Introducing OSLC, an open standard for interoperability of open source development tools

Olivier Berger, Sabri Labbene, Madhumita Dhar, Christian Bac

Télécom SudParis
Département INF
9, rue Charles Fourier
91011 Évry Cedex, France
Tel: +33 1 60 76 45 32 – Fax: +33 1 60 76 47 80 – email: olivier.berger@it-sudparis.eu

Abstract: In the COCLICO project, efforts have been made to improve interoperability of software development forges, to help integrate them better in the modern software quality process of organizations. In order to improve such interoperability, implementation of the *Open Services for Lifecycle Collaboration* (OSLC) specifications has been conducted (in particular the *Change Management* domain specifications). The OSLC-compatible open source adapters we have developed for the Jenkins Continuous Integration server and the FusionForge bug-trackers have been integrated in a Web *mashup*, allowing demonstrating the interoperability potential of OSLC-CM in a continuous integration use case. Even though deployment of OSLC is still limited for open source tools in the Application Lifecycle Management (ALM) field, we believe it will offer great benefits and potential, for new complex and difficult problems, in particular for large scale bug tracking applications.

Keywords: ALM, standard, interoperability, OSLC, OSLC-CM, FusionForge, Jenkins, open source, free software

1. INTRODUCTION

In the COCLICO project [7], we have been working on improving the interoperability potential of the software development forges [4]. We have participated in the *Open Services for Lifecycle Collaboration* (OSLC) community [5] to help refine the OSLC-CM (*Change Management*) specifications and worked on implementing them for the forge's tracker interfaces in order to validate which level of standardization can be achieved practically with OSLC. In this introduction, we will first briefly give some elements of context on the efforts reported here.

1.1. Interoperability utopia

As stated in [2], Application Lifecycle Management (ALM) tools integration is a quest that most software development organizations have to consider, as no single tool covers all the needs of a particular project, or the peculiarities of all lifecycles or Quality Assurance (QA) processes. Cost control, Return On Investment and quality are directly impacted by the lack of interoperability of software development tools, which renders human actions and developed custom adaptors both expensive and error prone.

With proprietary tools deployed in an organization, there's hope that the vendor's technical choices can ensure some coherence between different tools offered in that vendor's portfolio. But it is rarely achievable to find all the needed features in one single vendor's offers, and it's often hard to replace all previous tools at once. There's a danger of vendor lock-in, on the other hand, that organizations may want to diminish when they want to keep control of future evolutions of their development environments.

With open source tools, there's more variety of choice, in principle, but there's rarely a full range of solutions that have been specified and devised to inter-operate with each-other, under the coordination of a standardization initiative. Still these tools tend to adhere closely to open standards, for interoperability's benefits.

Either case has its own advantages but whatever the licensing scheme, there are very few open standards that have yet emerged to ensure practical interoperability of tools in the ALM sector. The OSLC specifications seek to address this situation by proposing a set of integration specifications so that conforming tools can interoperate better with each-other.

1.2. Open Source Software forges

Historically, the forge vocabulary emerged with the *SourceForge.net* service that was started in 2001 [8], and which greatly helped the Open Source movement's momentum, in allowing developers to collaborate through the Internet. Not only have many forge *services* emerged in the last decade, but also forge *applications*, that an organization may deploy, in a self hosted way [4], such as FusionForge [6], Codendi, Redmine, Trac or may others.

In terms of interoperability, such forges often integrate already existing tools, such as mailing-list management tools, or software configuration management (*aka* version control systems), rendering them naturally interoperable. But if one saves the difficulty of integrating, say, a mailing-list manager and a revision control system (for commit notification, for instance), the forges are often too monolithic, and not really interoperable with other tools (like Continuous Integration servers, or developer's IDEs), or with each-other.

1.3. About the COCLICO project

The COCLICO collaboration project has been working for 2 years under the French cluster programme "pôles de compétitivité" (through the `System@tic` and `Minalogic` clusters), grouping 9 industrial and academic partners, with the help of public agencies funding[1].

Quoting its site's description, COCLICO

> *“aims to reinforce software forge communities by structuring an open source ecosystem for which a critical mass exists in France. The dynamics of Forge platforms development is now a key issue to address the needs to manage business development in terms of collaborative and distributed process.”*

The principal awaited results of the project are:

- Re-dynamization of the open source development community around FusionForge and the PlanetForge Website,
- Definition of an open integration model,
- Data integrity and confidentiality,
- Exchange of data in real-time between various forges,
- Features for industrial use and quality assurance (traceability of information, support of software engineering methodologies, interaction with the user's workstation), etc.

Interoperability issues are thus at the heart of the work packages of COCLICO.

1.4. Structure of this article

In a first section, we will briefly describe the nature of the OSLC effort and specifications, then, in a second section, we will describe our implementations of OSLC-CM in a continuous integration mashup use case, based on open source tools.

2. THE OSLC STANDARD

In line with the COCLICO project's goals [7], and our quest for interoperability solutions, we have studied the *Open Services for Lifecycle Collaboration (*OSLC) specifications [5], and devoted much effort to try and evaluate their benefits. We have not only started to experiment with the OSLC specifications, but have also contributed at times to their improvement. We also try and foster their adoption in the open source ecosystem, in order to maximize interoperability and standardization of exchanges between ALM applications, to allow greater flexibility, and a potential maximum integration for complex processes.

We have not investigated the whole of the OSLC specification domains, but focused mainly on OSLC-CM (Change Management), and its base, OSLC Core. The concrete implementations of the OSLC specifications in ALM tools are mainly available, to date, in proprietary tools. We have worked on taking advantage of the OSLC specifications, while deliberately following an alternative and complementary path, in delivering implementations as free and open source software. In a previous project called HELIOS [10], we worked on implementing an OSLC-CM V1 compatible service provider for the Mantis bugtracker. In COCLICO we've

1 The present paper was written as part of COCLICO activities [7], with financial support of the Paris Region council (http://www.iledefrance.fr/).

been reusing the same server code as a base, to implement an OSLC-CM V2 service provider for FusionForge [6].

2.1. OSLC specifications

OSLC is a proposed open standard for interoperability of applications in the ALM domain. Quoting "OSLC Core Specification Version 2.0" [9]:

> *"The Open Services for Lifecycle Collaboration (OSLC) initiative is creating a family of web services specifications for products, services and other tools that support all phases of the software and product lifecycle. The purpose of these specifications is to enable integration between products that support Application Life-cycle Management (ALM) and Product Life-cycle Management (PLM). Each part of the lifecycle or domain has its own group and specification, for example there are Change Management, Quality Management, Estimation & Measurement and more. Each of the domain specifications are built upon this core specification."*[2]

2.2. Technical scope

The OSLC specifications cover the communication protocols between different servers called *Service Providers,* and client applications, which collaborate via Web technology in order to deliver lifecycle processes for software development projects.

In addition to defining common services APIs, they impose or recommend the use of Web and W3C standards like REST, RDF (as XML or JSON), AJAX or OAuth, for integration of applications, setting up a common framework for interoperability. As an example:

> *"Each Service can provide* Creation Factories *for resource creation*, Query Capabilities *for resource query and* Delegated UI Dialogs *to enable clients to create and select resources via a web UI. Query Capabilities and Creation Factories may offer* Resource Shapes *that describe the properties of resources managed by the service. This is illustrated in Figure Fig 1.*

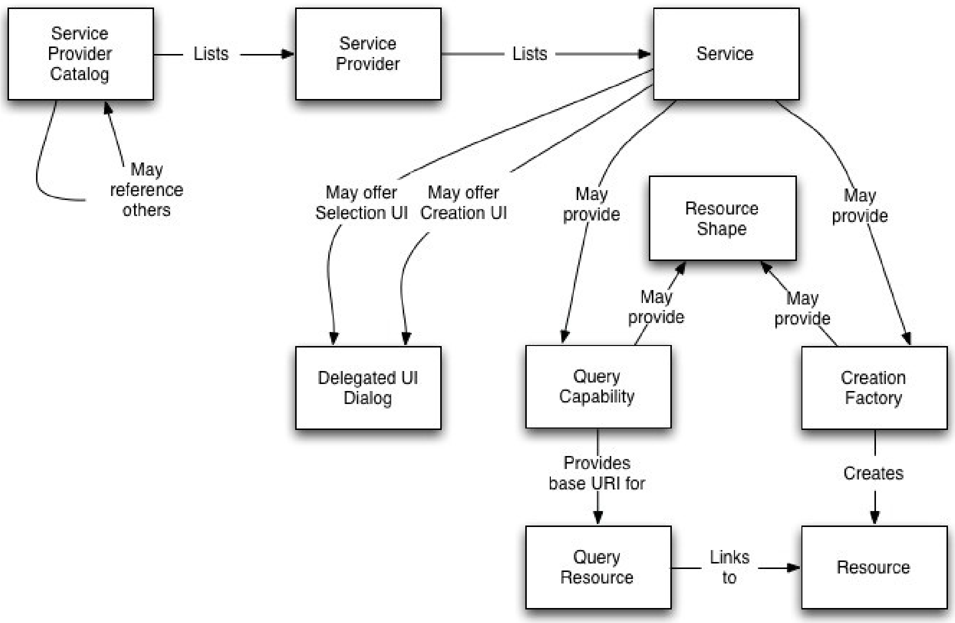


Fig 1: OSLC Core Specification concepts and relationships

The resource and property-based data model used in OSLC resources *is based on the Resource*

2 In the rest of the article, quotes marked in italics are excerpts from the OSLC specifications.

Description Framework (RDF)"

2.3. The OSLC initiative

The OSLC initiative is an open community, where all actors can contribute, provided they don't embed in their contributions elements of industrial property which would be restricted. OSLC was started initially by IBM/Rational and others in 2008 and became mature in early 2011 with finalization of the first set of version 2 specifications. Most actors in the initiative are US based (not surprising in the software area), and software vendors (proprietary applications mostly), or big user firms. As OSLC proposes an open standard, and is led by major actors of the software industry, this should in principle grant it some impact on the way tool integrations are done, even if, at the time of writing, its real impact on the market remains to be surveyed.

The participants work mainly through mailing-lists (and phone call meetings) and a Wiki, to write the specifications which are then published directly in the Wiki. The specifications are published under a Creative Commons license.

In addition to the specifications, a dedicated open source project was started to collaboratively develop support tools such as a test suite and an OSLC reference implementation in Java [11]. This project's code was initially released under the Apache Software License and hosted at SourceForge.net. More recently the Eclipse Lyo project [15] has been initiated by participants of this open source project (including authors of the present article), in order to give it more visibility (by hosting it under the Eclipse Foundation umbrella), and to foster more OSLC compatibility in the Eclipse tool chains.

For the past 2 years, Institut TELECOM, through Télécom SudParis, has been contributing to the OSLC specifications during the HELIOS project (mainly on OSLC-CM) and the current COCLICO project (on OSLC Core too).

2.4. Benefits of OSLC

The technologies chosen for the OSLC specifications (Web standards) are modern and quite interesting, both on a purely technical point of view, but also in terms of industrial strategy (through openness) chosen by some tool vendors. As such, OSLC is the result of a mature and documented "open innovation" initiative [2].

With our strong backgroung in Free/Libre/Open Source software (FLOSS), adopting and contributing to these specifications appears to us as particularly important, in the hope to help diminish the risks of lock-in effect, promote technical and semantic interoperability in a pragmatic context, and foster more general openness.

2.4.1. Truly open standard

Even though it was initially founded by IBM/Rational, and primarily implemented in releases of proprietary tools, OSLC seems to us a truly agnostic and particularly well written open standard. Its openness is guaranteed by the license of the specifications (Creative Commons) and the *patent non-assert covenant* required to be signed by contributors.

2.4.2. Tailored for interoperability

The "philosophical" grounds of OSLC are in line with the powerful and scalable World Wide Web architecture, and seek to make the "simplest things that work":

- *Build on the WWW. OSLC builds on the architecture of the WWW and follows the REST architectural pattern. This means that OSLC Services provide a uniform HTTP interface, OSLC URIs are stable and opaque and, in simple terms, OSLC works like the web.*
- *Keep things simple. Doing the simplest things that will possibly work means a couple of different things in regard to OSLC. It means starting with simple and existing concepts.[…]*
- *Accommodate different schemas. Because of the breadth of the OSLC domains, spanning lifecycle and platforms, OSLC has to work for systems with very different data schemas or no schemas at all. Flexibility is needed, […]*
- *Accommodate different representations. Different client platforms might require or at least prefer different representations. For example, in the browser a JSON or Atom format representation might work best. OSLC Services will all support RDF/XML and may support*

other formats including JSON, Atom and Turtle.

- *Align with the W3C Linked Data initiative. Instead of defining a new data model, OSLC's resource and property-value approach is based on industry standard Resource Description Framework (RDF) data model. This model allows OSLC to keep things simple, build on the WWW and accommodate different schemas.*

The choice of the base technologies promoted by OSLC (open Web technical standards) is, in our opinion, very well tailored to the establishment of an open industrial standard. In particular, this should, in principle, make it even more acceptable to the free and open source software development communities, which have always been adepts of open Web standards. A few pragmatic principles underlie the writing of the OSLC specifications, which, in our opinion, should ensure their success and a wide dissemination, with great impact on the market. Among these is the use of extensible semantic data representation formats based on RDF [12]. This naturally allows extending OSLC, for specificities of particular tools or contexts, making it a quite versatile standard.

2.4.3. Extensible and semantic exchange format: RDF

In OSLC, resources are exchanged, referenced, and identified, using RDF and the Linked Data principles. Below, are two examples of a fictional OSLC resource, a blog entry, represented either in XML or JSON (copied from OSLC specifications).

- XML variant (classical standard representation of RDF in XML):

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:oslc_blog="http://open-services.net/ns/bogus/blogs#">
    <oslc_blog:Entry
      rdf:about="http://example.com/blogs/entry/1">
      <dcterms:title>I love trash</dcterms:title>
      <dcterms:modified>2002-10-10T12:00:00-05:00</dcterms:modified>
      <dcterms:content>
         Anything dirty or dingy or dusty.
         Anything ragged or rotten or rusty.
      </dcterms:content>
      <dcterms:creator>
         <foaf:Person>
             <foaf:name>Oscar T. Grouch</foaf:name>
         </foaf:Person>
      </dcterms:creator>
   </oslc_blog:Entry>
</rdf:RDF>
```

- JSON more compact variant (particularly suited for mashup applications in Javascript, using OSLC Core formatting conventions for representing RDF: XML processing in Javascript browser based applications is not particularly efficient):

```
{
   "prefixes" : {
      "oslc": "http://open-services.net/ns/core#",
      "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
      "foaf" : "http://http://xmlns.com/foaf/0.1/",
      "dcterms" : "http://purl.org/dc/terms/"
   },
   "rdf:type" : { "rdf:resource" : "http://open-services.net/ns/bogus/blogs#Entry" },
   "rdf:about" : "http://example.com/blogs/entry/1",
   "dcterms:title" : "I love trash",
   "dcterms:modified" : "2002-10-10T12:00:00-05:00",
   "dcterms:content" : "Anything dirty or dingy or dusty. \\nAnything ragged or rotten or
rusty.",
   "dcterms:creator" : {
      "foaf:name" : "Oscar T. Grouch"
   },
}
```

2.5. Limitations of OSLC

OSLC offers a generic base with quite low profile semantics, at least in the current versions of the specifications.

Alone, OSLC is not enough to define complex business processes, but it's a necessary paving stone for practical interoperability. For instance, for bug-tracker tools, there's yet no lifecycle modeling specified in OSLC-CM V2 trying to offer a standard meta-model of a bug's life. Much room is left for innovators, implementers, to take advantage of that standard, still keeping added value for each offering.

We hope that, due to our efforts serving as an example, more open source projects will notice OSLC and start implementing support in their tools; still, that's only wishful thinking so far. It would actually be sad if its origin in the sector of the proprietary vendors should render it difficult to notice or even to accept by open source projects, as it has very interesting technical and standardization qualities.

2.6. Other efforts

OSLC is certainly not perfect, but to our knowledge, there's no other competing alternative. Of course one may find *de-facto* standards imposed by dominant vendors in one sector of the ALM landscape, but they are then naturally dismissed in our eyes, for lack of any FLOSS implementation.

Even if competing proposals were to be issued, in our opinion, the very use of the Web standards at the heart of OSLC render it particularly unavoidable, in particular with the advent of the Web of data / Linked Data emergence that is starting to become noticed [16].

3. A CONTINUOUS INTEGRATION OSLC-CM MASHUP USE CASE

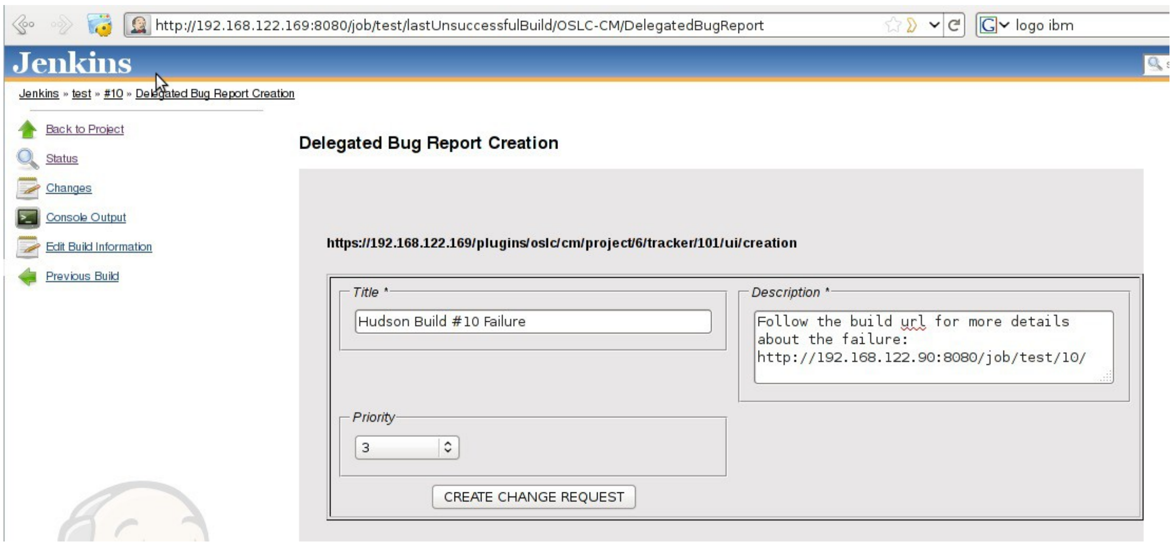
In the results of COCLICO, we have tried to assemble the necessary components of a continuous integration use case, taking advantage of the OSLC-CM specifications. Our use-case explores the situation of a Continuous Integration server running software builds (and tests) automatically. Tests may fail, so the server has to be interfaced to a bug-tracker, so that such failures can be tracked, among other problems, in the same tool used by forge users or members of the project.

We chose FusionForge trackers as the bug-tracker, which is complemented with an OSLC-CM compatible server we developed. The Continuous Integration server, which acts as an OSLC-CM consumer, is Jenkins [14] (or Hudson, as the plugin we developed should work on each of the original or forked variant of the tool).

OSLC specifications standardise two kinds of APIs that can be used in such a case. First, a problem may be reported automatically by Jenkins whenever a build fails (or, depending on its threshold and other configuration settings, only once in a while), using the OSLC-CM *Creation factory* endpoint, through an HTTP POST, resulting in full creation of a report. Or, such failures may be selectively reported only when a user, logged-in to Jenkins, notices a build failure worth to report. Then he/she may use the *Delegated creation dialogs* (standardized by OSLC) in order to use a Web *mashup* between the FusionForge tracker's Web interface and Jenkins dashboards.

3.1. Delegated bug report creation mashup

The working prototype that we have developed in COCLICO is composed of two elements: a Jenkins plugin and a FusionForge tracker OSLC-CM server (a screenshot is provided in Figure Fig 2).


http://192.168.122.169:8080/job/test/lastUnsuccessfulBuild/OSLC-CM/DelegatedBugReport
logo ibm
Jenkins
Jenkins » test » #10 » Delegated Bug Report Creation
Back to Project
Status
Changes
Console Output
Edit Build Information
Previous Build
CREATE CHANGE REQUEST

Delegated Bug Report Creation

https://192.168.122.169/plugins/oslc/cm/project/6/tracker/101/ui/creation

Title *: Hudson Build #10 Failure

Description *: Follow the build url for more details about the failure: http://192.168.122.90:8080/job/test/10/

Priority: 3

Fig 2: Jenkins and FusionForge mashup

On the request of a user logged into Jenkins and browsing its Web interface for build failures, the Jenkins plugin is able to connect to FusionForge, through Oauth (managing access permissions on behalf of a FusionForge user), to fetch a prefilled bug report. It then renders the full bug creation form of FusionForge into an *iframe* (in grey background, in Fig 2) allowing to complement the prefilled details of the build with user's anotations (such as assigning the bug to someone). Upon validation by the user, the bug report will be created in FusionForge's database, and its reference notified to Jenkins.

The Jenkins plugin is mainly composed of an AJAX mashup OSLC-CM consumer, that we have developed with JQuery. It is available on Gihub under an open source license [17]. The FusionForge OSLC-CM server/provider is developed using the Zend Framework MVC components to offer a REST API conforming to OSLC, and which displays the Web interface for delegated bug creations. It also relies on the OAuth authentication plugin also developed in COCLICO. All of this is available in the FusionForge trunk development branch under the GNU GPL license [6].

We haven't tested interfacing Jenkins with other OSLC-CM compatible bug trackers yet. This remains to be tested, but should of course work as well as with FusionForge, as is the goal of the OSLC standardization effort. One particular benefit of such a setup is to allow Jenkins and FusionForge to operate over the Internet, without having to create "fake" bug creation accounts in the tracker for Jenkins to connect. With Oauth [18], the CI server connects on behalf of a real FusionForge user who has explicitely chosen to delegate it some of its privileges. Thus, Jenkins may for instance offer the ability to the reporting user to assign a report to a colleague, maybe, even though it wouldn't have privileges to remove any other existing reports.

4. INTER-LINKED DEVELOPMENT FACTS

With the use of RDF and Linked-Data approach [13] in OSLC compatible services, and with their deployments on public forge services hosting open source development projects, not only will it be possible to integrate better the tools used by these projects, but also, allow novel analysis or large scale Quality Assurance process [3].

For instance, large scale bug tracking will be facilitated not only by the deployed interoperable bug trackers, but also by the interlinkable nature of the bug reports. Semantic annotations will become inherently part of the interlinking of reports, to track duplicates, or similar bug reports. This will be particularly important for instance in the open source distributions ecosystem, where many packagers work on similar programmes, but in independent distributions bug-trackers. Their users face similar problems all the time, but communication between these people isolated in their project's silos is not easy. Once all similar reports in different bug-trackers can be navigated, problems will be much more quickly and efficiently solved, improving greatly the QA process of the FLOSS ecosystem. A further elaboration on this potential is developed in [1], even though, at that time, implementations based on OSLC hadn't yet been fully tested.

5. CONCLUSION

We have demonstrated the usefulness and the potential of the *Open Services for Lifecycle Collaboration (*OSLC) specifications through the integration of Jenkins and FusionForge OSLC compatible server and consumer, achieving a well suited mashup solution for integration of a Continuous Integration service with a bug-tracker. Even though OSLC was born among proprietary vendors, it was devised with truly open standard principles, reusing existing Web standards, which make it particularly acceptable by Free / Libre / Open Source tools implementers.

We believe OSLC should eventually become a natural standard for interoperability of open source ALM tools, including software forges. Still, such interoperability scenarios will only deliver their potential once OSLC compatible services will be available and deployed in the tools actually used by the software development projects. To date, sadly, very few open source projects have yet shown interest in OSLC.

We hope that our efforts will serve as an example for other open source projects, and that the availability of OSLC compatible connectors in Eclipse Mylyn, and a reference implementation through Eclipse Lyo will foster OSLC adoption. Only if vendors and developers learn and implement OSLC will we see real and concrete pragmatic interoperability. Then only, can we hope to implement new uses like Inter-Linked development facts applications available, for instance for solving large scale QA challenges like global bug tracking in the entire open source ecosystem.

REFERENCES

[1] Olivier Berger, Valentin Vlasceanu, Christian Bac, Quang Vu Dang, Stéphane Laurière: Weaving a semantic web across oss repositories: Unleashing a new potential for academia and practice. International Journal of Open Source Software and Processes (IJOSSP), 2010.

[2] Scott Bosworth, Carl Zetie: The business value of open collaboration. Technical report, IBM Rational, 2010. http://open-services.net/html/ opencollab.pdf.

[3] Aftab Iqbal, O. Ureche, M. Hausenblas, and G. Tummarello: LD2SD: Linked Data Driven Software Development. In 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 09), Boston, USA, 2009.

[4] Dirk Riehle, John Ellenberger, Tamir Menahem, Boris Mikhailovski, Yuri Natchetoi, Barak Naveh, Thomas Odenwald: “Open Collaboration within Corporations Using Software Forges.” IEEE Software, vol. 26, no. 2 (March/April 2009). Page 52-58.

[5] OSLC community Website: http://open-services.net/

[6] FusionForge project Website: http://fusionforge.org/

[7] COCLICO project Website: http://www.coclico-project.org/

[8] Sourceforge: http://sourceforge.net

[9] OSLC Core specification: http://open-services.net/bin/view/Main/OslcCoreSpecification

[10] HELIOS project: http://heliosplatform.sourceforge.net/

[11] OSLC open source implementation: http://oslc-tools.sourceforge.net/

[12] Resource Description Framework (RDF) W3C standard: http://www.w3.org/RDF/

[13] Linked Data paradigm: http://www.w3.org/standards/semanticweb/data

[14] Jenkins project Website: http://jenkins-ci.org/

[15] Eclipse Lyo project: http://www.eclipse.org/projects/project_summary.php?projectid=technology.lyo

[16] “Tim Berners-Lee on the next Web” TED talk: http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html

[17] Jenkins plugin for OSLC-CM: https://github.com/jenkinsci/oslc-cm-plugin

[18] The OAuth protocol: http://oauth.net/