Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 2. pp. 331–338.

Web Ontology for Software Package Management

Péter Jeszenszky

Debreceni Egyetem, Informatikai Kar

Abstract

In this paper the author presents an OWL web ontology that defines classes and properties to represent software package metadata in RDF. As part of this work a collection of software tools is provided that support various software package formats and extracts metadata as RDF triples, providing a rich source of information for Semantic Web applications.

As demonstrated, this framework can be used to publish the full metadata content of a software package repository as linked data, thus allowing repositories to become part of the Semantic Web. The framework is also intended to serve as a foundation for a unified software package dependency analyzer tool, and maybe for a unified package management tool that can support arbitrary package formats.

Keywords: software package, metadata, package management system, dependency, Semantic Web, web ontology, RDF, linked data

1. Introduction

Applications typically consist of files that must be copied to specific locations of the file system during installation. Many other tasks may be necessary to be performed before the application can be run. For example, compilation is required if the application is available in source code form, that may be quite difficult. Setting file permissions and ownership is also critical for security.

To make installation easier vendors often distribute their software products together with custom installers. Third party Windows software is usually distributed such way.

Keeping installed software up-to-date is another problem. Windows systems have automatic update functionality, MS software and third party drivers can be updated in this way. Third party Windows applications may also have such capability.

Linux distributions are usually based on free and open source software. They consist of a kernel and a collection of packages. Free software can be packaged uniformly, packages are available from repositories over network. Tools are provided to install new software and to maintain installed software. Ian Murdock, founder of the Debian Linux distribution stated that package management is "the single biggest advancement Linux has brought to the industry" [1].

Modern Linux distributions and Unix-like operating systems consist of hundreds or thousands of software packages. The handling of software packages on these systems is carried out by a package management system.

A package management system typically uses a specific software package format and provides a wide range of functionality, such as automatic package updates and handling dependencies amongst software packages automatically.

Package management can be used to maintain an entire system up-to-date, including the operating system. Unix-like systems may use several different solutions. Many of them are based on the the RPM package format (for example, CentOS, Fedora, Mandriva Linux, openSUSE and Red Hat Enterprise Linux). Many others are built on the .deb package format (for example, Debian, Ubuntu and KNOPPIX). Gentoo Linux and its derivatives, FreeBSD and others of the family also have their own package management system.

Applications may also provide package management to extend their functionality via add-on packages. An example is the open source R statistical computing environment [2] that has its own software package format and package management system. Application developers can also benefit from the advantages of package management. Many programming languages and environments provide tools to install programming libraries using package management techniques. For example, Apache Maven [3] is an open source project management and comprehension tool that revolutionizes Java development providing full featured package management.

Package management is a widely used technique. There are many different applications and implementations but they also have a lot in common. For example, packages have the following important characteristic feature: they are rich sources of metadata. The main goal of this work is to make these data sources available to Semantic Web applications. We show that package metadata can be published as Linked Data. The result can be considered as a contribution to the building of the emerging Semantic Web. As part of the work several OWL web ontologies have been developed. These ontologies provide a general framework to model package interdependencies. They are intended to serve as a foundation for a unified software package dependency analyzer tool, and maybe for a unified package management tool that can support arbitrary package formats.

The paper is organized as follows. Section 2 introduces the basic concepts of package management. Section 3 gives a description of our framework to publish package metadata as Linked Data. Section 4 presents the OWL ontology behind the framework. Section 5 summarizes further development plans of the author. Finally, Section 6 concludes.

2. Basic concepts

2.1. Software package

The term software package refers to a unit of distributable and installable software. A software package is usually provided as a single archive file that contains computer software to be installed. Packages always have a name and a version number. If a package is distributed as an archive, these are usually appear in the filename. They usually, but not necessarily, represent an application or a service. For example, the `filesystem` Fedora RPM package contains the basic directory layout for an operating system. Obviously, this package does not provide any kind of application.

Packages may contain source code or binaries, together with additional data files. They usually contain metadata, such as the full name of the package, its version number, description and license, and the list of its prerequisites. Packages may require a specific hardware and software environment, as well as many other packages to be installed beforehand. The latter requirements are often referred as dependencies. Embedding metadata is file format specific.

2.2. Package management system

A package management system is an application that provides functionality to install packages automatically and uniformly, and to other related tasks, such as to remove and upgrade installed packages. It usually uses a specific package format, such as RPM or .deb. It usually maintains a local database that stores metadata about installed packages. Package management systems may provide varying levels of convenience. Sometimes a distinction is made between low-level and high-level package management tools. High-level tools are often based on low-level tools providing a more convenient user interface and extra functionality. For example, a high-level tool can download and install package dependencies automatically.

2.3. Repository

Vendors often use network repositories to distribute their packages. This is the usual way of package distribution in the Linux world. Software repositories are locations where collections of packages are available for installation. They are usually accessed by package management systems over network, but can also be available on CD/DVD. Network repositories may use advanced access methods, or they can simply be an FTP or HTTP server.

Repositories usually provide only the most up-to-date version of packages. They often maintain a database of available packages. The database stores metadata gathered from the individual packages. Little if any additional data is provided. Packages may be classified into groups, but no information is supplied about the abstract entity behind related package versions. The database can be used to perform search or to extract dependency information quickly.

3. Publishing package metadata as Linked Data

The term Linked Data [4, 5] refers to a style of publishing RDF on the Web, that is based on the use of dereferenceable URIs. In our case, this allows users and applications to navigate between packages following RDF links that represent dependencies and other relationships, giving people the feeling of browsing (browsing the Semantic Web). This kind of RDF publishing is a current and popular topic. Many data sources are available as Linked Data sets, see [6] for a comprehensive list. The authors personal favorite is the DBpedia [7] project that provides Wikipedia content as Linked Data. The increasing number of Linked Data sets and the fact that they can be interconnected make Semantic Web become a reality!

As part of the work the author has been developed tools to retrieve metadata from packages and to transform this information to RDF. Currently, RPM and Debian packages are supported, as well as R packages.

Debian packages are archive files that are created with the Unix `ar` utility. Such an archive contains two additional compressed `tar` archives, one for the files and one for metadata. Debian uses so-called control files to store metadata in the second archive. These are plain text files that have a simple structure and contain metadata field names and values [8].

R packages are gzipped `tar` or ZIP archive files that contain the files of the package in a specific directory layout. Similarly to Debian packages, metadata is stored in Debian control files in the archive [9].

RPM packages are binary files that use a special data structure called the header to store all available information about the package [10].

In the case of Debian and R packages metadata can be extracted easily using unpacking and text file parsing. RPM header information is stored binary that requires a bit more sophisticated processing. Fortunately, programming languages may have support to work with RPM packages. The functionality of these tools is also available as a RESTful web service [11] that accepts a package URI and returns metadata in RDF.

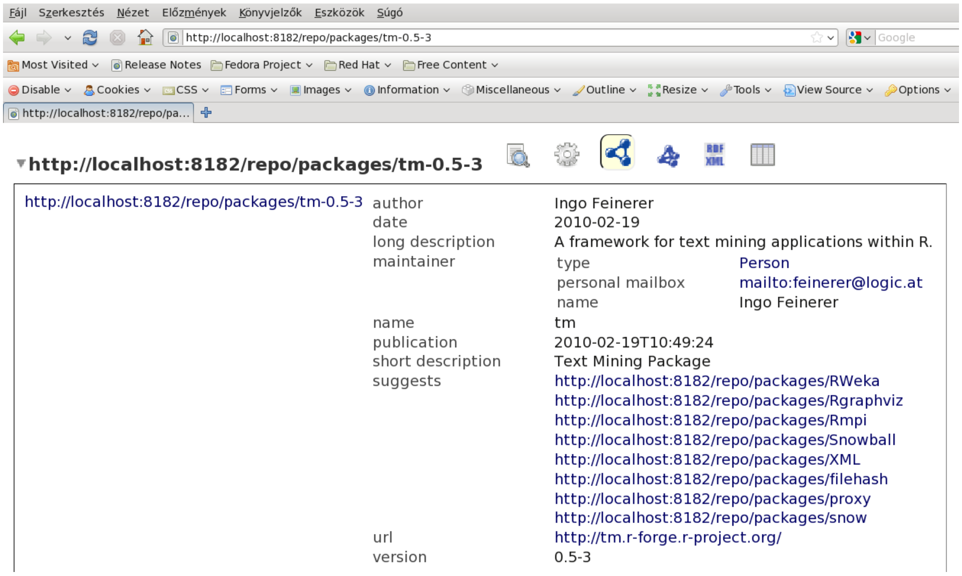
The above mentioned utilities operate on single files and they do not produce Linked Data and are not suitable to explore dependencies and other relationships between packages. In order to achieve our goal, packages must be considered in context. In our case that context is a repository. Package management systems based on any of the three above mentioned formats support repositories.

Our system provides linked data views of Yum RPM repositories [12] and CRAN-style repositories holding R packages [13]. The linked data front-end to a repository is based on the repository database that is converted into an RDF database. The RDF database can be accessed via a SPARQL endpoint or using a RESTful web service.

Yum databases are complete, as they contain all available information about packages. On the other hand, CRAN repository indices are partial and contain only the most important metadata items for each package. In this case the index is used to build an initial RDF database that can be populated with missing information on demand automatically. To provide all available information about a package the web service can perform on the fly metadata extraction from packages in the repository. Once a package is requested all of its metadata is available in the RDF database.

Any Linked Data browser can be used to access the Linked Data view of a repository. Figure 1 shows the presentation of RDF metadata in the Tabularor Firefox extension [14].

The system is implemented in Java and is based on the following frameworks and libraries: Restlet Framework [15], Redline RPM [16], Jena Semantic Web Framework [17], Joseki SPARQL Server for Jena [18].


Fájl Szerkesztés Nézet Előzmények Könyvjelzők Eszközök Súgó
http://localhost:8182/repo/packages/tm-0.5-3
Google
Most Visited Release Notes Fedora Project Red Hat Free Content
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
http://localhost:8182/repo/pa...
RDF XML

▾http://localhost:8182/repo/packages/tm-0.5-3

http://localhost:8182/repo/packages/tm-0.5-3	author	Ingo Feinerer		
	date	2010-02-19		
	long description	A framework for text mining applications within R.		
	maintainer		type	Person
			personal mailbox	mailto:feinerer@logic.at
			name	Ingo Feinerer
	name	tm		
	publication	2010-02-19T10:49:24		
	short description	Text Mining Package		
	suggests	http://localhost:8182/repo/packages/RWeka		
		http://localhost:8182/repo/packages/Rgraphviz		
		http://localhost:8182/repo/packages/Rmpi		
		http://localhost:8182/repo/packages/Snowball		
		http://localhost:8182/repo/packages/XML		
		http://localhost:8182/repo/packages/filehash		
		http://localhost:8182/repo/packages/proxy		
		http://localhost:8182/repo/packages/snow		
	url	http://tm.r-forge.r-project.org/		
	version	0.5-3		

Figure 1: Browsing a CRAN repository with the Tabulator Firefox extension.

4. Web ontology for software package management

The tools that are discussed in Section 4 use specialized metadata vocabularies to represent information about packages in RDF. As a result of this work several OWL web ontologies are available to define these vocabularies for the following package formats: .deb, RPM, and the package format used by the R statistical computing environment.

The package format specific RDF vocabularies served as a basis to construct a unified OWL web ontology, that is flexible enough to support all three formats. File format specific ontologies and also the unified ontology are in OWL 1 DL.

It was a challenging task to develop an appropriate ontology-based model that can be used in conjunction with existing package management systems. Challenges to be addressed include:

Challenge 4.1. Many different versions of a package may be available at the same time. Packages that have the same name but different version numbers contain different versions of the same abstract entity, for example, an application. The model should provide a means to express that different packages are versions of the same abstract entity. Package management systems do not provide any information about these abstract entities, as metadata is always provided in an individual package and applies to that specific version.

Challenge 4.2. Packages may depend not only on packages, but also on features that are provided by other packages. Such a feature may represent a file or may stand for an abstract capability. A package may provide several different features, and many different packages may require the same feature similarly. For example, RPM uses this dependency model. In RPM terminology features are called capabilities.

Challenge 4.3. Several types of relationships may exist between packages. For example, the R statistical computing environment handles the following relationships: Depends, Imports, Suggests, Enhances. There is also a SystemDependency metadata element to specify dependencies that are external to the R system.

Challenge 4.4. Version numbers may be used in dependencies. A package may depend on a specific version of another package, for example, may require version 1.3. It may omit version number in the dependency, meaning that any version is appropriate. Finally, a minimal or maximal required version may also be specified, for example, the package may require version 1.3 or later.

Our OWL ontologies provide appropriate solutions to the above problems. Moreover, the unified ontology supports all three formats. The main classes of the unified ontology are the following:

Feature Represents features that may be provided or required by packages.

PackageFeature A subclass of Feature whose instances represent abstract entities that are embodied in the packages (see Challenge 4.1).

File A subclass of Feature that represents files that may be provided or required by packages.

VirtualFeature A subclass of Feature that represents features intended to be provided by many different packages. The name of such a feature can be any arbitrary string (e.g. `webserver`).

SystemFeature A subclass of Feature that represents features that are external to the package management system. Packages may require, but never provide SystemFeatures.

Package Represents installable packages.

Questions and problems to be answered in package management may include the following:

- Can a repository satisfy the dependency requirements of a package?
- Determine the packages that must be installed in order to install a given package!
- What are those external dependencies that are required by packages in the repository but are not provided?

Unfortunately, the model has limited query capabilities. Some useful information may be extracted and inferred from the model, but the above problems are beyond the limitations of OWL 1 and SPARQL. Specialized tools and algorithms may be required, RDF may be useful as a representation only.

Some file format specific features are also not supported. For example, Debian packages may specify alternative dependencies, that are currently not available within our framework. That means that a package may require package A or B, moreover, restrictions on their versions may also present in the dependency. Representing alternatives in a semantically meaningful way is definitely a problem. Although RDF provides the rdf:Alt container to represent alternatives, neither RDF nor OWL provide formal semantics.

5. Further work

The author is currently working on porting the ontologies to OWL 2. The new features of OWL 2 [19], such as property chains and the extended datatype capabilities may be useful to build a more powerful model. Forthcoming versions of the framework will also provide support for other package formats. The work presented in the paper is intended to serve as a basis for a general dependency analyzer framework.

6. Conclusions

This paper presents an unique framework that makes package metadata available to Semantic Web applications. The framework supports RDF metadata extraction from individual package files, and can process entire repositories, too. Currently, only a few package formats and repositories are supported (namely the RPM format and Yum repositories, the package format of the R statistical computing environment and CRAN-style repositories), but can be extended to work with other package management systems. The main contribution of the work is that the presented framework can turn package repositories into valuable Linked Data sets.

References

[1] MURDOCK, I., How package management changed everything (2007). http://ianmurdock.com/solaris/how-package-management-changed-everything/

[2] The R Project for Statistical Computing http://www.r-project.org/

[3] Apache Maven http://maven.apache.org/

[4] BERNERS-LEE, T., Linked Data (2006). http://www.w3.org/DesignIssues/LinkedData.html

[5] Linked Data – Connect Distributed Data across the Web http://linkeddata.org/

[6] Linking Open Data on the Semantic Web http://esw.w3.org/TaskForces/CommunityProjects/LinkingOpenData/DataSets

[7] DBpedia http://dbpedia.org/

[8] THE DEBIAN POLICY MAILING LIST, Debian Policy Manual (2010). http://www.debian.org/doc/debian-policy/policy.pdf.gz

[9] R DEVELOPMENT CORE TEAM, Writing R Extensions (2010). http://cran.r-project.org/doc/manuals/R-exts.pdf

[10] FOSTER-JOHNSON, E., RPM Guide (2005). http://docs.fedoraproject.org/drafts/rpm-guide-en/

[11] FIELDING, R. T., TAYLOR, R. N., Principled Design of the Modern Web Architecture, *ACM Transactions on Internet Technology* Vol. 2, No. 2 (2002), 115–150.

[12] Yum Package Manager http://yum.baseurl.org/

[13] R DEVELOPMENT CORE TEAM, R Installation and Administration (2010). http://cran.r-project.org/doc/manuals/R-admin.pdf

[14] The Tabulator Extension http://dig.csail.mit.edu/2007/tab/

[15] Restlet – RESTful Web framework for Java http://www.restlet.org/

[16] Redline Java RPM library http://code.google.com/p/redline-rpm/

[17] Jena Semantic Web Framework http://jena.sourceforge.net/

[18] Joseki – A SPARQL Server for Jena http://joseki.sourceforge.net/

[19] GOLBREICH, C., WALLACE, E. K., EDS. OWL 2 Web Ontology Language: New Features and Rationale (2009). http://www.w3.org/TR/owl2-new-features/

Péter Jeszenszky
4032 Debrecen, Egyetem tér 1., Hungary
e-mail: jeszenszky.peter@inf.unideb.hu