Living Book – An Interactive and Personalized Book*

Peter Baumgartner Margret Gross-Hardt
Anna B. Simon

Institut für Informatik
Universität Koblenz–Landau
Koblenz, Germany

E-mail: {peter,margret,annab}@uni-koblenz.de

1 Introduction

In the era of information technology, providing high quality and state of the art educational material is quite a challenge and requires the use of new media. The In2Math project at the University of Koblenz-Landau develops in this context the *Living Book*; Living Book means personalized user oriented educational material together with interactive components. The Living Book aims at supporting the fundamental parts of undergraduate classes in theoretical computer science. The main goal is to support the active, explorative and self-determined learning in lectures, tutorials and selfstudy.

This paper describes the main aspects of the Living Book and explains the concepts and ideas behind it as well as the techniques used to realized these concepts. As the title of the paper already indicate, the main aspects that have driven the development of the Living Book are: personalization of content according to the user's needs, and support of explorative and interactive learning techniques by means of embedded systems.

We rely on the idea of combinating the advantages of printed books, e.g. excellent readability of mathematical and other scientific texts, with the flexibility and actuality of web based techniques. The statical sequential structure of classical textbooks is overcome by the possibility of a student to dynamically select those subjects she is

*The project is financed by the German Ministry of Education and Research, BMBF (www.bmbf.de) within the program "New Media in Education".

interested in or she needs to know in order, say, to prepare for an exam. The electronic book is organized as a collection of small units, so called slices, representing units of knowlegde. Furthermore, meta data describe the dependencies between different slices. This knowledge together with the user's selection of subjects is used to generate a personalized book that fits the user's needs. Different users therefore might work with different personalized books; also, one user might have different personalized books related to various learning situations.

A number of mathematical systems extend the functionality of the teaching material and realizes the *Living* Book. Problem generators provide a repository for training; problem checkers control the students' solution, etc. The lecturer and the student might use these systems in different interactive settings and related to different teaching and learning goals. All systems are integrated seamlessly in the book; neither student nor lecturer do have to spend time in learning different interfaces of the underlying systems which might be quite time consuming and sometimes frustrating if the use of the system is complex. Instead, the Living Book offers an easy to use web based interface.

In order to achieve high quality layout of the electronic material, and also allow a user to print some parts of the material, LaTeX is used for typesetting with PDF as the target format, which in turn is used for displaying the material in a web based environment. LaTeX in combination with PDF on the one hand side guarantees excellent display of scientific content, as e.g. mathematical formulae, and other side allows to print (part of) the documents with high printing quality.

Currently, the Living Book "Logic for Computer Science" (author: Prof. U. Furbach, Universität Koblenz-Landau) is developed and has been used during lectures and tutorials, already. Due to the fact that this kind of electronic material is quite new and experience still has to be collected, regular evaluations are taking place [Simon and Valerius, 2002].

This paper describes the main ideas and concepts behind the Living Book and presents the techniques used to achieve these goals.

1.1 Structure of the document

The Living Book relies on two main components: Slicing Information Technology and Interactivity. Section 2 presents the slicing technology, Section 3 presents the interactive part of the Living Book. The paper is summarized in Section 4.

2 Slicing Information Technology

The Living Book is based on the so called *Slicing Information Technology (SIT)* for the management of personalized documents (cf. `http://www.slicing-infotech.de/en/index.php`). With SIT, a document, say a mathematical text book is separated once, as a preparatory step into a number of small units, such as definitions, theorems,

proofs, and so on. The purpose of the *sliced* book then is to enable authors, instructors and students to produce personalized teaching or learning materials based on a selective assembly of units. In particular, the slicing technology relieves the user from considering a document as a static sequence of document components but allows for dynamic retrieval of parts of the book according to the user's needs: for instance, if a lecturer wants to use the teaching material for different learning groups or if a students needs the material in different learning situations.

Once the reader is entering the portal of the book in the web, she can login with her account and gets the entry page of the book: this is depicted in figure 1.

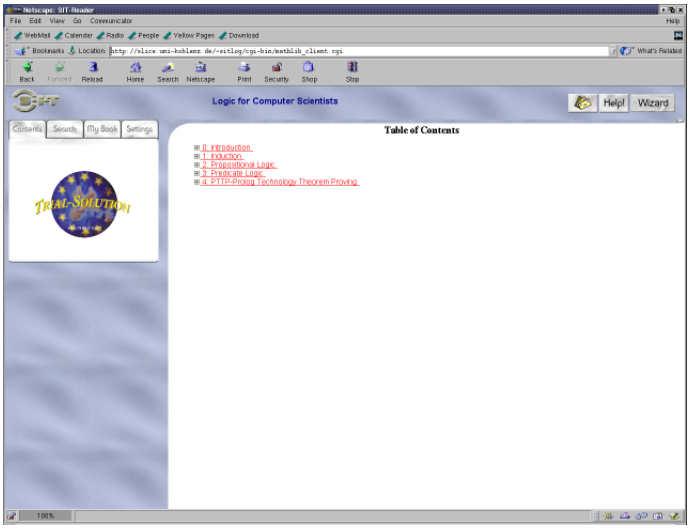


Figure 1: Entry page of *Logic for computer scientists*.

Besides many other features, the reader can decide to read e.g. the definition of "Semantics in Predicate Logic", which is the slice number `3/2/3`. If the reader is marking these units – as is depicted in Figure 2 – and is then clicking the "read"' button on the left of the screen, she will get a PDF document containing just this part of the book.

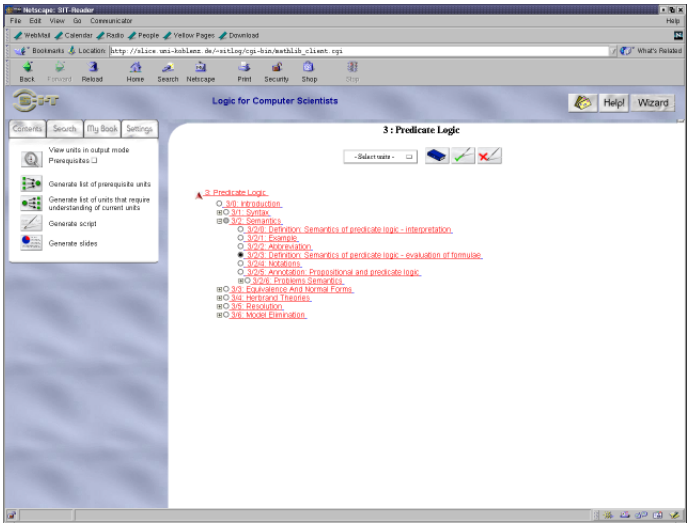


Figure 2: Selection of a unit.

If she now discovers that she does not understand the definition because it relies on some other terms introduced before, she can decide to include all the material necessary for the understanding the units presented until now. Again, after clicking the appropriate button she gets the view shown in Figure 3, offering all the material which is computed as prerequisites for this definition.

After marking all or some of these units, she can decide to get the corresponding

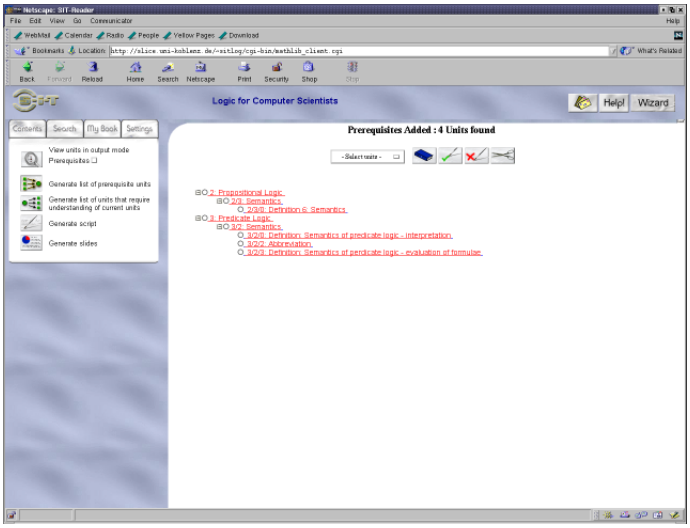


Figure 3: A personalized view by including prerequisites of the selected unit.

PDF document; it is partially depicted in Figure 2.

In a similar way the reader can choose to include all material which uses the slices marked until then, e.g. to understand how and where a certain property is used in the rest of the chapter or the entire book. She furthermore can store information, which parts of the book she do not want to get presented anymore, because she knows it very well. All such reasoning and computations are only possible, because besides the LaTeX sources of the units there is also a lot of meta-data stored on the server. This includes besides a glossary and keywords also information about relations of a unit to other units. Hence, we have not only the text of the book, we have an entire knowledge base about the material, which can be used by the reader.

2.1 Metadata describe slices and their relationships

Slices are associated with metadata that describe, for instance, the *type* of a slice, which might be *example, definition, proof, connecting text*, etc. Additionally, metadata contain (among other things) keywords attached to slices. All this information, together with a thesaurus allows for sophisticated searches within the documents. A user therefore has the option to assemble a document containing mainly examples and omitting proofs; this combination might be quite useful in getting a first understanding of a new subject because the user does not have to cope with theorems and proofs in the first place. The thesaurus, furthermore, provides the means to retrieve similar concepts related to a certain term.

Metadata, however, does not only describe types of information content but also relationships between different slices. This knowledge is used in assembling document components together with their presuppositions. As described in the previous section, if a reader decides to read a single unit, e.g. a certain theorem, she can request the system to deliver also those units that are needed to understand the theorem. By exploiting metadata, the Living Book will deliver all units explicitly selected by the user or derived by evaluating dependencies between units, and present these as a

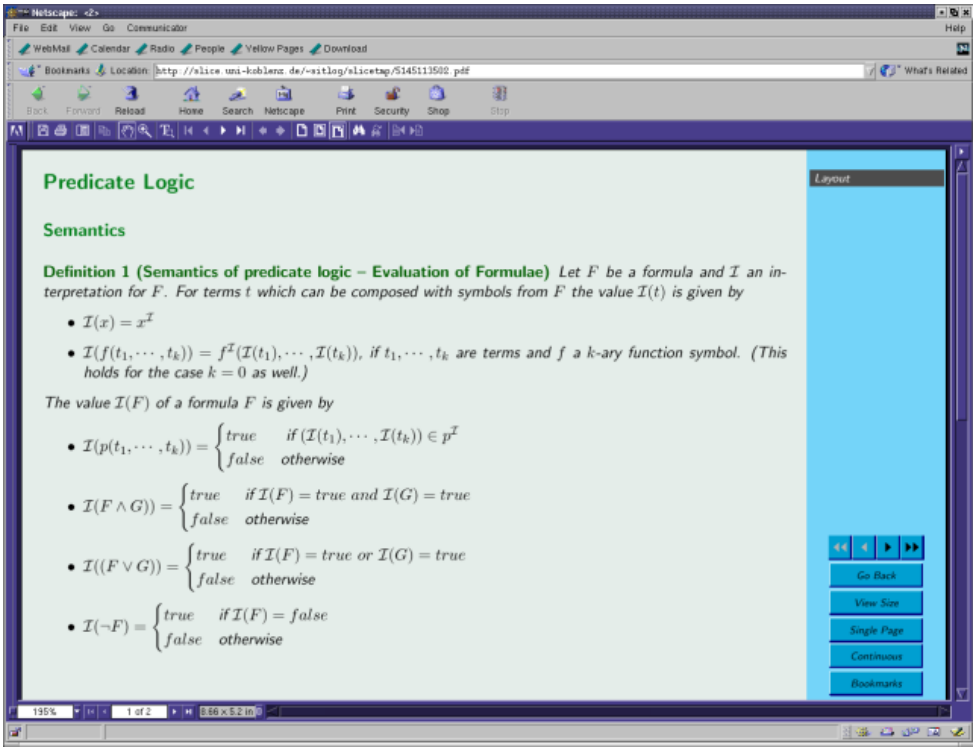


Figure 4: A PDF view of the result in Figure 3.

continuous text in PDF.

3 Interactivity in the Living Book

One of the goals we are aiming at, is the support of explorative learning. This means, students have control about accessing the learning material from different angles and different levels of detail. They also have options how to solve certain problems. A problem can be solved manually or by means of some system; for instance, a truth table generator can help in deciding if a certain logical formula is satisifiable.

Within the PDF textbook, interactive elements allow the user to make use of different mathematical or logical systems. The student does not have to learn different system interfaces, but uses a uniform interface for all embedded systems.

Currently, the Living Book integrates the following tools:

- *Formulae Database*: A set of "current" formulas is maintained by the system. There are operations to add and delete formulas, and also to store and retrieve the current set under a chosen name. Another means to obtain formulae is the following.

- *Formulae Generator*: This tool generates propositional formulae that can be used in various exercises. Thereby, the Living Book memorizes which formulae already have been generated for a certain user. These formulae will not presented a second time.

- *Truth Table Generator*: This system takes a propositional formula as input and generates a truth table. It is used in the first part of the logic course where the students learn about propositional formulae, satisifiabitlity and model generation.

- *Tableau prover*: This system is used to test satisfiability of propositional formulas. The salient feature of this system is a graphical, tree based visualization of the computed tableau. The visualization yields an illustrative and explanatory description of the steps executed in finding a proof (or a counterexample).

  A possible application scenario is to quiz students to formalize in propositional logic, say, a logical puzzle, and use the Tableaux prover to "debug" their solution. Another scenario is described below in Section 3.1.

- *CNF Transformation.* This system takes a propositional formula as input and generates its clause normal form (CNF). It can be used by the students to understand clauses and the semantic equivalence between propositional formulae and CNF representation. Also, the CNF representation is the input format accepted by most automated deduction systems, and the CNF obtained by this tool can be fed directly into the deduction systems in the Living Book.

- *Resolution systems.* The resolution calculus is the most prominent method for reasoning with CNF formulas. There are two respective systems included: a naïve one, and the state of the art theorem prover OTTER. The naïve one closely corresponds to the resolution calculus as introduced in the logic course. Since it is not optimized, it can be used for small examples only, and it thus demonstrates the need for optimized techniques (which are present in OTTER).

- *PROTEIN prover.* The PROTEIN model elimination prover takes CNF as input, and it returns a tableau similar as mentioned above; it is an alternative to using OTTER, and it can be used to demonstrate the need for more than one system.

Basically, the Living Book and all the systems embedded are used not only by students but also by instructors to explain different topics.

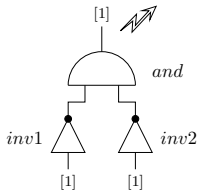
Currently, the mentioned interactions are extended from propositional to first-order predicate logic, which is another core topic in the logic course.

3.1 Interactivity - by example

This section describes the use of the interactive components in the Living Book by means of an example. The domain is the logic-based diagnosis of electrical circuits.

The theory behind logic-based diagnosis is sufficiently explained in the course. The purpose of the use of an interactive system here is to train students in modeling an application domain and let them use an automated theorem prover to solve some task from the application domain (i.e. finding diagnosis) .

When treated within a (classroom) exercise, the modelling of the application would presumably be prepared in a stepwise way, thus gradually introducing more complexity and more involved questions to solve with the interactive component. There is no need to describe these steps here in detail, and we are thus concentrating on the final step.



To mediate the principles of logic-based diagnosis, a microscopic example is sufficient. The underlying circuit is depicted in the figure on the left. Notice that the observed output ("[1]") contradicts the output expected from the given input vector ("[1] - [1]"). The students are expected to describe this circuit by formulas of propositional logic. Of particular importance is to take abnormal behavior into account.

A correct formalization is depicted in the upper half of the screenshot in Figure 3.1, and (parts of) the user input leading to it is depicted in the lower half.

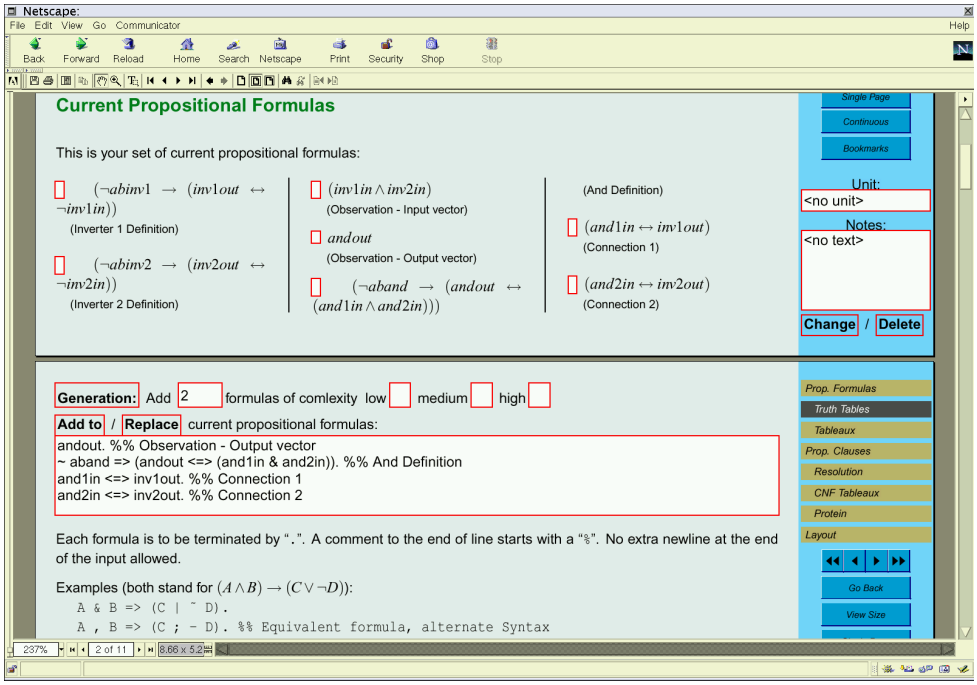


Figure 5: The formula editor, showing a formalization of the circuit in Figure 3.1.

Now, various interactive components could be used to solve the task at hand, which

is to determine possibly faulty components of the circuit that are consistent with the observed behavior. The most natural interactive component to use here, however, is the *Tableau prover*. After invocation, a tableau for the given formulas is constructed. Tableaux are tree data structures, and it is easy to read off certain logical properties from them. In particular, from inspecting the branches the diagnosis task can be solved. In order to enable students to do so, tableau are typeset by some LaTeX tree drawing package. The result is delivered in PDF and can be inspected as usual. Figure 3.1 shows an overview of the tableau, and Figure 3.1 shows a zoomed-in view.

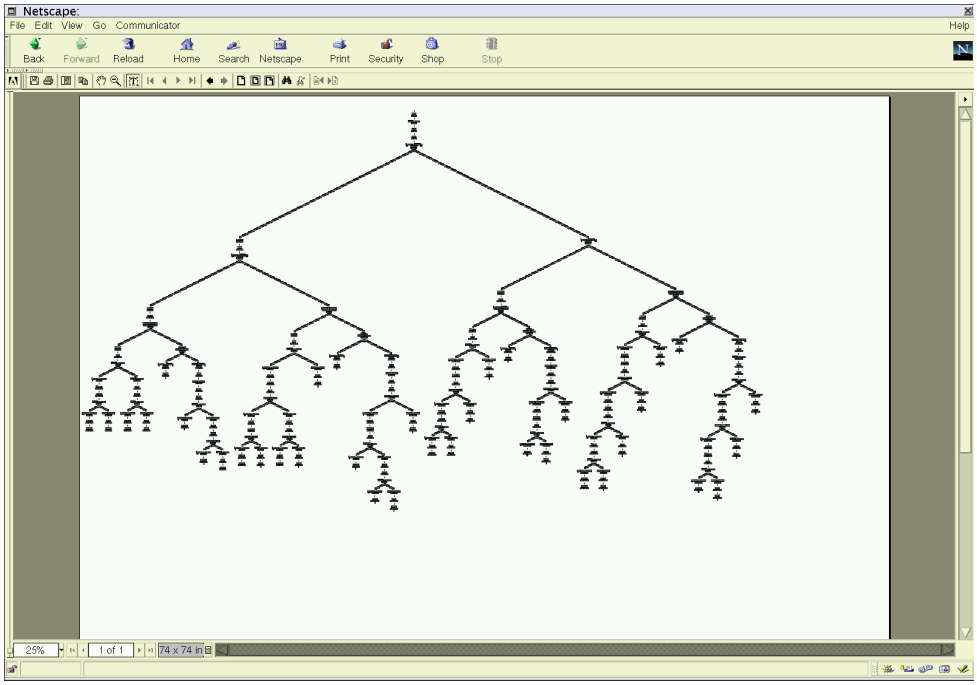


Figure 6: The tableau resulting from the formulas in Figure 3.1 (overview).

The kind of interaction as just outlined is currently explored within the logic course held in the summer term 2002. First experiences are very encouraging, and a more formal evaluation will be carried out at the end of the course.

3.2 Architecture

The Living Book is a distributed system with a web based client interface and two servers realizing the main parts of the system: The content of the Living Book is provided by the SIT-Server. The SIT-Server maintains the units of one or multiple related books and keeps track of the user profiles and their personalized views onto the learning material. The SIT-Server communicates with the Interaction Server which

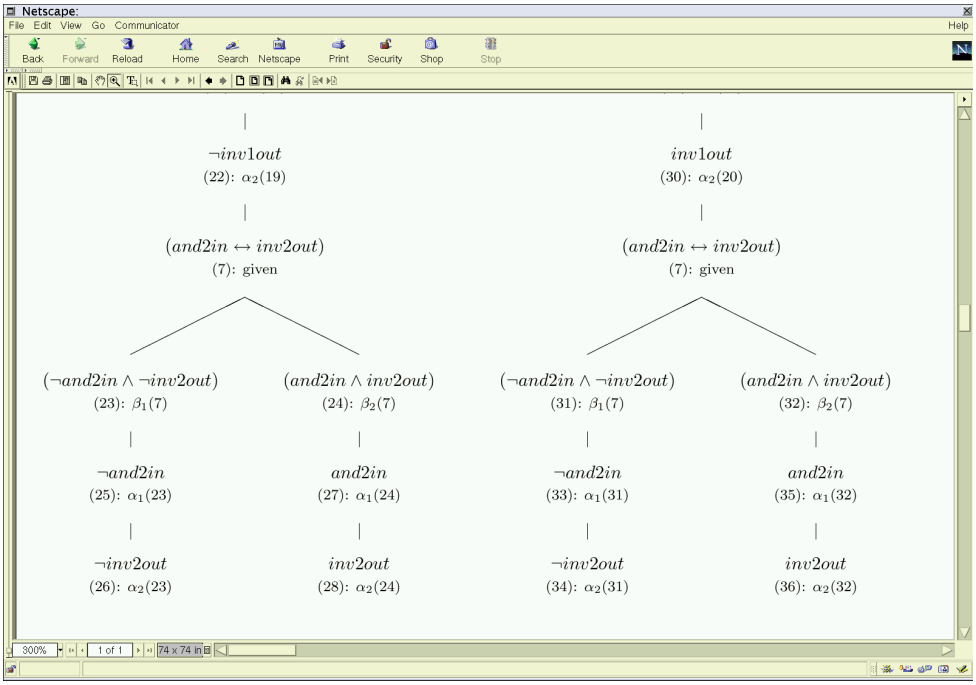


Figure 7: A zoomed-in view of the tableau in Figure 3.1.

is responsible for access to different mathematical and logic based systems embedded into the learning material.

The Interaction Server is responsible for integration of interactive components related to the Living Book. This server maintains the database of interactive system components from one and even possible from multiple books. The Interaction Server wraps the different mathematical and logic systems and provides one common interface to different connected systems. Furthermore it maintains the user profiles related to the interaction with the Living Book. Figure 3.2 illustrates the process architecture.

An incoming request is handled by the SIT-Server; if the requested units by the user contain interactive ones, the Interaction Server gets appropriate requests and retrieves content provided by interactive services. These services (yellow boxes) either are running within the Interaction Server or run remotely and can be accessed, e.g. via the internet. The Interaction Server generates LaTeX fragments representing e.g. the result of some problem execution and returns these fragments to the SIT-Server. There, all units, no matter, if they were generated dynamically from the Interaction Server or retrieved from the knowledge based are put together resulting in one LaTeX document which is transformed into PDF and transfered to the user.

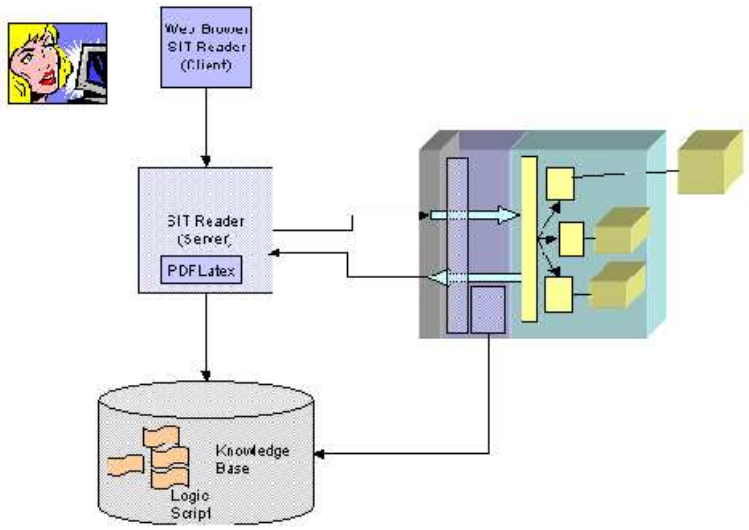


Figure 8: Process arrchitecture of the Living Book.

4 Summary

In this paper we have presented the Living Book. The Living Book stands for personalized learning and teaching material as well as for interactive use of it.

One challenge in supporting interaction and new media within learning and teaching material for natural sciences is that these materials often contain many textblocks with formalized content. In2Math's goal is to provide readable and high quality presentations of these contents.

Documents in the Living Book are typeset by the author with the best available typesetting program, LaTeX, and delivered in the PDF format (Portable Document Format). This is a must, because mathematical and logic formulas appear very frequently. LaTeX/PDF is used for both the online usage of the system through a web browser and for the printed material. For both usages, the presentations are tailored respectively.

It poses quite an implementational challenge to take LaTeX/PDF as the base technology for dynamically changing online material. However, we think that the benefits (Quality of presentation, avoiding media breaks) are worth the effort.

Notice that we do not rely on, e.g., a LaTeX to HTML converter. This would not be feasible for various reasons. Beyond inferior quality, poor scalability of the resulting document, the conversion times, and thus the turn-around times for interactions, would be beyond acceptability.

The material and the system has been used during lectures already and also has been evaluated by means of questionnaires filled out by the students [Simon and

Valerius, 2002]. The overall feedback has been quite encouraging and we plan for further systems that are embedded into the material.

References

[Simon and Valerius, 2002] Anna Simon and Marianne Valerius. User Requirements – Lessons Learned from a Computer Science Course. Fachberichte Informatik 5–2002, Universität Koblenz-Landau, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2002.