Deliverable 1.4

Early LOD2 Stack Prototype

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due Date of Deliverable</td>
<td>31/08/2011</td>
</tr>
<tr>
<td>Actual Submission Date</td>
<td>03/09/2011</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP 1, Requirements, System Design and LOD2 Stack Prototype</td>
</tr>
<tr>
<td>Task</td>
<td>T1.4</td>
</tr>
<tr>
<td>Type</td>
<td>Report &amp; Source Code</td>
</tr>
<tr>
<td>Approval Status</td>
<td>In Review</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Number of Pages</td>
<td>24</td>
</tr>
<tr>
<td>Filename</td>
<td>D1.4 Early Stack Prototype.docx</td>
</tr>
</tbody>
</table>

Abstract:
This report will describe the LOD2 Stack and the LOD2 demonstrator. These are the first steps towards integrating the components in one easy to use collection.

The information in this document reflects only the author’s views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided “as is” without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.
### History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>28/08/2011</td>
<td>First version</td>
<td>TenForce</td>
</tr>
<tr>
<td>0.1</td>
<td>04/03/2011</td>
<td>Review</td>
<td>ULEI</td>
</tr>
<tr>
<td>0.2</td>
<td>05/03/2011</td>
<td>Review / small fixes</td>
<td>ULEI</td>
</tr>
</tbody>
</table>

### Author List

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Name</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>TenForce</td>
<td>Bert Van Nuffelen</td>
<td><a href="mailto:Bert.Van.Nuffelen@tenforce.com">Bert.Van.Nuffelen@tenforce.com</a></td>
</tr>
<tr>
<td>TenForce</td>
<td>Bastiaan Deblieck</td>
<td><a href="mailto:bastiaan.deblieck@tenforce.com">bastiaan.deblieck@tenforce.com</a></td>
</tr>
<tr>
<td>ULEI</td>
<td>Sebastian Tramp</td>
<td><a href="mailto:tramp@informatik.uni-leipzig.de">tramp@informatik.uni-leipzig.de</a></td>
</tr>
<tr>
<td>ULEI</td>
<td>Sören Auer</td>
<td><a href="mailto:auer@informatik.uni-leipzig.de">auer@informatik.uni-leipzig.de</a></td>
</tr>
<tr>
<td>TenForce</td>
<td>Johan De Smedt</td>
<td><a href="mailto:Johan.de-smedt@tenforce.com">Johan.de-smedt@tenforce.com</a></td>
</tr>
</tbody>
</table>
Executive Summary

This document outlines the LOD2 stack (http://stack.lod2.eu) as Debian package repository. It describes how to build Debian packages for new components and how to contribute a package to the repository. It also describes an early prototype of the LOD2 demonstrator. This demonstrator aims at supporting end-users in the publication and consumption of Linked Data. This is done by offering easy to use actions and supporting the information flow between the different components. The current version is the first release of this integration process. In WP6 (the follow up work package of WP1) the integration will be further elaborated.
## Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DAK</td>
<td>Debian Archive Kit</td>
</tr>
<tr>
<td>EDOAL</td>
<td>Expressive and Declarative Ontology Alignment Language</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract, Transform, Load</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>LOD</td>
<td>Linked Open Data</td>
</tr>
<tr>
<td>LSL</td>
<td>Link Specification Language</td>
</tr>
<tr>
<td>NIF</td>
<td>NLP Interchange Format</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>OLAP</td>
<td>OnLine Analytical Processing</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>RDFa</td>
<td>Resource Description Framework in attributes</td>
</tr>
<tr>
<td>RDFS</td>
<td>RDF Schema</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>SKOS</td>
<td>Simple Knowledge Organisation System</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-Oriented Architecture</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>WYSIWYG</td>
<td>Acronym for ‘What You See Is What You Get’</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **INTRODUCTION**.................................................................................................................. 7
2. **LOD2 STACK**..................................................................................................................... 8
   2.1 DEBIAN PACKAGES CONCEPTS AND CREATION............................................................... 9
       2.1.1 Debian packages’ main concepts ................................................................................. 9
       2.1.2 Example of meta-packaging: OntoWiki ....................................................................... 10
       2.1.3 Creating Debian packages for your component .............................................................. 11
       2.1.4 Licensing ..................................................................................................................... 11
       2.1.5 Inter-component integration requirements .................................................................... 11
   2.2 LOD2 DEBIAN REPOSITORY .............................................................................................. 11
       2.2.1 Contributing to the repository .................................................................................... 11
   2.3 INSTALLATION & SETUP OF THE LOD2 STACK................................................................. 12
       2.3.1 Remote server or command line installation ................................................................. 13
       2.3.2 Desktop computer ....................................................................................................... 14
       2.3.3 Virtual Environments .................................................................................................. 16
       2.3.4 Cloud Services ............................................................................................................ 17
       2.3.4.1 Clients ....................................................................................................................... 17
       2.3.4.2 API Cloud ................................................................................................................ 17
       2.3.4.3 Execution Cloud ...................................................................................................... 18
       2.3.4.4 Storage Cloud ........................................................................................................... 18
   2.4 PREINSTALLED LOD2 STACK SYSTEMS .......................................................................... 18
   2.5 AUTOMATIC TESTING ....................................................................................................... 18
   2.6 THE LOD2 DEMONSTRATOR ............................................................................................ 18
       2.6.1 Available components ................................................................................................. 19
       2.6.2 Dependencies between components ............................................................................ 21
3. **PLANNING** .......................................................................................................................... 21
4. **CONCLUSION** ..................................................................................................................... 23
5. **BIBLIOGRAPHY** .................................................................................................................. 24
List of Figures

Figure 1: Envisioned support of different stages of the Linked Data life-cycle by the LOD2 Stack............. 7
Figure 2: Example DEB-package dependency tree (OntoWiki)........................................................................ 10
Figure 3: Screenshot of Synaptic (1)........................................................................................................... 14
Figure 4: Screenshot of Synaptic (2)........................................................................................................... 15
Figure 5: Screenshot of Ubuntu Software Center......................................................................................... 16
Figure 6: Example cloud setup.................................................................................................................... 17
Figure 7: Welcome page of the LOD2 demonstrator................................................................................... 19
Figure 8: Dependency graph of the LOD2 demonstrator............................................................................ 21

List of Tables

Table 1: Dependency graph of the LOD2 demonstrator............................................................................. 21
1. Introduction

This document outlines the version 1.0 of the LOD2 stack and the LOD2 demonstrator. The LOD2 stack is an integrated distribution of aligned tools which support the whole life cycle of Linked Data from extraction, authoring/creation over enrichment, interlinking, fusing to maintenance (see Figure 1). The LOD2 Stack comprises new and substantially extended existing tools from the LOD2 partners and third parties. The major components of the LOD2 Stack will be open-source in order to achieve a quick penetration and deployment and it will scale to knowledge bases with hundreds of billions of triples and large numbers of concurrent users (supported by its deploy-ability on clusters of commodity hardware and existing cloud infrastructures).

![Linked Data Lifecycle Diagram]

Figure 1: Envisioned support of different stages of the Linked Data life-cycle by the LOD2 Stack.

Our agile, iterative software development approach (with a first release after year 1, overall 5 iteration cycles – 3 for the LOD2 Stack and 2 for the use case adaptations) aims at ensuring that the stack fulfils a broad set of user requirements and thus pushes the transition to a Web of Data. The stack is designed to be versatile, for all functionality we will define clear interfaces, which enable the plugging in of alternative third-party implementations.

In order to fulfill these requirements the architecture of the LOD2 Stack is based on three pillars:

1. **Software integration and deployment using the Debian packaging system.** The Debian packaging system is one of the most widely used packaging and deployment infrastructures and will facilitate the packaging and integration as well as the maintenance of dependencies between the various LOD2 Stack components. Using the Debian system will also allow to facilitate the
deployment of the LOD2 stack on individual servers, cloud or virtualization infrastructures.

2. **Use of a central SPARQL endpoint and standardized vocabularies for knowledge base access and integration** between the different tools of the LOD2 stack. All components of the LOD2 Stack will access this central knowledge base repository and write their findings back to it. In order for other tools to make sense out of the output of a certain component it is important to define vocabularies for each of the stages of the Linked Data life-cycle.

3. **Integration of the LOD2 stack user interfaces based on a REST enabled Web Applications.** Currently the user interfaces of the various tools are technologically and methodologically quite heterogeneous. We do not plan to resolve this heterogeneity, since each tool’s UI is specifically tailored for a certain purpose. Instead, we plan to develop a common entry point for accessing the LOD2 Stack UI, which will then forward a user to a specific UI component provided by a certain tool in order to complete a certain task.

These three pillars comprise the methodological and technological framework for integrating the very heterogeneous LOD2 Stack components into a consistent framework. In this document we describe a first version of the 3 pillars. We describe how to build Debian package and how to contribute them to the LOD2 stack repository (pillar 1). The LOD2 demonstrator forms the common entry point which guides users to the right tool for the task under consideration (pillar 3). The knowledge flow between the components is the ability of the LOD2 components to interact with SPARQL endpoints to read data from and to write data to.

For a more exhaustive overview of the different components and their functionalities we refer to the Deliverable D1.3.

### 2. LOD2 Stack

One of the major problems for software producers and in particular open source software producers is to create a deployment system for the software they create, which allows easy and hassle-free installation of the components.

For the LOD2 project, we decide to utilize the Debian package management system for this task. Debian is a Linux based operating system which is very popular and influential due to its stable and reliable releases. Debian and its applications can be deployed on a wide variety of hardware ranging from very small devices to complete cloud infrastructures very easily.

Debian is used as a base for many other distributions, for example Ubuntu - which recently got high attraction. A crucial part in that success was the construction of a high quality packaging system which allows e.g. to express hardware requirements and dependencies between packages. For this reason, we selected Ubuntu as the Operating System, nowadays one of the most popular Linux distributions. This choice allows us to play on long term stability by requiring that a component should be able to run w.r.t. the Ubuntu 10.04 release, supported by [www.ubuntu.com](http://www.ubuntu.com) until April 2013. At the same time we can make packages on more recent OS libraries by using one of the half-yearly releases.

In the following we describe the LOD2 Debian repository and the processes which have been setup to bring the components of the LOD2 stack together in a central repository. We also describe the automated validation process we set up to check whether new components and their updates do not break the system’s integrity.
Debian packages concepts and creation

2.1.1 Debian packages’ main concepts

In the Debian package management system, software is distributed in architecture-specific binary packages and architecture-independent source code packages. A Debian software package consists of two archives:

- An archive which holds the control information of that package, and
- An archive which represents the software itself.

The control information of a Debian package will be indexed and merged together with all other control information from other packages available for the system. This information consists of descriptions and attributes for:

- The software itself (e.g. licenses, repository links, name, tagline, ...),
- Its relation to other packages (dependencies and recommendations),
- The authors of the software (name, email, home pages), and
- The deployment process (where to install, pre and post install instructions).

The most important part of this control information is the definition of the relations to other software. This allows the deployment of a complete stack of software with one action. The following dependency relations are commonly used in the control information (1):

- **Depends**: This declares an absolute dependency. A package will not be configured unless all of the packages listed in its Depends field have been correctly configured. The Depends field should be used if the depended-on package is required for the depending package to provide a significant amount of functionality. The Depends field should also be used if the install instructions require the package to be present in order to run.
- **Recommends**: This declares a strong, but not absolute, dependency. The Recommends field should list packages that would be found together with this one in all but unusual installations.
- **Suggests**: This is used to declare that one package may be more useful with one or more others. Using this field tells the packaging system and the user that the listed packages are related to this one and can perhaps enhance its usefulness, but that installing this one without them is perfectly reasonable.
- **Enhances**: This field is similar to Suggests but works in the opposite direction. It is used to declare that a package can enhance the functionality of another package.
- **Conflicts**: When one binary package declares a conflict with another using a Conflicts field, dpkg will refuse to allow them to be installed on the system at the same time. If one package is to be installed, the other must be removed first.

All of these relations may restrict their applicability to particular versions of each named package (the relations allowed are <<, <=, =, >= and >> for strictly earlier, earlier or equal, exactly equal, later or equal and strictly later, respectively). This is useful to force the upgrade of a complete software stack.

In addition to this, dependency relations can be set to a list of alternative packages. In such a case, if any one of the alternative packages is installed, that part of the dependency is considered to be satisfied. This is useful if the software depends on a specific functionality on the system instead of a concrete package (e.g. a mail server or a web server).

Another use case of alternative lists are meta-packages. A meta-package is a package which does not contain any file or data to be installed. Instead, it has dependencies to other packages or lists of
alternative packages. There are several use cases for meta-packages. An example is given in the next section.

2.1.2 Example of meta-packaging: OntoWiki

To build an appropriate package structure, the first step is to inspect the manual deployment of the software, its variants and the dependencies of these variants.

OntoWiki is a browser-based collaboration and exploration tool as well as a application for linked data publication. There are two clusters of dependencies: the runtime environment and the backend. Since OntoWiki is developed in the script language PHP, its architecture is independent but needs a web server running PHP on it. More specific, OntoWiki needs PHP5 running as an Apache 2 module. Other configurations, e.g. PHP running with the fastcgi module, are neither tested nor supported. OntoWiki currently supports two different back-ends which can be used to store and query RDF data: Virtuoso and MySQL. Virtuoso is also part of the LOD2 stack while MySQL is a standard package in all Debian-based systems. In addition to OntoWiki, the user can use the OntoWiki command line client owcli and the DL-Learner from the LOD2 stack to enhance its functionality.

![Figure 2: Example DEB-package dependency tree (OntoWiki)](image)

Some explanation about this figure:
- Boxes are part of the LOD2 stack,
- Ellipses are part of the Debian base system,
- Dashed forms are meta-packages,
- Relations \(\Rightarrow\) Depends (D), Depends alternative list (A), Conflicts (C) and Suggests (S).

The tree is far away from being complete since every component also depends on libraries and additional software which is omitted here.

Given this background information, we can start to plan the packaging. We assume that users either use MySQL or Virtuoso as a backend on a server, so the first decision to take, is to split this functionality in two packages: ontowiki-mysql and ontowiki-virtuoso.

These two packages are overtopped by the meta-package ‘ontowiki’, which requires either ontowiki-mysql or ontowiki-virtuoso and which can be used by other LOD2 stack packages to require OntoWiki. Since
both the MySQL backend and the Virtuoso backend version use the same system resources, we need to declare them as conflicting packages.

2.1.3 Creating Debian packages for your component

The process of making a Debian package depends on the applications' source code and building process. The main tool is build. Many of the LOD2 applications currently in the LOD2 are java web applications deployed via tomcat. If one uses maven to build the application, the build process of the Debian package can be integrated in maven. There exist plugins that support the construction of Debian packages but not all are so easy to configure due lack of documentation. In

http://code.google.com/p/lod2-stack/source/browse/#svn%2Ftrunk%2Fdocuments

the document demonstrator_implementation.pdf describes how the lod2demo package is being built by maven.

2.1.4 Licensing

There is no specific license imposed by the LOD2 stack repository on the components except that it has to be an open-source license which allows end-users to use, configure and interact with the contributed component. Every component-owner is responsible for validating that his tool satisfies its chosen open-source license.

2.1.5 Inter-component integration requirements

The contribution of the component as a Debian package (or set of Debian packages) to the LOD2 stack repository is the first step in the integration process. A next level is to ease the communication between the different components. For that we standardize upon communicating via SPARQL using SPARQL endpoints. Hence, we require for integration the components input and output to be a SPARQL endpoint description w.r.t a graph. With that the basic glue between the components in the current version of the LOD2 stack is made. In the future more requirements could come like to allow external programs to share and edit the component functionality configuration.

LOD2 Debian repository

Apart from creating single packages, all LOD2 stack software will be shared at a single repository. A Debian repository is a directory where you can find software which you can install on Debian-based systems. Repository directories are accessible via FTP or HTTP and consists of the Debian packages as well as index files for the repository.

In order to create a valid package repository, the index needs to be maintained after uploading new packages. The Debian Archive Kit (DAK, http://wiki.Debian.org/DakHowTo) is a collection of tools that work together to manage a Debian package repository.

2.2.1 Contributing to the repository

The repository is publicly available at http://stack.lod2.eu/. The repository forms a trusted group of components. This is achieved by a signing procedure while uploading a new (version of a) component. The component owner signs the component package with its private key. This signature is validated with his public key known by the repository. To contribute a component to the LOD2 Debian repository, the component owner must create a key pair for signing the package and also a key for the secure connection to the repository. The GPG key pair can be recreated with
01: gpg --gen-key

The public key is exported with
02: gpg --export -a > publickey.asc

The ssh-key for the upload can be generated as follows
03: ssh-keygen -t dsa

All these public keys are sent to the LOD2 stack repository responsible (support-stack@lod2.eu). When added to the trusted key ring one can upload of the component package using the tool dput. The local configuration for dput is the following (it should be stored in the file ./dput.cf)

04: [lod2]
    fqdn = stack.lod2.eu
    method = scp
    login = packaging
    incoming = /var/www/stack.lod2.eu/deb/mini-dinstall/incoming

The upload command is
05: dput lod2 <packet>.changes

where <packet>.changes is the changes file created during the package creation.

For the signing and upload procedure the distribution identifier in the changelog file (describing the changes of the package from version to version) must be set to lod2. Hereunder one sees and extract of the changelog of the LOD2 demonstrator.

06: lod2demo (1.1.13) lod2; urgency=low
    [ Bert Van Nuffelen (TenForce/LOD2) ]
    * add intro page
    * apply more mockup style

After the Debian package is uploaded the whole repository has to be signed by one trusted partners.

### Installation & setup of the LOD2 stack

In general, deploying the LOD2 software stack or parts of it is very easy and comfortable when it’s based on the Debian package management system. There are only two execution steps to install LOD2 stack software:

1. Add LOD2 stack package repository to the systems repository list and update the repository index.
2. Install wanted software packages by using a graphical or text-based package management application.

The following subsections describe some deployment details on different platforms.
2.3.1 Remote server or command line installation

Installing Debian software on a remote server means that administrators have to use command-line front-ends to the package management system. The following command line applications are commonly used on remote servers (2):

- **apt-get** is the most basic package management tool and the preferred command line front-end for non-interactive package management. The procedure described below uses this one.
- **aptitude** is the most versatile package management tool and the preferred text front-end for interactive package management. It offers a full screen interactive text user interface, an enhanced package resolver and an enhanced search function. It is most suitable for the daily package management.

The first step is the installation of the LOD2 repository package.

```
07: wget http://stack.lod2.eu/lod2repository_current_all.deb
08: sudo dpkg -i lod2repository_current_all.deb
09: sudo apt-get update
```

This registers the LOD2 stack repository for the local package manager. As the LOD2 stack component Sig.ma EE depends on the Oracle-Sun Java suite - and this Java package is not always per default accessible - one has to update the repository with the following:

```
10: sudo add-apt-repository 'deb http://archive.canonical.com/ lucid partner'
11: sudo apt-get update
```

where ‘lucid’ is the Ubuntu release. During the Oracle-Sun Java package installation acceptance confirmation of the license is requested.

Next, one can install the lod2demo package. This will install the whole LOD2 stack as it depends on all components.

```
12: sudo apt-get install lod2demo
```

If MySQL and Virtuoso are not installed yet, the root passwords for Virtuoso and MySQL will be requested. Please use the root password “dba” for Virtuoso for now (we will change this soon). In case this is chosen differently -- best done when the intention is to make the machine public accessible -- then one has to adapt the configuration of OntoWiki. In the short future a release will be ready where this is lifted.

The LOD2 demonstrator is now accessible at [http://localhost:8080/lod2demo](http://localhost:8080/lod2demo).

In order to make all functionality to which the LOD2 demonstrator refers visible one has to do some manual post configuration.

- Virtuoso iSPARQL package must be activated manually via the conductor interface. If it is not activated then the menu option querying via isparql is not working properly.

The default configuration assumes a local usage of the LOD2 stack, not accessible for the outside world. If one wants to expose the LOD2 demonstrator or one of the components, we advise to setup a proxy. For instance one can use the apache2 server as proxy.

To expose the LOD2 demonstrator, its configuration graph has to be adapted to the location of the different components. The LOD2 demonstrator configuration file is the graph http://localhost/lod2demo/configuration, which is stored in Virtuoso. It contains the hostname prefix of the tools. Per default the value is
"http://localhost:8080". To make the tools accessible via a public LOD2 demo interface, this value has to be changed to the URL of the system.

2.3.2 Desktop computer

Installing Debian-based software on desktops is typically done with a graphical package manager. Currently, Synaptic is usually used on Debian/Ubuntu systems. It can be used to install, remove and upgrade software packages as well as to add repositories.

![Screenshot of Synaptic](image.png)

*Figure 3: Screenshot of Synaptic (1)*

*(Shows the list of package repositories and functions to add and delete them from the package index.)*
Figure 4: Screenshot of Synaptic (2)
(Shows list of LOD2 stack packages by filtering on 'LOD2' in the descriptions)

The above figures show screenshots of Synaptic in action. Features of Synaptic include (3):
- Install, remove, upgrade and downgrade single and multiple software packages
- System-wide upgrade (e.g. from a stable to a testing distribution)
- Manage package repositories (see Figure 3)
- Find packages by name, description and several other attributes (see Fehler! Verweisquelle konnte nicht gefunden werden.)
- Select packages by status, section, name or a custom filter
- Sort packages by name, status, size or version
- Browse available online documentation related to a package
- Download the latest change log of a package
- Lock packages to the current version
- Force the installation of a specific package version
- Undo/redo selections
- Get screenshots of applications (from http://screenshots.debian.net)

Using Synaptic is easy but some users, e.g. Linux beginner, can have problems with the pure amount of packages available. On a standard Ubuntu distribution there are over 32,000 packages available, where nearly 1500 are installed with the base system. Finding specific software can be problematic if you do not know what software exists. Another deployment option is to use a web browser for searching and installing. Debian-based browser installations understand the apt: link type. To simplify the deployment
even more, we will in addition add these links to our resource pages on http://lod2.eu so that desktop users can install LOD2 software instantly.

In addition to Synaptic, there are alternative front-ends for the Debian software package system:

- **KPackage** is a front-end for the KDE Desktop environment (http://www.general.uwa.edu.au/u/toivo/kpackage/).
- **The Ubuntu software Center** (https://launchpad.net/software-center) is a simple front-end for Linux beginners.

![Screenshot of Ubuntu Software Center](image)

**Figure 5: Screenshot of Ubuntu Software Center**

*(Shows the list of packages having LOD2 in their description.)*

### 2.3.3 Virtual Environments

Virtual environments (hardware virtualizations) are often used to simplify the operating system landscape in data centers and as such achieve a server consolidation. Doing this, multiple small physical servers can be replaced by a larger physical server to increase the utilization of costly hardware resources such as CPU’s.

Each operating system running on a physical server is converted to a distinct operating system running inside a virtual machine. This virtual machine emulates the hardware for the guest system. Typical **software for running virtual machines** is:

- **VirtualBox**, an x86 virtualization software package, purchased by Sun Microsystems, and now developed by Oracle.
- **VMWare**, a virtualization-software server suite developed and supplied by VMware, Inc.
- **Xen**, a virtual machine monitor originally developed by the University of Cambridge.

All these solutions are able to emulate i686 hardware to host a Debian/Ubuntu-based operating system as a guest. This virtual system is treated as a normal remote system. LOD2 software can be installed both with command-line and graphical tools on these systems.
In addition to that, a virtual machine is planned which can be downloaded and started and which acts as a showcase for all LOD2 stack software.

2.3.4 Cloud Services

As a next step in the development of virtual hardware systems, cloud computing recently got a momentum. Cloud computing is computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services (4).

One major player in the cloud computing service business is Amazon.com. Amazon.com’s cloud computing platform is centered around the Amazon Elastic Compute Cloud (EC2). Since EC2 supports Debian as an operating system environment (S), deploying the LOD2 stack in that cloud is as easy as deploying it on a remote server. Other services as Rackspace (http://rackspace.com) support Debian/Ubuntu guest systems as well.

Setting the Web Services in the Cloud also results in:

- **Cheap Processing Power:**
  The parallelization of cloud computing, queries can be spread across multiple CPUs simultaneously.

- **Elastic Scale:**
  Demand for capacity can be arranged is a very flexible way.

- **Massive multi-tenancy:**
  We run a single instance used by a lot of clients. This results in a small provision, service and upgrade cost compared with a classical setup.

- **Service-Orientation:**
  The delivered services count, not the hardware.

![Figure 6: Example cloud setup](Based upon the Amazon Web Services)

2.3.4.1 Clients

All clients call the same set of REST services, among which:

- Application
- Dashboards
- Embeddable widgets (like the Google widgets)
- REST API: Provides full API access to backend services. This HTTP-based API can be simply used from any 3rd party application as well as from a plain browser.

2.3.4.2 API Cloud

The API cloud consists of a series of stateless REST-based services that are commonly called by clients. They are designed to be stateless and heavily load balanced to take advantage of cloud architecture, making elastically scaling easy for Administrators to meet client requests.
2.3.4.3 Execution Cloud

The execution cloud connects front-end services with customer data. These services support asynchronous longer-running processes such as ETL (Extract, Transform, Load) and report execution. These services are made available via an asynchronous queue to optimize performance between front-end APIs and client data.

2.3.4.4 Storage Cloud

Customer data is stored securely in separate data stores. Heavy use of on-demand caches to ensure performance is optimized for users.

Preinstalled LOD2 stack systems

We have installed the LOD2 stack on a public system: [http://demo.lod2.eu/lod2demo](http://demo.lod2.eu/lod2demo). With this demo environment publication of Linked Data can be explored easily without the need to have system administration knowledge.

For those who like to explore in more detail the LOD2 stack, we have created a VMware virtual machine on which we have installed the LOD2 stack. This virtual machine will be available at the VMware marketplace: [http://www.vmware.com/appliances/](http://www.vmware.com/appliances/) The exact access details will be announced later. Regularly new versions of the stack will be uploaded.

Automatic testing

The LOD2 stack is under development. We have configured two machines for the LOD2 consortium one to validate the above deployment procedure and one to explore with several partners jointly a problem on a shared machine. It is one of the first tasks of WP6 to finalize this setup.

The LOD2 demonstrator

The wide variety of available tools and components supporting the publication of Linked Data makes the choice for system architects not easy. This is the purpose of this demonstration prototype. It brings the tools together in one story. The demonstrator has also an internal LOD2 purpose. It serves as the workbench to explore the interaction between the LOD2 stack components. This information is necessary to construct predefined configurations for specific use case problems.

The first release of the demonstrator, described here, focuses on creating an overview of all components, not on the in depth exploration of each tool individually. Consequently, for problems were multiple solutions by different components exist we have chosen for the component for which the problem is its core business and was not yet present in the demonstration tour.

In Figure 19 the welcome page of the demonstrator is shown.
Figure 7: Welcome page of the LOD2 demonstrator

Via selecting one of the menu items in one of the top menu groups the user selects a situation he wants to explore. In the Extraction & Loading, one finds ways to upload RDF files in Virtuoso, extract RDF from XML documents (using XSLT) and SQL databases and extract RDF formatted annotations for natural language documents either w.r.t. a general knowledge base (DeRePen) or a controlled vocabulary (SKOS taxonomy in PoolParty).

The querying menu contains ways to use SPARQL for querying the knowledge stored in Virtuoso. In the Authoring tab one finds tools to edit the knowledge. The Linking menu contains ways to create links between knowledge bases. The Enrichment menu points to tools which support validation, error detection and repair possibilities for the stored knowledge. Finally, in the Online Tools and Services menu, a collection of online resources is shown. Many of them are maintained by LOD2 members.

Information sharing between the components is done via the shared connection to Virtuoso. The user can set in the configuration tab a current graph (via selecting one of a list). This current graph is the input for many tasks. It is also the default output target, although that can be changed by the user if necessary. The ability to interact with SPARQL endpoints for retrieving input data and writing results to is a necessary requirement for each LOD2 component.

In the future we also aim at building a single sign on process so that a user has full access to the components with one authentication request. This task will be taken to WP6. Furthermore we also will investigate how the configuration of the tools’ functionality can be more centralized.

2.6.1 Available components

At the moment of writing, the LOD2 component stack consists of

1) Components as locally installed Debian packages
   a) LOD2 demonstrator: http://localhost:8080/loz2demo
   b) Virtuoso
      i) Conductor: http://localhost:8890/conductor
ii) SPARQL endpoint: http://localhost:8890/sparql

c) OntoWiki and OWCLI: http://localhost/ontowiki

d) SILK: http://localhost:8080/silk

e) D2R: http://localhost:8080/d2r-cordis

f) Sig.ma EE: http://localhost:8080/sigmaee

g) ORE and DL-Learner: http://localhost:8080/ore

There are also Debian packages which contain public datasets. The purpose of these datasets is to offer users of the LOD2 stack interesting datasets locally to evaluate and explore the linked data publication opportunities. They can be of course be part of a linked data solution the user is building.

2) As online components

a) Spotlight

b) PoolParty

PoolParty is available as online service for the LOD2 consortium partners. In 2012, a public cloud version will be available. At the end of the project in 2014 an open source version will be packaged and contributed to the LOD2 stack.

Next to the LOD2 components which are locally installed, the LOD2 project contributes several public online resources of information:

- The LOD cloud as one single data source: http://lod.openlinksw.com
- The semantic web index: http://sindice.com
- CKAN: http://ckan.net and http://publicdata.eu
2.6.2 Dependencies between components

The current dependencies between the packages are depicted in the figure below.

![Dependency graph of the LOD2 demonstrator](image)

Figure 8: Dependency graph of the LOD2 demonstrator

3. Planning

During the first year of the project, WP1 initiated many tasks to enable the construction of the LOD2 stack environment. This report describes several decisions and procedures the partners have made during this period. The next 3 years the coordination and technical follow-up of the LOD2 stack is part of work package WP6.

While we have achieved to centralize all technology, we have in the next year try to centralize experience and best practices for linked data publication. For that the case work packages (7, 8 and 9) and WP6 must exchange information about needs, possible choices and results for the actual choice. This is reflected in the planning of WP6 for which we outline here.

<table>
<thead>
<tr>
<th>Start</th>
<th>End</th>
<th>What</th>
<th>Who</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M24</strong></td>
<td><strong>Sprint 13</strong></td>
<td>01.09.2011 - 26.10.2011</td>
<td>Monitor the changes made in the Use Cases / Requirements</td>
<td>TenForce, SWC, WKD, ExaLead, OKFN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01.09.2011 - 26.10.2011</td>
<td>Analyse dependencies &amp; refine architecture towards the Use Cases</td>
<td>TenForce, SWC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01.09.2011 - 26.10.2011</td>
<td>Freeze the changes on the LOD2 stack component requirements</td>
<td>All</td>
</tr>
<tr>
<td><strong>Sprint 14</strong></td>
<td>27.10.2011</td>
<td>26.12.2011</td>
<td>Align with OpenGov Use Case</td>
<td>TenForce, OKFN</td>
</tr>
<tr>
<td><strong>Sprint 16</strong></td>
<td>23.02.2012</td>
<td>25.04.2012</td>
<td>Analyse/verify first release of news &amp; media datasets</td>
<td>SWC, TenForce</td>
</tr>
<tr>
<td></td>
<td>23.02.2012</td>
<td>25.04.2012</td>
<td>Define the improvements to be made</td>
<td>TenForce</td>
</tr>
<tr>
<td><strong>Sprint 17</strong></td>
<td>26.04.2012</td>
<td>27.06.2012</td>
<td>Acceptance 2 of API's/interface components</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>26.04.2012</td>
<td>27.06.2012</td>
<td>Analyse acceptance feedback</td>
<td>TenForce</td>
</tr>
<tr>
<td></td>
<td>26.04.2012</td>
<td>27.06.2012</td>
<td>Define the improvements to be made</td>
<td>TenForce</td>
</tr>
<tr>
<td><strong>Sprint 18</strong></td>
<td>28.06.2012</td>
<td>30.08.2012</td>
<td>Vary initial release of adapted LOD2 stack for large Enterprises</td>
<td>ExaLead, TenForce</td>
</tr>
<tr>
<td></td>
<td>28.06.2012</td>
<td>30.08.2012</td>
<td>Update packages for API's/interface components</td>
<td>TenForce</td>
</tr>
<tr>
<td></td>
<td>28.06.2012</td>
<td>30.08.2012</td>
<td>Freeze packages for API's/interface components</td>
<td>TenForce</td>
</tr>
<tr>
<td></td>
<td>28.06.2012</td>
<td>30.08.2012</td>
<td>Demo &amp; Accept Prototype 1</td>
<td>TenForce</td>
</tr>
<tr>
<td><strong>M36</strong></td>
<td><strong>Sprint 19</strong></td>
<td>31.08.2012 - 25.10.2012</td>
<td>Monitor the changes made in the Use Cases / Requirements</td>
<td>TenForce, SWC, WKD, ExaLead, OKFN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.08.2012 - 25.10.2012</td>
<td>Analyse dependencies &amp; refine architecture towards the Use Cases</td>
<td>TenForce, SWC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.08.2012 - 25.10.2012</td>
<td>Review availability of the modules</td>
<td>TenForce</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.08.2012 - 25.10.2012</td>
<td>Freeze the changes on the LOD2 stack component requirements</td>
<td>All</td>
</tr>
</tbody>
</table>
WP7 and WP8 are work packages that investigate the Linked Data publishing processes. WP9 is of a different nature: it is about setting up an environment for describing and managing datasets. The components of the LOD2 tool stack are mostly applied in WP7 and WP8. However, the outcome of WP9, a public repository of datasets, is the component required to announce new datasets published using the LOD2 tool stack as public and open data sets. To aid this process we foresee the creation of a component in the LOD2 tool stack that eases the announcement and management of new linked datasets to the public repository. The definition of this package is a task of the first quarter of the first year of WP6.

The experiences and use cases will impact on the interplay between the available components. For common interaction flows, we intend to create downloadable configurations which will ease the setup of the LOD2 stack for those situations. In this way the learned experience will be documented.

We expect that components will be improved and new components will be added to the stack. Consequently, an updated version of the LOD2 demonstrator will be presented at the end of the first year of WP6.

Due to the extension of the LOD2 consortium with new members and a new use case work package the original planning for WP6 which is shown above will be revised. This is will be discussed during the plenary in Leuven (19-23 September 2011).
4. Conclusion

We have setup a Debian package repository as distribution platform of the LOD2 software components. These components are all accessible from the common entry point the LOD2 demonstrator. This work gives a firm ground to a) improve the interaction between the available components so that more Linked Data publication challenges can be tackled easier and b) to improve the distribution and ease the deployment of LOD components.
5. Bibliography


