

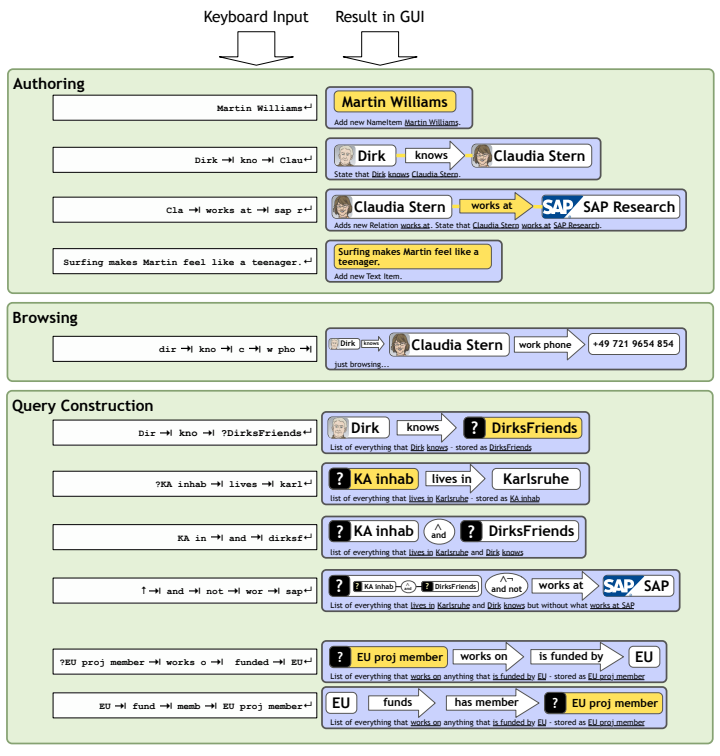


QuiKey – The Smart Semantic Commandline (a concept)

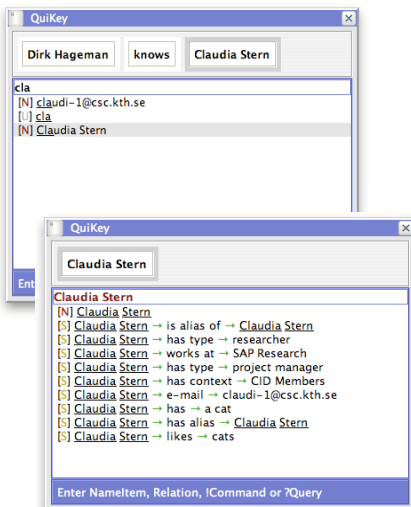
Abstract

QuiKey is a kind of smart semantic command-line that focuses on highest interaction-efficiency to browse, query and author semantic knowledge bases in a step-by-step manner. It combines ideas of simple interaction techniques like auto-completion, command interpreters and faceted browsing and integrates them to a new interaction concept. QuiKey forms a generic, extensible user interface for graph-structured knowledge bases.

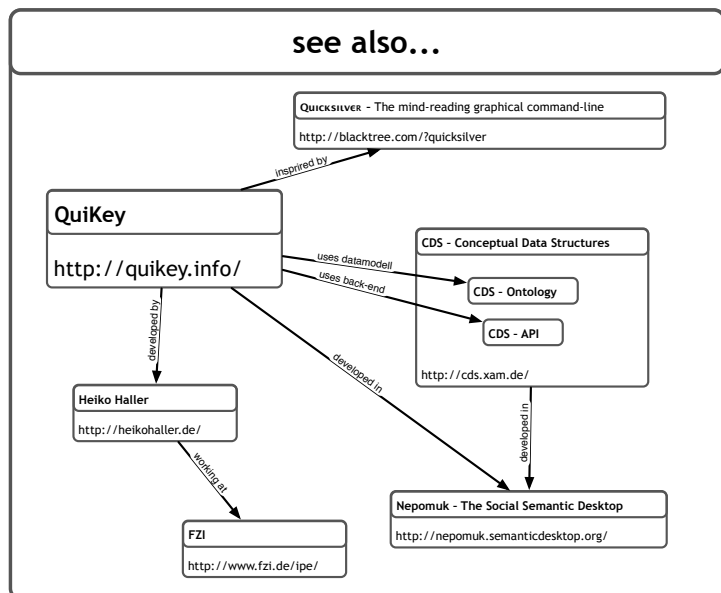
User Interface Mock-Ups



current implementation



see also...



QuiKey

Heiko Haller

Forschungszentrum Informatik (FZI), Germany
heiko.haller@fzi.de

Abstract. QuiKey is a kind of smart semantic command-line that focuses on highest interaction-efficiency to browse, query and author semantic knowledge bases in a step-by-step manner. It combines ideas of simple interaction techniques like auto-completion, command interpreters and faceted browsing and integrates them to a new interaction concept. QuiKey forms a generic, extensible user interface for graph-structured knowledge bases.

In knowledge bases like a semantic desktop, knowledge is typically be modelled in a formal and fine granular way. QuiKey provides a light-weight generic UI for browsing and editing them in such fine-granular ways. It also brings simple ways of constructing structured queries to not-so-technically-advanced users.

The current implementation of QuiKey is built on top of CDS [1], a lightweight top-level ontology designed to bridge the gap between unstructured content like informal notes and formal semantics like ontologies.

QuiKey is organised around the notion of *parts*. A part can be an existing item, a relation, a new text string or a command. Depending on the number, order and types of parts entered, it is decided what action to take.

Authoring To add a new text item to the knowledge base, it is enough to just type the text into the QuiKey console and press enter. To make statements about existing items, the statement can just be entered in a subject-predicate-object fashion, separated by tab-keys. Only that the user would not even have to type in the whole labels because parts that are already known are auto-completed. If not all three parts in such a statement are known strings, the respective items or relations are automatically added to the knowledge base. Like this, a knowledge graph can be woven in single simple steps in an ad-hoc fashion.

Browsing Simply navigating the knowledge base through its graph structure is done with QuiKey without even changing into a different mode: when a part



Fig. 1. Mock-up showing how both a new statement and relation are added.



Fig. 2. Mock-up of a simple and a nested modular query including generic description of its meaning.

has been selected, before the user types anything new to select the next part, existing contents that fit the part pattern are already displayed in the suggestion area and can be browsed in a way similar to faceted browsing.

Queries Constructing complex, possibly nested queries is difficult for non-expert users and every slight error in the syntax of the query makes the whole query fail or return unintended results. QuiKey tackles two common problems:

a) *Misspellings and syntax errors* are largely avoided because instead of requiring the user to write a whole query in some complicated syntax which is parsed later on, in QuiKey the query is constructed interactively, selecting from existing items and without the need of syntactical characters.

b) To facilitate modular construction of complex queries in a step-by-step manner, each query can be saved and referred to as a special query item. More complex queries can be constructed by combining existing query items:

The current implementation that is developed in the semantic desktop project nepomuk¹, is available as a working alpha prototype. While the currently used CDS middleware uses the nepomuk semantic desktop infrastructure as a back end, the general approach could also be used for plain rdf- or other graph-based knowledge bases. In the current implementation, a list of CDS:NameItems (named entities) is cached for fast auto-completion, while structural queries are converted to SPARQL. However, since the expressiveness of QuiKey's queries does not exceed EL++[2], there could also be optimised implementations that scale to large knowledge bases without slowing down user experience.

For further information on QuiKey, see <http://quikey.info/>.

References

1. Völkel, M., Haller, H.: Conceptual data structures (cds) – towards an ontology for semi-formal articulation of personal knowledge. In: Proc. of the 14th International Conference on Conceptual Structures 2006, Aalborg University - Denmark (2006)
2. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for horn description logics. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, Vancouver, British Columbia, Canada, AAAI Press (2007) 452–457

¹ <http://nepomuk.semanticdesktop.org/>, financed by the EU (IST-FP6-027705).