Knowledge Discovery Using a Service Oriented Web Application

Janez Kranjc, Vid Podpečan, Nada Lavrač
Department of Knowledge Technologies
Jožef Stefan Institute
Ljubljana, Slovenia
{*janez.kranjc,vid.podpecan,nada.lavrac*}*@ijs.si*

***Abstract*—The paper proposes a novel platform for knowledge discovery, which is based on modern web technologies, and is implemented as a web application. It is based on the principles of service-oriented knowledge discovery, and features interactive scientific workflows. In contrast to the few existing comparable platforms, our environment is suitable for any knowledge discovery task, offers advanced workflow construction including meta-workflows, can use any existing web service as a workflow processing component, and runs in all major web browsers and platforms, including mobile devices. The presented environment has been demonstrated on two use cases: a simple motivating use case built using Weka web services, and an advanced use case featuring a complex text mining scenario.**

***Keywords*-data mining; knowledge discovery; web application; web services**

I. Introduction

The development of modern knowledge discovery and data mining environments and tools has reached maturity. While traditional data analysis software supported a single or few algorithms, designed to mine highly specialized data, today's modern knowledge discovery systems provide a large collection of generic algorithm implementations, usually coupled with an easy-to-use graphical user interface. The importance of visual programming using scientific workflows is now also widely recognized, and all advanced knowledge discovery software offer some form of workflow construction and execution, as this is of crucial importance for conducting complex scientific experiments, which need to be repeatable, and easy to verify at an abstract level.

However, these so-called second generation systems have failed to benefit from the concepts of service-oriented architecture, and complex and geographically dispersed information and knowledge sources as well as algorithms and functions, publicly available on the web. Finally, today's knowledge discovery systems have also failed to bridge different operating systems and platforms, and are not able to fully utilize available server resources in order to relieve the client from heavy-duty processing and data transfer. As the general trend is shifting towards mobile devices and mobile computing, this effectively prevents the employment of such tools in modern mobile information environments.

The novel knowledge discovery platform presented in this paper was designed to overcome all the recognized deficiencies while retaining all important features of existing solutions. As such, our platform benefits from service-oriented technologies [1], the visual programming paradigm, as well as platform *and* software independent technologies.

Firstly, service-oriented architecture featuring web services as primal processing components enables parallelization, remote execution, and high availability by default. It provides access to large public (and proprietary) databases, enables easy integration of third party components (as services) and loose coupling, and supports not only distributed processing but also distributed development.

Secondly, the visual programming paradigm simplifies the construction of complex knowledge discovery scenarios by providing basic building blocks, which can be connected and executed, enables repeatability of experiments by saving constructed workflows and parameters, provides an intuitive structuring of complex parts of workflows by introducing the notion of meta-workflow (workflow of workflows), and makes the platform suitable also for non-experts due to the representation of complex procedures as sequences of simple processing steps.

Finally, as the platform and software independence can be achieved by using web technologies only, the platform relies on standards such as HTML, CSS, Ajax and JavaScript, and widely supported and accepted software solutions such as Apache and PHP.

To summarize, the presented platform offers a complete service-oriented workflow environment, suitable for any knowledge discovery task. The platform is truly independent as it is implemented in the form of a web application, which is accessible from any modern web browser.

The rest of the paper is structured as follows. Section II briefly presents the related work. In Section III, the design of the platform and its components are discussed in detail. The description of the initial widgets repository is presented in Section IV. In Section V, two data mining use cases are presented. Finally, Section VI summarizes the work and concludes the paper by suggesting directions of further work.

II. Related work

This section discusses the work related to the main concepts of the presented platform: workflow editing and execution environments, service-oriented approaches to knowl-

 ISBN: 978-1-61208-181-6

edge discovery and browser-based applications for knowledge discovery.

Many software solutions from different domains enable construction and execution of scientific workflows. Well known examples include Weka [2], Orange [3], KNIME [4] and RapidMiner [5] data mining environments. The most important common feature is the implementation of a *workflow canvas* where complex workflows can be constructed using simple drag, drop and connect operations on the available components. The range of available components typically includes data loading and pre-processing, data and pattern mining algorithms and interactive and non-interactive visualizations.

Even though such data mining software solutions are reasonably user-friendly and offer a wide range of components, there are many deficiencies, which limit their use. Firstly, all available workflow components provided by any of these platforms are specific and can be used in this particular platform only. Secondly, the described platforms are implemented as standalone applications and have specific hardware and software dependencies. Thirdly, in order to extend the range of available workflow components in any of these platforms, knowledge of a specific programming language is required. This also means that they are not capable of using existing software components, implemented as web services, and available freely on the internet.

In order to benefit from service-oriented architecture concepts, another group of software tools have emerged, which are able to make use of web services, and can access large public databases (some also support means of grid deployment and P2P computing). Environments such as Weka4WS [6], Orange4WS [7], Web Extension for RapidMiner, Triana [8], Taverna [9] and Kepler [10] allow for integration of web services as workflow components. However, with the exception of Orange4WS and Web Extension for RapidMiner, these environments are mostly specialized in domains like systems biology, chemistry, medical imaging, ecology and geology. Lastly, none of these applications is browser based thus still requiring specific hardware and software.

The last group of software tools capable of workflow construction, most similar to the presented environment, encompasses browser based applications such as the Oryx Editor [11] for modelling workflows and business processes, and the Galaxy [12] genome analysis tool. The Oryx editor, however, although designed similarly as the proposed environment, does not support the deployment of workflows as it is only a modelling tool. Also, the Galaxy web application is limited exclusively to the workflow components, provided by the project itself, and does not provide means for employing arbitrary web services and other information and computing resources found on the web.

III. Platform design

The presented platform consists of three layers as shown in Figure 1. The upper-most layer presents the parts of the platform which run on the client side. The middle layer is located on the server where the platform is hosted. The bottom layer consists of the remote resources which provide web services.

This section describes these layers in detail. The user interface is presented first. Secondly, the workflow engine is described. Finally, workflow components and the functionalities to import web services are explained.

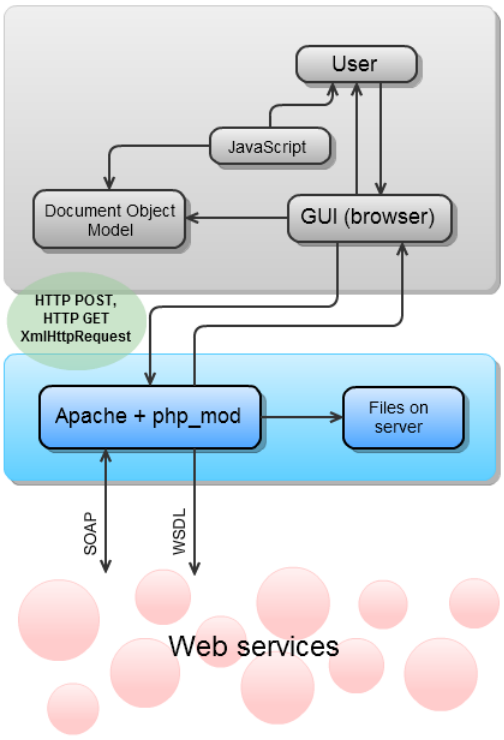

Figure 1. The three layered design of the platform. The upper-most layer represents the part of the platform executed on the client side. The middle layer is located on the server where the platform is hosted. The lower layer represents remote resources which provide web services.

A. The user interface

The graphical user interface was implemented in HTML. It consists of three main parts: the toolbar, the widget repository, and the canvas. A sample screenshot of the user interface is shown in Figure 2.

Two JavaScript libraries were used to implement the toolbar. The buttons were implemented using the jQuery UI library while their event listeners and handlers were implemented using the jQuery library [13]. The primary function of the toolbar is to start, execute, save, and load workflows, and to separate parts of the workflow.

 ISBN: 978-1-61208-181-6

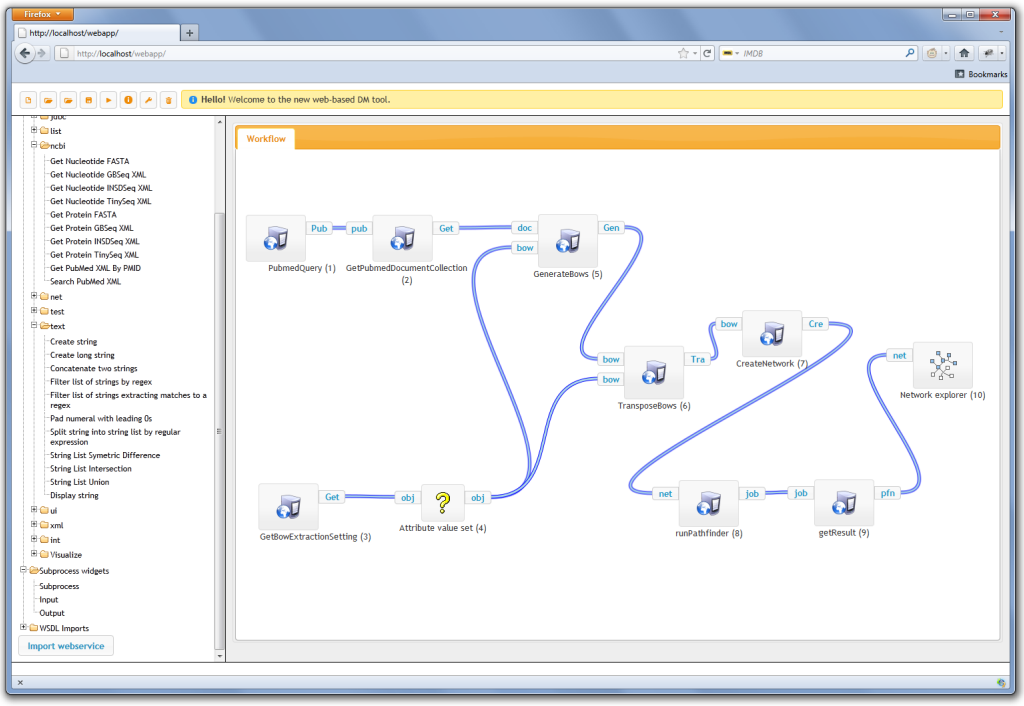

Figure 2. A screenshot of the environment in the Mozilla Firefox browser.

The second part of the graphical user interface is the widget repository, which provides a clickable list of available widgets. By clicking on an available widget, its instance appears on the canvas hosting the currently active workflow.

As the workflow canvas is the part which requires intensive user interaction, it was implemented in JavaScript. Each widget on the canvas is represented by a short HTML description, and a special function in JavaScript, which appends this HTML code to the document object model, is invoked whenever a widget from the repository list is clicked.

Widgets can be connected by clicking on an input or output. When both an input and an output are selected, an event is triggered, which checks for cycles in the workflow graph using the depth first search algorithm. If no cycles are detected, a connection is drawn and the corresponding widgets become connected.

The connections are graphically represented as Bézier curves, and implemented by dynamically adding HTML5 canvas elements to the document object model. To enable cross-browser functionality, the ExplorerCanvas library was used to simulate the canvas element in the Internet Explorer browser family. Finally, the selection and removal of connections are implemented using mouse and keyboard event handlers, respectively.

B. The workflow engine

The presented platform is able to execute workflows as well as separate widgets. A PHP script on the server corresponds to each widget. It is invoked from the graphical user interface using Ajax techniques.

The inputs of the widget are passed to the PHP script using an asynchronous HTTP POST request. When the results are available (or when an error occurs), a call-back function is called, which stores the results of the execution of the widget into the output variables in the underlying document object model. The PHP script may either return the data in a serialized form or issue a special command, which instructs the user interface to open a pop-up window for displaying the results (data visualization widgets utilize this functionality). The execution of multiple independent widgets simultaneously is assured by the asynchronous nature of POST requests, which essentially represent an equivalent to the multithreading programming paradigm.

The execution of the entire workflow is realized by a special JavaScript function, which iteratively searches for widgets whose predecessors have finished their execution, and executes them.

C. Web services as workflow components

Workflow components of the presented platform may be implemented as remote web services provided by a third party, or as PHP scripts located on the server hosting the platform.

 ISBN: 978-1-61208-181-6

Since web services are completely defined by their WSDL descriptions, the functionality to import web services was implemented in PHP by parsing the corresponding WSDL document. For every operation provided by a web service the PHP script returns an HTML description of the corresponding widget. In the user interface, this procedure is accessible through a button whose event handler queries the user for the location of a WSDL file, which is then imported and parsed, and a list of available widgets is presented to the user.

The user is allowed to decide about the role of each input of each operation. The input may be designated as a user interface input of the widget or as a widget input parameter. User interface inputs can be set by entering the values manually in the widget's graphical user interface while widget input parameters receive data from other widgets.

IV. The widget repository

In order to enable construction of scientific workflows implementing arbitrary knowledge discovery scenarios, an initial set of widgets is available to the user. The widgets belong to four distinct groups.

The first group of widgets enable data creation, manipulation and simple visualizations including creation of strings and integers, and joining strings and integers into arrays. Arithmetic integer operations are also implemented. These widgets are implemented as PHP scripts located on the server where the platform is running.

The second group of widgets consists of three widgets, which allow creating nested workflows (note that the depth is unlimited). A sub-workflow can be created by adding an instance of the sub-workflow widget on the canvas. On activation, the canvas view is switched to the sub-workflow (switching can also be achieved by clicking on the corresponding canvas tab). Additionally, two special widgets are used for assigning inputs and outputs of the current sub-workflow to carry the data from the parent workflow to the sub-workflow. By adding an input widget to the current sub-workflow, the corresponding sub-workflow widget gains an input, which is connected to the output of the input widget. The output widget operates similarly. This group of widgets is implemented in JavaScript and executed on the client side.

The third group of widgets consists of 35 implementations of the local services available in Taverna [9]. Because Taverna is written in Java and its local services in the Beanshell scripting language, they were implemented in PHP and integrated into the platform. These services include widgets that allow reading and writing files, re-encoding strings, executing SQL queries, querying public databases such as PubMed, accessing documents using HTTP, extracting and viewing images from websites, performing operations on strings and lists of strings, and manipulating XML files. Due to security issues in browsers, some services could not have been implemented in PHP, such as services that list files and folders of the user's computer and execute applications. Using the implemented collection of Taverna services our platform supports the majority of workflows created in Taverna.

Finally, the platform offers several data mining algorithm implementations from the Weka data mining environment, which have been made available as SOAP web services. The Orange4WS [7] data mining platform and its tools were used to implement these services, which enable easy and platform independent access to the latest Weka software. The actual SOAP web server hosting Weka services makes use of the JPype wrapper library [14], which allows calling arbitrary Java code from the Python interpreter. The services communicate by exchanging serialized Weka objects such as learners, classifiers, and datasets. While this approach does not allow client-side modification of these objects unless a Java client running Weka is used, it is currently the only feasible way to use Weka in a service-oriented environment as only a few Weka classes implement XML-based serialization according to PMML standard [15].

V. Use cases

This section demonstrates some of the abilities of the presented platform. Two use cases are presented and discussed. The first one is a simple, motivating use case where Weka web services are used to show the basic functionalities of the platform: composing a workflow of remote web services and local processing components, executing the workflow, and displaying the results. The second use case is an advanced example of a text mining workflow where a word graph obtained by querying a public database is pruned using a specialized graph algorithm (available as a web service) and visualized locally in a powerful interactive graph visualization component, provided by the platform.

A. A use case featuring Weka web services

The purpose of this use case is to demonstrate the use of Weka algorithms, available as web services (see Section IV). The J48 decision tree induction algorithm (J48) is cross-validated on the 1984 United States Congressional Voting Records dataset, which consists of 435 instances with 17 attributes and a binary class attribute, and is available in the Attribute-Relation File Format (ARFF).

First, the *Read Text File* widget is used to load the dataset into the platform. The widget provides a file chooser dialog where an arbitrary text file can be selected and uploaded to the server.

Since Weka web services communicate by exchanging serialized Weka objects, the dataset has to be transformed into a serialized Weka Instances object. A service from the repository of Weka services provides this functionality.

Then, to perform the cross-validation of a learning algorithm the Weka service for cross validation is used. It accepts serialized data, a serialized Weka learner (i.e., an

 ISBN: 978-1-61208-181-6

implementation of the learning algorithm), and the number of folds. The Weka service providing the J48 learner is connected to the cross-validation service, which can finally be executed. The cross-validation service provides several outputs, which return different reports generated by Weka. Here, Weka's cross validation summary is displayed using the *Display String* widget.

The complete workflow for performing cross validation using Weka services and displaying the results is shown in Figure 3. The workflow can be executed by clicking the execute workflow button. It invokes the workflow engine, which takes care of the validation, execution, and error reporting.

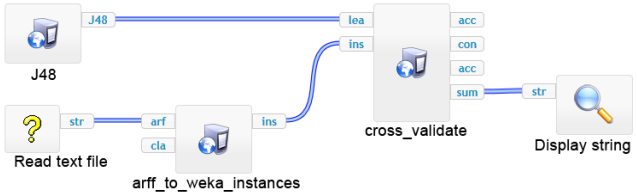

Figure 3. A workflow implementing cross validation of Weka algorithms.

B. An advanced text mining use case

This use case is built upon a collection of advanced services for text mining, graph analysis, and graph visualization. Its goal is to support the analysis of textual data by providing means for representing texts as graphs, graph pruning, and interactive graph visualization.

The information source for this use case is the well known PubMed database [16], a free database accessing the MEDLINE database of citations, abstracts and some full text articles on life sciences and biomedical topics. It was used for obtaining documents, relevant to the input query.

The resulting document corpus was then processed using text mining tools from the LATINO project [17], which is a software library implementing a range of data mining and machine learning algorithms with the emphasis on text mining and link analysis (components of the LATINO library components were provided as web services).

Using LATINO web services, the document corpus was transformed into a term network as follows. Firstly, it was tokenized and lemmatized, and transformed into the bag-of-words (BoW) vectors. Then, the network was produced using the following principle. Each link between two words represents a co-occurrence meaning that both words appear together in at least one document, whereas the link's weight represents the normalized number of co-occurrences through all documents. This weight is a similarity measure in the sense that two words (or concepts) linked with a higher weight are more similar (i.e. are more "connected" because they appear together more often) than two words with a lower weight.

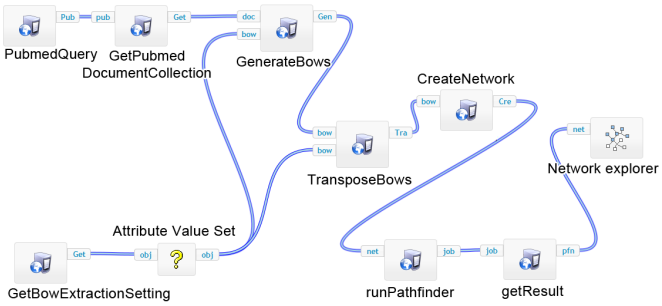

Figure 4. A workflow implementing the analysis of textual data using a public database and a collection of text mining and graph mining components.

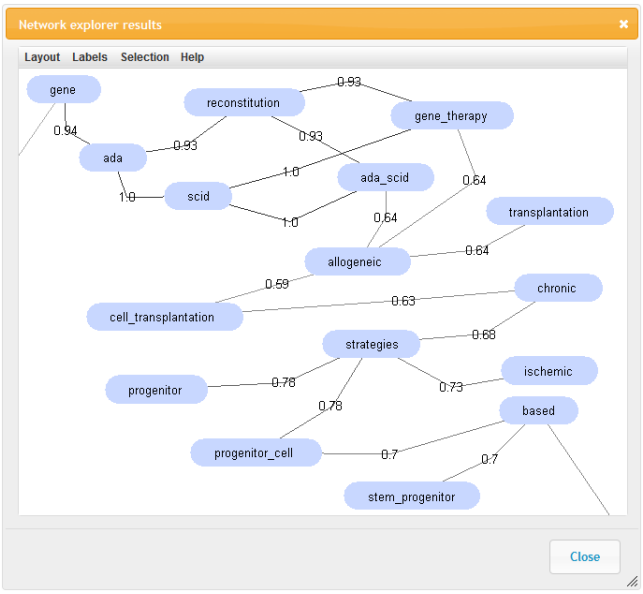

Figure 5. A part of the term graph obtained by querying PubMed with the query string *stem cell*. The graph is visualized using an interactive Java applet available as a widget in the presented platform.

Finally, the resulting weighted graph was pruned using the Pathfinder algorithm [18], a specialized algorithm for graph simplification (we omit the details of the algorithm as this is out of the scope of this paper). In order to transform the weights into dissimilarities required by Pathfinder the formula $w = 1/w'$ was applied where w is the original weight.

As a result, the pruned graph retained only the strongly linked concepts, which means that many less significant edges were removed thus improving its understandability and presentability. In the constructed workflow, as shown in Figure 4, an interactive graph visualization component was used, which enables user friendly exploration of large graphs. A zoomed part of the graph, obtained by querying PubMed with the query *"stem cell"*, is shown in Figure 5.

 ISBN: 978-1-61208-181-6

VI. Conclusion and futher work

The paper proposes a browser-based environment for service oriented knowledge discovery, which relies on modern web standards and widely supported and accepted software solutions. Coupled with the extreme versatility and power of web services, the proposed environment presents a new generation tool, ready to be used in any scenario or form of knowledge discovery, including mining of web and data streams thus surpassing all currently available knowledge discovery software tools. Moreover, the proposed environment is able to run in all modern web browsers, including those available on mobile devices, which presents great opportunities for its deployment and widespread use.

In summary, we have developed an open, general, and independent web application environment for knowledge discovery, which employs service-oriented technologies, and is ready to be used in any data and information analysis scenario.

In future, we plan to implement the process flow control widgets such as conditional branching and looping. We will also explore adding means of mining data streams as well as semi-automatic workflow construction based on planning algorithms, modern knowledge discovery ontologies, and systems for semantic annotation of web services. Finally, we plan to provide a public installation of the environment, a workflow repository, a community web site, and release the sources under an open-source license.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) within the context of the project FIRST, Large scale information extraction and integration infrastructure for supporting financial decision making, under grant agreement no. 257928.

References

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[2] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Amsterdam: Morgan Kaufmann, 2011.

[3] J. Demšar, B. Zupan, G. Leban, and T. Curk, "Orange: From experimental machine learning to interactive data mining," in *PKDD*, ser. Lecture Notes in Computer Science, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds., vol. 3202. Springer, 2004, pp. 537–539.

[4] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, "KNIME: The Konstanz Information Miner," in *GfKl*, ser. Studies in Classification, Data Analysis, and Knowledge Organization, C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, Eds. Springer, 2007, pp. 319–326.

[5] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "YALE: rapid prototyping for complex data mining tasks," in *KDD*, T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, Eds. ACM, 2006, pp. 935–940.

[6] D. Talia, P. Trunfio, and O. Verta, "Weka4WS: A WSRF-enabled Weka toolkit for distributed data mining on grids," in *PKDD*, ser. Lecture Notes in Computer Science, A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, Eds., vol. 3721. Springer, 2005, pp. 309–320.

[7] V. Podpečan, M. Zemenova, and N. Lavrač, "Orange4ws environment for service-oriented data mining," *The Computer Journal*, 2011.

[8] I. Taylor, M. Shields, I. Wang, and A. Harrison, "The Triana workflow environment: Architecture and applications," *Workflows for e-Science*, vol. 1, pp. 320–339, 2007.

[9] D. Hull, K. Wolstencroft, R. Stevens, C. A. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," *Nucleic Acids Research*, vol. 34, no. Web-Server-Issue, pp. 729–732, 2006.

[10] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, Jun. 2004, pp. 423–424.

[11] G. Decker, H. Overdick, and M. Weske, "Oryx – an open modeling platform for the bpm community," in *Business Process Management*, ser. Lecture Notes in Computer Science, M. Dumas, M. Reichert, and M.-C. Shan, Eds. Springer Berlin / Heidelberg, 2008, vol. 5240, pp. 382–385.

[12] D. Blankenberg, G. V. Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, and J. Taylor, "Galaxy: A Web-Based Genome Analysis Tool for Experimentalists," *Current protocols in molecular biology / edited by Frederick M. Ausubel ... [et al.]*, vol. Chapter 19, Jan. 2001.

[13] "jQuery: The write less, do more, javascript library," Last accessed January 2012. [Online]. Available: http://jquery.com

[14] "JPype - java to python integration," Last accessed January 2012. [Online]. Available: http://jpype.sourceforge.net/

[15] "PMML 4.0 - general structure of a pmml document," Last accessed January 2012. [Online]. Available: http://www.dmg.org/v4-0-1/GeneralStructure.html

[16] "Pubmed," Last accessed January 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/

[17] M. Grčar, "Latino - link analysis and text mining toolbox," Last accessed January 2012. [Online]. Available: http://sourceforge.net/projects/latino/

[18] A. Vavpetič, A. Batagelj, and V. Podpečan, "An implementation of the pathfinder algorithm for sparse networks and its application on text networks," in *Proceedings of the 11th International Multiconference Information Society*, 2009.

 ISBN: 978-1-61208-181-6