

# A Visual Interface for Building SPARQL Queries in Konduit

Knud Möller  
knud.moeller@deri.org

Laura Dragan  
laura.dragan@deri.org

Oszkar Ambrus  
oszkar.ambrus@deri.org

Siegfried Handschuh  
National University of Ireland,  
Galway (NUIG)  
Digital Enterprise Research  
Institute (DERI)  
siegfried.handschuh@deri.org

## ABSTRACT

This short demo description presents an extension to the *Konduit* tool for visual programming for the semantic desktop. Previously, Konduit required users to define filter components by manually specifying a SPARQL CONSTRUCT query. With the current work presented in this demo, we are exploring ways of building such queries visually, and aiding the use in doing it. We hope that this will make it easier to work with Konduit, which will thus appeal to more users.

## 1. INTRODUCTION

The Konduit tool<sup>1</sup> for generating simple RDF workflows (see Fig. 1) on the semantic desktop [1] is aimed at users who are capable of using complex software, yet don't have any programming experience, or are simply not willing to invest the time to program solutions for relatively simple, but repetitive and time-consuming knowledge tasks. However, writing SPARQL queries — an important component of Konduit — is still a considerable burden for those users if it has to be done manually and might deter them from using Konduit. For this reason, we have built a first version of a prototype to build SPARQL CONSTRUCT queries visually, thus hopefully improving the usability of Konduit significantly. If the research into visual interfaces of SPARQL queries turns out to be successful, we predict that those interfaces will also be useful in contexts outside of Konduit.

## 2. KONDUIT

In previous works [3] we have presented Konduit, a desktop-based tool to visually build simple workflows with RDF data. Konduit is integrated with the NEPOMUK KDE semantic desktop environment [2]. In NEPOMUK data from various desktop applications — e.g., contact details, document metadata or appointments and other events — as well as arbitrary other metadata is interlinked and represented in an integrated graph, and thus accessible through a uniform, system-wide interface. With Konduit, a knowledge worker can then use their desktop data, combine it freely with RDF data available on the Web, filter, merge, trigger functionality of other desktop applications and integrate arbitrary scripting code snippets into simple workflows which automate common, repetitive tasks. Examples for such tasks are generating reports from combined online and desktop

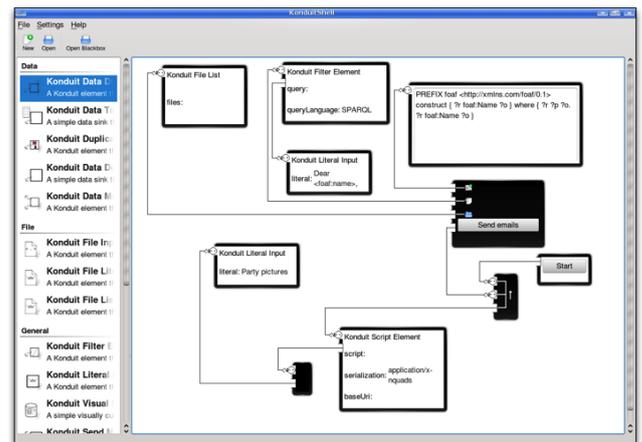


Figure 1: Interface of the Konduit tool

resources, sending emails with pictures of the last institute party to all members of the institute, etc.

### 2.1 SPARQL in Konduit

An important feature of Konduit is the possibility to define filter and transformer elements which modify and process incoming RDF into the desired output RDF. Technically, these elements are implemented as CONSTRUCT queries in the SPARQL<sup>2</sup> query language for RDF. In the initial version of Konduit, those queries had to be entered manually into a plain text box by the user (see Fig. 2).

The target user of Konduit is a knowledge worker with no or only basic programming skills, but good overall computer skills and a good understanding of the data they are dealing with. In other words, we are not (yet) looking at the proverbial “my mother”, but someone in the grey area between expert and naïve user. For such users, we consider defining a SPARQL query as feasible, but writing it manually is still tedious and error-prone. What is needed is a way to aid and guide users in defining those queries. In a first step, a syntactic support will already help semi-expert users in building queries faster and with fewer errors. A second step, which would abstract away from the actual syntax and model the query on a higher level, would make the system

<sup>1</sup><http://smile.deri.ie/projects/konduit>

<sup>2</sup><http://www.w3.org/TR/rdf-sparql-query/>

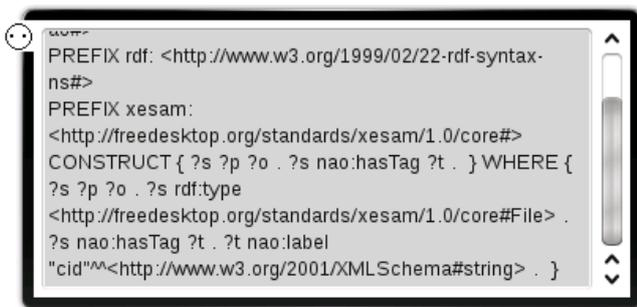


Figure 2: SPARQL CONSTRUCT element in Konduit

accessible to naïve users as well.

In this demo, we will present our current work in providing for basic syntactic support for query building. The first iteration of the interface is very simple, yet already very helpful in preventing errors and lowering the cognitive workload of the users. While the SPARQL text boxes from the previous version of Konduit still exist, the user is now presented with an additional helper window, in which they can assemble the query visually. From this interface, the query in the text box is generated.

Both the CONSTRUCT and the WHERE block of the query are built as series of triple patterns<sup>3</sup>, for which each component is entered into a text field. When each text field is filled, the pattern can be added to the list. This very basic functionality is further extended by the application of drop-down lists for predicates, as well as auto-completion. The drop-down lists are assembled by previously inspecting the current data in the NEPOMUK RDF store, while the auto-completion takes its completion suggestions from both the RDF store and the terms which have so far been entered as part of the query. Additionally, the user can also specify data types for literals from a selection box.

### 3. CONCLUSIONS AND OUTLOOK

The interface presented in this short paper is only a first step to provide non-expert users with a way to define SPARQL queries, and thus filter and manipulate data on the Semantic Web (or Desktop). However, because it prevents the user from errors, relieves them off typing effort and, according to our intuition, lowers the learning curve required for building queries, we think that an interface like this already provides significant advantages over manual query writing.

For further research, we will be looking into abstracting from the SPARQL syntax completely, and instead use an interface that is based on the visual representation of nodes and arcs instead, such as the “InteractiveSparqlQueryBuilder”<sup>4</sup> by OpenLink Software. However, an interesting research question would be to find how, for queries of differing complexity, a graph-based visualisation is actually more or less intuitive. An alternative idea would be an interface that applies the idea of Query by Example to SPARQL. Finally, a combination of approaches is worth investigating,

<sup>3</sup>The syntactic power of SPARQL is obviously restricted in this way.

<sup>4</sup><http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/InteractiveSparqlQueryBuilder>

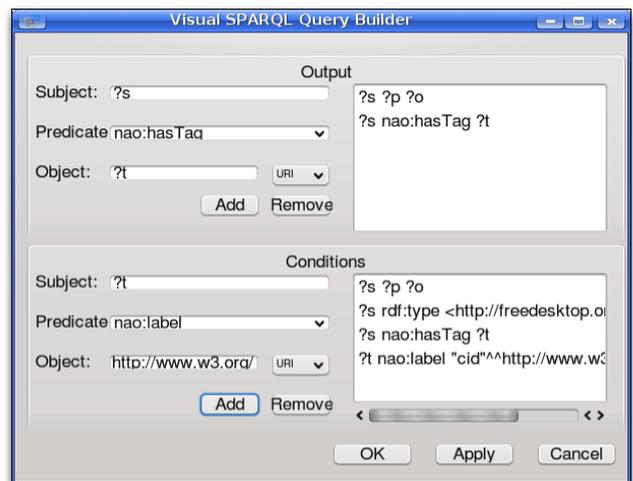


Figure 3: Building a SPARQL CONSTRUCT query in Konduit

where parts of the query (a simple graph pattern) would be represented graphically, whereas other aspects of SPARQL would still be edited in the syntax, but with UI support.

### Acknowledgments

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

### 4. 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] K. Möller, S. Handschuh, S. Trüg, L. Josan, and S. Decker. Demo: Visual programming for the semantic desktop with Konduit. In S. Bechhofer, editor, *Proceedings of the 5th European Semantic Web Conference (ESWC2008), Tenerife, Spain*, volume 5021 of *LNCIS*, pages 849–553. Springer, June 2008.