Licensing Services: An "Open" Perspective

Vincenzo D'Andrea and G.R.Gangadharan

Department of Information and Communication Technology,
University of Trento,
Via Sommarive, 14, Trento, 38050 Italy
{dandrea,gr}@dit.unitn.it

Abstract. Though service orientation is an incipient technology, the inherently infinite potentiality of services makes them to proliferate seamlessly, serving in myriad domains. Licensing of services enables to regulate the commercial use and modifications of service, retaining the copyright with owner of the service. With the growing influence of open source initiatives today, it becomes a significant topic to analyze 'open'ing services. In this paper, we present a concept of 'open service' and analyze the implications of open source approach on service licenses.

1 Introduction

Service oriented computing (SOC) is an emerging distributed systems paradigm, addressing the aspects of real world applications, crossing organizational and technical boundaries. With a vision of dynamically composing service oriented and non-service oriented applications, SOC continues to proliferate as a technology for connecting applications in a loosely coupled manner. Today, web services are being used as a component or utility and offer programmatic interfaces to applications. However, many available web services are not even considered as providing relevant business value. The majority of attention on SOC has been contemplated on its related technical standards and technology integration. Managerial issues and business strategy for implementing SOC have not been studied intensively.

One of the relevant issues from this perspective is the role of licensing for services. In the case of software, licensing is generally considered the way for extending property rights into software. Thus, software licensing [1] is considered to include all transactions between the licensor and the licensee in which the licensor agrees to grant the licensee the right to use some specific software or contents of information for a specific tenure under predefined terms and contracts.

In [2], the author describes a distributed software licensing framework using web services and SOAP. However, [2] addresses a framework using web services but does not address licensing of web services itself. The technical contracts of web services are described in [3], but business and legal contents of contracts are not considered. In [4], we had elaborated the dimensions of web services

Please use the following format when citing this chapter:
D'Andrea, V., and Gangadharan, G.R., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 143-154

differing from software and proposed an anatomy of a service license with a set of key negotiation issues.

As the foundations of open source regime rely on licenses, an approach inspired by open source could be considered during the process of conceptualizing licenses for services. The Free / Open Source Software (FOSS) approach protects the unconditional rights of modification and redistribution by the collaborating developers, making the source code freely available [5]. Freedom of distribution and freedom of modification are the core principles of open source licensing. To the best of our knowledge, the idea of making services 'open' is completely new and no previous work exists in this field. In this paper, we present a novel concept of licensing services, inspired by open source movement.

The rest of the paper is organized as follows. Section 2 introduces the concept of service oriented computing. Section 3 presents the distinguishing characteristics of services which preclude the direct adoption of software licenses for licensing services. Section 4 elaborates licensing of services, describing the issues of composition. A comprehensive description of what we mean by 'open' services is elucidated in Section 5. Section 6 describes the consequences of adoption of open principles in services paradigm, drawing some conclusions.

2 Service Orientation of Software

Most of the products fall in a continuum having pure service on a terminal point and pure commodity good on the other one [6]. Software, traditionally, has been perceived as a product, requiring possession and ownership, in order to receive the desired performance. Software-as-a-service [7] is a mechanism of renting software where users are subscribed to the software they use. SOC allows the software-as-a-service concept to expand to include the delivery of complex business process and transactions as a service, allowing applications to be constructed on the fly and services to be reused everywhere [8].

The idea of software composition and refinement instead of software development from scratch nowadays is elaborated to the platform-independent, distributed and standardized services paradigm [9]. In such paradigm, services reflect self-contained processes that can be described, published, discovered and invoked in a distributed environment, connecting people, processes, and applications. Services are intended to represent meaningful business functionality that can be federated with other services, to enhance more value to the business functionality.

The application of SOC model (see Figure 1) to web resources is manifested by web services to provide a loosely coupled model for distributed processing. Web services are the enabling technology, standardized to construct and integrate applications and organizational interfaces as services, using the Internet as the communication medium and open Internet-based standards [10]. A service is represented by an interface part defining the functionality visible to the external world as a means to access the functionality and an implementation

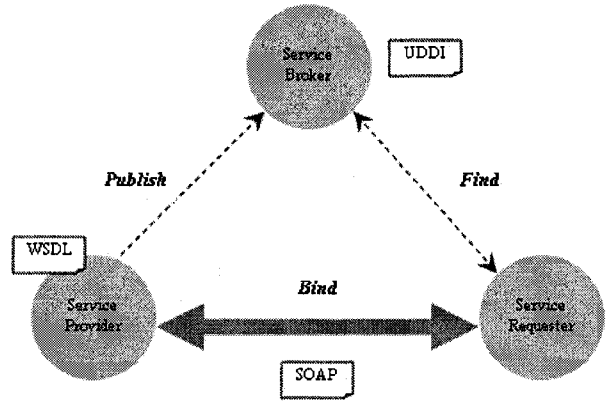

Fig. 1. Service Oriented Computing (Instances with Web Service)

part realizing the interface. The Web Services Definition Language (WSDL) is an XML based interface definition language, describing services as a collection of messages (abstract descriptions of the data being exchanged) and port types (abstract collections of operations), separated from their concrete network deployment or data format bindings. Directories of services are necessary in order to find services usable for a specific application. Universal Description, Discovery, and Integration (UDDI) enables publishing and accessing WSDL specifications in directories. Simple Object Access Protocol (SOAP) is a platform and language independent protocol, providing a way of communication between applications.

3 Dimensional Analysis of Software and Services

Software is an intangible asset, protected by copyright. Being a digital work, it can be vulnerable for perfect copying, and unlimited copies identical to the original can be made. Software is an experience good, whose value is not quantifiable without consumption. Thus, the socio-economic analysis of software signifies distribution strategies. While services (see Figure 2) present several similarities with software, we claim that it is not possible to adopt the software [11] and/or component [12] licensing models directly for licensing services. The reproduction of services could vary in the levels of interface, implementation, and execution (see Section 5 for details). Further, composition of services [13] is significant in reproduction of value added services. The following characteristics of services associated by functional and non-functional properties differing appreciably from software become the cornerstones for licensing of services:

Configurability: Generally software serves as a standalone application licensed by shrink-wrap or click-wrap licenses. In contrast, web services are not

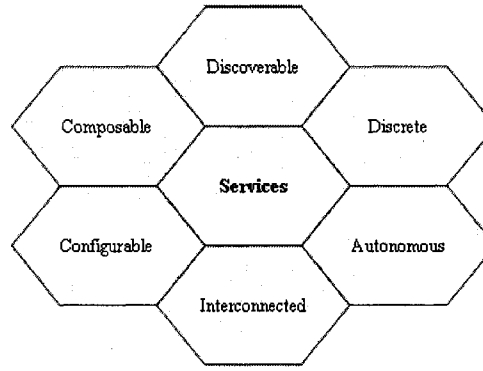

Fig. 2. Service Characteristics

targeted as standalone applications. The rationale behind web services is making network-accessible operations available anywhere and anytime. The counterpart of a software application in terms of services is a reconfigurable composition of distributed services. Service implementation may involve many steps, executed in distributed manner, supporting interoperability and location transparency [14]. In contrast to software components, consumers are not required to download them for local use.

Discreteness: Software ranges from small fragments to sophisticated applications. The separation of a software package will not be meaningful as it was originally intended to function. Similarly, services can also vary in complexity of functions. A service, as a self-contained software module, semantically encapsulates discrete functionality [8].

Autonomy: Unlike general software and components, services are connected to other services and clients using message based methods. They do not require knowledge of any internal structures or context at the client or service side. Thus, loose coupling allows service providers to modify the service interfaces, without impacting consumers.

Interconnectedness: Software programs run on infrastructures and consumers are responsible for maintaining the infrastructure on which the software executes. In case of services, functionality and reliability can be affected by problems in the network between consumers and services. The availability and performance of a service could not be directly guaranteed.

4 Towards Licensing Services

All the characteristics of SOC lead to composability, to form composite services by combining elementary and/or composite services. Service composition [15] is related to the implementation of a web service whose internal logic involves the invocation of operations offered by other web services. Services can be composed

(see Figure 3a) as a part of composite service, encapsulating individual services and exposing a different set of operations. Another perspective on composition (see Figure 3b) is by defining the invocation order of individual services [16]. Service composition allows a recursive process of composition of services i.e. a composed service can be composed with an other elementary and/or composite service. Thus, individual services can be composed up to any levels of hierarchies.

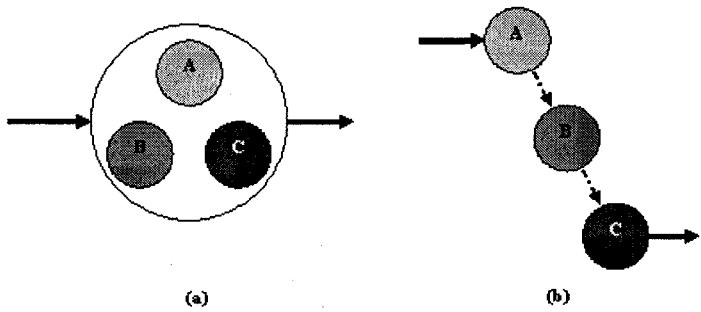

Fig. 3. Service Composition (a) by Encapsulation and (b) by Sequencing

Besides the functional operations, from the point of view of a service consumer, it is important to consider also other, non-functional, aspects of service provisioning, such as the cost or the reliability of a service. These aspects are collectively referred to as Quality of Service (QoS) or non-functional properties of a service. The QoS of a composite service is derived from the aggregation of QoS of each individual services, where the aggregation could be a simple combination such as adding the cost of individual services, or taking the maximum among the performances of the individual services to estimate the response time of a composite service. For other aspects, the combination requires the definition of a specific model, such as combining security aspects or reliability, availability, scalability and so on.

Analyzing the characteristics of services as discussed in Section 3, depicts the nature of services differing from software and/or components and rises a requirement for licensing services. Questions of ownership and distribution could impede composition, thereby impacting the reuse of services. Thus, the license of a service [4] is defined as not only the description of the terms and conditions for the use of service as in the case of software, but also a detailed description of clauses regarding reuse.

Though the concept of arbitrarily mixing and matching the services from different providers seems interesting, the basic clauses of service licenses would enforce certain terms and conditions on composition. To illustrate the issues that could arise in the context of licensing web services, we consider a simple

scenario where R is a restaurant service providing the following operations: R_0, information on location and opening hours; R_1, the facility for reserving table; R_2, a catalogue of specialty cuisines; R_3, a daily recipe for one of the specialty cuisine. Another service, F, a restaurant finder service uses R, for the following operations: F_1, a restaurant locator giving a list of restaurants close to a given location and using R_0 (as well as similar operations for other restaurants); F_2, for intermediating table reservation, using R_1; F_3, a daily recipe randomly selected among the recipes provided by the restaurants listed using F (in the case of R, it will use operation R_3). The license terms of R may deny the provision of R_3 to other services intended for providing recipe information exclusively or may require attribution for the use of R_3. The license terms of R can even require the same set of terms and conditions for any hierarchy of composed services, even the successive compositions use F. In this case, the license terms of F will have to comply with R, for the request and deny provision of F_3 to other services intended to provide the recipe information exclusively. Another restaurant service, S, has a similar set of operations S_0, S_1, S_2, S_3 as R, but having a different license that freely allows the use of operations anywhere. If F uses also S, then it could be possible to have a different license when F_3 presents a recipe chosen from S. Even in this simple scenario, it is apparent that the composition of licenses could easily bring to incompatibility between the composed services.

The license compatibility is a complex issue, requiring careful attention before attempting to merge licenses. The licensing of a composed service would be based on the licenses used in different service and the way they are combined together. As composition of services is established dynamically ('just-in-time') and composed service is created on-the-fly, the license of composed service would be program generated and needed to be validated by analyzing the licenses of composing services.

5 'Open'ing Services

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. According to [17], it refers to four kinds of freedom, for the users of the software:

1. The freedom to run the program, for any purpose (freedom 0).
2. The freedom to study how the program works, and adapting to the needs (freedom 1).
3. The freedom to redistribute copies (freedom 2).
4. The freedom to improve the program, and release improvements to the public, so that the whole community benefits (freedom 3).

Open source software, as a superset of free software, exists in a plethora of initiatives today, representing a variety of technology innovations and approaches [18]. Some of the key conditions of Open Source Definition (for authoritative definition, see [19]) are as follows:

1. The software should be freely redistributable.
2. The software must include source code, and must allow distribution in source code as well as compiled form.
3. The software must permit modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
4. The rights attached to the software must apply to all to whom the software is redistributed without the need for execution of an additional license.
5. The license must not discriminate against any person or group of persons or any field of endeavor.

Following FOSS definitions [17, 19], we define an 'Open Service' as follows:

1. An 'Open Service' should be free for use.
2. The source code of the interface (WSDL descriptions) as well as the implementation of an 'Open Service' should be available.
3. The service implemented by creating a new service using the source code and interface of an 'Open Service' should be freely distributable as an independent service. The modification of interface and implementation should be permitted.
4. The service using an 'Open Service' as part of a composite service should be freely distributable as an independent service, even when using a separate interface. The modification of interface and implementation should be permitted.
5. Derived services and modified services must be allowed and be capable of distribution.
6. The license must not discriminate against any person or group of persons or any field of endeavor.
7. The license agreement must provide an 'Open Service' "as is" with no warranties either to functional and/or non-functional properties or non-infringement of third party rights.
8. The license must not place restrictions on composition with other services and on distribution of composed services.

Open service perspective enhances the quality properties of a service, leveraging the availability of the source code and the right to modify it. Beyond composition, the 'open'ness of service makes the class of derivative service, a service being modified and re-distributed with more value addition.

Now, we exemplify the freedom and openness exclusively associated with 'open'ing of services, varying in the levels of interface and implementation and in the levels of composition and execution.

1. ***Service Usage***
   Service usage describes the freedom to execute a service by other applications, for any purpose. The basics of 'open'ing service allows the use (execution) of service by any other service oriented and/or non-service oriented applications, adhering the given open service license.

2. ***Service Implementation***
   With the opening of service, we are provided with the freedom to know how the service works and could be adapted to our needs, making the source code of service interface as well as service implementation freely available.
   a) A service is described by WSDL. Service orientation obligates WSDL code to be available publicly for service discovery, and composition.
   b) In addition, an 'open' service allows the availability of the source code of implementation (the real functionality of a service).
   c) The source code of a service wrapping the functionality of another proprietary software partially or fully, can be available publicly with service interface and implementation, except the source code of proprietary software being wrapped in the given service. Consider a spell checker service wrapping PWP[1] spell check API. As PWP is proprietary, its source code can not be available. However, a service can use the PWP spell checker API for spell checking operation. Thus, an 'open' service wrapping the PWP spell checker API allows users to read the source code of interface and implementation of the service except the source code of the wrapped system.
3. ***Service Redistribution***
   Service redistribution describes the freedom to distribute a service as a separate service. Further, any entity can create a new service which would use the interface of an 'open' service, without the need to implementing the service realization.
   a) **Separate and independent service: replica of an 'open' service:** Opening of service allows to create independent services, attributing to the 'open' service. Let S_A be an 'open' service providing a spell checking operation for words, say, *Spell*(*word*). Consider S_A provides this service by wrapping PWP spell checker API. Let S_B be an another independent service, providing the same *Spell*(*word*), created by replicating the source code of implementation and WSDL of the 'open' S_A. Albeit S_A and S_B are performing the same operations, S_A and S_B are two different services, executed separately.
   b) **Separate but dependent service with same interface:** This is a common scenario in SOC; our perspective stresses the attention on licensing aspects. 'Open'ing service adds value to a service by distributing the service, not requiring to implement the service again. Let S_B be a service providing a spell checking operation *Spell*(*word*) for words, using (copying) the WSDL interface *Spell*(*word*) of 'open' S_A. S_B is designed in such a way that *Spell*(*word*) of S_B directly invokes the operation of S_A, executing on the host of S_A.

   From a service consumer perspective, in both cases, S_A and S_B are providing exactly the same *Spell*(*word*) interface, thus they are interchangeable in an application on the consumer side. The two implementations of S_B are

[1] PWP is a fictitious name for a Proprietary Word Processor.

not distinguishable. Theoretically, there will not be any differences in performances of both the services, apart from possible network latency between S_A and S_B.

4. ***Service Derivation & Distribution***
   Service derivation and distribution offer the freedom to improve the service, and release improvements to the public, so that the whole community benefits. Opening of services allows to perform modifications on the WSDL interface and implementation of the service and thus, derived services are created. Derived services could be executed independently (together with separate interface and implementation) or could use the implementation of the parent service.
   a) **Separate and independent service: replica of an 'open' service with modified interface and implementation:** Now, consider the case similar to 3(a) with interface of the open service S_A be modified in S_B. The modified interface of S_B provides *Spell*(*sentence*) which composes a *parser*() and repeated invocation of the code derived from S_A, to access PWP API. Now, S_A and S_B are the different services, executing independently. *Spell*(*sentence*) of S_B is derived and improved version (having an own additional functionality *parser*()) of *Spell*(*word*) of S_A.
   b) **Separate but dependent service with modified interface and implementation:** Consider a service S_B similar to the case of 3(b), but with modified WSDL interface as well as implementation of the open service S_A. *Spell*(*sentence*) of S_B comprises a *parser*() and repeated invocation of the spell checking operation provided by S_A via the interface *Spell*(*word*). Thus, the word spell checking operation of S_B is executed in the host of S_A (invoking repetitively the service of *Spell*(*word*) of S_A) for spell checking of a given prose *Spell*(*sentence*).

The cases presented above are only a partial view of all the possible combinations of derivation (or not) from the source code, modification (or not) of the service interface, and relationship between services (compositional properties). Due to space constraints, the most common and significant cases of SOC have been illustrated and summarized in Table 1.

Table 1. Partial view of 'open'ing of services

Interface	Implementation	Composition	Case
Unmodified	Unchanged	No	3(a)
Modified	Derived	No	4(a)
Unmodified	Unchanged/Derived	Yes	3(b)
Modified	Unchanged/Derived	Yes	4(b)

6 Consequences and Conclusions

The 'open'ing of services significantly contributes to the development of new services from existing services by adding new operations. Consider S_X be an open service. S_Y could be developed by extending S_X in any of the ways discussed in Section 5, keeping S_Y as 'open'. A new service S_Z could be developed by incorporating S_Y, which in turn provides access to S_X operations. S_Z could even enhance S_X, with additional operations.

Free services inspired by FOSS licenses could make value addition by composition, resulting composed services as 'free'. Thus, free services (with free licenses) could create a chain effect on composition of services to be free, even if one of the composing service may be not 'free'.

Let S_P and S_Q be the two individual services of a composite service S. S_P and S_Q may be licensed by free or proprietary licenses not imposing restrictions on the use in a composition. The composition of S_P and S_Q inspired by FOSS scheme, is illustrated in Table 2. Making services free will be highly beneficial

Table 2. Service composition enriching 'Free Culture'

S_P	S_Q	$S = \{S_P, S_Q\}$
Free	Free	Free
Free	Proprietary	Free
Proprietary	Proprietary	Free

for government sectors, education, and non-profitable organizations to explore and enjoy the benefits of services.

'Open'ing services may raise an emergent question of how a service provider could profit by providing services. Many OSS business models are in practice of the community [20]. Some of these business models could be adaptable to the 'open' service context.

1. ***Support Seller:*** 'Open' services could adopt this scheme where revenue comes from media distribution, branding, training, consulting, and custom development.
2. ***Service Enabler:*** An 'open' service could be created and distributed primarily to support access to revenue-generating on-line services.
3. ***Sell It, Free It:*** Like traditional commercial softwares, services would begin their product life cycle as closed and then are converted as 'open' service when appropriate.
4. ***Brand Licensing:*** An 'open' service provider can charge other service providers/ aggregators/ consumers for the right to use its brand names and trademarks in creating derivative services.

Further, a copyright holder can release his/her works under any license, including multiple licenses and users of that work are allowed use under one of the licenses they choose [21]. Dual licensing is a business model for open source

software exploitation based on the idea of simultaneous use of both open source and proprietary licenses [22]. Several open source projects, including MySQL, Perl, and Qt use dual licensing for their business model. Following the dual licensing strategy, a service can be licensed under open source inspired license as well as a proprietary license.

According to GPL [23], the distribution of GPL'd software must include source code. A GPL'd application delivered as a web service is not actually distributed to the end user. Hence, in this case, the application license does not require to disclose the source code. The nature of web services allows users to interact with the application via an interface, without downloading the software. This can result against the 'freedom' of GPL, i.e. users consuming services without having access to the source code as delivered by the providers, retaining the rights to modify and distribute. More precisely, GPL acts on the source code, but not on the use of source code by a service. Consider a service wrapping FWP[2] instead of PWP (a Proprietary Word Processor). As FWP is a GPL'd software, a wrapper for FWP is also GPL'd code. However, GPL does not restrict the use of this FWP wrapper provided by a web service. Since, the service is using only the execution of FWP (not the source code of FWP), GPL does not effect the licensing of composite services based on 'FWP wrapper' service. Even the draft version of GPL3 [24] is silent about this issue.

Nowadays, standards are 'open' in SOC. But, the services developed using these standards are unfortunately 'closed'. If services are 'open', service consumers can add value beyond the concept of composition. Hence, we introduced the concept of open services in this paper and analyze the impacts of open source inspired licenses on SOC. The wedding of services with open source would be beneficial for both communities, spreading services 'open'ly. In our future work, we aim to embed formal licenses in services and make legally enforceable service composition.

References

1. Classen, W.: Fundamentals of Software Licensing. IDEA: The Journal of Law and Technology **37**(1) (1996)
2. Clarke, N.: Distributed Software Licensing Framework based on Web Services and SOAP. http://www.dsg.cs.tcd.ie/~dowlingj/students/clarken/clarken_02.pdf (May 2002)
3. Tosic, V., Pagurek, B.: On Comprehensive Contractual Descriptions of Web Services. In: Proceedings of the IEEE e-Technology, e-Commerce, and e-Service (EEE). (2005) 444–449
4. 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)

[2] FWP is a fictitious name for a free word processor.

5. 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
6. Wikipedia: Service. http://en.wikipedia.org/wiki/Services (Accessed on 27.12.2005)
7. Bennett, K., Layzel, P., Budgen, D., Brereton, P., Macaulay, L., Munro, M.: Service-Based Software: The Future for Flexible Software. In: Proceedings of the Asia-Pacific Software Engineering Conference (APSEC). (2000) 214–221
8. Papazoglou, M., Georgakopoulos, D.: Service Oriented Computing. Communications of the ACM **46**(10) (2003) 25–28
9. D'Andrea, V., Marchese, M., Gangadharan, G.R., Ivanyukovich, A.: Towards a Service Oriented Development Methodology. In: Proceedings of the Eighth World Conference on Integrated Design and Process Technology, Beijing, China. (2005)
10. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.: Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall PTR (2005)
11. Kendra, G.: The Anatomy of a Technology License. Michigan's Lawyer's Weekly **16**(34) (2002)
12. Chavez, A., Tornabene, C., Wiederhold, G.: Software Component Licensing: A Primer. IEEE Software **15**(5) (1998) 47–53
13. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services Concepts, Architectures, and Applications. Springer Verlag (2004)
14. Colan, M.: Service Oriented Architecture expands the vision of Web services. http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html (2004)
15. Alvarez, P.A., Baares, J.A., Ezpeleta, M.J.: Approaching Web Service Coordination and Composition by means of Petri Nets. In: Proceedings of the 3rd International Conference on Service Oriented Computing. (2005) 185–197
16. Dustdar, S., Schreiner, W.: A Survey on Web Services Composition. International Journal of Web and Grid Services **1**(1) (2005) 1–30
17. Free Software Foundation: The Free Software Definition. http://www.fsf.org/licensing/essays/free-sw.html (Accessed on Jan. 2006)
18. Brown, A., Booch, G.: Reusing Open Source Software and Practices: The Impact of Open Source on Commercial Vendors. In: Proc. of 7th International Conference on Software Reuse. (2002) 123–136
19. Open Source Initiative: The Open Source Definition. http://opensource.org/docs/definition.php (Accessed on Jan. 2006)
20. Raymond, E.: The Magic Cauldron. http://www.catb.org/ esr/writings/magic-cauldron/magic-cauldron.html (1999)
21. Wikipedia: Dual license. http://en.wikipedia.org/wiki/Dual_license (Accessed on 29.12.2005)
22. Valimaki, M.: Dual Licensing in Open Source Software Industry. Systemes d' Information et Management (2003)
23. Free Software Foundation: GNU General Public License. http://www.gnu.org/copyleft/gpl.html (Accessed on Jan. 2006)
24. Free Software Foundation: GNU General Public License Version 3. http://gplv3.fsf.org (Accessed on Jan. 2006)