Patterns for Licensing Web Services*

G.R. Gangadharan, University of Trento, Trento, Italy
Michael Weiss, Carleton University, Ottawa, Canada
Vincenzo D'Andrea, University of Trento, Trento, Italy

Abstract

The trend towards providing software as a service has required us to rethink the way software is licensed. There are different types of proprietary and open source licenses for software. However, the nature of web services differs significantly from traditional software and software components, which prevents the direct adoption of their respective licenses. As web services can be accessed and consumed in a variety of ways, there is a also spectrum of licenses for web services. We have mined existing licenses for web services for commonalities, and present the different licensing options in the form of patterns.

1 Introduction

Increasingly, software is provided as a service. Specific examples include the Google web services, the Amazon cloud computing service, or the StrikeIron data services. Though web services are software, they differ from traditional software in many ways:

- Services are executed in a hosted infrastructure. Consumers do not install applications, and a new class of issues is created by the accessing the service over a network.
- Services are designed to facilitate reuse. Services abstract from language and platform-specific aspects of the underlying software (loose coupling).
- Services encourage composition. Service composition allows consumers to build coarse grained services by combining finer grained services to any level of hierarchy.
- Services are data-driven applications ("data is the next Intel Inside" [1]). We need to distinguish between the use of a service as software, and the use of the data it provides.

These differences prevent the direct adoption of licenses for traditional software and software components [2]. By a license, we refer the terms and conditions that accompany a piece of software or a service. The party defining those terms is known as the licensor. Licensing principles reflect the overall business value of software to its producers and consumers. Licensing is often also used to provide protection to software producers for their intellectual property rights, and thereby becomes a source of revenue and a tool for business strategy. Licenses for web services reflect the differences between traditional software and web services: they govern

the execution, reuse and composition of the services. Although of great practical relevance, licensing of the data provided by services is out of scope for this paper.

In this paper, we classify web service licenses as proprietary or open [3]. Proprietary software licenses allow the execution of the software (including components) in the licensee's computing environment. Open source licenses allow consumers to view, modify, and share the source code and redistribute the software either for commercial and/or non-commercial purposes. However, for web services, we also need to consider their execution/usage. Execution of a web service refers to access/use (invocation) of the web service by another service.

A web service has an interface part, which defines the externally visible functionality (and typically some non-functional properties), and an implementation part, which realizes the interface. The opaque nature of services often hides the details of operations from service consumers. A consumer can be restricted from either seeing anything beyond the interface, or understanding how a service is composed from other services. Which parts of a service are made accessible to the consumer in addition to the (always accessible) interface, and which additional rights the consumer is given, is determined by the terms of the license.

Below, we describe patterns for licensing proprietary services and open service. The relationship between these patterns is shown in the pattern roadmap in Figure 1. The diagram uses circles to represent the common context shared by a group of patterns, and rounded rectangles to represent patterns. This gives us a way to refer to a group of patterns which solve related problems, for example, all the patterns relating to limiting execution. The audience for these patterns includes managers of companies who offer software as a service, and developers who need to understand the business impact of those licenses. Our focus is, therefore, on the strategic aspects of licensing web services, not on the underlying technology.

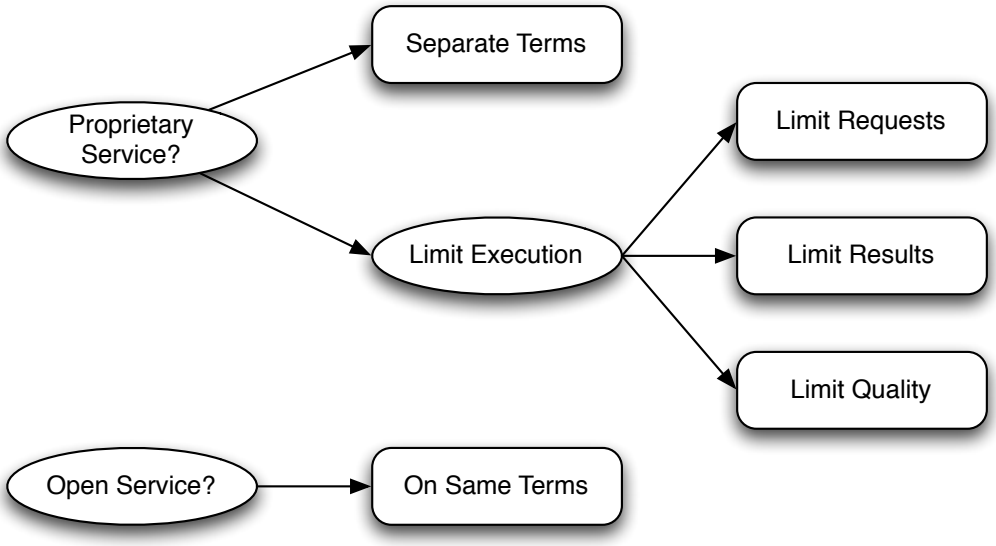

Figure 1: Roadmap for web service licensing patterns

2 Separate Terms

2.1 Example

Banca Indica offers a Daily Exchange Rate web service through which consumers can calculate foreign exchange rates based on the daily rates offered by Banca Indica. It also keeps a historical data record of exchange rates based on Daily Exchange Rate, which can be accessed via another web service, Historical Exchange Rate. Banca Indica offers Daily Exchange Rate as a free service and the Historical Exchange Rate service at 1 euro per use.

2.2 Context

A service provider that offers web services.

2.3 Problem

How can a service provider use the terms and conditions of a service license to attract consumers of its free service to subscribe to its premium service?

2.4 Forces

Free services provide an incentive to users to upgrade to paid-for services. They allow consumers to evaluate the service, before they make a decision to purchase the service. However, the free services should not be sufficient for power users, who would be willing to pay for a more advanced service. It must be beneficial for those users to upgrade. Providing free services is also generally not free for the service provider itself (development costs, hosting fees, sublicenses). Hence, the cost of providing the free service must be carefully balanced against any potential revenue from premium services, and non-monetary benefits (reputation).

2.5 Solution

License free and premium services under separate terms and conditions. For example, free web services can be licensed under one set of terms that specify which services (or service operations of a single service) are offered for free, and under what restrictions they are provided (see also Limit Execution). Paid-for services can be offered under terms that define their usage terms (which are more attractive to power users) and pricing terms.

2.6 Example Resolved

Banca Indica offers the Daily Exchange Rate service under the following terms and conditions:

1. The service cannot be composed with other services.[1]
2. The service can be used without a fee.

Banca Indica also provides an Historical Exchange Rate service under a different set of terms and conditions, which are more attractive to power users:

[1] This example should not suggest that, in general, free services cannot be integrated into other applications. The payment and composition terms of a license are orthogonal to one another.

1. The service can be composed.
2. The service requires a payment of 1 euro per use.

2.7 Consequences

Providing different versions of a service under separate terms gives the service provider control over how the service is used. The restrictions of the free service will not satisfy more demanding consumers, who will want to upgrade to the premium version. Yet, while they are evaluating the service, these same restrictions (eg non-commercial use) may be acceptable. However, when providing a version of the service for free, there is a risk that the service provider overestimates the potential demand, and its offering is satisfactory to all consumers.

2.8 Known Uses

The general Amazon Web Services (AWS) Licensing Agreement[2] states:

> The services covered by this Agreement include both free services that AWS and its affiliates (referred to together herein as "we" or "us") make available for no fee, for the purpose of promoting sales on the Amazon.com website and related websites and for other purposes (the "Free Services"), and services that we make available for a fee (the "Paid Services").

The Amazon Simple Queue Service (SQS) is a premium service provided by Amazon that offers a reliable, scalable and hosted queue for storing messages as they travel between computers. Below, we quote from the pricing options for the Amazon SQS service:

> Pay only for what you use. There is no minimum fee.
>
> **Requests**
>
> USD 0.01 per 10,000 Amazon SQS Requests (USD 0.000001 per Request)
>
> Amazon SQS requests are CreateQueue, ListQueues, DeleteQueue, SendMessage, ReceiveMessage, DeleteMessage, SetQueueAttributes and GetQueueAttributes.
>
> **Data Transfer**
> USD 0.10 per GB - all data transfer in
> USD 0.18 per GB - first 10 TB / month data transfer out
> USD 0.16 per GB - next 40 TB / month data transfer out
> USD 0.13 per GB - data transfer out / month over 50 TB
>
> Data transfer "in" and "out" refers to transfer into and out of Amazon SQS. Data transferred between Amazon SQS and Amazon EC2 is free of charge (i.e., USD 0.00 per GB).

Hence, the general AWS Licensing Agreement lists the terms and conditions common to all free and paid services. Pricing terms for premium services are listed separately.

[2] http://www.amazon.com/AWS-License-home-page-Money/b?ie=UTF8&node=3440661

2.9 See Also

Separate Terms is not specific on how the service provider should select the terms for the different versions of the service. The Limit Requests, Limit Responses, and Limit Quality patterns discuss ways how the service provider can limit service execution.

License patterns for proprietary software have been described by Kaminski and Perry [5]. They can be used by developers to select an appropriate license type for their software.

Segmenting customers into free and premium is an application of Segmented Customer [4].

3 Limit Execution

Rather than a pattern, this section describes a common context for the following three patterns. The common starting point for these patterns is a service provider that uses Separate Terms to attract users to its service by offering different licenses for high-end and low-end versions of the service. Now, the provider needs to devise license restrictions that create an incentive for users to upgrade the service, once they have had a chance to evaluate the service. The service may also not be ready for real deployment, but the provider wants to create attention around the service without creating a wrong impression of the potential capabilities of the service.

The solution involves imposing constraints on the execution of the web service. The patterns in the next three sections suggest different ways how this can be done: Limit Requests, Limit Results, and Limit Quality. These strategies could also be applied in combination.

4 Limit Requests

4.1 Example

A foreign currency exchange service Xenon provides buy and sell rates of large-value transactions in global currency markets. Any registered service consumer can make requests (invocation to the Xenon service) up to 10 times per day.

4.2 Context

A service provider that uses Separate Terms for free and premium versions of its service.

4.3 Problem

How can a service provider use license terms to control the execution a web service?

4.4 Forces

The number of times a service can be "freely"[3] executed should be more than sufficient for light use or use during development, but it should not allow heavy use of the service.

4.5 Solution

Impose constraints on the number of invocations of the web service. These restrictions may include the number of times a web service can be executed, predefined purposes for which it can be used, the type of user (e.g. free to academic institutions), or the level of payment.

4.6 Example Resolved

The Xenon service includes the following terms to restrict the invocation to no more than 10 times per day:

> You should not use more than 10 invocations per day to the Xenon service.

4.7 Consequences

A user who wishes to execute the service under regular load conditions will need to subscribe to a premium version of the service. During development the limited version is sufficient. A possible drawback of the pattern is that, because it is "free", the limited version may attract so much attention that the provider does not have enough capacity to handle the requests.

4.8 Known Uses

StrikeIron offers different payment schedules for its web services: monthly subscription, annual subscription, or one-time purchase. Its service agreement[4] states:

[3] Instead of considering free vs. paid-for version of service, we can apply the same reasoning to a tiered pricing scheme.

[4] http://www.strikeiron.com/info/faqs.aspx

> A paid subscription is an agreement between consumers and StrikeIron to pay a specified amount of money over a specified amount of time, in exchange for a specified amount of accesses (or hits) to the web service. [...] A 'hit' is the term used in the StrikeIron Marketplace to refer to a counter that is decremented every time a subscriber invokes an operation by accessing and activating the web service.

4.9 See Also

This pattern can be used in combination with other patterns that Limit Execution.

5 Limit Results

5.1 Example

GelPub provides access to a set of online articles in the field of Computer Science. Any users can access the service, but for a given query, the number of results displayed are limited to 10 articles. The remaining search results will be available only to paid subscriptions.

5.2 Context

A service provider that uses Separate Terms for free and premium versions of its service.

5.3 Problem

How can a service provider use license terms to control the execution a web service?

5.4 Forces

The amount of data returned by a service for "free" should be sufficient to evaluate the service. However, as the value of the service lies in its data, it should be difficult to replicate the data that the service has. While the dynamic nature of the data could ensure that obtaining the full data is of limited value, even under these circumstances we would like to be able to charge a premium for more complete results than are available for free, or at a low charge.

5.5 Solution

Specify constraints that restrict the amount of data returned by the web service.

5.6 Example Resolved

The free version of the GelPub service restricts the number of results per query:

> You cannot access information beyond the 10th result for any given query.

5.7 Consequences

The number of results returned by the service are sufficient for light use. However, it is not possible for the user to obtain the full data. Hence, the value of the data is maintained. One possible drawback is that limited results may not give potential consumers the impression that the service is also of limited use. These user will not return, or upgrade to the full version.

5.8 Known Uses

The Google Web Services Licensing Agreement[5] is specified as follows:

> You can retrieve a maximum of 10 results per query, and you cannot access information beyond the 1000th result for any given query.

[5] http://code.google.com/apis/soapsearch/

5.9 See Also

This pattern can be used in combination with other patterns that Limit Execution.

6 Limit Quality

6.1 Example

MicroSync offers a on-demand financial web services. It delivers real-time stock quotes to paid subscriptions. MicroSync also provides stock quotes with a 20 min delay to anyone for free.

6.2 Context

A service provider that uses Separate Terms for free and premium versions of its service.

6.3 Problem

How can a service provider use license terms to control the execution a web service?

6.4 Forces

Some consumers may require a service to deliver high quality data. Others may prefer to trade lower quality data for a lower cost of the service. A provider that offers the same high quality service to all its consumers at one price, foregoes the opportunity to charge demanding consumers a premium, and will be perceived as too expensive by less demanding users.

6.5 Solution

Specify constraints that impose different levels of service quality. You can also impose restrictions that affect the quality of the resulting composed service. For example, you may want to restrict the right to publish the results of the service, as a means of protecting your data.

6.6 Example Resolved

MicroSync delivers real-time (high quality) stock quotes to consumers for a higher fee. The license clauses of these service may differ from the license clauses for a MicroSync service that provides delayed (low quality) data. For example, a MicroSync web service delivering real-time stock quotes may deny composition of this web service with other services.

6.7 Consequences

To be able to provide high quality data (for example, on-time delivery of critical data that changes continuously), a provider may have to make considerable investments. Naturally, this creates an incentive to offer those services at a higher price. These services may attract and retain consumers for whom quality is a top priority. However, this also creates a risk that the demand for the service may not be high enough to justify the investments.

6.8 Known Uses

Xignite offers two versions of its stock quote service: XigniteRealTime which provides real-time stock quotes for U.S. equities, and XigniteQuotes which delivers delayed quotes. The

XigniteRealTime service is offered at a higher price (both require subscription). However, XigniteRealTime also imposes a restriction on how the stock quotes can be used:[6]

> You cannot display real-time information on public web sites.

On the other hand, XigniteQuotes can be displayed on a public web site.[7]

6.9 See Also

This pattern can be used in conjunction with other patterns that Limit Execution.

[6] http://preview.xignite.com/xRealTime.asmx
[7] http://preview.xignite.com/xQuotes.asmx

7 On the Same Terms

7.1 Example

Spells is an open web service providing a spell checking operation for words. A new independently executable web service Spells Mirror is created by modifying the interface and implementation of Spells. Spells Mirror provides a functionality to split a given sentence and another functionality to spellcheck a word by reusing the operation of Spells. Spells Mirror is derived from Spells and is value-added as it provides its own additional functionality. Since Spells is an open web service, its license clauses determine the amount of control its creators can exercise over value-added services that are derived or modified from Spells.

7.2 Context

A service provider that offers web services.

7.3 Problem

How do we prevent proprietary lockup of open web services when they are composed?

7.4 Forces

Users should be able to modify a service, or derive new services from the service. However, in order to avoid license forking, we would like to prevent that the new service is licensed differently from the parent service. In this way, the value-added by the changes can benefit the whole community created around the web service. This benefit needs to be balanced against the need of service providers to generate profit from their service offerings.

7.5 Solution

Include a condition in your license that derivations of the service or modifications must be licensed under the same terms (a sharealike clause). A web service license with a clause similar to the "ShareAlike" clause of the Creative Commons license requires value-added web services to be licensed under the same terms and conditions. These clauses prevent others from turning value-added web services into closed services, if the parent web service is open.

Opening a web service means making the source code of the service implementation available in addition to the source of the service interface.[8] Inspired by how open source software is licensed, an open web service allows access to the source code of its interface as well as its implementation, allowing freely distributable composite and derivative services.

An open web service can expect another web service that uses ths service to reflect the same terms and conditions. Though open source software licenses do not discriminate among the uses of a software, the dynamic binding and execution of web services can enforce certain restrictions on the execution/usage of open services similar to Limited Execution.

7.6 Example Resolved

If Spells is released with a license clause that includes a sharealike clause, then

[8] A web service interface is, in a trivial sense, always available.

- Spells Mirror should be an open web service.
- Spells Mirror should be licensed under the same license as the one that Spells has.

Under this scenario, any value additions to the open web service Spells remains open, thus benefitting the community and avoiding forking of the license.

7.7 Consequences

Sharealiking (ie specifying a sharealike clause in the service license) prevents license forking, and benefits the community by returning user contributions to the community. However, another service provider could build a value-added web service based on a different open web service with more or less similar functionality. In this case, the parent web service may fail to retain users. Hence, using a sharealike clause carries the risk that it is perceived as too strong, and it may motivate others to derive from services that are less restrictive.

7.8 Known Uses

The GNU Affero General Public License[9] (AGPLv3) is a free, copyleft license for software and other kinds of works, specifically designed to ensure modifications or derivations of software are returned to the community in the case of network server software (ie a service).

> You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:
>
> [· · ·]
>
> d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

WikiDot[10], a farm of Wiki sites, uses the GNU AGPLv3. It requires releasing any changes to the service implementation. Funambol[11], a leading provider of mobile 2.0 messaging software built on open source stacks, offers its services under a GNU AGPLv3 license.

7.9 See Also

Open source license patterns for traditional software have been described by Kaminski and Perry [6]. The rights granted in an open source software license range from basic access to the source code of the software to the rights to make copies and distribution of the software.

[9] http://www.fsf.org/licensing/licenses/agpl-3.0.html
[10] http://www.wikidot.com
[11] http://www.funambol.com

8 Acknowledgements

Our sincere thanks go to our shepherd, Tim Wellhausen, for his probing comments.

References

[1] Musser, J., O'Reilly, T.: Web 2.0 Principles and Practices. O'Reilly Radar. O'Reilly (2007)

[2] D'Andrea, V., Gangadharan, G.R.: Licensing Services: The Rising. In: Proceedings of the IEEE International Conference on Internet and Web Applications and Services (ICIW'06), Guadeloupe, French Caribbean. (2006) 142–147

[3] Merges, R., Menell, P., Lemley, M.: Intellectual Property in The Technological Age. Aspen Publishers, New York (2003)

[4] Kelly, A.: More Patterns for Technology Companies Product Development. In: Proceedings of the European Conference on Pattern Languages of Programs (EuroPLOP). (2007)

[5] Kaminski, H., Perry, M.: Pattern Language for Software Licensing. In: Proceedings of the Tenth European Conference on Pattern Languages of Programming (EuroPLoP). (2005)

[6] Kaminski, H., Perry, M.: Open Source Software Licensing Patterns. In: Proceedings of the Sixth Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP). (2007)