Software Co-development in the Era of Cloud Application Platforms and Ecosystems: The Case of CAST

Dimitrios Kourtesis, Konstantinos Bratanis, Dimitris Bibikas, Iraklis Paraskakis

Software Co-development in the Era of Cloud Application Platforms and Ecosystems: The Case of CAST

Dimitrios Kourtesis, Konstantinos Bratanis, Dimitris Bibikas, Iraklis Paraskakis

[1] South-East European Research Centre,
International Faculty, The University of Sheffield
Thessaloniki, Greece
{dkourtesis, kobratanis, dbibikas, iparaskakis}@seerc.org

Abstract. Interest around cloud computing has been growing quite rapidly during the past few years, and the model of cloud computing is evolving into an indispensable component of innovation strategy across the software industry. We are witnessing a paradigm shift that will have a profound impact on software platforms and ecosystems and will give rise to new forms of software co-development. In this paper we make a first attempt to discuss the evolution of the relationship between software co-development, platforms and ecosystems in the era of cloud computing, and the role of cloud application platforms. We present the case of a cloud application platform designed to support advanced forms of software co-development, and to foster the emergence of a novel type of software ecosystem. As demonstrated, cloud application platforms can be designed in a way that facilitates the emergence of new forms of hierarchical cloud-centric software ecosystems.

Keywords: Co-development; Software Ecosystems; Cloud Application Platforms; Platform as a Service; PaaS.

1 Introduction

Software co-development represents a form of collaborative product development [1, 2] that has been gaining more and more attention. For many years, vendors have been practicing the development of commercial software products in relative isolation from others in the same industry. At some point however, they started realising the benefits of partnerships beyond their obvious role for software distribution, and started opening their products to co-development. Large-scale software products (e.g. operating systems) started to transform from single-vendor projects into platforms for co-development and software ecosystems [3, 4].

As a term, software co-development is used rather loosely to refer to several different models of collaboration in creating software—ranging from limited outsourcing partnerships to large-scale networks for open innovation. In this paper we appeal to a notion that is closer to the latter, and examine how co-development as a practice is affected by the advent of cloud application platforms.

The contribution of this paper is twofold. First, we discuss the concept of software co-development in relation to software platforms and software ecosystems, and provide our view on how the relationship between these concepts evolves in the era of cloud computing, giving contemporary examples of major cloud-platform-centric environments for software co-development. Second, we present the case of a cloud application platform that was designed with the objective of supporting advanced forms of software co-development [5]. We place emphasis on the features of the platform that are particularly aimed at making this possible, and discuss implications with respect to the future of co-development on cloud application platforms.

2 Software Co-development and Software Ecosystems

In the past, software products were largely created by vendors in relative isolation from their wider community [6]. At some point, however, software companies started becoming aware of the benefits of external collaborations and networked operations [7]. Software vendors realised that by bringing more partners into their development process (and by being involved into others' supply chains), they could gain increased functionality and keep customers loyal with less capital investments [8]. The previously "fixed" supply chain model of collaboration in the software industry started giving way to a fuzzy partnership approach, where virtually infinite numbers of partners could add value upon a central product [9]. Large-scale software products started to transform from single-vendor projects into platforms for co-development and software ecosystems.

In this new context, the platform provides a central coordination mechanism for software development. Irrespective of the degree of separation between the platform core and each member of the network, there can be many advantages for everyone involved: decreased software and business development costs, quicker time-to-market, improved focus, reduced complexity, and of course, economic profit [6]. In some cases a software platform is open for all interested partners to commit their resources - as in free and open source projects like Apache, Linux, etc. In other cases, a platform is closed and owned by a central partner who controls access levels and/or contributions by third parties (e.g. Facebook/Apps, iPhone/Appstore, etc). As long as there can be benefits to the network [7], individual developers and companies will continue to incorporate their contributions to the platform core, making them available for further use by other parties. Open collaboration between companies in the software industry is evolving into a standard practice.

3 Software Co-development in the Context of Cloud Computing

The models of Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), represent new ways of thinking about the delivery of computing capabilities within the emerging paradigm of cloud computing. In all its different forms, cloud computing has been gaining more and more attention during the past few years, and is rapidly evolving into an indispensable component of

innovation strategy across the software industry. We are witnessing a paradigm shift that will have a profound impact on software platforms and ecosystems as we know them, and will give rise to new forms of software co-development. A predominantly important trend is the rise of cloud application platforms.

3.1 Cloud Application Platforms

Cloud application platforms offer a combination of some form of computing infrastructure that is made accessible over the internet, and a set of tools and services which allow developers to create applications and have them deployed and executed over that infrastructure. They are often referred to as *aPaaS* (application-Platform-as-a-Service) [12], so as to avoid confusion with other types of PaaS offerings which have different objectives, such as *iPaaS* (integration-Platform-as-a-Service) or *bPaaS* (business-process-Platform-as-a-Service) [11].

For companies interested in creating new applications and making them commercially available in the form of SaaS offerings, adopting a cloud application platform carries many benefits. Development of applications against a platform of this kind allows a significant portion of the effort traditionally required for engineering, distributing and maintaining web applications to be shifted to the provider of the platform. This in turn allows application developers to concentrate on what they know best, i.e. on their domain-specific problems and solutions, rather than the setup and operation of a supporting infrastructure.

Seen from a platform provider's perspective the value proposition of the aPaaS model is different. Most importantly, it allows software vendors to realise new models of partnership and co-development while leveraging their potentially existing partner networks. In many cases, the goal for vendors engaging in this model is to transform one of their core products into a platform that fosters the emergence of a software ecosystem. In other cases the goal is to create an ecosystem for software co-development that doesn't revolve around a central product or application domain.

Results from recent surveys suggest that the market around this cloud computing model is still immature and fast-changing [12]. However, it is anticipated that the vendors who will succeed in creating ecosystems with a critical mass of developers will also attract a large community of users, particularly those who, in addition to richness of software features, also seek safety in numbers [11].

3.2 Cloud-platform-centric Software Ecosystems

In the following paragraphs we provide an overview of some examples of software ecosystems centred on the platform offerings of major cloud service providers.

Force.com is a cloud application platform offered by Salesforce.com – presently the leading SaaS vendor in the domain of Customer Relationship Management (CRM). Force.com supports the development of custom applications by providing a comprehensive stack of database, integration, logic and user interface capabilities on top of the core technology used in the CRM environment of Salesforce.com. The custom applications can be used either independently or as extensions of the core

CRM. A developer can publish their application to the AppExchange application marketplace, allowing end-users to find it and buy it. The form of co-development enabled by Force.com is the creation of custom applications by third-parties which introduce new features to the platform provider's core CRM product.

In 2010 Salesforce.com acquired Heroku, the dominant cloud platform for developing applications in the Ruby programming language. Except for Ruby, Heroku supports several other technologies such as Java, Python, and Scala. Heroku offers an add-on provider program for third-party Independent Software Vendors (ISVs). Third-party ISVs can use a self-service portal and development kit in order to offer their services as add-ons to the Heroku platform. As such, the Heroku platform is co-developed by being continuously extended with more features (services) that other developers can use for creating new applications on the platform.

Google Apps is another popular software ecosystem. Google Apps provide APIs to ease the integration between third-party applications with the core Google Apps (e.g., Google Docs, Google Calendar and others). The third-party applications can be either provisioned by a third-party infrastructure, or developed against the Google Apps Engine. The Google Apps Engine is a cloud platform for developing applications in various programming languages, such as Java, Python and Go. Third-party ISVs can make their applications available in the Google Apps ecosystem by publishing them to the Google Apps Marketplace. In short, the form of co-development is allowing third-party ISVs to build solutions that interact with one or more of Google's core products, and are mostly hosted and executing outside Google.

Windows Azure is a cloud application platform for developing software applications using the .NET Framework. An integral part of the platform is the Windows Azure Marketplace. Windows Azure offers a third-party ISVs scheme which aims to help ISVs bring SaaS solutions to the market faster. Publishing their SaaS applications and datasets in the marketplace allows ISVs to reach a global market of customers using an integrated environment that provides comprehensive management of their services (e.g., self-service on-boarding, creation of terms of use and trial offers). In this co-development model, third-party ISVs partner-up with the Windows Azure platform to co-develop new SaaS products.

4 The CAST Platform

The CAST project was a collaborative EU-supported research effort that begun in 2009 and finished in 2011[1]. It was set up to investigate the engineering challenges associated with realising a cloud application platform that enables the development and delivery of on-demand (SaaS) business applications. One of the central requirements for the CAST platform's design was to ensure that the way in which development and delivery will be carried out will promote positive network effects [14] and will foster the emergence of a software ecosystem around the platform. Instrumental in achieving this design goal was to employ an appropriate model of software co-development that maximises collaboration and reuse of resources.

[1] CAST: Enabling Customisation of SaaS Applications by Third parties (www.cast-project.eu)

4.1 CAST Concepts and Terminology

Before discussing the model of software co-development employed in the CAST platform, it is necessary to introduce some key concepts and terminology.

CAST platform solutions. A *solution* is defined as a complete enterprise software application that targets a specific application domain or market niche (e.g. customer relationship management for French insurance companies, or event management for Greek exhibition centres). It is deployed on the CAST platform and made available to end-users as on-demand software (SaaS). A solution is not manifest as executable artefacts – there are no code binaries in a solution, just metadata. This is because a solution is effectively a logical bundle of finer-grained components which provide the actual functionality.

CAST platform apps. The finer-grained components that solutions are composed of are called *apps*. Each app within a solution provides a highly-specialized function. An app can be data-centric or process-centric. A data-centric app provides the implementation for creating, viewing, editing and storing a custom-built data object (for example, an employee's record, or a project's timesheet). A process-driven app provides the implementation for supporting an end-user in carrying out a sequence of tasks (for instance, supporting a sales employee for mass-importing customer addresses from a spreadsheet file). Apps can (albeit are not required to) affect all of the platform's runtime layers. That is, an app may define new data object types on the data layer, new business operations on the business logic layer, and new user interface elements on the presentation layer. An app's behaviour can be extended by creating so called app extensions which interface with the app at designated extension points. An app extension is therefore not a standalone component, but functions as a plug-in to one or more apps.

External services. Apps and app extensions may rely on external services to deliver part of their functionality. By external services we refer to systems that are deployed and executing outside the platform and are accessible over the Web, through a programmatic interface (i.e. REST and SOAP Web services). The ability to use Web services enables the developers of solutions to leverage already existing (and tried) solutions for particular specialized tasks within their apps. For example, an app or app extension for contact management could invoke an external service to perform email address validation for a particular contact, or to obtain stock quote information for a contact's company.

4.2 Development and Delivery on the CAST Platform

The platform constructs presented above represent a generic model of abstraction that can be applied to a wide range of cloud application platforms. But how do these constructs map to specific roles in a cloud platform ecosystem? Who creates and who extends those constructs in the context of co-development?

Figure 1 illustrates the mapping between platform constructs and the different ecosystem roles through an abstract example.

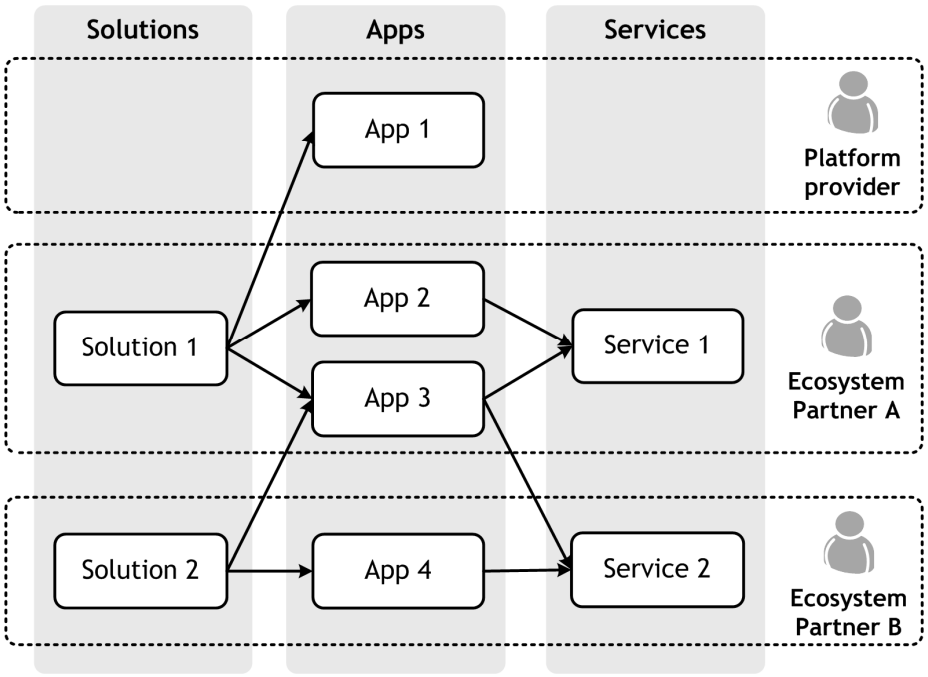

Fig. 1. Example mapping of platform constructs to ecosystem roles

Apps and app extensions may be built both by the platform provider and by third-parties (ecosystem partners A and B). In order to help developers to bootstrap their work, the platform provider may build a number of apps that target functionality that is rather common in business applications, such as document management (App 1). Any partner that needs to use a built-in app is allowed to configure it for the needs of a particular solution (Solution 1). Alternatively, apps and app extensions can be developed from the ground-up by an ecosystem partner (Apps 2, 3, 4). Optionally, those apps can depend on external services (Apps 2, 3, 4) which may not necessarily be owned by the same ecosystem partner (App 3, Service 2). In any case, as soon as a third-party app, app extension, or external service is added to the platform it can be made available for other partners to reuse in their own works (opting-out of reuse could theoretically be offered as an option).

Composing built-in and third-party apps (and app extensions) into solutions is the responsibility of ecosystem partners (Solution 1, 2). In creating a solution package ecosystem partners are also specifying how the appearance and behaviour of the included apps should be customised (at run-time) for the particular solution at hand. This is done by defining solution-specific constraints on the apps. Since an app can be part of more than one solution (App 3), different constraints can be active for a particular app depending on the execution context. A typical constraint is a domain-specific restriction of the allowed range of values for some field (which may be unbounded in the general data model for an app). For example, data validation rules for fields such as a postal code or a vehicle license plate can be customized differently depending on the country a solution targets.

4.3 CAST Platform Model of Co-development

Enabling developers to build applications by mixing and matching components contributed by third parties within an ecosystem is an increasing trend in the space of cloud application platforms [15]. A distinctive characteristic of the CAST platform, however, is that it allows developers to create applications (solutions) by reusing not only low-level services offered by the provider of the platform or third-parties, as is usually the case, but also entire applications (apps) developed and deployed by third-parties. The third-party apps to be reused can be customised to fit new needs, integrated with external systems via Web services, and combined into a package that is resold as a distinct on-demand business application (solution).

This model allows co-development relationships to be formed not only among the platform provider and individual ecosystem partners, but most importantly, among ecosystem partners themselves. This gives rise to a model of many-many co-development relationships, as opposed to the traditional model of one-to-one collaboration. Figure 2 provides an illustration of the two alternative models.

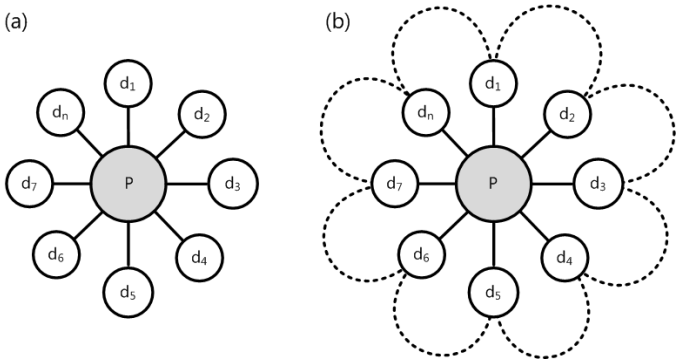

Fig. 2. Alternative models of software co-development: one-to-one (a) vs. many-to-many (b) co-development relationships (P=platform; d_i=developer)

The model of co-development that CAST employs allows members of a software ecosystem to be linked not only to the central platform provider, but to each distinct partner. Each partner can be directly associated with other ecosystem members and can become a centre around which others develop their own work, resulting in a multi-centric co-development environment. This approach allows for fundamentally new forms of collaboration within a cloud-centric software ecosystem.

5 Conclusions

In this paper we attempted a first discussion on how the relationship between co-development, platforms and ecosystems evolves in the era of cloud computing, and specifically how cloud application platforms contribute in shaping the future of software co-development. We have also presented the model of co-development employed by CAST—a cloud application platform designed with the objective of

supporting collaboration between a PaaS provider and an ecosystem of SaaS developers.

As we have demonstrated, cloud application platforms have some novel characteristics that are of great interest in relation to software co-development. Firstly, cloud application platforms embody both the core software artefact on which co-development is centred—i.e. the core software product that partners extend and/or out of which new products materialize, but also the central mechanism for coordinating the ecosystem and the software co-development process. Secondly, as demonstrated by the case of CAST, cloud application platforms can be designed in a way that allows co-development relationships to be formed not only among the platform provider and individual ecosystem partners, but most importantly, among ecosystem partners themselves. The combination of those two characteristics with the agility intrinsic in the cloud computing paradigm has far reaching implications for software co-development. Most notably, it allows ecosystems to be rapidly formed not only around the platform core, but also around contributions by third-parties, thus giving rise to new forms of hierarchical cloud-platform-centric software ecosystems.

References

1. H. Chesbrough, K. Schwartz. Innovating Business Models with Co-Development Partnerships. Research Technology Management 50(1). (pp. 55-59). (2007).
2. G. Buyukozkana, J. Arsenyanb. Collaborative product development: a literature overview. Production Planning & Control: The Management of Operations, 23(1). (pp. 47-66). (2012).
3. M.H. Meyer, R. Seliger. Product Platforms in Software Development. Sloan Management Review, vol. 40, no. 1, pp. 61-74 (1998).
4. D.S. Evans, A. Hagiu, R. Schmalensee. A Survey of the Economic Role of Software Platforms in Computer-based Industries. CESifo Economic Studies, vol. 51, no. 2, pp. 189-224 (2005).
5. D. Kourtesis, V. Kuttruff, I Paraskakis. Optimising development and deployment of enterprise software applications on PaaS: The CAST project. In ServiceWave 2010 Workshops (pp. 14–25). LNCS, 6569/2011. Springer. (2010).
6. Enabling Open Collaboration with Development Partners. CollabNet. (2007).
7. G. Parker, M. Van Alstyne. Managing Platform Ecosystems. In Proceedings of the 29th International Conference on Information Systems (pp. 1-24). (2008).
8. J. Bosch. From software product lines to software ecosystems. In Proceedings of the 13th International Conference on Software Product Lines. LNCS. Springer. (2009).
9. D.G. Messerschmitt, C. Szyperski. Software Ecosystems, Understanding an Indespensable Technology and Industry. The MIT Press, Cambridge. (2003).
10. M. Pezzini, B.J. Lheureux. Integration Platform as a Service: Moving Integration to the Cloud. Gartner Research. (2011).
11. Y.V. Natis, B.J. Lheureux, M. Pezzini, D.W. Cearley, E. Knipp, D.C. Plummer. PaaS Road Map: A Continent Emerging. Gartner Research. (2011).
12. J.R. Rymer, S.Ried. The Forrester Wave: Platform-as-a-Service for App Dev and Delivery Professionals, Q2 2011. Forrester Research. (2011).
13. D.S. Evans, A. Hagiu, R. Schmalensee. Invisible Engines: How Software Platforms Drive Innovation and Transform Industries. MIT Press. (2006).
14. D.C. Chou. Rise of the Cloud Ecosystems. MSDN Blogs. 16 Mar. 2011.