Chapter 21

Framework for Managing Features of Open Service Ecosystems

Toni Ruokolainen
University of Helsinki, Finland

Lea Kutvonen
University of Helsinki, Finland

ABSTRACT

The recent increased use of Internet, social media, and networked business mark a development trend where software-based services flow to the open market for enabling service-oriented networked business. Our vision is that in the future, organizations and individuals collaborate within open service ecosystems. An open service ecosystem is characterized especially by the autonomy of its entities, its evolution with respect to available services and collaboration types, and dynamic establishment of collaborations. For facilitating collaboration establishment in open service ecosystems features of services and cooperation facilities, and feature inter-dependencies need to be governed rigorously. Towards this purpose we have established a framework for unambiguous description of service ecosystem features. The framework comprises a conceptual model which provides especially a categorization of features, and a formalization of the conceptual model as a meta-model for service ecosystems. We show that the corresponding feature categories have their specific roles and semantics as part of different ecosystem elements and in different phases of service ecosystem processes.

INTRODUCTION

The recent increased use of Internet, social media, and networked business mark a development trend where software-based services flow to the open market. Technological approaches like SaaS, SOA, and Web Services present tools and architectures for this: they provide protocols for accessing remote functionality encapsulated to a business-relevant units, declared available though service registries and manifests of service functionality, requirements for messaging platform support, information representation and semantics, and

DOI: 10.4018/978-1-61350-432-1.ch021

choreography (protocol) for exchanges in utilising the service.

However, this situation is uncontrolled and uncontrollable in several ways. First, the trustworthiness of the services marketed is unknown, as there is no guaranteed knowledge (facts) about their properties. The clientele is left to rely on declarations by the service providers. The declarations carry several risk aspects. The semantic of the declaration may be obscure due to the lack of shared vocabulary for describing service behaviour in functional and nonfunctional aspects exists. Furthermore, the declarations can be biased, as the cost of inaccurate declarations is not sufficient as an incentive.

Second, the interoperability between independently developed services is immature, especially in terms of nonfunctional properties. There is no commonly accepted framework for functionality and selectable properties or property management for those functions. Middleware platforms have built-in support for various transparency properties (e.g., location and access, data representation, transactionality) and various security technologies (e.g., encryption, non-repudiation), but as the groupings of properties differ, the interworking challenges still exist. Furthermore, the concepts of nonfunctional properties commonly refer to platform services, but in modern social networking and inter-enterprise collaboration scenarios, business and user oriented properties (such as policies for governing joint behaviour, pricing schemes, privacy preservation declarations) are relevant requirements.

Third, the current platforms weakly support collaboration management or concepts required for it. Concepts of contracts, parties, authority and ownership, policies and breaches of contracts causing sanctions are necessary for the different kinds of networked collaborations.

As a partial solution to these challenges, software ecosystems have become popular as a means for producing software applications more efficiently for heterogeneous clientele with varying requirements. A software ecosystem is typically based on a software platform provided by an organization. The platform is then used by internal and external developers for implementation of applications (Bosch & Bosch-Sijtsema, 2010). Software ecosystem strategy is utilized by companies such as Amazon [1] or Nokia [2] for establishing communities of developers and clientele over their own corresponding platforms.

While the software ecosystem approach emphasises the software production challenges, the open use of services from the open marketplace is stressed by service ecosystem approaches. A service ecosystem is an environment for creating and managing service-based collaborations, such as virtual organizations or service mash-ups, from services provided by a community of service providers. Service ecosystems exist currently especially in form of platform provider specific Software-as-a-Service (SaaS) -environments. The typical service ecosystems available currently are closed, meaning that the methods and technologies used for providing new services are pre-determined by the hosting environment, and service compositions and collaboration networks are determined statically during service development. Such closed ecosystems can not be applied in domains where services are to be provided and managed by autonomous entities, or when service collaboration networks are to be established dynamically on demand.

The challenge still remains to provide an environment where several service-oriented software engineering (SOSE) methodologies and distributed teams could produce services that easily can be organised into collaborations managed by dynamic contracts, because linkage between these two sides is missing.

The main architecture design must address a more complex situation where the clientele and the ecosystem itself have potential conflicts of interest in details, but still, the members of the ecosystem have incentives for collaboration both