Demo: Visual Programming for the Semantic Desktop with *Konduit*

Knud Möller[1], Siegfried Handschuh[1], Sebastian Trüg[2], Laura Josan[1], and Stefan Decker[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
{knud.moeller, siegfried.handschuh, laura.dragan, stefan.decker}@deri.org
[2] Mandriva S.A., France
strueg@mandriva.com

Abstract. In this demo description, we present *Konduit*, a desktop-based platform for visual programming with RDF data. Based on the idea of the semantic desktop, non-technical users can create, manipulate and mash-up RDF data with Konduit, and thus generate simple applications or workflows, which are aimed to simplify their everyday work by automating repetitive tasks. The platform allows to combine data from both Web and desktop and integrate it with existing desktop functionality, thus bringing us closer to a convergence of Web and desktop.

1 Introduction

With the Semantic Web gaining momentum, more and more structured data becomes available online. The majority of applications that use this data today are concerned with aspects like search and browsing. However, a greater benefit of structured Web data is its potential for reuse: being able to integrate existing Web data in a workflow relieves users from the investment of creating this data themselves[3]. When it comes to working with data, users still rely on desktop-based applications (exceptions such as Google Docs only serve to support this rule), which are embedded in a familiar environment, using familiar UI metaphors. Web-based applications either simply don't exist, or have shortcomings in terms of usability. What's more, web-based applications can only access Web data, and do not integrate with data that a user might already have on their own desktop, let alone with other applications on the user's desktop. Even considering that it may be benefical for users to publish some desktop data on the Web, releasing all their data to the Web is not an option, since this may raise significant privacy issues. Instead, what is needed is a way of accessing structured Web data from the desktop, integrate it with existing desktop-data and applications and work with both in a unified way.

The advent of the Semantic Desktop [1] through projects such as *NEPOMUK* [2] now opens up new possibilities of solving this problem of integrating

[3] Nicely illustrated in the idea of *TCO - Total Cost of Ontologies*, e.g., http://www.w3.org/2005/Talks/1110-iswc-tbl/ (12/12/2007)

data and functionality from both Web and desktop. On the Semantic Desktop, data is lifted from application-specific formats to a universal format (RDF) in such a way that it can be interlinked across application boundaries: emails can be linked to calendar events, address book contacts to pictures or PDF documents, electronic plane tickets to tasks in a task management system. This allows new ways of organizing data, but also new views on and uses of arbitrary desktop data. To use a Web 2.0 term, data can now be *mashed-up*. What is more, because desktop data is now available in a Web format, it can also be interlinked and processed together with genuine Web data: e.g., a book editor could now query a SW-enabled conference website for the contact details of all authors who wrote papers about a specific topic (assuming that this data is available, such as e.g. on `http://data.semanticweb.org`), mash this data locally with a mail template and use his preferred mail client to send a call for contribution to all those authors. While the unified data model makes this scenario easier than it previously was, implementing it would ordinarily still require an experienced developer, who would use a full-fledged programming language to create applications that manipulate and visualize RDF data. With current tools, casual or naïve users would not be able to perform such tasks.

In this demo, we will present *Konduit*, a software that allows users to build simple applications and workflows which can create, manipulate, mash-up and visualize RDF data. Konduit is based on the ideas of a semantic desktop, combined with the principles of UNIX pipes, which has been a central part of UNIX and its derivatives since 1973, when it was introduced by M. Doug McIlroy. In order to allow non-expert users to work with RDF, Konduit is realized as a form of visual programming, meaning that it is *"a computer system whose execution can be specified without scripting"* [3][4]. In a sense, Konduit and similar systems are also related to the concept of data-flow programming [4], in which a program consists of a series of components with inputs and outputs, which become active when all of their inputs are valid.

We will outline an example scenario which illustrates the motivation behind building Konduit in Sect. 2, followed by a presentation of Konduit itself in Sect. 3 and a discussion of what will be contained in the actual demonstration in Sect. 4.

2 Motivating Example

The following example is inspired from a real-life scenario which recently occurred at our institute (names have been changed) and illustrates what Konduit can do for the user: The SemBar Institute needs to prepare a report for its funding agency, which has to show how well the SemBar researchers are connected to their research community. Among other things, co-authorship is a good indicator for this. For this reason, SemBar's scientific director Mary asked secretary Jim to do the following:

[4] Of course, that does not mean that the user is prohibited from extending the system with scripting, e.g., in the form of custom SPARQL queries.

– *Compile a list of all researchers.* To get this information, Jim accesses his electronic address book, in which he has cleverly organized all SemBar employees into groups like researchers, administrative staff, technical staff, etc.
– *For each researcher, compile a list of recent publications.* Unfortunately, SemBar does not yet have an internal list of publications (it's Jim's task to compile such a list, after all), so Jim has to resort to a web-based service like `http://data.semanticweb.org`.
– *For each publication, compile a list of co-authors.* Jim will have to spend a while to manually create these lists.
– *Organize all co-authors into one list and send it to Mary by email.* Jim has to manually remove duplicate and send the list using his mail client.

This scenario highlights a number of important aspects that our approach addresses, and illustrates how a tool such a Konduit can be used by Jim to aid him in his work:

– **Accessing and processing desktop data:** Since Jim is using a semantic desktop, his address book data is available as RDF and can therefore be processed by *Konduit*.
– **Accessing and processing Web data:** Service such as `http://data.semanticweb.org` expose their data as RDF, which means that our system can use it in the same way as desktop data.
– **Merging desktop and Web data:** Since both kinds of data sources use a unified data model, Konduit can simply mash both together.
– **Using desktop functionality:** Since our approach is desktop-based, we can easily access and integrate the functionality of arbitrary desktop applications (such as Jim's preferred email client).

3 Konduit

Konduit is a desktop-based visual programming environment that presents the user with a working environment not unlike a drawing application, and is as such similar to systems like Yahoo Pipes[5]. However, where Yahoo Pipes is restricted to web-based data in news feed-like form, Konduit works with any kind of RDF source, both from the Web and the user's desktop.

Within the Konduit environment, users can choose from a set of ready-made building blocks, which are organized into various groups (e.g., data-sources or operators). Each component has a number number of inputs and zero or more outputs. The user can drag the components onto the "drawing" area, move them around, connect outputs to inputs, set parameters and in this way build simple applications. Konduit defines two basic element types: *Source* and *Sink*. Source elements have an arbitrary number of sockets on which they provide data. Sink elements define plugs which can be plugged into the sockets to define the flow of the data. An element can also be a source and a sink at the same time. This is

[5] `http://pipes.yahoo.com/` (12/12/2007)

especially useful for elements that modify a stream of data (filter it, combine two streams into one, split the data, enrich the data with meta information, etc.). Based on these two simple types, a number of other components are available, such as various application adapters, SPARQL-based filter components, etc.

A Konduit workflow always has a start and an end point[6], represented by a source and sink element, respectively. The actual flow of data is activated from the end point, by clicking the activation button on a sink component (e.g., the "Send" button in Fig. 1). The connectors of each building block within Konduit are restricted to allow input or output of RDF — in this way, any group of components which have been combined to a workflow will also always have RDF input or output. As a result, combinations of components can be combined into complex elements and become part of the library of components the user can choose from.

An example of how Konduit looks like is given in Fig. 1. The screenshot shows a simple Konduit project, which will send mails to all members of a specific project group, combining desktop sources such as the address book and project data, a SPARQL construct query and functionality of a mail client in one workflow.

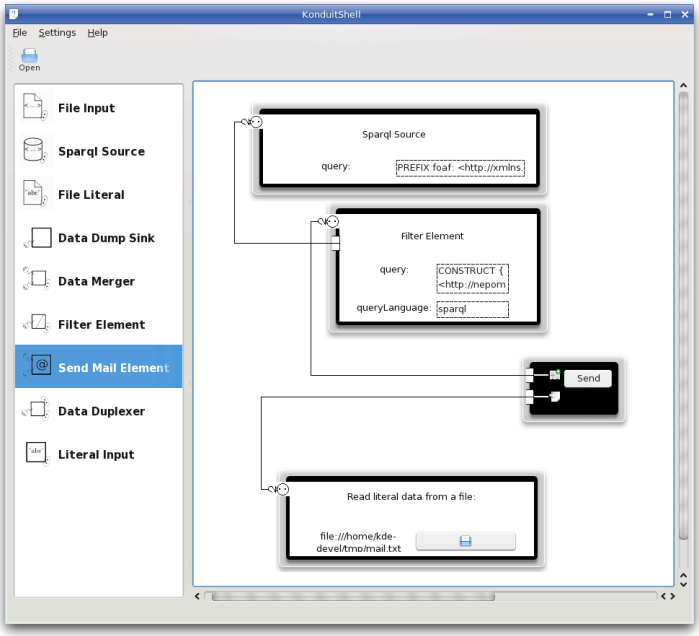

Fig. 1: A workflow for sending mails to all members of a project in *Konduit*

[6] Since the flow of data can be split, and different data flows can be combined, a Konduit workflow can also have multiple start and end points.

3.1 Implementation

Konduit is implemented as a desktop-based application for KDE 4 Linux, and is based on the Plasma engine[7]. Each component is realized as a plugin into the Konduit platform. Since components are also Plasma applets, designing and implementing new plugins for Konduit is quite straightforward (from the point of view of a KDE developer). The RDF functionality of Konduit makes use of the semantic desktop features that come as part of Nepomuk-KDE[8] implementation in the upcoming KDE 4, especially the Soprano RDF framework[9].

4 The Demo

During the demo, the audience will have the opportunity to get a live, hands-on experience of Konduit. The demo will consist of a running Nepomuk KDE installation from a fictitious user (one of the *personas* from the NEPOMUK project[10]) with various datasets that ordinary desktop users would find on their desktop, i.e. a contact database, calendar events, emails, textual documents, etc. A number of example scenarios will be prepared (such as the mailer example in the screenshot, or the more complex example in Sect. 2) to showcase what can be done with Konduit. However, visitors will be invited to experiment with and test the system in any way they like: access other data, re-order and re-connect components, interact with other application functionality adaptors, etc.

Acknowledgements

The work presented in this paper was supported (in part) by the Líon project supported by Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and (in part) by the European project NEPOMUK No FP6-027705.

References

1. S. Decker and M. R. Frank. The networked semantic desktop. In *WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web*, 2004.
2. T. Groza, S. Handschuh, K. Möller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjonsdottir. The NEPOMUK project - on the way to the social semantic desktop. In T. Pellegrini and S. Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 201–211. JUCS, 2007.
3. T. Menzies. Visual programming, knowledge engineering, and software engineering. In *Proc. 8th Int. Conf. Software Engineering and Knowledge Engineering, SEKE*. ACM Press, 1996.
4. L. Orman. A multilevel design architecture for decision support systems. *SIGMIS Database*, 15(3):3–10, 1984.

[7] `http://plasma.kde.org/` (10/01/2008)
[8] `http://nepomuk-kde.semanticdesktop.org/` (12/12/2007)
[9] `http://soprano.sourceforge.net/` (10/01/2008)
[10] `http://nepomuk.semanticdesktop.org/` (10/01/2008)