SCIENCE AND TECHNOLOGY FACILITIES COUNCIL

Tools and Guidelines for Preserving and Accessing Software as a Research Output

---

Report III: Tools

A Shaon, JC Bicarregui

4/1/2009

Science & Technology Facilities Council e-Science

Contents

Science & Technology Facilities Council
e-Science

1 Introduction and Rationale

The experience of applying the framework for significant properties of software (Report I) to the BADC WFS/GeoServer (Report II, Section 4) has shown that it is currently necessary to have considerable knowledge of both the framework and software in question to accurately map the software to the framework. This indicates a need for tooling to facilitate the creation of significant properties by providing guidelines which, for example, explain the underlying concepts of the framework in a user-friendly manner. The **Significant Properties Editing and Querying for Software** (SPEQS) is a "proof-of-concept" demonstration of such a tool.

SPEQS enables recording, editing and querying significant properties of software projects directly from within the Eclipse environment. The rationale is to demonstrate how capturing preservation information can be incorporated within the software development lifecycle to aid its long-term preservation in future. Therefore, SPEQS is intended to provide software developers with an easy-to-access interface and guidelines within their development environment for efficiently recording significant properties of software during its development. This approach of enabling the developer(s) of a software project to record its significant properties is envisaged to contribute towards ensuring the accuracy of the information recorded. From a wider perspective, this approach should also help increase awareness of the importance of significant properties of software for its long-term preservation across various software developer communities that use Interactive Development Environments (IDE)s, such as Eclipse.

SPEQS has been developed as a plug-in for Eclipse – a widely used Open Source interactive software development environment.

2 Architectural Overview

The underlying architecture of SPEQS has been designed specifically to facilitate recording and querying information based on the framework for significant properties of software (Report I). SPEQS uses an ontology representation of the framework, written in OWL (Web Ontology Language)[1] for recording significant properties (SPs) in RDF[2] format and querying the recorded SPs using SPARQL[3], the query language for RDF. The SPEQS architecture consists of four principal components: ***the SP Editor, the SP Query Interface, the SPEQS Data Store*** and ***a software repository***, such as Subversion[4] and SourceForge[5] (Figure 1).

[1] http://www.w3.org/TR/owl-features/
[2] http://www.w3.org/RDF/
[3] http://www.w3.org/TR/rdf-sparql-query/
[4] http://subversion.tigris.org/
[5] http://sourceforge.net/

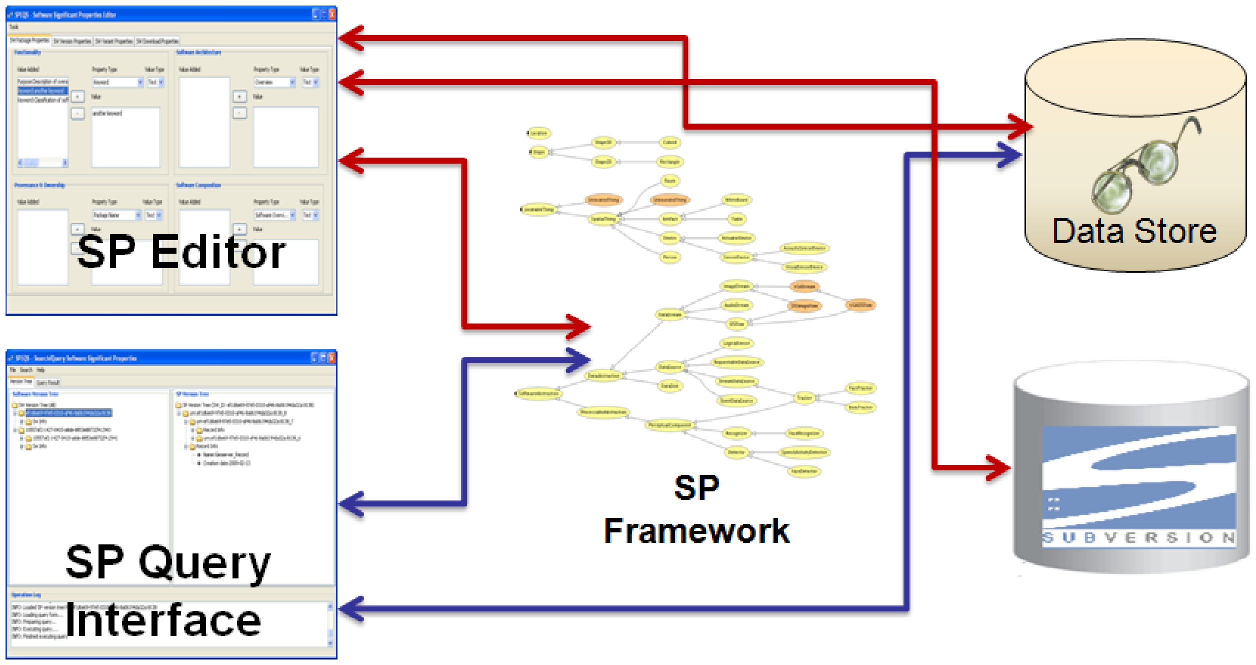


Figure 1: An Architectural View of SPEQS

The SP Editor and the SP Query Interface are Graphical User Interfaces (GUIs) for enabling recording and updating of SPs and querying the recorded SPs respectively. The SPEQS Data Store is a relational database that consists of a RDF Triple Store[6] for storing SPs and a standard data storage section for storing other meta-information (e.g. developer name, creation date etc.) associated with the software. SPEQS currently supports MySQL[7] and PostgreSQL[8] as the SPEQS Data Store.

SPEQS interacts with software repositories and management systems, such as Subversion, for keeping track of changes made to software and ensuring accurate and consistent association with its significant properties. It is envisaged that integration of SPEQS with commonly used software repositories and IDEs (e.g. Eclipse) would promote and facilitate recording of SPs of software by its developer(s) during its development. At present, SPEQS only supports Subversion based software.

3 A Walk Through

The following subsections provide an overview of the key features and functional entities of SPEQS presented in Figure 2.

[6] Databases specially configured to store and enable querying large RDF models.
[7] http://www.mysql.com/
[8] http://www.postgresql.org/

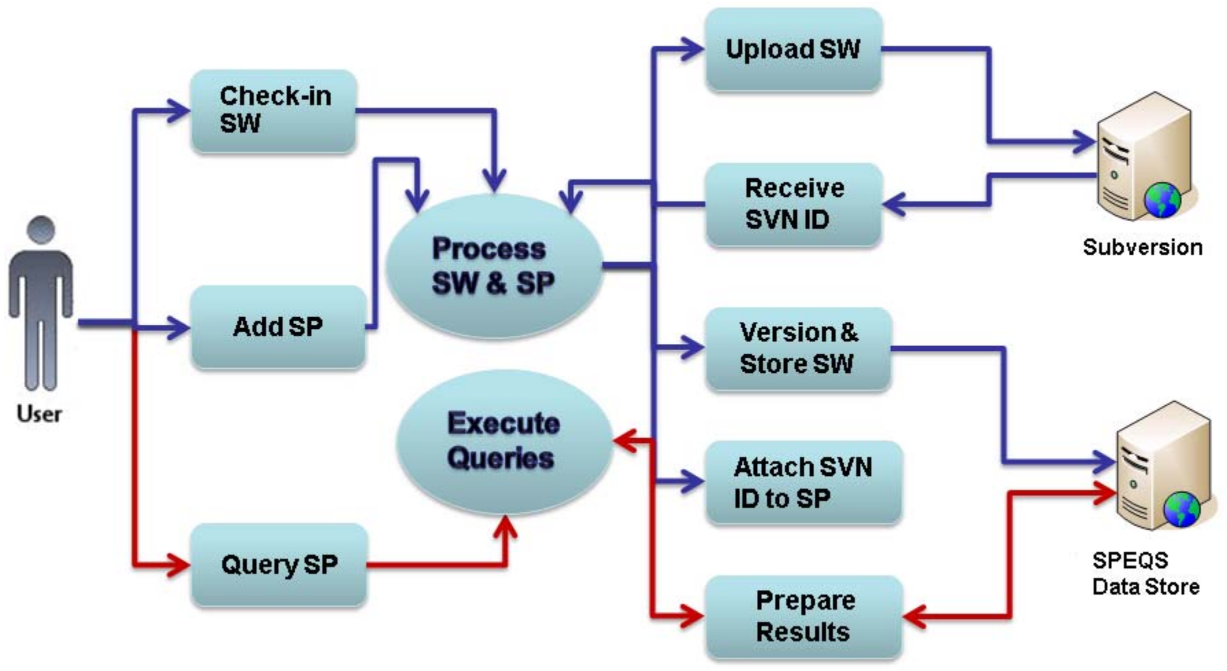
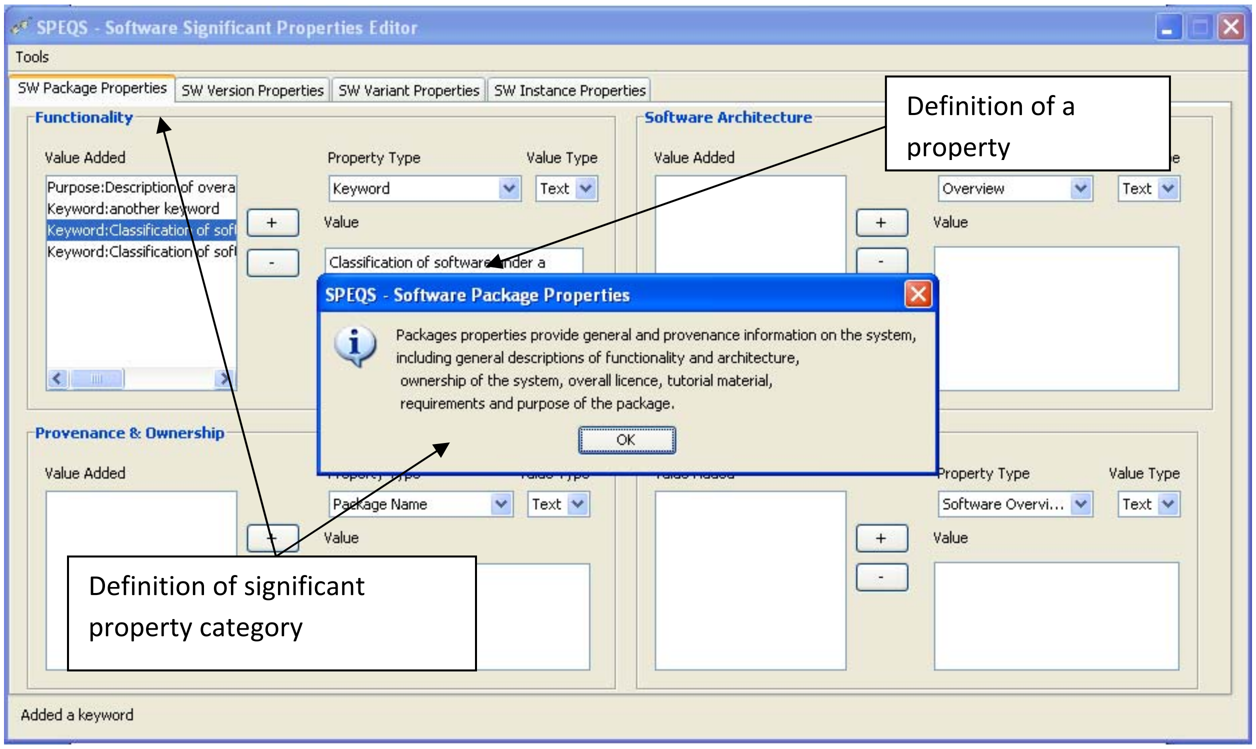


Figure 2: Functional Entities of SPEQS

3.1 Adding Significant Properties to Software

Let us consider a scenario where the developer of a Subversion-based software project, say, “svn-test” is using SPEQS to record significant properties of the software from within his Eclipse environment. To do this, the developer selects the project in the Eclipse project explorer and subsequently invokes the SP Editor (Figure 3), a graphical interface specifically designed for this purpose, from the SPEQS menu of Eclipse. Using the SP Editor, he asserts values to different significant properties of “svn-test” as defined in the framework for significant properties of software (Report I). For example, to assert a value to the “purpose” property of the “svn-test” software, the developer selects the property from a drop down menu, followed by the type of value to be asserted (i.e. text or URI), enters the value in a text box and finally clicks on the “add” button of the SP Editor. It should be noted that SPEQS would automatically determine and assert values to some of the significant properties of “svn-test”, such as Programming Language, Operating System and Subversion Source URL. However, the developer could change or update these automatically generated values as required.



Purpose:Description of overa
Keyword:another keyword
Keyword:Classification of soft
Keyword:Classification of soft

Classification of software under a

SPEQS - Software Package Properties

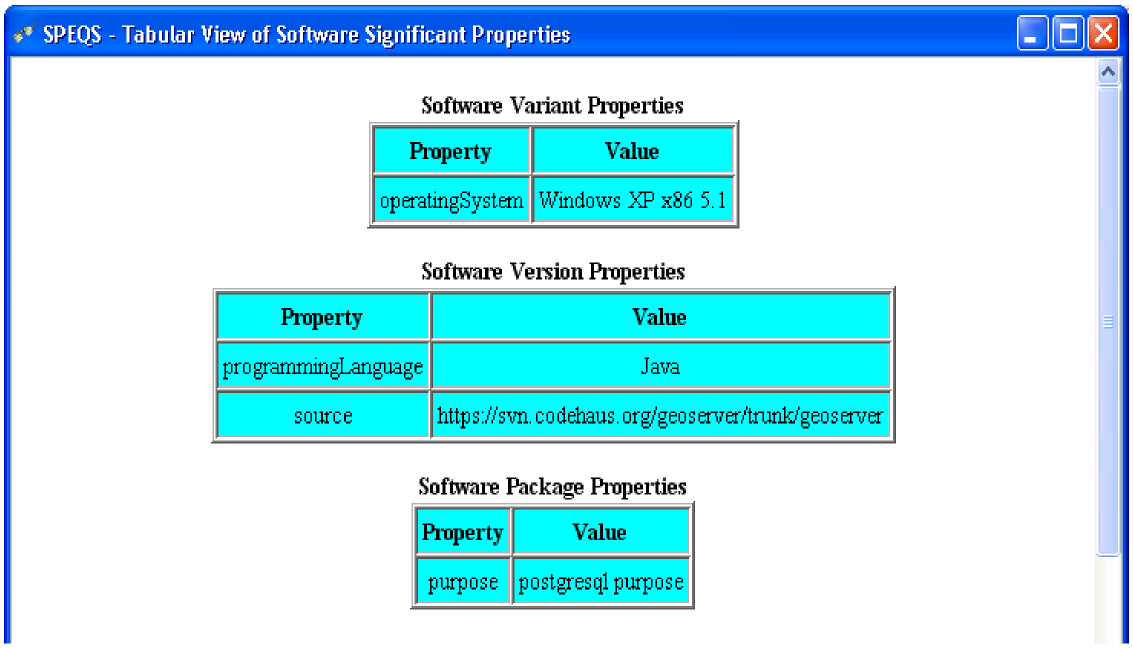
Packages properties provide general and provenance information on the system, including general descriptions of functionality and architecture, ownership of the system, overall licence, tutorial material, requirements and purpose of the package.

Added a keyword

Figure 3: The Significant Properties Editor of SPEQS

When adding significant properties to "svn-test", the developer may query the intended meaning of a particular significant property directly from the SP Editor by selecting it from the corresponding drop down menu. The SP editor then displays a brief text explaining the intended connotation of the property in the text field below the drop down menu (Figure 3). In addition, the developer could also view a general definition of a category of significant property by right clicking on the corresponding Tab in the SP editor and selecting the **?** option (Figure 3).

Furthermore, the developer may also visualise all asserted properties during the SP adding operation by selecting a SP visualisation option from the SP editor. The current version of the SP Editor enables three ways to visualise a SP record: as a Table (Figure 4), as a Graph and as RAW RDF source code.


SPEQS - Tabular View of Software Significant Properties

Software Variant Properties

Property	Value
operatingSystem	Windows XP x86 5.1

Software Version Properties

Property	Value
programmingLanguage	Java
source	https://svn.codehaus.org/geoserver/trunk/geoserver

Software Package Properties

Property	Value
purpose	postgresql purpose

Figure 4: Tabular View of a SP Record

After asserting values to different significant properties of "svn-test", the developer has the option to either store the SP record in the SPEQS data store or save it in a file as a draft for adding more properties or making changes later. In this particular scenario, the developer decides to store the SP record that he has created, in the SPEQS data store and invokes a very simple "Commit" interface from the SP editor. Using this interface, the developer associates additional metadata, such as his/her name and name of the record, to the SP record. After this, he commits the SP record to the SPEQS Data Store. SPEQS will then create a record for "svn-test" in the SPEQS Data Store and store the new SP record as the current version of SP record associated with "svn-test". SPEQS will use a combination of the subversion Identifier and revision number of "svn-test" to identify it within the SPEQS Data Store. As for the SP Record that the developer has associated with "svn-test", SPEQS will assign an identifier based on the SPEQS id of "svn-test". For example, if the SPEQS id for "svn-test" is "*12erer-335fds-fggeer3:8787*", then the SPEQS identifier for its SP record will be "*urn:12erer-335fds-fggeer3:8787_1*"

Now, if the developer had made any changes to the "svn-test" project prior to adding significant properties to it, he would also be able to commit those changes at the time of committing the added SP record to the SPEQS Data Store. In such a case, SPEQS would attempt to commit "svn-test" to Subversion first before tying to store it and its corresponding SP record in the SPEQS data store.

3.2 Updating Significant Properties of Software

In the above scenario, having stored the SP record in the SPEQS data store, the developer decides to make some changes to the record. Therefore, he re-invokes the SP Editor from the SPEQS menu of his Eclipse with the "svn-test" selected in the Eclipse project explorer. This time, SPEQS displays the SP editor automatically populated with all asserted properties in the existing SP record. This enables the developer to update and/or remove any of the existing properties as well as adding new ones. The developer makes some changes to the record and subsequently commits the updated record to the SPEQS Data Store. In this case, SPEQS marks the updated record as the current version of SP record associated with the "svn-test" project and appropriately links it with the previous version.

3.3 Exploring Software and Significant Properties Version Tree

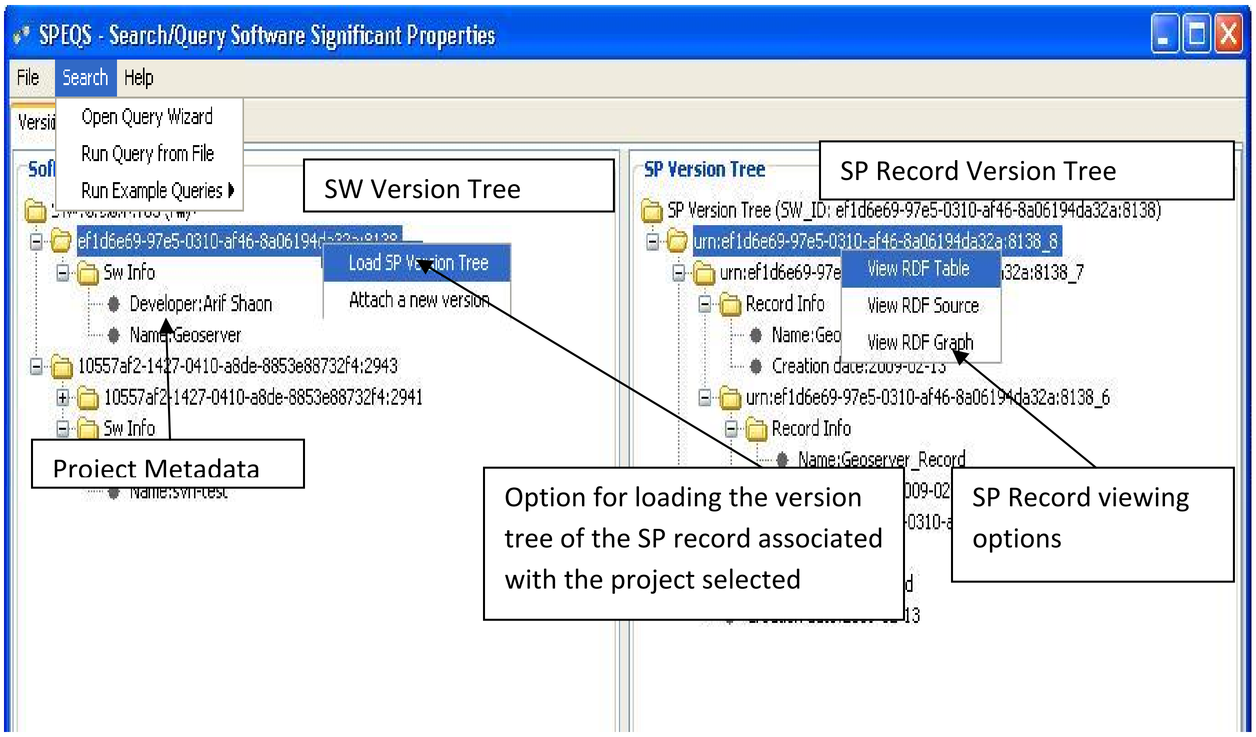
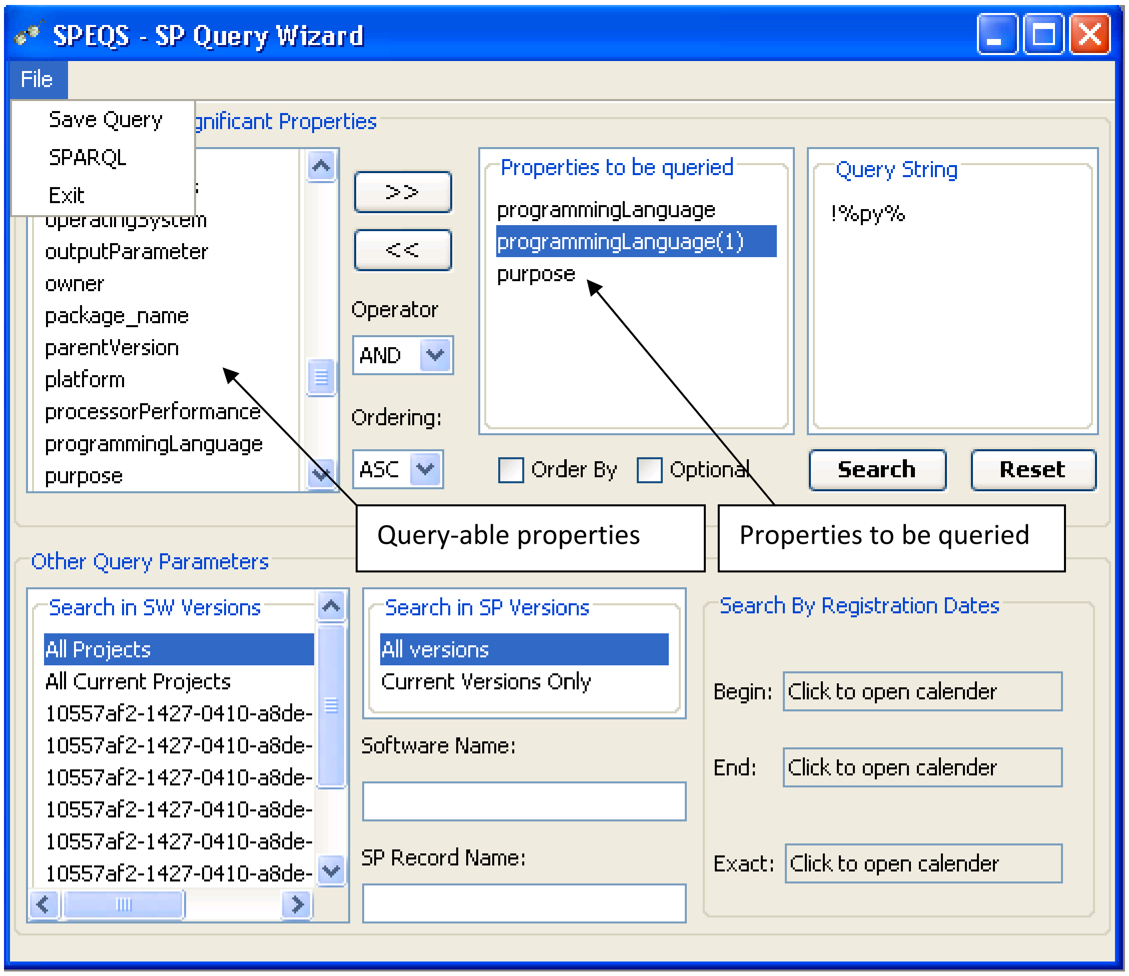

SPEQS - Search/Query Software Significant Properties
File
Search
Help
Open Query Wizard
Run Query from File
Run Example Queries
SW Version Tree
SP Version Tree
SP Record Version Tree
SP Version Tree (SW_ID: ef1d6e69-97e5-0310-af46-8a06194da32a:8138)
urn:ef1d6e69-97e5-0310-af46-8a06194da32a:8138_8
Load SP Version Tree
Attach a new version
View RDF Table
View RDF Source
View RDF Graph
Sw Info
Developer:Arif Shaon
Name:Geoserver
10557af2-1427-0410-a8de-8853e88732f4:2943
10557af2-1427-0410-a8de-8853e88732f4:2941
Record Info
urn:ef1d6e69-97e5-0310-af46-8a06194da32a:8138_6
Name:Geoserver_Record
Project Metadata
Option for loading the version tree of the SP record associated with the project selected
SP Record viewing options

Figure 5: SPEQS Version Tree for Software and corresponding SP Records

After updating the SP record associated with "svn-test" and committing it to the SPEQS Data store, the developer may explore the version trees of both "svn-test" and the corresponding SP record through an interactive graphical interface of SPEQS as illustrated in Figure 5. Using the version tree, the developer can also view a particular SP record in one of the three formats mentioned earlier.

3.4 Querying Significant Properties of Software



Properties to be queried: programmingLanguage, programmingLanguage(1), purpose

Query String: !%py%

Operator: AND

Ordering: ASC

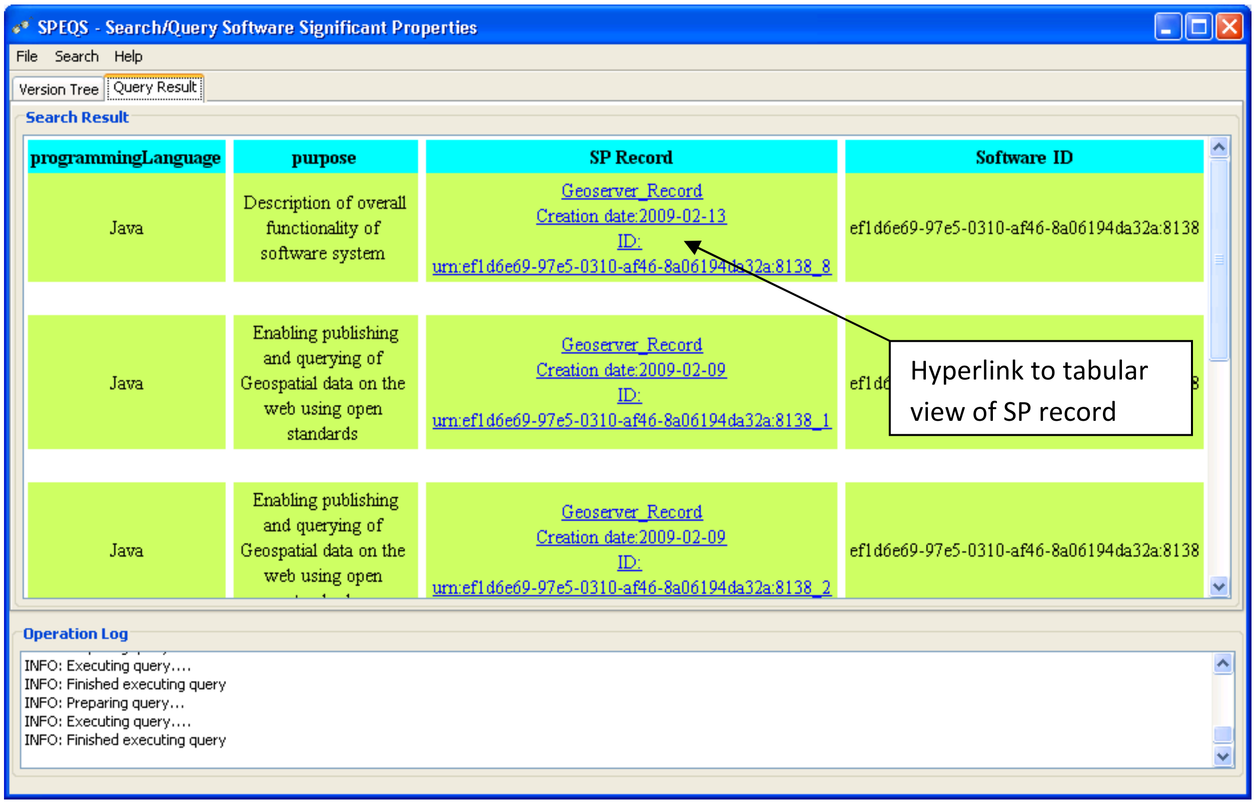
Begin: Click to open calender

End: Click to open calender

Exact: Click to open calender

Figure 6: SPEQS Query Wizard

The developer then decides to query the SP record he has just created and stored, using the SPEQS Query Wizard (Figure 6). This query wizard enables construction of both simple and complex queries against the SP records stored in the SPEQS Data Store. In short, using the query wizard, the developer would be able to select the properties to be queried as well as specifying conditions to be met for a particular query. The developer could also have his own SPARQL queries executed through SPEQS using the SPARQL editor of the query wizard (Figure 6).


SPEQS - Search/Query Software Significant Properties
File Search Help
Version Tree
Query Result
Search Result
Hyperlink to tabular view of SP record
Operation Log

programmingLanguage	purpose	SP Record	Software ID
Java	Description of overall functionality of software system	Geoserver_Record Creation date:2009-02-13 ID: urn:ef1d6e69-97e5-0310-af46-8a06194da32a:8138_8	ef1d6e69-97e5-0310-af46-8a06194da32a:8138
Java	Enabling publishing and querying of Geospatial data on the web using open standards	Geoserver_Record Creation date:2009-02-09 ID: urn:ef1d6e69-97e5-0310-af46-8a06194da32a:8138_1	ef1d6 8
Java	Enabling publishing and querying of Geospatial data on the web using open	Geoserver_Record Creation date:2009-02-09 ID: urn:ef1d6e69-97e5-0310-af46-8a06194da32a:8138_2	ef1d6e69-97e5-0310-af46-8a06194da32a:8138

```
INFO: Executing query....
INFO: Finished executing query
INFO: Preparing query...
INFO: Executing query....
INFO: Finished executing query
```

Figure 7: Query Results Representation in SPEQS

Having constructed his queries using the wizard, the developer submits them for execution. SPEQS executes the queries and display the results in a tabular format as shown in Figure 7. The table containing the results of the developer's query clearly indicates in two separate columns, the SP record and Software version to which a particular row of the table corresponds. The SP record names displayed in a search result table are in fact hyperlinks that would enable the developer to view the corresponding SP record in a tabular format (Figure 4).

4 Conclusions

SPEQS demonstrates the underlying concept of a tool that would enable recording and querying significant properties of software from within its development environment. Additionally, it aims to promote awareness of the role of significant properties in long-term preservation of software among software developers. The current version of SPEQS provides user-friendly means of recording, editing and querying significant properties of Subversion based software projects directly from within the Eclipse environment. It also provides users with suitable guidelines for accurately recording significant properties of software. Thus, SPEQS demonstrates the feasibility of providing effective guidance through suitable tooling for accurately annotating software with its significant properties. More importantly, it shows how capturing preservation information can be incorporated within the software development lifecycle to aid its long-term preservation.

However, there is still considerable scope for further improvement in SPEQS. In particular, SPEQS needs to incorporate an efficient mechanism for semantically validating values asserted in a SP record. This could involve integrating SPEQS with a suitable controlled vocabulary. Furthermore, the SPEQS Data Store should be subjected to effective long-term preservation technique, e.g. by integrating it with an efficient long-term preservation archive, to ensure longevity of the SP records. Additionally, to cater for a wider range of software projects, SPEQS would benefit from incorporating support for other widely used software development environments, such as NetBeans[9] and other software repositories, such as SourceForge and CVS[10].

Despite these shortcomings, we believe that SPEQS is already a significant step towards a comprehensive software system that would facilitate capturing, validating, querying and preserving significant properties of software.

[9] http://www.netbeans.org/

[10] http://ximbiot.com/cvs/wiki/Main%20Page