The Journal of Systems and Software 114 (2016) 69–81

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Quality assurance in software ecosystems: A systematic literature mapping and research agenda

Jakob Axelsson*, Mats Skoglund

Software and Systems Engineering Laboratory, Swedish Institute of Computer Science (SICS), PO Box 1263, SE-164 29 Kista, Sweden

ARTICLE INFO

Article history:
Received 12 March 2015
Revised 11 December 2015
Accepted 11 December 2015
Available online 18 December 2015

Keywords:
Software ecosystems
Quality
Verification
Testing

ABSTRACT

Software ecosystems are becoming a common model for software development in which different actors cooperate around a shared platform. However, it is not clear what the implications are on software quality when moving from a traditional approach to an ecosystem, and this is becoming increasingly important as ecosystems emerge in critical domains such as embedded applications. Therefore, this paper investigates the challenges related to quality assurance in software ecosystems, and identifies what approaches have been proposed in the literature. The research method used is a systematic literature mapping, which however only resulted in a small set of six papers. The literature findings are complemented with a constructive approach where areas are identified that merit further research, resulting in a set of research topics that form a research agenda for quality assurance in software ecosystems. The agenda spans the entire system life-cycle, and focuses on challenges particular to an ecosystem setting, which are mainly the results of the interactions across organizational borders, and the dynamic system integration being controlled by the users.

1. Introduction

Software ecosystems are an increasingly popular model for software development and the associated business aspects. Traditionally, companies work in a fairly closed setting in which software systems are provided as more or less monolithic entities. In a software ecosystem, this changes to an open environment where a community of developers from different organizations gather around an open or semi-open platform, and thrive from each other in a coopetition. This gives a potential for numerous benefits, including improved customer offers through the use of the innovation potential in the ecosystem (Iansiti and Levien, 2004) and reduced complexity of development by allowing composability (Bosch and Bosch-Sijtsema, 2010a).

Many of the examples of successful software ecosystems come from the IT domain, and include general applications such as PC operating systems, mobile phone apps, etc. (Bosch, 2009; van der Schuur et al., 2011). There are also initial signs of software ecosystems emerging in more critical applications, based on embedded systems, which are to a much higher degree tailor-made for a specific purpose (Eklund and Bosch, 2014). For all software products, quality is of high importance, and for critical embedded systems, on which the lives of people may depend, it is a non-negotiable characteristic. For some embedded systems, certain aspects of quality are even formal requirements that must be fulfilled to obtain certification from authorities.

Our current research deals with ecosystems for federated embedded systems, a concept where traditional embedded systems can be dynamically extended with plug-in software components (Axelsson and Kobetski, 2013) which can be provided by external developers. In a previous case study around the characteristics of such an ecosystem, quality assurance was identified as one of the most important factors (Axelsson and Kobetski, 2014; Axelsson et al., 2014), and some indication was given about important factors that contribute to quality in the embedded domain. However, according to Manikas and Hansen (2013) and Barbosa et al. (2013), there appears to be a general lack of knowledge about the relation between quality assurance and software ecosystems.

The purpose of this paper is therefore to: (a) provide an initial characterization of the state-of-the-art in the area, as well as (b) provide directions for further research needs in the form of a research agenda for the field.

To investigate the relationship between quality assurance and software ecosystems scientifically, it is necessary to first provide some clarity what is meant by these two rather broad terms. Thus, the following subsections deal with defining the terms software ecosystems and quality assurance. This also provides a theoretical framework that defines the scope of the study. After this, a more exact definition of the research questions in the paper will be given, together with an overview of the remainder of the paper.

* Corresponding author. Tel.: +46 72 734 29 52.
E-mail address: jakob.axelsson@sics.se (J. Axelsson).

http://dx.doi.org/10.1016/j.jss.2015.12.020
0164-1212/

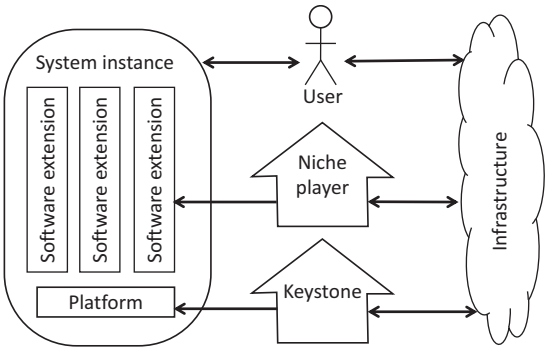

Fig. 1. Technical and organizational concepts in a software ecosystem.

1.1. Definition of software ecosystems

Although most of the work on software ecosystems use the term in similar ways, there is no definition which is universally agreed upon. In a recent systematic literature review (Manikas and Hansen, 2013), it is found that 44% of the papers do not state any definition at all. Among those that do provide a definition, the most common one (27%) used is the one by Jansen et al., who define a software ecosystem as "a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently under-pinned by a common technological platform or market and operate through the exchange of information, resources and artifacts" (Jansen et al., 2009).

Another commonly used definition (14% of the papers in the above study) is provided by Bosch and Bosch-Sijtsema: "A software ecosystem consists of a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs" (Bosch and Bosch-Sijtsema, 2010a, b).

Based on their review, Manikas and Hansen (2013) identify common elements in the various definitions, and propose the following definition, which is also the one we will use in this paper:

Definition. A *software ecosystem* is the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services.

The actors in the ecosystem are commonly divided into a *keystone* and a set of *niche players* (Iansiti and Levien, 2004). In the conceptual model used in the analysis in this paper, the keystone is responsible for developing the *platform* and also plays an important role in the success of the ecosystem through orchestration. The niche players develop *software extensions* which execute on top of the platform, and the extensions can be added by the *user* in different combinations. A *system instance*, whose quality we are interested in, thus consists of a platform combined with a set of different extension components (Jansen and van Capelleveen, 2013). To support the interaction between the actors, and between actors and systems, an *infrastructure* is needed. All these are illustrated in Fig. 1. (Some ecosystems may not map easily to this conceptual model, and in those cases, other challenges may be present when it comes to quality assurance.)

1.2. Definition of quality and quality assurance

The focus of this paper is quality assurance, but to able to define that, a discussion of quality in general is needed. Whereas software ecosystems are a relatively new phenomenon, product quality has been discussed for many decades. Already in the 1930s, in his effort to characterize quality and provide quality control mechanisms, Shewhart identified that "there are two common aspects of quality: one of them has to do with the consideration of the quality of a thing as an objective reality independent of the existence of man. The other has to do with what we think, feel or sense as a result of the objective reality. In other words, there is a subjective side of quality" (Shewart, 1931). In software engineering, a similar difference was much later captured in a description of validation vs. verification, where validation checks if we are building the right product, and verification if we are building the product right (Boehm, 1979). Both these definitions concern primarily how the product is performing during its operational use, and this is also the emphasis of this paper, where we focus on how well the system under operation fulfills the needs or requirements as defined (implicitly or explicitly) by various stakeholders.

For software systems, the word "quality" is however often used in a different and broader sense, which also covers many aspects related to the efficiency of development, maintenance, evolution, etc. of the system. The well-established quality standard ISO 9126 and its follower ISO 25010 are examples of this, where two of the quality characteristics are maintainability, defined as "the capability of the software product to be modified", and portability, defined as "the capability of the software product to be transferred from one environment to another" (ISO/IEC, 2001; ISO/IEC, 2011). Those are aspects which are beyond the primary scope of this paper, and instead our focus is quality in use, even though all aspects are sometimes intertwined with each other.

In addition, much research related to software ecosystems has a relation to software architecture, a field where it is common practice to discuss about the "quality attributes" of an architecture. The architecture is in itself an abstraction of the real product, and not all quality in use aspects are considered since they may relate more to details at a lower level than to the architecture. Also, the quality attributes of the architecture usually go beyond the quality in use, and emphasize how to manage the product during its development and maintenance phases, in order to reduce complexity.

The definition of quality we will use in this paper is as follows:

Definition. The *quality* of a software product is how the system under operation fulfills the needs and requirements as defined (implicitly or explicitly) by its stakeholders.

Given this definition of quality, quality assurance can be defined as follows:

Definition. *Quality assurance* is the set of activities carried out to ensure that the system has sufficient quality.

This includes activities for ensuring that the software product was designed to be fit for its intended purpose, i.e. validation, but also that mistakes in the implementation are eliminated, i.e. verification. However, quality assurance cannot be seen as an isolated activity, but rather as a set of activities that take place during different phases of the product's life-cycle. This life-cycle can be described in many ways, and one example is provided in the standard ISO12207 (ISO/IEC/IEEE, 2008) for software life-cycle processes. This standard defines four high-level process areas, namely agreement processes; organizational project-enabling processes; project processes; and technical processes, with sub-processes defined for each of them. For example, the technical processes include activities such as requirements definition; system architectural design; implementation; integration; and operations. Quality assurance related activities can occur in any of these processes.

Product quality has a relationship with the much broader concept of *ecosystem health* used by many software ecosystem researchers. This concept is defined as the ability of the ecosystem to endure and remain variable and productive over time (Manikas and Hansen, 2013), or alternatively as longevity and a propensity