Free/Open Services: Conceptualization, Classification, and Commercialization

G.R.Gangadharan[1], Vincenzo D'Andrea[1] and Michael Weiss[2]

[1] Department of Information and Communication Technology, University of Trento, Via Sommarive, 14, Trento, 38050 Italy
{gr,dandrea}@dit.unitn.it
[2] School of Computer Science, Carleton university, 1125 Colonel By Drive, Ottawa, K1S 5B6, Canada
weiss@scs.carleton.ca

Abstract. The concept of Free/Open Services (F/O-Services) emerges by bringing together services with Free/Open Source Software (FOSS). F/O-Services enable the creation of transparent composite services collectively and allow people and other services to access them. This paper extends the concept of F/O-Services beyond the level of open interfaces, analyzing the associated licensing interpretations and exploring the notion of open service dependencies. Further, the paper overviews the business models for F/O-Services as a part of this social mechanism of exchange.

1 Introduction

Software has, traditionally, been perceived as a product, requiring possession and ownership, in order to receive the desired performance. The common model for software use is to install and execute on a computer owned by the user or his/her organization. This model is overridden by Software-as-a-Service [1], a mechanism for renting software where users subscribe to the software they use. Service oriented computing (SOC) allows the software-as-a-service concept to expand to include the delivery of complex business processes and transactions as a service, allowing applications to be constructed on the fly and services to be reused everywhere and by anybody [2]. The two important motivations for opening interfaces through services are as follows:

- The trend toward componentization and commoditization of business functionality [3, 4] means that a component-based business will focus on its value-added functionality, and outsource non-value-added functionality.
- Opening interfaces leverages external innovation, as amply demonstrated by the great variety of mash-ups built using the Google Maps interfaces [5].

In general service consumers have no access to the implementation details of a service, including whether or not a service uses other services, and what are these other services. An approach similar to Free/Open Source Software (FOSS)

Please use the following format when citing this chapter:

Gangadharan, G.R., D'Andrea, V. and Weiss, M., 2007, in IFIP International Federation for Information Processing, Volume 234, Open Source Development, Adoption and Innovation, eds. J. Feller, Fitzgerald, B., Scacchi, W., Sillitti, A., (Boston: Springer), pp. 253–258.

that opens the availability and accessibility of the source code of services would significantly enhance the understandability of the service composition process (including data and control flow) and allow the creation of derivative services.

Free/Open Source Software (FOSS) is an encompassing term for the development models, legal terms, and sociological issues associated with this novel software paradigm [6].The acronym FOSS combines the views of Free Software and Open Source Software and implies the commonalities between these approaches. Inspired by the success of FOSS, we have conceptualized and analyzed the implications of Free/Open Services (F/O-Services) in [7]. In this paper, we extend our previous work by adding service dependencies to the notion of F/O-Services and elaborate the explicit expression of dependencies in licenses.

2 Free/Open Services

A service is represented by an interface part defining the externally visible functionality (and typically some non-functional properties [8]) and a realization part implementing the interface [9]. Generally a service would be available when invoked by another service/entity, but remains idle until the request arrives. Services provide universal interoperability, manifested by the web-like network of services created by the composition of lower level services into higher level services [10]. Composite services could be created dynamically based on functional and non-functional requirements. Individual services can be replaced in case of malfunctions or due to the changes of requirements. A truly dynamic service oriented system can achieve software evolvability. The dynamics and composability of services are at the core of service orientation[1].

The opaque nature of services often hides the details of operations from the service consumer. The consumer could neither see anything beyond the interface nor understand about the services being composed in a composite service.

A F/O-Service is inspired from FOSS concepts and is characterized by the following principles [11]:

- The source code of the interface (WSDL descriptions) as well as of the implementation should be available.
- The service should be allowed for modification and the modified services should be freely distributable.
- The service should allow derivation and should be freely reimplementable and executable.

A F/O-Service allows the access to the source code of interface as well as its realization, making composite services and derivative services. We extend the F/O-Service philosophy by introducing the term dependency.

[1] In this paper, we do not explore the implications of dynamic nature of service composition.

We define dependency between services as the description of the interactions of a service with other services. Interactions do not have a direction per se, but a dependency does. A dependency link is directed from the service user to the service provider. Consider a service A, which composes the services B, C, and D. Further, these services compose E, F, G, and H. Given the service A, we could not understand what the services are being composed in A. If we make

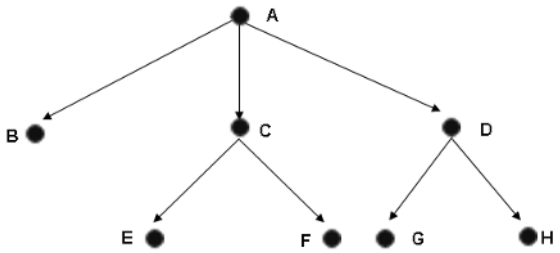

Fig. 1. Dependencies of a service

the dependencies of services open, we could achieve a service, whose service internals are completely exposed to the consumer. In Figure 1 circles and arrows represent services and their dependencies. From the given dependency graph, we could recognize the complete hierarchy of composed services. This approach is quite similar to white box description of components [12].

There are at least two notions of openness in the context of dependencies[2]:

1. The service declares which services it uses, but this does not imply a right for the consumers to invoke those services directly, if the service provider wishes to protect the intellectual property inherent in the composition and
2. The service allows others to reuse the relationships with other services it has. Restrictions imposed by the component services apply, of course.

These notions are not all-encompassing. For instance, there may be intellectual property rights attached to the selection of services during composition.

3 Free/Open Services Classification

Like software, a service is also an asset transferring an inherent value from the provider to the recipient. A service is a self-contained implementation of specific operations with a well-defined interface. However, the nature of services [7] precludes the direct adoption of software licenses for services.

Copylefting is the process of imposing copyright law to remove limitations on distribution and modifications, requiring to preserve the same freedoms in

[2] Opening dependencies implies only the provision of a list of composed services, and differs from fully exposing the application logic used to compose them.

the modified versions. From the perspective of service providers and developers, copylefting of services could be seen as a restriction imposed on the new service, that allows the value addition solely with the same conditions as the original. However, from the perspective of the service consumer, copylefting could be viewed as an ultimate guide for using any value added services inheriting from a particular copylefted service.

Free/open source software, in general, could be either copylefted or not. However, in addition to the dimension of source code, F/O-Services have another dimension: execution/usage. Unlike software, services are not resident in the recipient's environment. Though FOSS licenses do not discriminate in the uses of a software, the dynamic binding and execution of services could enforce certain restrictions for the execution/usage of F/O-Services. These restrictions could be of several kinds, for example,

- a service should be allowed to be executed for a certain number of times;
- a service should be used for certain purposes (for example, academic use);
- a service should require payment from the consumer for execution.

Now, considering the continuum of execution/usage with respect to copylefting in association with the openness of source code, we classify the F/O-Service licenses as follows[3]:

Unbounded Licenses belong to the most permissive family of F/O-Service licenses. These licenses allow licensees to use the service (and its source code) for any purposes without any restrictions. These licenses are wholly unrestricted for any kind of value addition of services.

Disjoint Licenses require no copylefting on any value addition but imposing restrictions on usage/execution of the service. These licenses are disjoint (unrelated) from the licenses of parent services.

Confined Licenses are the family of licenses where execution/usage may be restricted and could allow any kind of value addition, provided the value added services could be licensed under the same family. These licenses enrich the F/O-Services community.

Accessible Licenses refer to the family of licenses where execution/usage may be unrestricted and could allow any kind of value addition, with the requirement of copylefting.

The taxonomy of F/O-Service licensesis tabulated as shown (see Table 1):

Table 1. Taxonomy of F/O-Service Licenses

	Execution Restricted	Execution Unrestricted
Copylefting	*Confined Licenses*	*Accessible Licenses*
Non copylefting	*Disjoint Licenses*	*Unbounded Licenses*

[3] We support the rights of a service provider to make use of any license for their service, and highly recommend that service providers obtain appropriate legal advice regarding their selection of a service license.

4 Free/Open Service Business Strategies

Business strategies are the specifications of complex real world descriptions for managing a business by an organization in a sustainable way. There exists a wide range of business models for FOSS [13] as well as for services [14].

A service may be available at no cost. However, it could motivate the consumer to purchase something. Like traditional commercial software, services could begin their product life cycle as closed and then could become open when appropriate. Further, F/O-Services could adopt a complementary service/product scheme where revenue comes from media distribution, branding, training, consulting, and custom development. Also, by following the dual licensing strategy, a service can be licensed under both an open source inspired license and a proprietary license.

Service hosting is another business strategy for F/O-Services. A F/O-Service provider could host the services defined by others, thus making a viable business opportunity. A service host provides the capacity for executing a F/O-Service.

The intermediary and shared infrastructure models [15] can also be adapted to F/O-Services. One type of intermediary is a service aggregator. It adds value by composing other services so that a new functionality arises that was not available before. Intermediaries may also add value through the pre-selection of component services and managing their quality. A shared infrastructure service is an open service jointly developed by service users or providers for their common usage. In this case, it is more economic to share the development costs rather than developing the capabilities provided by the services individually.

5 Concluding Remarks

Though the standards of services are open, the domain of SOC is not enjoying the freedom of openness till now. As freedom of distribution and freedom of modification are the core principles of free and open source licensing, we think an approach inspired by FOSS for conceptualizing the service licenses would be beneficial to the services community. To the best of our knowledge, [11] is the first published work heralding the rise of F/O-Services. Following this work, very recently, there are few informal blogs or writings scattered in the Internet [16, 17]. However, these works neither conceptualize F/O-Services completely nor intend to formalize the representation of licenses for F/O-Services. exploring the concept of dependencies.

Exposing the service dependencies could matter to the service consumer for a number of reasons, such as the consumers could deny a service that composes a service from a particular service provider. Service providers fail to get sufficient guidance for the implementation from the inadequate interfaces (the partial information conveyed through current service interfaces) [10]. Similarly, service consumers could make incorrect assumptions about the service implementation given these inadequate interfaces. The opening of service increases the quality

of service integration by reducing the number of composition errors due to misinterpretation of service interfaces, and failures due to hidden side-effects, thus reducing the cost of developing and maintaining composite services.

References

1. Elfatatry, A., Layzell, P.: Negotiating in Service Oriented Environments. Communications of the ACM **47**(8) (2004) 103–108
2. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services Concepts, Architectures, and Applications. Springer Verlag (2004)
3. Sanford, L., Taylor, D.: Let Go to Grow: Escaping the Commodity Trap. Prentice Hall (2005)
4. Cherbakov, L., Galambos, G., Harishankar, R., Kalyana, S., Rackham, G.: Impact of Service Orientation at the Business Level. IBM Systems Journal **44**(4) (2005) 653–666
5. Mulholland, A., Thomas, C., Kurchina, P.: Mashup Corporations: The End of Business as Usual. Evolved Technologist Press (2006)
6. Feller, J., Fitzgerald, B.: A Framework Analysis of the Open Source Software Development Paradigm. In: Proc. of the 21st Annual International Conference on Information Systems. (2000) 58–69
7. D'Andrea, V., Gangadharan, G.R.: Licensing Services: The Rising. In: Proceedings of the IEEE Web Services Based Systems and Applications (ICIW'06), Guadeloupe, French Caribbean. (2006) 142–147
8. Wang, G., MacLean, A.: Software Components in Contexts and Service Negotiations. In: CBSE Workshop. (1999)
9. Papazoglou, M., Georgakopoulos, D.: Service Oriented Computing. Communications of the ACM **46**(10) (2003) 25–28
10. Weiss, M., Esfandiari, B., Luo, Y.: Towards a Classification of Web Service Feature Interactions. Computer Networks **51** (2007) 359–381
11. D'Andrea, V., Gangadharan, G.R.: Licensing Services: An "Open" Perspective. In: Open Source Systems (IFIP Working Group 2.13 Foundation Conference on Open Source Software), Vol. 203, Springer Verlag. (2006) 143–154
12. Szyperski, C.: Component Software: Beyond Object Oriented Programming. ACM Press, New York (1998)
13. Raymond, E.: The Magic Cauldron. http://www.catb.org/ esr/writings/magic-cauldron/magic-cauldron.html (1999)
14. Hagel, J.: Out of the Box: Strategies for Achieving Profits Today and Growth Tomorrow Through Web Services. Harvard Business School Press (2002)
15. Weill, P., Vitale, M.: Place to Space: Migrating to E-business Models. Harvard Business School Press (2001)
16. Slashdot: Web Services and Open Source at OSCON. http://developers.slashdot.org/article.pl?sid=06/07/26/1537213 (Posted on July 26, 2006)
17. log.ometer.com: Log for July, 2006. http://log.ometer.com/2006-07.html (Posted on July 29, 2006)