TRIAL-SOLUTION
Project Deliverable D7

Knowledge Management System

Peter Baumgartner
Universität Koblenz-Landau
Institut für Informatik
Rheinau 1
D-56075 Koblenz

Email: peter@uni-koblenz.de

March 28, 2002

Contents

1 Introduction

The Knowledge Management System (KMS) is a component of the TRIAL-SOLUTION tool set. The task of the KMS is to compute assemblies of new documents from elementary document units ("slices") stored in a database. The assembly is done in a selective way, based on the user's interest, preferences and knowledge.

In order to appreciate the role of the KMS, some information about the TRIAL-SOLUTION project seems helpful.

About the TRIAL-SOLUTION Project

TRIAL-SOLUTION stands for Tools for Reusable, Integrated, Adaptable Learning - Systems/-standards for Open Learning Using Tested, Interoperable Objects and Networking. It is a project funded by the EU as part of its Information Society Technologies Programme (IST), which is a major theme of research and technological development within the EU's Fifth RTD Framework Programme.

In brief, in the TRIAL-SOLUTION project, mathematical textbooks are separated into small units and stored in a database for later retrieval. The general idea is to combine the advantages of books with the flexibility of databases (more information about the TRIAL-SOLUTION project can be found at `http://www.trial-solution.de/`).

About the Knowledge Management System

The KMS is *not* a standalone system. It interfaces with other tools of the TRIAL-SOLUTION tool set. Quoting from the Annex 1 of the TRIAL-SOLUTION contract,

> "The KMS takes the metadata delivered by the Metadata Assignment System (D6) and information about the intentions, preferences and pre-requisites of the user as specified by the Component Design Document (D5) and combines them in order to infer new metadata and to propose collections of learning objects to the user. It works fully automatic."

A word on terminology: in the sequel, when speaking of "computing assemblies" we mean the computations that yield the just mentioned, combined learning objects, where "learning objects" means "documents".

Computing assemblies is a complex task. In order to master the complexity, we employ Artificial Intelligence methods, in particular methods from the areas of logic programming and automated deduction.

In the heart of the KMS is a an automated deduction system – the krhyper[1] deduction system – specifically designed (but applicable in other context as well) for our purpose. The assembly tasks are specified by *logic programs* and fed into krhyper. Logic programs thus substitute traditional imperative, object-oriented programs, or database programs (e.g. SQL) that would have to be written otherwise.

This approach gives benefits that are not obtainable with the traditional techniques. The main benefits arise from the purely *declarative*[2] nature of logic programs, which are the following:

- Logic programs are easier to understand - there is no explicit control flow in the program.
- Our logic programs are compositional: they can be combined in a plug-and-play manner to add or remove functionality.
- Our logic programs are comparatively short.
- As a consequence of the previous items, logic programs can be more easily adapted and maintained.

It should be emphasized that the techniques we employed are rather general and applicable in different context as well. This is, because the KMS works on meta-data *only* and knows nothing about the particular subject contents. Conceivable reuse of the techniques covers the assembly of arbitrary objects from specifications under a rule-based constraint system, as e.g. in product configuration or analysis, combination and retrieval of information in the "Semantic Web".

Here is some actual code from the KMS in order to get a flavor of the logic programs we use. It states two cases of how to derive an "interesting unit". "interesting units" are at a middle layer in the computational hierarchy. If an "interesting unit" passes successfully further tests, it names a slice that goes into the final assembly:

```
interesting_unit(Book/Unit) :-
        %% first case: the unit may be selected explicity:
        selected_unit(Book/Unit),
        %% see if it stems from a book enabled in preferences:
        book(Book),
        %% see if it has attached at least one keyword that is unknown:
        unitKeyword(Book/Unit,Keyword),
        unknown(Book/Unit,Keyword).

interesting_unit(Book/Unit) :-
        %% first case: a topic 'Keyword' was derived as interesting
        interesting_topic(Keyword),
```

[1]"krhyper" stands for *k*nowledge *r*representation *hyper* tableau.

[2]In *declarative* programming, it is specified *what* shall be computed – not *how*.

```
%% we have to consider units that carry this keyword:
unitKeyword(Book/Unit,Keyword),
%% see if it stems from a book enabled in preferences:
book(Book),
%% see if this keyword is unknown:
unknown(Book/Unit,Keyword).
```

This text, however, is not about the techniques underlying the KMS (Scientific publications in this regard are [**?**; **?**]). It rather describes the system requirements, the architecture and the application programming interface of the Knowledge Management System .

2 System Requirements

Hardware: disk storage $\approx$ 50 MBytes , main memory $\geq$ 256 MByte.

The KMS runs on any Unix-like hardware platform that supports Eclipse Prolog and Perl.

Software: **Eclipse Prolog**, version 5.2 or higher. Eclipse Prolog is available from IC Parc, London, England, free of charge for academical sites (cf. `http://www-icparc.doc.ic.ac.uk/eclipse/`).

The **Perl** programming language system, version 5.6.1 or higher.

The **Krhyper deduction system**. It is publicly available from `http://www.uni-koblenz.de/~peter/KRHYPER/`.

The **Book metadata and control programs**. They are available from the author on request.

3 System Architecture

3.1 Overview

We are going to describe the overall system architecture now. Figure 1 depicts the architecture graphically.

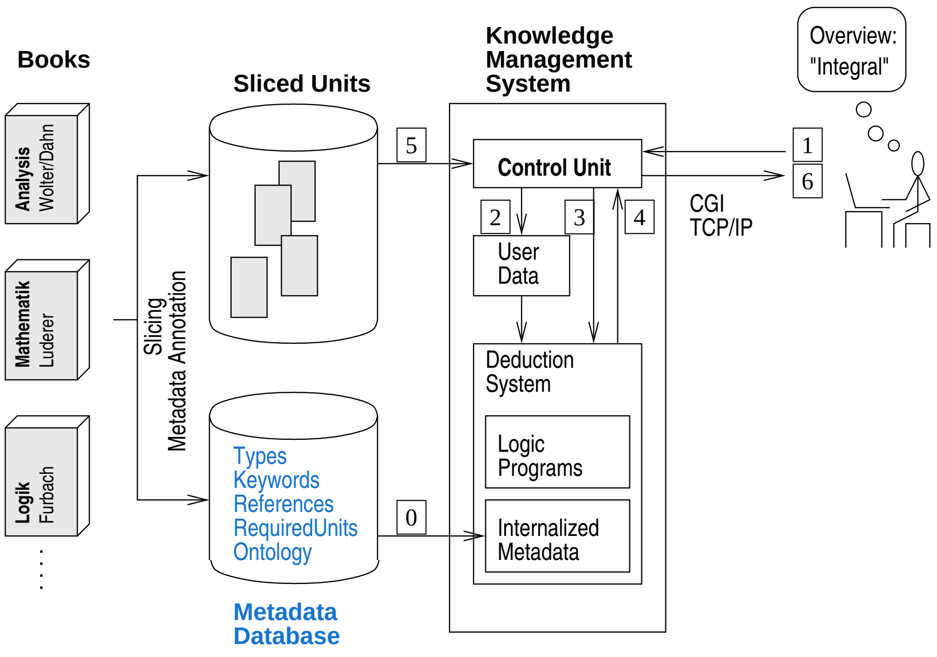

Figure 1: Knowledge Management System architecture.