assertions about quality of service and of experience. In our cooking metaphor, such data collection corresponds to the host that probes the ingredients or meal and interviews the guests. Probes include looking, taking a smell, and tasting the food and asking guests whether they like its appearance and taste. The host uses this data to understand whether the food meet the desired quality standards and to understand the guest's experience with it. Some of these properties can be derived from the corresponding measurement. For example, bad smell can be an indicator for bad food. Other properties can be inferred from indirect measurements. For example, whether the food was good can be inferred by asking the guests about their opinion. Similarly, experienced cooks are able to accurately predict the guest's experience based on the just tasting the food. The assessment of the ultimate success is different, though. As hosts, we would define it as whether the guests are eating or not. This can be assessed by observing whether the guests are eating or not.

Figure 2 illustrates the application of these measurements to software that is used by a human user. The human user corresponds to the guest, the software to the meal, and the host to the software provider. Software analytics are applied at the software, and empirical inquiries performed with users. Both of them allow collecting data for assessing quality of the software, quality of the user experience. Also, either of them also allows assessing the ultimate success of the software: whether it is used or not.

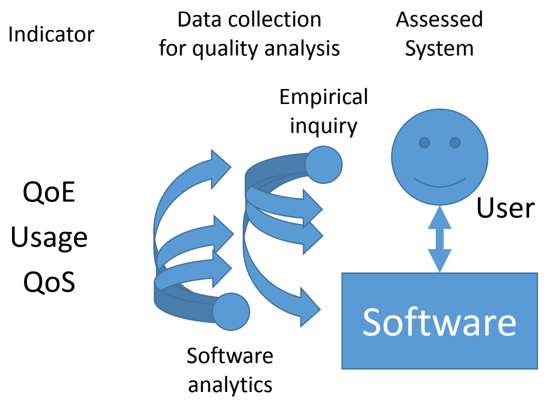

Figure 2: Measurement model: software analytics and empirical inquiry to assess QoS, QoE, and usage of software.

A substantial amount of work exists to understand how to assess software quality with analytics. Many address a selection of the software quality characteristics outlined by ISO/IEC 25010. The most common analytics are time and error-based.

The most common empirical inquiry determines a score of user opinion. Table 1 gives an overview of existing work on how measurement are used to assess software quality. It excludes software qualities that affect stakeholders other than users. Table 2 illustrates the same idea: how measurements will be used to assess the impact of software on the user.

Table 1: Measurement of Software Quality

	Time	**Error**	**OS/MOS**
Functional suitability			[23, 24], [25]
Performance	[26]		
Reliability	[27]	[28], [29]	
Security	[30]	[30]	
Usability			[23, 24]

Evaluation of functional suitability of a software is usually performed by functional testing. However the result of functional suitability is reflected in terms of functional acceptability from user's perception. It can be reflected even in usage analytics [25]. As example, during a software use, unnecessary functions will be understood from click a stream that is an implication of functional inappropriateness.

Other aspects of software quality, usually called Quality of Service (QoS), are performance [31] and reliability [32]. QoS usually refers to system components and network delivery capacity. It concerns time behavior, resource utilization, and capacity aspects, in addition to availability, frequency of failures, fault tolerance rate, and recoverability time. Attributes such as throughput, loss ratio, jitter, packet error rate, response time, delay and availability time are vital for measuring in the network layer, and the transport layer between two machines [28, 33]. Servers are measured by essential attributes of load rate, error rate, response time, peak response time, server up time, resource (i.e. CPU, memory, and disk) utilization, and threads [34]. In the application layer, statistics about page errors, frame rate, call success rate and the quality of outputs such as audio, video, and files are identified to measure QoS [35]. Finally, security of an application/component affects the solution health [36, 37]. The Attacks attribute is used to combat security issues such as DOS or malware attacks [38].

A time dependent attribute has the largest coverage for an end-to-end software health assessment. User perceived quality is dominated by response time and waiting time [39, 40]. The perception of quality on the user is typically measured by the Mean-Opinion-Score (MOS)[41]. Availability of the software solution is measured by infinite response time. The response time of an intrusion tolerant system with the steady-state availability is monitored for the security assessment [30]. Therefore response time can be a suitable candidate that simulates waiting time, availability as well as security. Error attribute provides further support for the assessment of software health in security, availability and fault tolerance.

Table 2: QoE Measurements mapping to Quality in Use

	Time	**Error**	**OS/MOS**
Effectiveness			
Efficiency		↵	
Satisfaction			
Freedom from Risk			

B. Composition Model

The composition model describes how data is collected to combine assertions about quality of service and of experience. In our cooking metaphor, such composition corresponds to the host that combines and cooks the ingredient into a meal that is served to the guests. The host uses heating and combination to process the ingredients into a result of value higher for the guests than the inputs that were used. The quality of the inputs and the host's own work affect the quality of the results. The results are at most as good as the worst of the inputs that was used. Skilful preparation of the meal and presentation of it to the guests, however, can increase the value of the whole meal well beyond the sum of the inputs.

Figure 3 shows a software composition model that allows describing the solution shown in Figure 1. Nodes such as the private cloud contained in the secure zone, which again is contained in the hospital correspond to instances of the infrastructure. Patient Data Analysis and Electronic Health Record are two instances of software that run on the private cloud infrastructure. Not shown in Figure 1 are the generic enablers that the Patient Data Analysis contains. The clinician is a user that uses a browser, which communicates with the Patient Data Analysis and the Electronic Health Record software.

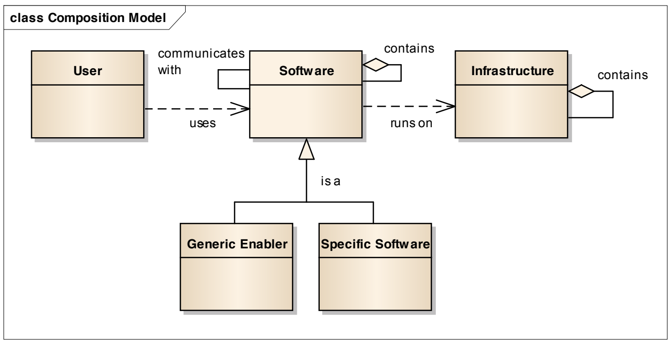

Figure 3: Composition Model

The composition model allows propagation of quality properties. Such propagation can be expressed in rules that are evaluated with an instance of the composition model (Figure 1 is such an instance). They determine how a property of one entity, for example a failure of an infrastructure, affects the rest of the software system. A set of availability-related rules would state that failure of infrastructure implies that any dependent software and user will experience the failure. Software that runs on reliable infrastructure, however, would not be affected by the failure. Similarly, a set of performance-related rules would state that the total roundtrip time for a user interaction corresponds to the aggregated time behavior of software, run on the respective infrastructure, and communication channels. Depending on criticality of quality of service and experience, the set of rules can be completed and refined.

C. Lifecycle Model

The lifecycle model describes how software and service infrastructure come into existence and evolve. The evolution stages then give raise to possible quality assurance actions. In our cooking metaphor, the cook would perform quality assurance actions based on the evolution stages of ingredients and the meal. He would look for ingredients that are made available to him on the market. Preferably he would turn to ingredients with trusted quality, for example as indicated by certification labels awarded to some ingredients. In addition, he would touch and take a smell of some of them to assess their quality. Once in the kitchen, he would process and combine the ingredients into a meal. The meal undergoes quality assurance in the kitchen before it is made available to the guests. Once these guests have received the meal, they look at it and take a smell (presumably with delight) before they decide to eat it.

Figure 4 shows a lifecycle model that allows explaining how software is developed, delivered, integrated, and made available as a solution for the healthcare environment shown in Figure 1. Each supplier, indicated by the letter in parentheses in Figure 1, has developed, tested, and released software or infrastructure. The integrator then has performed acceptance testing of the sourced software and infrastructure in his own premises and integrated them into the solution that Figure 1 describes. Again the integrator tested and released the software solution, before performing site acceptance testing and initiating its usage.

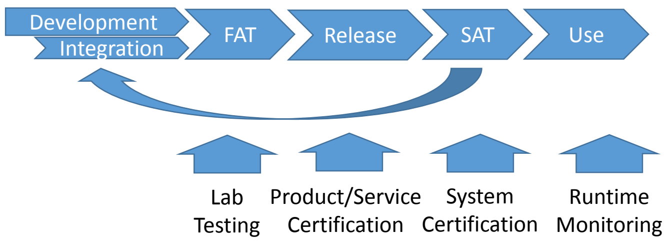

Figure 4: Software Lifecycle Model. FAT = factory acceptance test. SAT = site acceptance test.

The lifecycle model describes quality assurance actions that is performed at each respective lifecycle stage. Factory acceptance testing includes testing of the software in the supplier's laboratory environment. Software release is accompanied with certification. Such certification is standard practice of application stores such as Google Play and iTunes [42]. Site acceptance testing is performed by the consumer of the released software in a laboratory environment that is as close to the real-world environment as possible. In the healthcare environment, site acceptance testing of software systems is accompanied IEC 80001 and ISO/IEC 27000 [43]. Systems that have passed all these quality assurance hurdles are put into use, where they continue to be monitored [19].

Each quality assurance action involves collection of analytics and possibly empirical data as described by the measurement model. The collected data updates earlier predictions made with the help of the composition model. Such updating allows validation of the prediction and increases confidence in whether the final solution actually meets its quality objectives or not.

The combination of the measurement and composition models enable early prediction. The lifecycle model allows planning for step-wise improvement of these prediction, hence reducing the risks of the final test of where a solution is being used in a real-world environment and the achieved quality of experience level determines success or failure of the system.

IV. Summary and Conclusion

While important for any software, quality assurance is particularly critical for acceptance and successful use of heterogeneously sourced systems. Such systems integrator components from parties that the system integrator has little control over. As a consequence, the risk and the corresponding need for trust much higher than when a single-source software is developed.

This paper introduces a holistic approach for quality assurance of heterogeneously sourced systems. It is based on three models that together allow quality of experience prediction and step-wise validation of these predictions with real-world measurements. The measurement model describes how analytics and empirical data is collected and used for assertion of quality of service and experience. The composition model describes how measurements are propagated through the composed system to estimate overall quality of service and experience. The lifecycle model describes quality assurance actions that are used for validation of system quality.

The paper represents an important step towards unifying the so far separated disciplines of software engineering and performance evaluation in telecommunication systems. It contributes with a QoS and QoE measurement-based approach to managing quality while a software system is constructed. The paper explains the approach in depth with the metaphor of a host that prepares a delicious meal to guests. An exemplar taken from the FI-STAR project is taken to describe how the approach is transferred into a real-world environment.

Future work includes validation of the approach. Analysis of software architectures will be used for refining the composition model. A literature review will be performed for constructing a rule base for QoS and QoE assessment and prediction. Empirical inquiries about engineering process will be used to evaluate the composition model and refine the description of quality assurance practices. A particular focus will be given to the healthcare environment, where quality assurance is particularly important as it may decide on death or life.

V. Acknowledgment

This work has been co-sponsored by the European Commission, DG CNECT, through the FI-PPP Integrated Project FI-STAR.

VI. References

[1] P. Le Callet, S. Möller, and A. Perkis, "Qualinet White Paper on Definitions of Quality of Experience (2012)," European Network on Quality of Experience in Multimedia Systems and Services, Lausanne, SwitzerlandMarch 2013 2013.

[2] C. Thuemmler, S. Fricker, B.-J. Koops, M. Fiedler, E. Kosta, A. Schneider, *et al.*, "Norms and Standards in Modular Medical Architectures," presented at the Submitted to 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (IEEE Healthcom 2013), Lisbon, Portugal, 2013.

[3] K. Pohl, G. Böckle, and F. Van Der Linden, *Software product line engineering: foundations, principles, and techniques*: Springer, 2005.

[4] European-Commission. (2013, 2013-07-18). *FI-PPP | Future Internet PPP*. Available: http://www.fi-ppp.eu

[5] K. U. R. Laghari and K. Connelly, "Toward total quality of experience: A QoE model in a communication ecosystem," *Communications Magazine, IEEE*, vol. 50, pp. 58-65, 2012.

[6] FI-STAR_project. (2013, 2013-07-18). *Home: FI-STAR*. Available: http://www.fi-star.eu
[7] M. Glinz, "A Risk-Based, Value-Oriented Approach to Quality Requirements," *IEEE Software*, vol. 25, pp. 34-41, 2008.
[8] ISO/IEC, "Systems and Software Quality Requirements and Evaluation," vol. ISO/IEC FDIS 25010, ed: ISO/IEC, 2010.
[9] H.-B. Kittlaus and P. Clough, *Software Product Management and Pricing*: Springer, 2009.
[10] M. Fiedler, T. Hossfeld, and T.-G. Phuoc, "A Generic Quantitative Relationship between Quality of Experience and Quality of Service," *IEEE Network*, vol. 24, pp. 36-41, 2010.
[11] B. Regnell, R. Berntsson Svensson, and S. Olsson, "Supporting Roadmapping of Quality Requirements," *IEEE Software*, vol. 25, pp. 42-47, 2008.
[12] S. Khirman and P. Henriksen, "Relationship between quality-of-service and quality-of-experience for public internet service," in *In Proc. of the 3rd Workshop on Passive and Active Measurement*, 2002.
[13] T. Menzies and T. Zimmermann, "Software Analytics: So What?," *IEEE Software*, vol. 30, pp. 31-37, 2013.
[14] ISO/IEC, "Systems and Software Engineering - Measurement Process," vol. ISO/IEC 15939, ed, 2007.
[15] F. Fotrousi, K. Izadyan, and S. Fricker, "Analytics for Product Planning: In-Depth Interview Study with SaaS Product Managers," presented at the IEEE 6th International Conference on Cloud Computing (Cloud 2013), Santa Clara, CA, USA, 2013.
[16] O. Baysal, R. Holmes, and M. Godfrey, "Developer Dashboards: The Need for Qualitative Analytics," *IEEE Software*, vol. 30, pp. 46-52, 2013.
[17] J. Czerwonka, N. Nagappan, W. Schulte, and B. Murphy, "CODEMINE: Building a Software Development Data Analytics Platform at Microsoft," *IEEE Software*, vol. 30, pp. 64-71, 2013.
[18] D. Zhang, S. Han, Y. Dang, J.-G. Lou, H. Zhang, and T. Xie, "Software Analytics in Practice," *IEEE Software*, vol. 30, pp. 30-37, 2013.
[19] R. Musson, J. Richards, D. Fisher, C. Bird, B. Bussone, and S. Ganguly, "Leveraging the Crowd: How 48,000 Users Helped Improve Lync Performance," *IEEE Software*, vol. 30, pp. 38-45, 2013.
[20] G. Guest, A. Bunce, and L. Johnson, "How many interviews are enough? An experiment with data saturation and variability," *Field methods*, vol. 18, pp. 59-82, 2013/07/16/09:08:13 2006.
[21] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Boston, USA: Kluwer Academic Publishers, 2000.
[22] S. Fricker, "Software Product Management," in *Software for People: Fundamentals, Trends and Best Practices*, A. Maedche, A. Botzenhardt, and L. Neer, Eds., ed: Springer, 2012, pp. 53-81.
[23] P. Lew, L. Olsina, and L. Zhang, "Quality, quality in use, actual usability and user experience as key drivers for web application evaluation," in *Web Engineering*, ed: Springer, 2010, pp. 218-232.
[24] J. Kirakowski and M. Corbett, "SUMI: The software usability measurement inventory," *British journal of educational technology*, vol. 24, pp. 210-212, 1993.
[25] S. Elbaum, S. Karre, and G. Rothermel, "Improving web application testing with user session data," in *Proceedings of the 25th International Conference on Software Engineering*, 2003, pp. 49-59.
[26] S. Ran, "A model for web services discovery with QoS," *ACM Sigecom exchanges*, vol. 4, pp. 1-10, 2003.
[27] M. Merzbacher and D. Patterson, "Measuring end-user availability on the web: Practical experience," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, 2002, pp. 473-477.
[28] R. Shuping, "A model for web services discovery with QoS," *SIGecom Exch.*, vol. 4, pp. 1-10, 2003.
[29] Z. Zheng and M. R. Lyu, "An adaptive QoS-aware fault tolerance strategy for web services," *Empirical Software Engineering*, vol. 15, pp. 323-345, 2010.
[30] B. B. Madan, K. Gogeva-Popstojanova, K. Vaidyanathan, and K. S. Trivedi, "Modeling and quantification of security attributes of software systems," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, 2002, pp. 505-514.
[31] J. Burby and S. Atchison, *Actionable web analytics: using data to make smart business decisions*: Wiley. com, 2007.
[32] M. Houtermans, T. V. Capelle, and M. Al-Ghumgham, "Reliability Engineering & Data Collection," in *Systems, 2007. ICONS'07. Second International Conference on*, 2007, pp. 42-42.
[33] K. Hyun-Jong, L. Dong Hyeon, L. Jong Min, L. Kyoung-Hee, L. Won, and C. Seong-Gon, "The QoE Evaluation Method through the QoS-QoE Correlation Model," in *Networked Computing and Advanced Information Management, 2008. NCM '08. Fourth International Conference on*, 2008, pp. 719-725.
[34] N. Bhatti and R. Friedrich, "Web server support for tiered services," *Network, IEEE*, vol. 13, pp. 64-71, 1999.
[35] K. Mintauckis, "Empirical studies of Quality of Experience (QoE): A Systematic Literature Survey," 2010.
[36] S. Lindskog and E. Jonsson, "Adding Security to Quality of Service Architectures," in *Proceedings of the SS-GRR Conference*, 2002.
[37] A. Hamam, M. Eid, A. El Saddik, and N. D. Georganas, "A quality of experience model for haptic user interfaces," in *Proceedings of the 2008 Ambi-Sys workshop on Haptic user interfaces in ambient media systems*, 2008, p. 1.
[38] P. Yadav and G. Gupta, "Depleting Clouds," *International Journal of Engineering*, vol. 2, 2013.
[39] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler, "Waiting times in quality of experience for web based services," in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*, 2012, pp. 86-96.
[40] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *Services-I, 2009 World Conference on*, 2009, pp. 693-700.
[41] ITU, "ITU-T P.800," in *Mean Opinion Score(MOS) terminology*, ed: TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 2003.
[42] S. Jansen and E. Bloemendal, "Defining App Stores: The Role of Curated Marketplaces in Software Ecosystems," presented at the 4th International Conference on Software Business, Potsdam, Germany, 2013.
[43] C. Thuemmler, S. Fricker, O. Mival, D. Benyon, W. Buchanan, A. Paulin, *et al.*, "Norms and Standards in Modular Medical Architectures," presented at the IEEE HealthCom 2013, Lisbon, Portugal, 2013.