University of Crete
School of Sciences and Engineering
Computer Science Department

Personalized
Rule-based E-Learning
Using Semantic Web Technologies

Zebide Akkus

Master of Science Thesis

Heraklion, November 2006

Πανεπιστήμιο Κρήτης
Σχολή Θετικών και Τεχνολογικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

Σύστημα εξατομικευμένης ηλεκτρονικής μάθησης, βασισμένο σε κανόνες, με χρήση τεχνολογιών Semantic Web

Εργασία που υποβλήθηκε από την
Zebide Akkus
ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Zebide Akkus
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Εισηγητική Επιτροπή:

Γρηγόρης Αντωνίου Καθηγητής, Επόπτης

Δημήτρης Πλεξουσάκης Αναπληρωτής Καθηγητής, Μέλος

Βασίλης Χριστοφίδης Αναπληρωτής Καθηγητής, Μέλος

Δεκτή:

Παναγιώτης Τραχανιάς
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών
Ηράκλειο, Νοέμβριος 2006

Personalized Rule-based E-Learning

Using Semantic Web Technologies

Zebide Akkus

Master of Science Thesis

Computer Science Department, University of Crete

Abstract

Σύστημα εξατομικευμένης ηλεκτρονικής μάθησης, βασισμένο σε κανόνες, με χρήση τεχνολογιών Semantic Web

Zebide Akkus

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Περίληψη

Ευχαριστίες

Αισθάνομαι την ανάγκη να εκφράσω ένα μεγάλο ευχαριστώ στον επόπτη καθηγητή μου κ. Γρηγόρη Αντωνίου, για την ουσιαστική καθοδήγηση και συμβολή του στην ολοκλήρωση της παρούσας εργασίας .Το ενδιαφέρον και η κατανόηση που έδειχνε για κάθε πρόβλημα με έβγαλαν από αδιέξοδο σε πολλές και διαφορετικές περιστάσεις. Θα ήθελα να τον ευχαριστήσω σε προσωπικό επίπεδο για την συμπαράστασή του στις δυσκολίες που προέκυψαν σε αυτό το διάστημα.

Επίσης θα ήθελα να ευχαριστήσω τον καθηγητή κ. Δημήτρη Πλεξουσάκη για την συμμετοχή του στην εξεταστική επιτροπή και για τις συμβουλές που μου έδωσε για την ολοκλήρωση της εργασίας μου. Επιπλέον, όποτε χρειαζόμουν κάποια άλλη συμβουλή ή βοήθεια, έδειχνε πάντα ουσιαστικό ενδιαφέρον.

Παράλληλα, θα ήθελα να ευχαριστήσω τον καθηγητή κ. Βασίλη Χριστοφίδη για την προθυμία του να συμμετάσχει στην εξεταστική μου επιτροπή.

Ακόμα, θέλω να ευχαριστήσω τον υποψήφιο διδάκτορα Αντώνη Μπικάκη για τη βοήθειά του σε όλη τη διάρκεια εκπόνησης της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω το Πανεπιστήμιο Κρήτης και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Ερευνάς για τις γνώσεις και τις εμπειρίες που μου πρόσφεραν όλα αυτά τα χρόνια.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Contribution of the Study

Online courses, web-based education, computer supported training and even virtual university are already wide used terms. All of them represent e-learning which is growing very fast both in educational end corporate environment. In particular, e-learning systems that are offered via the world-wide web can be considered as specific web-based information systems with a focus on the provision of knowledge to learners. Traditional computer-based learning environments are often driven by prescriptive programs that allow the learner to input information, however the responses to that input are prescribed and predetermined. In essence they are "*closed*" systems. A good example of this approach would be the placing courseware on a web server to be accessed by remote students, which would suit the prescriptive pattern of a taught course. In our estimation, an e-learning environment should be "*open*"; that is to say it can be adapted by learners or trainers to the particular needs of learners, teams or groups of learners from different surroundings or cultures. It is modular in order to facilitate its adaptation, updating or its re-engineering. The today's needs call for an e-learning environment capable of supporting a dynamic learning process, concerning learners and instructors who *share* knowledge and both contribute to a shared cognition.

In this context, several works [65], [66], [67] have proposed the use of the Semantic Web technologies that provide the needed infrastructure to build a dynamic distributed learning environment. The use of ontologies to model knowledge enables the creation of semantic relations among resources publicly available via the www and the standard query facilities, that current Semantic Web languages provide, enable their retrieval according to the user information need.

Especially, to describe and implement personalized e-Learning in the semantic web, there are at least three related research areas which contribute: open hypermedia, adaptive hypermedia, and reasoning for the semantic web. Open hypermedia [51], [52], [53], [54], [55], [56] is an approach to relationship management and information organization for hypertext-like structure servers. Key features are the separation of relationships and content, the integration of third party applications, and advanced

hypermedia data models allowing, e.g., the modelling of complex relationships. Adaptive hypermedia has been studied normally in closed worlds, i.e. the underlying document space / the hypermedia system has been known to the authors of the adaptive hypermedia system at design time of the system. To open up this setting for dynamic document or information spaces, approaches for so called open corpus adaptive hypermedia systems have been discussed [57], [58]. The relation of adaptive hypermedia and open hypermedia has for example been discussed in [59]. Finally, several works [57], [60], [61], [62], [63], [64] exploring the usefulness of Semantic Web query and rule languages for an e-learning environment.

Several challenges have to be faced in order to facilitate an open dynamic e-learning environment. *Incomplete* knowledge and *conflicts* are two main issues emerging in this context. The former is actually a result of the open world assumption, e.g., we do not expect an e-learning system to collect all the knowledge that it needs to adopt to a specific user needs/interests, while the latter stems either from the fact that different knowledge sources may present the same domain of discourse in different ways or from different conceptualization/cognition of individual. We should also stress that *personalization* and *recommendation* tasks further imply the management of incomplete knowledge, e.g. by considering the partially known preferences of a group of users that is closer to a specific one. Moreover, decisions, concerning personalization/recommendation, made at a time may become invalid later, after the consideration of a new piece of knowledge. A formal language with well-understood meaning to tackle these challenges is the Defeasible Logic. To the best of our knowledge, our work is the first *combining the advantages of Semantic Web with Defeasible Logic reasoning in the domain of e-Learning*.

In this work we elaborate on the design and implementation of a personalized rule-based e-learning system using Semantic Web technologies, so that remote agents can connect and query remote system resources. Our system can be extended to support intelligent agent communication and/or automatic ontology merging between different resource descriptions. In particular, a potential user of our system can navigate between personalized Web pages, view links, theory, examples and exercises according to their subjects, prerequisites or related subjects, as well as to user knowledge level. The learner knowledge level for each subject is deduced by the reasoning module. This module uses logic over online RDF descriptions, to conclude

or guess the user knowledge level, depending on learner answers to exercises on related subjects. The reasoning module makes also recommendation to the learner recommending the most appropriate content to focus his attendance.

Furthermore, descriptions of knowledge domains and educative material in RDF and XML, support the sharing of them between multiple educational centers, and description of learner attributes gives the ability to a learner attending lectures to multiple learning sources simultaneously, to share a common personalized user profile between them. Any education center can use its own educative material while using or extend parts of the material from other educational centers.

Additionally, for reasoning with inconsistent or incomplete information, which is a common phenomenon in these cases, we use defeasible logic. Its nonmonotonic behavior supports easy revision of system hypothesis about user knowledge on specific subjects when data is considered, without having inconsistencies. Defeasible rules were also used to describe make recommendation rules.

1.2 Thesis organization

In section 2 we provide background theory and related work. More specifically, we define the terms “e-learning” and “semantic web”. Subsequently, we present in detail defeasible logic and discuss its applicability to e-learning systems.

We also present all the related work in e-learning.

In section 3 we present the architecture of our implementation of e-learning. We show the e-learn RDFS schema we use, along with examples. Consequently, we describe in detail every module of the system – in particular, the reasoning module.

In section 4 a concrete example for the usage of our system is presented.

We conclude in section 5 with a brief discussion of the characteristics of our implementation and possible extensions of this approach.

2 Background Theory and Related Work

2.1 Adaptive E-learning

2.1.1 E-learning

The term e-learning originates from electronic learning and is often used as another term for web-based learning, online learning or distance learning. However, there are differences in the meaning of these terms. Thus, they cannot be used as interchangeable synonyms.

There are still discussions about the definition of the term e-learning. [1] states that e-learning represents just one part of the learning process. It has to be completed by e-teaching while the overall process is called e-education. However, the common meaning of e-learning includes the overall process as well and within this thesis only the term e-learning is used.

According to [2], e-learning is defined as follows:

"E-learning is mostly associated with activities involving computers and interactive networks simultaneously. The computer does not need to be the central element of the activity or provide learning content. However, the computer and the network must hold a significant involvement in the learning activity."

As the quotation mentions, e-learning implies the usage of computers for learning purposes. Concerning web-based learning, which is restricted to deliver the content over the World Wide Web (WWW), e-learning does not specify the transmission method. Online learning is connected to available learning materials in a computer environment, while not demanding a network. Distance learning is the "oldest" term and does not require the use of computers or networks. Distance learning includes the interaction between learners or students within a class over a distance for example, receiving the course materials by mail and learning at home [2] According to [3], e-learning has two main facets: the first is relative to using technology to support distance learning, the second is concerned with enhancing the learning experience with the help of information technology. In the first case the learners and the instructors can be physically separated (they never or rarely meet for face-to-face lectures, discussions, etc.) and thus all the learning process is technology-

mediated. In the second scenario the traditional learning approaches can be supported with complementary services, like online delivery of the learning materials, support for collaborative work, virtual communities etc. In many cases both aspects are simultaneously present. The goals of e-learning systems and the functionalities they offer can differ: the needs and goals of know-how transfer in an industrial company are quite different from the educational needs of a university. The functionalities can be broadly grouped in four categories: access to resources (data), specific e-learning services, common services and presentation. We intend to first list the main services and then discuss how these services must be modified with the introduction of small ubiquitous devices.

a) Resources

- Support of learning objects (LO) – any digital material, link to other resources, active element (like simulations etc.). Breaking the educational content into small pieces allows modularity and reusability of the content. These chunks of digital resources can be rearranged in modules, like lectures and courses. To facilitate this process they are usually described by additional metadata (as prescribed by the LOM standard).
- Support for Learning Metadata – Repositories for metadata can help to catalogue learning objects, and facilitate search and reuse.
- Quizzes and questions: lecturers can create a pool of questions and answers to be used both for automatic formal examination (summative assessment) or self-assessment of the students.

b) E-learning specific services

- Content management services – In general any e-learning system has the notion of Course and Lecture. A course can be composed by collection of resources: syllabus, one or many lectures, a structure for describing lecture sequence, forum, board, etc. A lecture is usually composed by many resources: presentation, exercise, additional material. All these components should be organized and accessed through a proper engine. There could be searchable directories of courses, programs, etc.

- Assessment - one of the main advantages of computer-supported learning is the automation of some important processes. The self-assessment is one example. The pool of questions/answers and a suitable engine allow automatic generation of different versions of tests and quizzes and also automatic checking of the results, evaluation of performance and comparison with others' results.
- Knowledge management (KM) – today most e-learning systems do not really support knowledge management services. KM in general aims at extraction, summarization and organization of explicit or tacit knowledge from data sources (e.g. Web, e-mails, chats, etc.). Application of KM to e-learning can be of vital importance in companies, while in university context (where most of the knowledge to be acquired by the students is explicit and formalized) it can be a useful but less relevant addition.
- Tools to support learners and tutors in managing their learning resources - some systems allow different users to have their own workspace and to upload personal resources (links, documents, notes, etc.), or to markup learning material.

c) Common services

- Support of different actors (students, teachers, tutors, administrator and guests), and integration with the company (university) information system. Different users typically have different levels of permissions. Unregistered users (guests) can have some (typically very limited) level of access to the platform.
- Collaboration tools: synchronous (chat rooms, shared applications, whiteboards, web-cast, audio- or video-conference, role games, simulations) and asynchronous (FAQ, forums, wikis, blogs, message/news boards, e-mail, mailing lists). Usually few different services are offered for communication between users of the system (learners, lecturers, tutors, mentors). Some of these tools are mainly meant to support cooperative work, while others aim at sharing and accessing important or topical information.

2.1.2 Adaptive Systems

An adaptive system adapts itself or another system to various circumstances. The process of adaptation is based on user's goals and preferences. These properties of the user are stored in a user model. The user model is hold by the system and provides information about the user like for example, knowledge, goals, etc. A user model gives the possibility to distinguish between users and provides the system with the ability to tailor its reaction depending on the model of the user [4].

In the context of e-learning, adaptive systems are more specialized and focus on the adaptation of learning content and the presentation of this content. According to [5], an adaptive system focuses on how the knowledge is learned by the student and pays attention to learning activities, cognitive structures and the context of the learning material. In Figure 1, the structure of an adaptive system, according to [4], is shown. The system intervenes at three stages during the process of adaptation. It controls the process of collecting data about the user, the process of building up the user model (user modelling) and during the adaptation process.

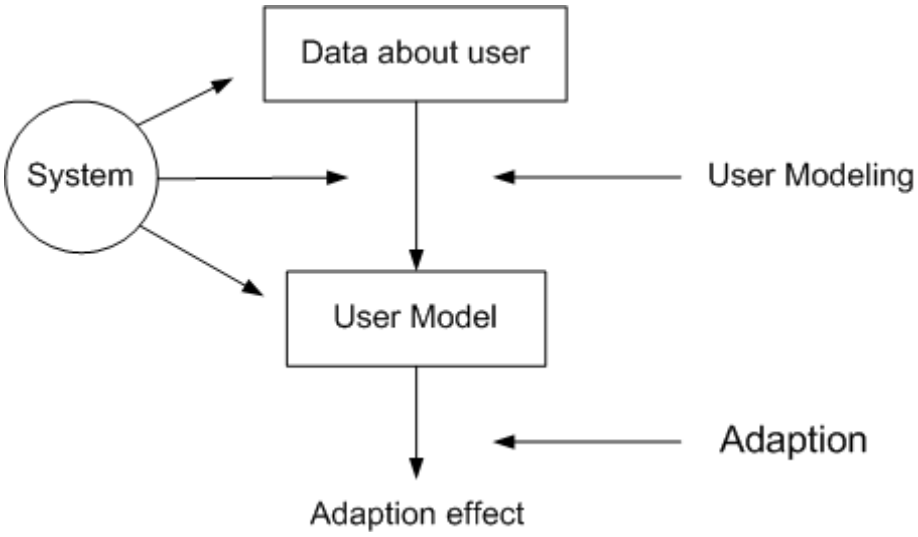

Figure 1: The Structure of an Adaptive System [4]

Beside this structure of an adaptive system, there exist several other models [6] lists the Benyon and Murray's model, the Oppermann's model and the Jameson's model. An adaptive system for e-learning is called an adaptive e-learning system. This restricts the purpose of an adaptive system to the field of e-learning.

An adaptive e-learning system is described, according to [7], as follows:

"An adaptive e-learning system is an interactive system that personalizes and adapts e-learning content, pedagogical models, and interactions between participants in the environment to meet the individual needs and preferences of users if and when they arise."

Thus, an adaptive e-learning system takes all properties of adaptive systems. To fit the needs for the application in the field of e-learning, adaptive e-learning systems adapt the learning material by using user models.

In the following section adaptive e-learning systems are described in more detail.

2.1.3 E-learning Theoretical Approaches

Theoretical approaches describe the different possibilities of adaptive instruction. Since adaptive instruction has a history of more than 100 years, the approaches are listed in chronological order beginning with the oldest approach.

Macro-adaptive Approach

Early attempts to personalize instruction to learners took place on the so-called macro-level. The students were grouped or classified by grades. This grouping resulted in a homogeneous evaluation of the learners and had minimal effects on the adaptation because the groups received different instructions very seldom. To better accommodate different student abilities, the macro-adaptive approach was invented in the early twentieth century, where the adaptation of instruction is concerned on a macro-level as well. Within the macro-adaptive approach, alternative instructions are computed, based on a few main components such as learning objectives, levels of detail and delivery system. The selection of the appropriate instruction is mostly based on the student's instructional goals, general abilities and achievement levels in the curriculum structure [8].

According to [9], the selection of instructions (i.e., activities) depends on learning objectives such as compensate students' weaknesses or developing new skills and student aptitudes. These aptitudes are categorized into three types, namely intellectual abilities and prior achievement, cognitive and learning styles and academic motivation and personality.

Aptitude-treatment Interaction Approach

The aptitude-treatment interaction (ATI) approach adapts instructional strategies to students' aptitudes. This strategy recommends different types of instructions for students with different characteristics. [5] lists the most important characteristics as intellectual abilities, cognitive styles, learning styles, prior knowledge, anxiety, achievement motivation, and self-efficiency.

ATI further offers the user full or partial control over the learning process. The user is able to control the style of the instruction or the way through the course. Three levels of control are defined, complete independence, partial control within a given task scenario and fixed tasks with control of pace. Studies have shown that the learner's aptitudes influence the learning result when offering different levels of control of the instruction to the learner. For example, students with low prior domain knowledge get better results if this control is limited [9].

Micro-adaptive Approach

Learning needs during instruction are used by the micro-adaptive approach to adapt the instruction. Theses needs are examined and an appropriate prescription is generated. Compared to the pretask measurements of the macro-adaptive and the ATI approach, the micro-adaptive approach is rather based on on-task measurements. The student behaviour and performance are observed by measuring e.g., response errors, response latencies and emotional states.

The first model for the micro adaptive approach is the idea of programmed instructions and was originally applied by Pressey in the year 1926. Through the usage of technology, a number of different micro-adaptive instructional models have been developed. These models differ from the programmed instruction idea by applying a specific model or learning theory. [8] lists following existing models: the mathematical model, the trajectory model, the Bayesian probability model and the structural and algorithmic approach.

According to [5], in case of the micro-adaptive approach adaptive e-learning is separated in two main processes, the diagnostic process and the prescriptive process. The first step (the diagnostic process) is used to characterize the learner by identifying the aptitudes or the prior knowledge and to formulate the task. Secondly, the

interaction between the learner and the task is optimized by adapting the learning content to the student's aptitudes and actual performance.

Constructivistic-collaborative Approach

The constructivistic pedagogical approach focuses on how an e-learning system can be integrated into the learning process. The learner takes an active role in the process of learning, where the knowledge is constructed by experiences in the specific knowledge domain according to the constructivistic learning theory.

Another major part of this approach is the employment of collaborative technologies, where the pedagogical approach of collaborative learning activities is integrated. Five characteristics of effective collaborative learning are identified by [10], namely participation, social behaviour, performance analysis, group processing and conversation skills and primitive interaction. To enable a learning success through collaborative technologies, these five characteristic should be available to the learner.

2.1.4 E-learning Types of Systems

This section describes types of systems with the help of the theoretical approaches introduced in Sub-section 2.1.3. Starting with macro-adaptive systems, intelligent tutoring systems and adaptive hypermedia system are presented.

Macro-adaptive Instructional Systems

As already mentioned in Sub-section 2.1.3, the macro-adaptive is the oldest approach where students were simply tracked by grades of ability tests. Macro-adaptive instructional systems where developed to tailor the instruction to the learner's abilities. [8] mentions the Burke plan, Dalton plan and Winnetka plan as early systems applying the macro-adaptive approach. Within these systems the students were able to go through the learning material at their own pace. In 1963, the Keller plan was developed at the Columbia University. The Keller plan is a macro-adaptive system where the instructional process was personalized for each student [5]. It was the first macro-adaptive system used at many colleges and universities all over the world. Until around 1985 several other macro-adaptive instructional systems were developed.

The examples macro-adaptive instructional systems given so far should demonstrate the history of adaptive e-learning and its application. These systems were applied in many schools and universities by providing only weak adaptation.

Computer-managed Instructional Systems (CMI)

An exceptional position takes the Computer-managed Instructional Systems (CMI). CMI systems provide many macro-adaptive instructional features offering the instructor possibilities to monitor and control the learning activities of the student. Further, CMI systems integrate features of micro-adaptive models (e.g., prediction of student learning needs). This makes CMI systems more effective concerning adaptive e-learning compared to pure macro-adaptive systems [8].

Intelligent Tutoring Systems

Intelligent Tutoring Systems (ITS) are adaptive instructional systems applying artificial intelligence (AI) techniques. The goal of ITS is to provide the benefits of one on-one instruction automatically and cost effectively [11]. As in other instructional systems, ITS consist of components representing the learning content, teaching and instructional strategies as well as mechanisms to understand what the student does or does not know. In ITS, these components are arranged into the expertise module, the student-modelling module, the tutoring module and a user interface module (see Figure 2) [12]. The expertise module evaluates the performance of the student and generates instructional content. The student modelling module represents the user's current knowledge and estimates his reasoning strategies and conceptions. This information is used by the ITS to determine, how the teaching process should continue. The tutoring module holds information for the selection of instructional material. This information describes how this material should be presented and when. The user interface module is the communication component that controls interaction between the student and the system.

ITS apply the micro-adaptive model since the decision about learning diagnosis and instructional prescriptions are generated during the task. Further, the combination with aptitude variables allows the expertise module to generate conditions for instructions based on the learner's characteristics. [5]

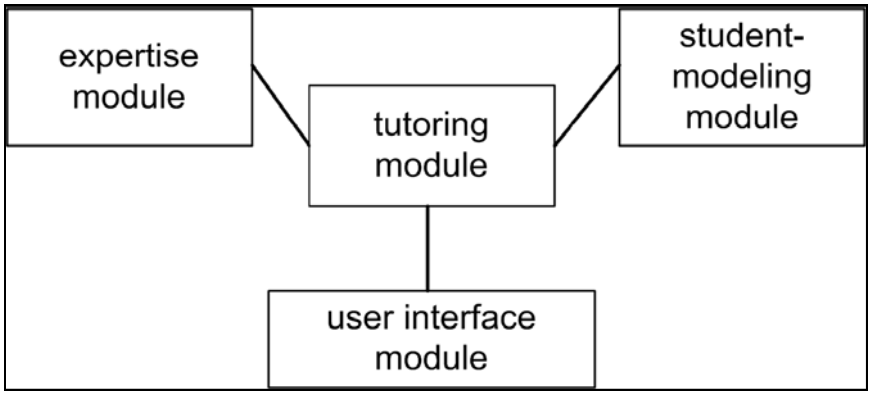

Figure 2: Components of an ITS [12]

A variety of AI techniques are used to represent the learning and teaching process. For example, some ITS systems capture topic related expertise in rules. This enables the ITS to generate problems on the fly, combine and apply rules to solve the problems, assess each learner's understanding by comparing the software's reasoning with them, and demonstrate the software's solutions to the participants. The development of an expert system that provides comprehensive coverage of the subject material remains the major problem for ITS.

Adaptive Hypermedia Systems

The development of Adaptive Hypermedia Systems (AHS) can be traced back to the early 1990s. The hypermedia model was extended by utilizing user models. AHS are inspired by ITS and try to combine adaptive instructional systems and hypermedia-based systems. [13] describes the definition of AHS as follows:

"By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user."

Thus, three criteria are satisfied by AHS. First, the system is based on hypertext or hypermedia. Second, a user model is applied and third, the system is able to adapt the hypermedia by using this user model. So far, AHS have been used in educational systems, e-commerce applications, information systems and help systems.

[14] distinguishes between two different types of AHS regarding adaptation methods. The first group, which deals with adaptive presentation, provides an adaptation of the content that can be presented in different ways or orders. The content can be adapted to various details, difficulty, and media usage to satisfy users with different needs, background knowledge, interaction style and cognitive

characteristics. Adaptation of the navigation is provided by the so called adaptive navigation support group. It can be implemented as direct guidance, adaptive hiding or re-ordering of links, link annotation, map adaptation, link disabling and link removal. [15]

The introduction of hypermedia and the Web has had a great impact on adaptive instructional systems but there are some limitations of AHS. According to [8], there was little empirical evidence for the effectiveness of AHS. [16] states, that if prerequisite relationships are omitted or are just wrong, the user may be directed to pages that cannot be understood by him because of necessary prior knowledge in this domain. Another drawback is that the same page might look different if this page is visited again. When the document is adapted to a developing user model, each time a user visits a particular page again, it may look different. This can cause confusion and loss of orientation for the user.

[16] concludes, that AHS has the potential to provide the users with freedom regarding the navigation through the instruction. Further the users can ensure that the presented learning material is relevant and can be understood by him.

Adaptive Educational Hypermedia Systems

A subtype of AHS are the adaptive educational hypermedia systems (AEHS). As implied in the name, AEHS are applied in the context of education. This type of systems is based on the AHS. The hyperspace of AEHS is kept very small since the documents are related to a specific topic. The focus of the user modelling is on the domain knowledge of the learner. [13]

According to [17], an AEHS consists of a document space, a user model, observations and an adaptation component. The document space belongs to the hypermedia system and is enriched with associated information (for example annotations, domain graphs or knowledge graphs). The user model stores, describes and infers information, knowledge and preferences about a user. Observations represent the information about the interaction between user and AEHS. These observations are used for updating the user model. Rules for adaptive functionality (if for example a document should be suggested for learning or to generate learning paths) and adaptive treatment (arrange links to further documents depending on the needs of a particular user) are provided by the adaptation component.

2.2 Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation".

This is an informal definition for the Semantic Web, given by Berner-Lee in the May 2001 American article "The Semantic Web" [18]. The Semantic Web is an extension of the World Wide Web in which both data and its semantic definition can be processed by computer programs. The next generation of the Web will combine existing Web technologies with knowledge representation formalisms in order to provide an infrastructure allowing data to be processed, discovered and filtered more effectively on the Web. A set of new languages organized in a layered architecture will allow users and applications to write and share information in a machine-readable way, and will enable the development of a new generation of technologies and toolkits. This layered architecture of the Semantic Web is often referred to as the Semantic Web tower [18].

2.2.1 The Semantic Web Tower

The Semantic Web tower (Figure 3) is a work in progress, where the layers are developed in a bottom-up manner. The so far defined languages in the bottom-up order include: XML, RDF, RDF Schema and Web Ontology Language OWL. The next step in the development of the Semantic Web will be the logic and proof layer. In the next sections we will briefly describe the basic layers of the tower.

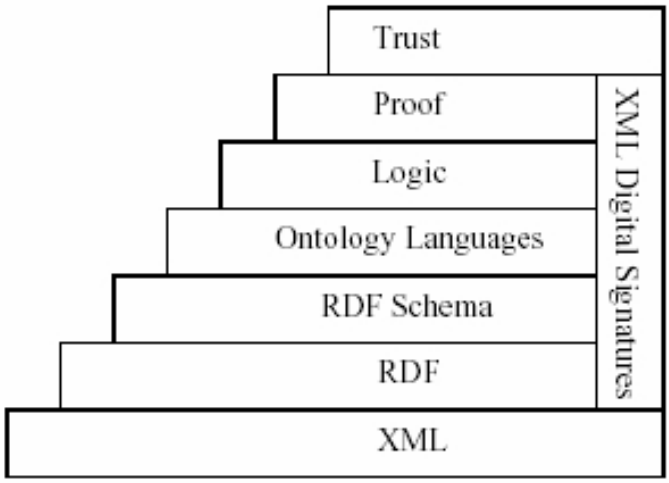

Figure 3: The Semantic Web Tower

The Representation Layer

The base language of the Semantic Web tower is XML [19], the dominant standard for data encoding and exchange on the web. Essentially the data represented in XML can be seen as labelled ordered trees. Such trees are encoded as XML documents with the parenthesis structure marked by tags. In the context of the Semantic Web XML will be used for encoding any kind of data, including the meta-data, describing the meaning of application data. Such meta-data will be described by the languages of the next layers of the Semantic Web tower.

Several mechanisms have been proposed for defining sets of XML documents. A standard one is the XML Schema language [20]. The elements of this language, called XML schemas, are XML documents. Thus, an XML schema is an XML document defining a (usually infinite) set of XML documents. This makes possible automatic validation of a given XML document d with respect to a given schema s, that is automatic check, whether or not d is in the set of documents defined by s.

The syntax of the languages of the next layers of the Semantic Web is also defined in XML. This means that the constructs of these languages are encoded as XML documents, and can be validated against the language definitions by standard validators. However, alternative syntaxes, better suitable for the human, can be provided and can be used as a starting point for defining the semantics of these languages.

The XML Namespaces [21] and Uniform Resource Identifiers [22] are important standards used in XML and therefore also in the upper layers of the Semantic Web, which are encoded in XML. They make it possible to create unique names for web resources. In the upper layers of the Semantic Web such names may be used as logical constants.

RDF and Ontology Languages

The idea of the Semantic Web is to describe the meaning of web data in a way suitable for automatic reasoning. This means that a descriptive data (meta-data) in machine readable form is to be stored on the web and used for reasoning. The simplest form of such description would assert relations between web resources. A more advanced description, called ontology, to be shared by various applications, would define concepts in the domain of these applications. Usually an ontology

defines an hierarchy of classes of objects in the described domain and binary relations, called properties.

The Semantic Web tower introduces language layers for describing resources and for providing ontologies:

The Resource Description Framework (RDF) [23] makes it possible to assert binary relations between resources (identified by URI's), and between resources and literals, which are strings. Such assertions have the form of triples, called statements. The elements of a triple are called subject, predicate (or property), and object. Usually they are URI references; the object may also be a literal. A triple can be seen as a kind of an atomic formula with a binary predicate. However, the vocabulary of RDF does not distinguish predicate symbols from logical constants: the predicates of RDF sentences may also appear as subjects and objects. In addition, RDF allows reification of a statement which can then for example be used as the subject of another statement.

For describing hierarchies of concepts RDF is extended with some built-in properties interpreted in a special way. The extension is called RDF Schema [24].

Statements of RDF Schema (RDFS) make it possible to define hierarchies of classes, hierarchies of properties and to describe domains and ranges of the properties. RDFS allows defining simple ontologies without using advanced features of RDF, like reification.

The emerging Web Ontology Language OWL [25] builds-up on RDFS introducing more expressive description constructs. However, as explained in [26], defining an expressive ontology language as a semantic extension of RDFS may lead to paradoxes. The design of OWL takes this into account. OWL has three increasingly expressive sublanguages: OWL Lite, OWL DL and OWL Full. *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. The complexity of computing ontology entailment is also lower for OWL Lite, than OWL DL. *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite

time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with *description logics* [27]. *OWL Full* is meant for users who want very high expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary.

The Top Layers

The top three layers of the Semantic Web tower are: the logic layer, the proof layer and the trust layer. The *logic* layer is used to enhance the ontology language further, and to allow writing application-specific declarative knowledge. The *proof* layer involves the actual deductive process, as well as the representation of proofs in Web languages and proof validation. Finally *trust* will merge through the use of *digital signatures*, and other kinds of knowledge, based on recommendations by agents we trust, on rating and certification agencies and on consumer bodies. Being located at the top of the pyramid, trust is a high-level and crucial concept: The Web will only achieve its full potential when users have trust in its operations (security) and the quality of the information provided.

2.2.2 The Role of the Rules

Rules constitute the next, not yet developed language level over the ontology languages in the Semantic Web tower. The arguments supporting the need of rules in the Semantic Web include the following:

- Rules appear naturally in many applications, e.g. business rules, policy specifications, service descriptions, database queries and many others. It is desirable to have a web rule language for expressing them for web applications.
- Rules provide a high-level description, abstracting from implementation details; they are concise and simple to write. They are well-known, understood

by non-experts, and well integrated in the mainstream Information Technology.

- The ontology languages are designed to describe concepts of the application domains, but are not sufficiently expressive for describing some aspects of applications, expressible in rule languages, e.g. composition of relations, extensively used in database query languages.

The ongoing discussion on rules for the Semantic Web seems to indicate that a family of rule languages may be needed rather than one language, since different applications require different kind of rules. The effort to define such languages and to enable Web-based interoperability between various rule systems and applications has been undertaken by the RuleML Initiative [28]. In general, the role that the rule systems are expected to have in the development of the Semantic Web is twofold:

(a) they can serve as extensions of, or alternatives to, description logic based ontology languages; and

(b) they can be used to develop declarative systems on top (using) ontologies.

Possible interactions between description logics and monotonic rule systems were studied in [29]. Based on that work and on previous work on hybrid reasoning [30], it appears that the best one can do at present is to take the intersection of the expressive power of Horn logic and description logics; one way to view this intersection is the Horn-definable subset of OWL.

2.2.3 The Role of Nonmonotonic Rule Systems

One of the issues that have recently attracted the concentration of the developers of the Semantic Web, is the nature of the rule systems that should be employed in the logic layer of the Semantic Web tower. Monotonic rule systems have already been studied and accepted as an essential part of the layered development of the Semantic Web. Nonmonotonic rule systems, on the other hand, seem also to be a good solution, especially due to their expressive capabilities. The basic motives for using such systems are:

Reasoning with Incomplete Information: [31] describes a scenario where business rules have to deal with incomplete information: in the absence of certain information some assumptions have to be made which lead to conclusions not supported by

classical predicate logic. In many applications on the Web such assumptions must be made because other players may not be able (e.g. due to communication problems) or willing (e.g. because of privacy or security concerns) to provide information. This is the classical case for the use of nonmonotonic knowledge representation and reasoning [32].

Rules with Exceptions*:* Rules with exceptions are a natural representation for policies and business rules [33]. Priority information is often implicitly or explicitly available to resolve conflicts among rules. Potential applications include security policies [34][35] , business rules [31] , personalization, brokering, bargaining, and automated agent negotiations[37].

Default Inheritance in Ontologies*:* Default inheritance is a well-known feature of certain knowledge representation formalisms. Thus it may play a role in ontology languages, which currently do not support this feature. [38] presents some ideas for possible uses of default inheritance in ontologies.

The following example is used to represent default inheritance in ontologies: Elephants are grey, with the exception of the royal elephants, which are white. We can restate the previous statement by saying that:

- Elephants are grey, except for royal elephants.
- Royal elephants are white.
- All royal elephants are elephants.

By applying a strict form of inheritance we should infer that any instance of the class royal elephant should be grey because it is a subclass of the class elephant.

However, we know that the property, colour, should be filled with the value, white, for any instance of the class royal elephant. This situation leads naturally to the idea of inheritance by default. We can model inheritance by default by means of non–classical logic. For instance, the above statement can represented in Default Logic as:

$$\frac{elephant(x) : \neg royalElephant(x)}{grey(x)}$$

A natural way of representing default inheritance is rules with exceptions, plus priority information. Thus, nonmonotonic rule systems can be utilized in ontology languages.

Ontology Merging: When ontologies from different authors and/or sources are merged, contradictions arise naturally. Predicate logic based formalisms, including all current Semantic Web languages, cannot cope with inconsistencies.

Some of the mismatches that may occur when someone tries to crate a single ontology by merging two different ontologies with overlapping parts are:

- Same concepts are represented by different names (synonym terms); e.g. term "car" in one ontology and term " automobile " in another ontology.
- The same term is used with different meaning (homonym terms); e.g. term "conductor" has different meaning in music domain than in electrical engineering domain.
- Values in ontologies may be encoded in different formats; e.g. distance may be described as miles or kilometres, or date may be represented as "dd/mm/yyyy" or as " mm-dd-yy"
- Mismatch between part of the domain that is covered by the ontology, or the level of detail to which that domain is modelled, e.g. one ontology might model cars but not trucks. Another one might represent trucks but only classify them into a few categories.

If rule-based ontology languages are used (e.g. DLP [29]) and if rules are interpreted as defensible (that is, they may be prevented from being applied even if they can fire) then we arrive at nonmonotonic rule systems. A sceptical approach, as adopted by defensible reasoning, is sensible because does not allow for contradictory conclusions to be drawn. Moreover, priorities may be used to resolve some conflicts among rules, based on knowledge about the reliability of sources or on user input. Thus, nonmonotonic rule systems can support ontology integration.

2.2.4 Semantic Web Query Languages

Several ontology query languages have been proposed during the last 6 years. They differ in: a) the ontology/metadata standard for which the language has been

proposed, b) the data model used for capturing the generated description bases and ontologies and the language of origin on which the query language has been based, c) the ability of the language to support functional composition of queries, d) their orthogonality, which indicates whether the language permits any kind of data as input and output of queries and e) their generality, i.e., whether the language exploits all the primitives of the ontology/metadata model. Bellow, we focus on RDF query languages.

ICS-FORTH RQL

RQL [74], developed in the context of the EU projects C-Web (IST-1999-13479) and MesMuses (IST-2001- 26074), is a typed, declarative query language for querying RDF description bases following a functional approach a la OQL. It is defined by a set of basic queries and iterators, which can be used to build new ones through functional decomposition. RQL relies on a formal graph model that enables the interpretation of superimposed resource descriptions by representing properties as self-existent individuals and introducing a graph instantiation mechanism that permits multiple classification of resources. It adapts the functionality of semi-structured or XML query languages to the peculiarities of RDF (i.e., labels on both graph nodes and edges, taxonomies of labels) but, foremost, it extends this functionality by uniformly querying both resource descriptions and (meta)schemas. In particular, the novelty of RQL lies in its ability to smoothly combine ontology and data querying while exploiting - in a transparent way - the taxonomies of labels and multiple classification of resources. Thus, users are able to query resources described according to their preferred ontology, while discovering in the sequel how the same resources are also described using another classification ontology (schema). RQL can compose schema paths to perform more complex ontology navigation, a kind of query not expressible in existing languages with ontology querying capabilities, while it supports generalized path expressions featuring variables on both labels for nodes (i.e., classes) and edges (i.e., properties). Furthermore, it features set-based queries and supports XML Schema data types, grouping primitives, aggregate functions and arithmetic operations on data values.

TRIPLE

TRIPLE[75] language is an RDF query, inference, and transformation language, developed as a joint work by Stefan Decker (Stanford University Database Group) and Michael Sintek (DFKI GmbH Kaiserslautern, Knowledge Management Department and Stanford University Database Group). TRIPLE's layered and modular nature, based on Horn Logic and F-Logic, aims to support applications in need of RDF reasoning and transformation, i.e., to provide mechanisms to query web resources in a declarative way. However, contrary to many other RDF query languages, TRIPLE allows the semantics of languages on top of RDF, such as RDF Schema and Topic Maps, to be defined with rules, instead of supporting the same functionality with built-in semantics. Wherever the definition of language semantics is not easily possible with rules (e.g., DAML+OIL), TRIPLE provides access to external programs, like description logics classifiers. Thus, two different kinds of layers are supported: syntactical extensions of Horn Logic to support basic RDF constructs, like resources and statements, and modules for semantic extensions of RDF, like RDF Schema], OIL [and DAML+OIL, implemented either directly in TRIPLE or via interaction with external reasoning components, such as a DL classifier. In particular, TRIPLE provides native support for resources and namespaces, abbreviations (e.g., isa:=rdf:SubClassOf), models (sets of RDF statements), reification and rules with expressive bodies (full First Order Logic syntax). TRIPLE also allows Skolem functions, which, when used in rules, can be used to transform one or several models (i.e., a set of RDF statements) into a new one, a functionality especially useful for ontology mapping or integration. Furthermore, instead of subject, predicate or object definitions, TRIPLE permits the usage of path expressions. For example, we can define (horn) rules that search for documents with a specified subject. TRIPLE provides a human-readable ASCII-syntax, as well as an RDF-based syntax for exchanging queries and rules, e.g., between communicating agents.

SPARQL

SPARQL[76] is a Semantic Web candidate recommendation presently undergoing standardization by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium. An RDF graph is a set of triples; each triple consists of a subject, a predicate and an object. These triples can come from a variety of sources. For instance, they may come directly from an RDF document; they may be inferred

from other RDF triples; or they may be the RDF expression of data stored in other formats, such as XML or relational databases. The RDF graph may be virtual, in that it is not fully materialized, only doing the work needed for each query to execute.

SPARQL is a query language for getting information from such RDF graphs. It provides facilities to:

- extract information in the form of URIs, blank nodes and literals.
- extract RDF subgraphs.
- construct new RDF graphs based on information in the queried graphs.

As a data access language, it is suitable for both local and remote use.

2.2.5 Semantic Web Database Storage

The necessity for ontology building, annotating, integrating and learning tools is uncontested. However, the sole representation of knowledge and information is not enough. Human information consumers and web agents have to use and query ontologies and the resources committed to them, thus the need for ontology storage and querying tools arises. However, the context of storing and querying knowledge has changed due to the wide acceptance and use of the Web as a platform for communicating knowledge. New languages for querying (meta)data based on web standards (e.g., XML, RDF) have emerged to enable the acquisition of knowledge from dispersed information sources, while the traditional database storage techniques have been adapted to deal with the peculiarities of the (semi)structured data on the web.

ICS-FORTH RDFSuite

The ICS-FORTH RDFSuite [68], [69], partially supported by EU projects C-Web and MesMuses (IST-2001- 26074), is a suite of tools for RDF metadata management, addressing the need of RDF metadata processing for large-scale Web-based applications. It consists of tools for parsing, validating, storing and querying RDF descriptions, namely the Validating RDF Parser (VRP), the RDF Schema Specific DataBase (RSSDB) and the RDF Query Language (RQL). RSSDB is a persistent tool for loading resource descriptions in an object-relational DBMS (e.g.,

PostgresSql) by exploiting the available RDF schema knowledge. It preserves the flexibility of RDF in refining schemas and/or enriching descriptions at any time whilst it can be customized in several ways (as opposed to triple-based repositories) according to the specificities of both the manipulated RDF descriptions (i.e., schemas) and the underlying RDF application queries. Its main goal is the separation of RDF schema information from data information, as well as the distinction between unary and binary relations holding the instances of classes and properties. Querying of stored RDF descriptions is accomplished by the query module, which implements the RQL language for performance reasons, the module pushes as much as possible query evaluation to the underlying DBMS, while benefiting from robust SQL3 query engines and DB indices. The RQL module is easy to integrate with web application servers and it is easy to couple with other commercial ORDBMS.

Sesame

Sesame [70], [71], [72], an RDF Schema-based Repository and querying facility, is being developed by Administrator Nederland as one of the key deliverables in the European IST project On-To-Knowledge. It is a system consisting of a repository, a query engine and an administration module for adding and deleting RDF data and Schema information. It supports expressive querying of RDF data and schema (ontology) information, using the RQL query language and understands the semantics of most of the RDF Schema classes and properties. Thus, it supports the basic inference needed for supporting RDF Schema, such as transitivity of subClassOf- and subPropertyOf-properties. The RQL implementation of Sesame is slightly different from the ICS-FORTH RDFSuite's, since the interpretation of the RDF Schema differs in the two cases and the Sesame's query engine does not support all features of RQL. To facilitate querying, Sesame supports the storage of large quantities of RDF and RDF Schema information. The RDF is parsed using the SiRPAC parser, and stored in the Object-Relational DBMS PostgreSQL. A public demo server running Sesame is available for experimentation.

Jena

Developed by the Hewlett-Packard Company, Jena [73] is a collection of RDF tools written in Java that includes: a Java model/graph API, an RDF Parser (supporting an N-Triples filter), a query system based on RDQL , support classes for

DAML+OIL ontologies and persistent/in-memory storage on BerkeleyDB or various other storage implementations. Due to its storage abstraction, Jena enables new storage subsystems to be integrated. To facilitate querying, Jena provides statement-centric methods for manipulating an RDF model as a set of RDF triples and resource-centric methods for manipulating an RDF model as a set of resources with properties, as well as built-in support for RDF containers. Jena contains Joseki RDF server, a server accepting SOAP and HTTP requests to query RDF resources. Latest version of Jena and Joseki support SPARQL

2.3 Defeasible Logic

2.3.1 Nonmonotonic Reasoning

One of the issues that have recently attracted the concentration of the developers of the Semantic Web is the nature of the rule systems that should be employed in the logic layer of the Semantic Web tower. Monotonic rule systems have already been studied and accepted as an essential part of the layered development of the Semantic Web. Nonmonotonic rule systems, on the other hand, seem also to be a good solution, especially due to their expressive capabilities.

Nonmonotonic reasoning is a subfield of Artificial Intelligence trying to find more realistic formal models of reasoning than classical logic. In common sense reasoning one often draws conclusions that have to be withdrawn, when further information is obtained. Thus, the set of conclusions does not grow monotonically with the given information. The latter phenomenon, nonmonotonic reasoning methods try to formalize.

In a monotonic logic system, given a collection of facts D that entail some sentence s (s is a logical conclusion of D), for any collection of facts D' such that $D \subseteq D'$, D' also entails s. In other words: s is also a logical conclusion of any superset of D.

In a nonmonotonic system, the addition of new facts can reduce the set of logical conclusions. So, if s is a logical conclusion of D, it is not necessarily a conclusion of any superset of D. Two of the basic characteristics of nonmonotonic systems are: adaptability (ability to deal with a changing environment), and the ability to reason under conditions of uncertainty. In other words, such systems are capable of

adding and retracting beliefs as new sets of information is available, and reasoning with an incomplete set of facts.

Defeasible logic, which was introduced by Donald Nute [77] is a representative language of nonmonotonic reasoning. In general, a defeasible theory (a knowledge base in defeasible logic) consists of five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation.

Facts are indisputable statements, for example, "Tweety is an emu". Written formally, this would be expressed as:

$$emu\ (tweety)$$

Strict Rules are rules in the classical sense: whenever the premises are indisputable (e.g., facts) then so is the conclusion. An example of a strict rule is "Emus are birds". Written formally:

$$emu\ (X) \rightarrow bird\ (X)$$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is "Birds typically fly"; written formally:

$$bird\ (X) \Rightarrow flies\ (X)$$

The idea is that if we know that something is a bird, then we may conclude that it flies, unless there is other, not inferior, evidence suggesting that it may not fly.

Defeaters are rules that cannot be used to draw any conclusions. Their only use is to prevent some conclusions. In other words, they are used to defeat some defeasible rules by producing evidence to the contrary. An example is "If an animal is heavy then it might not be able to fly". Formally:

$$heavy\ (X) \rightsquigarrow \neg flies\ (X)$$

The main point is that the information that an animal is heavy is not sufficient evidence to conclude that it does not fly. It is only evidence that the animal may not be able to fly. In other words, we do not wish to conclude ¬flies (X) if heavy (X); we simply want to prevent a conclusion flies (X).

The superiority relation among rules is used to define priorities among rules, i.e., where one rule may override the conclusion of another rule. For example, given the defeasible rules

$$r:\ bird\ (X) \Rightarrow flies\ (X)$$

s: brokenWing (X) ⇒ ¬flies (X)

which contradict one another, no conclusive decision can be made about whether a bird with broken wings can fly. But if we introduce a superiority relation > with s > r, with the intended meaning that s is strictly stronger than r, then we can indeed conclude that the bird cannot fly.

Notice that a cycle in the superiority relation is counterintuitive. In the above example, it makes no sense to have both r > s and s > r. Consequently, we focus on cases where the superiority relation is acyclic.

Another point worth noting is that, in Defeasible Logic, priorities are local in the following sense: two rules are considered to be competing with one another only if they have complementary heads. Thus, since the superiority relation is used to resolve conflicts among competing rules, it is only used to compare rules with complementary heads; the information r > s for rules r, s without complementary heads may be part of the superiority relation, but has no effect on the proof theory.

A more formal definition of Defeasible Logic and a proof theory are given in the next section.

2.3.2 Formal Definition

In this thesis we restrict attention to essentially propositional Defeasible Logic. Rules with free variables are interpreted as rule schemas, that is, as the set of all ground instances. If q is a literal ~q denotes the complementary literal (if q is a positive literal p then ~q is ¬p; and if q is ¬p, then ~q is p).

Rules are defined over a language (or signature) Σ, the set of propositions (atoms) and labels that may be used in the rule. In cases where it is unimportant to refer to the language of D, Σ will not be mentioned.

A rule r: A(r) $\rightsquigarrow$ C(r) consists of its unique label r, its antecedent A(r) (A(r) may be omitted if it is the empty set) which is a finite set of literals, an arrow $\rightsquigarrow$ (which is a placeholder for concrete arrows to be introduced in a moment), and its head (or consequent) C(r) which is a literal. In writing rules we omit set notation for antecedents, and sometimes we omit the label when it is not relevant for the context. There are three kinds of rules, each represented by a different arrow. Strict rules use $\rightarrow$, defeasible rules use $\Rightarrow$, and defeaters use $\rightsquigarrow$.

Given a set R of rules, we denote the set of all strict rules in R by Rs, the set of strict and defeasible rules in R by Rsd , the set of defeasible rules in R by Rd , and the set of defeaters in R by Rdft . R[q] denotes the set of rules in R with consequent q.

A superiority relation on R is a transitive relation > on R. When r1 > r2, then r1 is called superior to r2, and r2 inferior to r1. Intuitively, r1 > r2 expresses that r1 overrules r2, should both rules be applicable. Typically we assume > to be acyclic (that is, the transitive closure of > is irreflexive).

A defeasible theory D is a triple (F, R, >) where F is a finite set of literals (called facts), R a finite set of rules, and > an acyclic superiority relation on R. D is called decisive if the atom dependency graph of D is acyclic.

2.3.3 Proof Theory

A conclusion of D is a tagged literal and can have one of the following four forms:

- + Δ q which is intended to mean that q is definitely provable in D.
- − Δ q which is intended to mean that we have proved that q is not definitely provable in D.
- +∂ q which is intended to mean that q is defeasibly provable in D.
- −∂ q which is intended to mean that we have proved that q is not defeasibly provable in D.

If we are able to prove q definitely, then q is also defeasibly provable. This is a direct consequence of the formal definition below. It resembles the situation in, say, default logic: a formula is sceptically provable from a default theory T = (W, D) (in the sense that it is included in each extension) if it is provable from the set of facts W.

Provability is defined below. It is based on the concept of a derivation in D = (F,R,>). A derivation is a finite sequence P = (P(1),…,P(n)) of tagged literals satisfying the following conditions (P(1..i) denotes the initial part of the sequence P of length i):

$+\Delta$: If $P(i+1) = +\Delta q$ then either

$q \in F$ or

$\exists r \in Rs[q]\ \forall a \in A(r): +\Delta a \in P(1..i)$.

Figure 4: Definite Provability in Defeasible Logic

That means, to prove $+\Delta q$ we need to establish a proof for q using facts and strict rules only. This is a deduction in the classical sense - no proofs for the negation of q need to be considered (in contrast to defeasible provability below, where opposing chains of reasoning must be taken into account, too).

To prove $-\Delta q$, i.e., that q is not definitely provable; q must not be a fact. In addition, we need to establish that every strict rule with head q is known to be inapplicable. Thus for every such rule r there must be at least one antecedent a for which we have established that a is not definitely provable ($-\Delta a$, Fig. 5).

$-\Delta$: If $P(i+1) = -\Delta q$ then

$q \notin F$ and

$\forall r \in Rs[q]\ \exists a \in A(r): -\Delta a \in P(1..i)$.

Figure 5: Definite Non-provability in Defeasible logic

It is worth noticing that this definition of nonprovability does not involve loop detection. Thus if D consists of the single rule $p \rightarrow p$, we can see that p cannot be proven, but Defeasible Logic is unable to prove $-\Delta p$.

$+\partial$: If $P(i+1) = +\partial q$ then either

(1) $+\Delta q \in P(1..i)$ or

(2) (2.1) $\exists r \in Rsd[q]\ \forall a \in A(r): +\partial a \in P(1..i)$ and

(2.2) $-\Delta \sim q \in P(1..i)$ and

(2.3) $\forall s \in R[\sim q]$ either

(2.3.1) $\exists a \in A(s): -\partial a \in P(1..i)$ or

(2.3.2) $\exists t \in Rsd[q]$ such that

$\forall a \in A(t): +\partial a \in P(1..i)$ and $t > s$.

Figure 6: Defeasible Provability in Defeasible Logic

To show that q is provable defeasibly ($+\partial q$ Fig. 6) we have two choices: (1) We show that q is already definitely provable; or (2) we need to argue using the

defeasible part of D as well. In particular, we require that there must be a strict or defeasible rule with head q, which can be applied (2.1). But now we need to consider possible "attacks", i.e., reasoning chains in support of ~q. To be more specific: to prove q defeasibly we must show that ~q is not definitely provable (2.2). Also (2.3) we must consider the set of all rules which are not known to be inapplicable and which have head ~q. Essentially each such rule s attacks the conclusion q. For q to be provable, each such rule s must be counterattacked by a rule t with head q with the following properties: (i) t must be applicable at this point, and (ii) t must be stronger than s. Thus each attack on the conclusion q must be counterattacked by a stronger rule.

The definition of the proof theory of Defeasible Logic is completed by the condition $-\partial$. It is nothing more than a strong negation of the condition $+\partial$.

$-\partial$: *If* $P(i+1) = -\partial q$ *then*

(1) $-\Delta q \in P(1..i)$ *and*

(2) (2.1) $\forall r \in Rsd[q]\ \exists a \in A(r): -\partial a \in P(1..i)$ *or*

(2.2) $+\Delta \sim q \in P(1..i)$ *or*

(2.3) $\exists s \in R[\sim q]$ *such that*

(2.3.1) $\forall a \in A(s): +\partial a \in P(1..i)$ *and*

(2.3.2) $\forall t \in Rsd[q]$ *either*

$\exists a \in A(t): -\partial a \in P(1..i)$ *or* $t \not> s$.

Figure 7: Defeasible Non-provability in Defeasible Logic

To prove that q is not defeasibly provable, we must first establish that it is not definitely provable. Then we must establish that it cannot be proven using the defeasible part of the theory. There are three possibilities to achieve this: either we have established that none of the (strict and defeasible) rules with head q can be applied (2.1); or ~q is definitely provable (2.2); or there must be an applicable rule s with head ~q such that no possibly applicable rule t with head q is superior to s (2.3).

The elements of a derivation P in D are called lines of the derivation. We say that a tagged literal L is provable in D = (F, R, >), denoted $D \vdash L$, if there is a derivation in D such that L is a line of P. When D is obvious from the context we write $\vdash L$.

2.3.4 Dr-Prolog

Dr-Prolog [46] is a system for defeasible reasoning on the Web and the main characteristics of this system are the following:

- Its user interface is compatible with RuleML [28], the main standardization effort for rules on the Semantic Web.
- It is based on Prolog. The core of the system consists of a well-studied translation [47] of defeasible knowledge into logic programs under Well-Founded Semantics [48]. This declarative translation distinguishes our work from other implementations [49], [50].
- The main focus is on flexibility. Strict and defeasible rules and priorities are part of the interface and the implementation. Also, a number of variants are implemented (ambiguity blocking, ambiguity propagating, conflicting literals; see below for further details).
- The system can reason with rules and ontological knowledge written in RDF Schema (RDFS) or OWL. The latter happens through the transformation of the RDFS constructs and many OWL constructs into rules. Note, however, that a number of OWL constructs cannot be captured by the expressive power of rule languages.

As a result of the above, DR-Prolog is a powerful declarative system supporting (a) rules, facts and ontologies; (b) all major Semantic Web standards: RDF(S), OWL, RuleML; and (c) monotonic and nonmonotonic rules, open and closed world assumption, and reasoning with inconsistencies.

2.4 Using Defeasible Logic on E-Learning

Any e-learning system should focus on the needs of the learner, given his or her practical experience and further support a personalised learning process, empowering the user to choose his / her own learning pathway**,** As Jonassen, Mayes and McAleese [39] note, the environment should take the form of an open learning system which is “need driven, learner-initiated and conceptually and intellectually engaging.

An e-learning environment should fulfil an information or knowledge construction need of the learner. This should be based on the interests and experience of the user. We see the needs-driven approach as an essential feature of e-learning, and view hypertext technology as a means by which individual users can interact effectively within an environment.

An e-learning system should support personalised learning trajectories, which consider of individual experiences and build on a learner's prior knowledge. This means that the environment supports the pro-activity of the learner in building knowledge, by considering individual characteristics and helping the learner to integrate available knowledge – transforming information into knowledge. An open system should encourage the development of knowledge and skills that will enable learners to search find and process information adequately; it must facilitate the development of transfer abilities as well as a high level of autonomy in the learning process [39].

This description departs from the traditional use of technology in course delivery, which has followed an 'instructor- centred' approach. Traditional computer-based learning environments are often driven by prescriptive programs that allow the learner to input information; however the responses to that input are prescribed and predetermined. In essence they are "*closed*" systems. A good example of this approach would be the placing courseware on a web server to be accessed by remote students, which would suit the prescriptive pattern of a taught course. In our estimation, an e-learning environment should be "*open*"; that is to say it can be adapted by learners or trainers to the particular needs of learners, teams or groups of learners from different surroundings or cultures. It is modular in order to facilitate its adaptation, updating or its re-engineering. The today's needs call for an e-learning environment capable of supporting a dynamic learning process, concerning learners and instructors who *share* knowledge and both contribute to a shared cognition.

Several challenges have to be faced in order to facilitate an open dynamic e-learning environment. *Incomplete* knowledge and *conflicts* are two main issues emerging in this context. The former is actually a result of the open world assumption, e.g., we do not expect an e-learning system to collect all the knowledge that it needs to adopt to a specific user needs/interests, while the latter stems either from the fact that different knowledge sources may present the same domain of discourse in

different ways or from different conceptualization/cognition of individual. We should also stress that *personalization* and *recommendation* tasks further imply the management of incomplete knowledge, e.g. by considering the partially known preferences of a group of users that is closer to a specific one. Moreover, decisions, concerning personalization/recommendation, made at a time may become invalid later, after the consideration of a new piece of knowledge.

A formal language with well-understood meaning to tackle these challenges is the Defeasible Logic. Defeasible Logic can be used in e-learning to the following topics

Personalization/Adaptation/Recommendation Rules

In e-learning applications, there are cases that need to be expressed by rules, like rules trying to determine user knowledge on a subject, recommendation rules, or rules to adapt learning content on user needs. According to [36] key properties that executable specification languages and systems should possess include

- *Expressive power*: the language should be rich enough to represent the rules, and the main ways in which these rules interact with one another.
- *Naturalness of expression*: moreover, the representations should reflect the rules in a natural, transparent way. This property is crucial for the maintainability of e-commerce business rules, and for the simplification of update processes.
- *Declarativity*: the language should have clear semantics, and the meaning allocated to specifications should correspond to intuitive ideas. This property is crucial for making nonspecialists comfortable with the language, and thus for the success of the approach in practice.
- *Formality* is needed to be able to analyse the behaviour of the rules, identify anomalies, run hypothetical cases, etc.
- *Computational efficiency*: reasoning mechanisms are needed to run the specifications in acceptable time.

To fulfil these aims it is natural to look for languages and techniques from artificial intelligence, and in particular from the area of knowledge representation. The most fundamental and well known knowledge representation language is *predicate*

logic. It has been extensively studied, has clear semantics, and is supported by automated reasoning techniques. But it falls short as an appropriate basis for our purposes in electronic commerce on two accounts: its contra positive interpretation of rules, and its inability to reason with conflicts.

Given the rule "If a learner has enough knowledge on subject A then consider that he has enough knowledge on subject B", and the decision that user has not enough knowledge on subject B, predicate logic would conclude that user has not enough knowledge on subject A. But this use of the above rule is unlikely to be intended.

This problem may be overcome by using *rule-based declarative languages* which apply such rules in one direction only. The second difficulty, though, is more serious. Some rules can conclude base on some information,(e.g. comparing profile of Learner 1 with profile of Learner 2 and using collaborative filtering) that Learner 1 has enough knowledge on subject A, and some other rules can conclude that has not (e.g. Learner 1 answers to exercises relative to subject A)

Now obviously there is a conflict between the three rules. Its natural representation in predicate logic would 'collapse': it would sanction any conclusion (including, for example, granting a 100% discount to all customers).

Nonmonotonic reasoning [32] comprises knowledge representation approaches that deal with incomplete and conflicting information. This family incomplete and conflicting information. This family also includes rule-based approaches. Rule-based nonmonotonic approaches are appropriate for the modelling rules for e-eleanring because:

(1) E-learning rules can be naturally mapped to rules (rules in the *logical* sense).

(2) If two rules that can be applied lead to conflicting conclusions none of them fires. This behaviour is referred to as *scepticism*. It prevents the inference of contradictory conclusions, as would happen in predicate logic-based approaches.

(3) Often the outcome in (2) is unsatisfactory: even though two rules may lead to conflicting conclusions, one rule may be stronger than the other. This preference of one rule over another may be based on implicit principles (such as higher authority, recency, specificity (a rule about the specific case at hand should usually be considered stronger than a more general rule covering more cases) etc.) or explicit

preference formulated in the body of rules (for example, a rule may be declared to be an exception to another rule). Based on these reasons we propose the use of (logical) *rules and priorities* as the language to model and reason with e-learning rules

Defeasible logic integrates all these concepts. Its language consists of (strict and defeasible) rules and a superiority relation on the set of rules. Defeasible logic is sceptical and follows appealing principles of reasoning.

Compared to other nonmonotonic logics, defeasible logic has the additional very important advantage of its relatively low computational complexity, making it preferable for applications that use very large rule sets. According to Maher 7[45], inference in propositional defeasible logic has linear complexity, which is much lower than the computational complexity of sceptical default reasoning, sceptical autoepistemic reasoning and propositional circumscription, which is Π_2^p - complete.

Now we will mention an example where the defeasible logic can be used at personalizing an e-learning system. Suppose we have some exercise and students. Every exercise can belong to many subjects and can have many prerequisite subjects. The following rules will help to decide whether an exercise will be shown to a student.

r1: if the student has grade > 7 at one of the subjects of the exercise then the exercise will be shown to the student

r2: if the student has grade < 3 at one of the subjects of the exercise then the exercise will **not** be shown to the student

r3: if the student has an average grade > 8 at the prerequisite subjects then the exercise will be shown to the student

r4: if the student believes (tell it explicitly to the system) that he doesn't know at least one subject of the exercise then the exercise will not be shown to the student

r2 > r1

r3 > r2

r4 > r3

The above rules written in defeasible logic:

r1: rank(Student, Subject, Grade), Grade > 7, belong(Exercise, Subjedt) □ show(Exercise, Student)

r2: rank(Student, Subject, Grade), Grade < 3 , belong(Exercise, Subjedt) □ □(show(Exercise, Student))

r3: average_rank(Student, Exercise, Prerequisite_subjects,Grade), Grade>8 □ show(Exercise, Student)

r4: belives_unknown(Student, Subject), contains(Exercise, Subject) □ □(show(Exercise, Student))

r2 > r1

r3 > r2

r4 > r3

Suppose a student S1 has grade > 7 at a subject of the exercise EX1, grade <3 at another subject of the same exercise, the average grade of all the subjects is > 8 and the student believes the he knows all the subject of EX1 then the exercise will be shown to the student. This happens because r3, r2 and r1 fire but r3 is stronger than the other.

We describe a set of recommendation rules for our system, written in defeasible logic in 3.9.

Ontology merging

An E-learning system uses a pedagogical ontology to describe knowledge domains. A learner can be attended to lectures provided from different educational sources around the world and there is a need for the learner to have a common e-learning profile. So there is a need for automatically merging pedagogical ontologies describing similar knowledge domains. When ontologies from different authors and/or sources are merged, contradictions arise naturally. Predicate logic based formalisms, including all current Semantic Web languages, cannot cope with inconsistencies. If rule-based ontology languages are used and if rules are interpreted as defeasible (that is, they may be prevented from being applied even if they can fire) then we arrive at nonmonotonic rule systems. A sceptical approach, as adopted by defeasible reasoning, is sensible because it does not allow for contradictory conclusions to be drawn. Moreover, priorities may be used to resolve some conflicts among rules, based on knowledge about the reliability of sources or on user input. Thus, nonmonotonic rule systems can support ontology integration.

Now we will mention an example where the defeasible logic can be used when merging two ontologies.

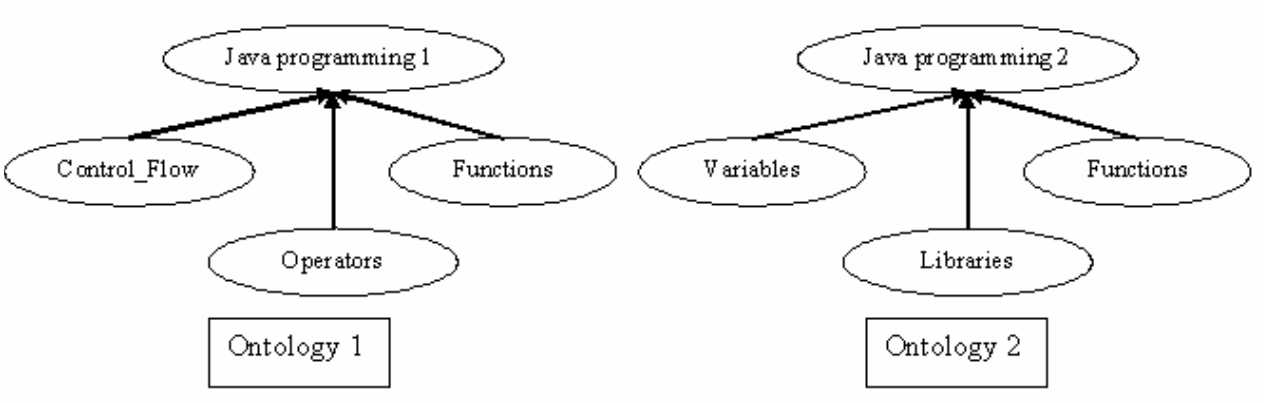

Figure 8: Ontology 1 and Ontology 2 from different universities

We have two ontologies (Ontology 1, Ontology 2) which are part of the collection of ontologies that are used at two different universities. Suppose a student (Learner_1) has attended Java programming lesson at both universities and has obtained grades at some subjects of the Java programming lesson in both universities e.g. at the first university: Control_Flow 5, Operator 8, Functions 9 and at the second university Variables 7, Libraries 6, Functions 3. If the ontologies shown in figure 8 merge, a conflict may arise at data level because the same student (Learner_1) has two different grades at the subject "Functions". This conflict can be solved by using rules of defeasible logic which help us decide which of the two grades is more "important". For example suppose the system has to decide whether an exercise that has as prerequisite the subject "Function" must be displayed to Learner_1 (an exercise is displayed to a student if his grades to all prerequisites are greater than 7). The problem arises because there exist two different grades for the same subject. If the rules of defeasible logic can conclude that one grade is more "important" than the other the system can decide if the exercise will be displayed.

2.5 Related Work

To describe and implement personalized e-Learning in the semantic web, there are at least three related research areas which contribute: open hypermedia, adaptive hypermedia, and reasoning for the semantic web. Open hypermedia is an approach to relationship management and information organization for hypertext-like structure servers. Key features are the separation of relationships and content, the integration of

third party applications, and advanced hypermedia data models allowing, e.g., the modelling of complex relationships. In open hypermedia, data models like FOHM (Fundamental Open Hypertext Model) [51] and models for describing link exchange formats like OHIF (Open Hyper Interchange format) [52] have been developed. The use of ontologies for open hypermedia has e.g. been discussed in [53]. Here, an ontology is employed that clarifies the relations of resources. On base of this ontology, inference rules can derive new hypertext relations. In [54] the open hypermedia structures are used as an interface to ontology browsing. The links at the user interface are transformed to queries over ontology. Thus links serves as contexts for particular user.

The question whether conceptual open hypermedia is the semantic web has been discussed in [55]. In [56], a metadata space is introduced, where the openness of systems and their use of metadata are compared. On the metadata dimension (x-axis), the units are the use of keywords, thesauri, ontologies, and description logic. The y-axis describes the openness dimension of systems starts from CD ROM / file system, Internet, Web, and ends with Open systems.

Adaptive hypermedia has been studied normally in closed worlds, i.e. the underlying document space / the hypermedia system has been known to the authors of the adaptive hypermedia system at design time of the system. As a consequence, changes to this document space can hardly be considered: A change to the document space normally requires the reorganization of the document space (or at least some of the documents in the document space). To open up this setting for dynamic document or information spaces, approaches for so called open corpus adaptive hypermedia systems have been discussed [57], [58]. The relation of adaptive hypermedia and open hypermedia has for example been discussed in [59].

[60] uses an ontology for adaptive functionality. Such an ontology can be derived using the "updated taxonomy of adaptive hypermedia technologies" in [57]. Reasoning over these distributed ontologies is enabled by the RDF-querying and transformation language TRIPLE. In Elena Project[61] there is a logic-based approach to educational hypermedia using TRIPLE, a rule and query language for the semantic web[62].

Related approaches in the area of querying languages for the semantic web can be found, e.g., in [63]. Here, a rule-based querying and transformation language for XML is proposed. A discussion of the interoperability between Logic programs and ontologies (coded in OWL or DAML+OIL) can be found in [64].

Blochl et al. [65] proposed an adaptive learning system which can incorporate psychological aspects of learning process into the user profile to deliver individualized learning resource. The user profile is placed in multi-dimensional space with three stages of the semantic decisions: cognitive style, skills and user type. However, both the means to acquire user's feedback and the algorithms to update user profile have not been addressed in the presentation.

SPERO [66] is a personalized e-learning system based on the IEEE Learning Technology Systems Architecture (LTSA). It could provide different contents for the foreign language learners according their interests and levels. The problem of SPERO system is that it is largely using questionnaires and e-surveys to build user profiles, which costs the users too much extra work.

In [67] the emerging theory of "Trialogical Learning" focus on the social processes by which learners collectively enrich/transform their individual and shared cognition. According to TL, knowledge creation activities rely heavily on the use, manipulation and evolution of shared knowledge artefacts externalizing a body of (tacit or explicit) knowledge. By representing their cognitive structures or Knowledge practices under the form of artefacts, individual learners can interact with themselves as well as with external tools (e.g., computers, information resources) to negotiate the meaning of concepts and signs embodied in these artefacts and thus reach a common understanding of the problem at hand. We could there- fore consider as cornerstone of trialogical learning the notion of shared objects of activity, a notion that is quite general to accommodate the requirements of various application contexts.

3 Implementation Architecture

3.1 Architecture Overview

Our system uses ontology models to describe knowledge domains (Pedagogical ontology), educative material (theory, exercises, examples, links) *(Content ontology)* and

the learner attributes relative with them *(user ontology)*. These models are described using RDF, as instances of the e-learn RDFS schema.

In order to describe a knowledge domain in an ontology model, it's possible to extend an existing knowledge model, to combine parts of several models, or to construct a new model, as instances of e-learn RDFS schema. Any RDF document using e-learn RDF schema is compatible to our-learning system and can be used from it without any changes. *User ontology* is stored permanently in RDF storage database.

The education material is described in XML documents. It is divided in *Tutorials*, every tutorial contains *Pages* and every page contains *document elements.*

E-learn Web servlet transforms e-learn XML documents to personalized Web pages, using a reasoning module over RDF descriptions. Each document element contained in a tutorial, can be presented or be omitted, according to the user knowledge level, giving the final personalized Web content. Any document element must belong to one of the following classes: *Theory, exercises (multiple choice* or *text), examples, links* and its content is described in XML format. A *document element* can be present to more than one *tutorials.*

The user of an e-learning application based on our system, can navigate between personalized Web pages, view links, theory, examples and exercises according to their subjects, prerequisites, or related subjects, as well as according to his knowledge level. The learner knowledge level for each subject is deduced by the *reasoning module*. This module uses logic over online RDF descriptions, to conclude or guess the user knowledge level, depending on learner answers to exercises on related subjects. It also uses *defeasible rules* to *recommend* to the user some of the visible document elements, asking him to focus his attendance. Learner profile information is stored to permanen*t RDF database storage*. Authorized clients can query the RDF database using *e-learn query servlet* and can update its content using *e-learn RDF storage Java servlet.* Reasoning module can query *e-learn servlets*, using the SPARQL query language.

Learner specific content stored in database is published to Semantic Web using *Semantic Web Publisher servlet.* So authorized users can retrieve learner specific information. The output of the local servlet can be used by any other remote installation of our system, or from local *reasoning module.* This feature gives to any educational unit using our system, the ability to retrieve information about students from other educational units. It also gives to the students the ability to be attended to personalized courses from different educational centers all over the world.

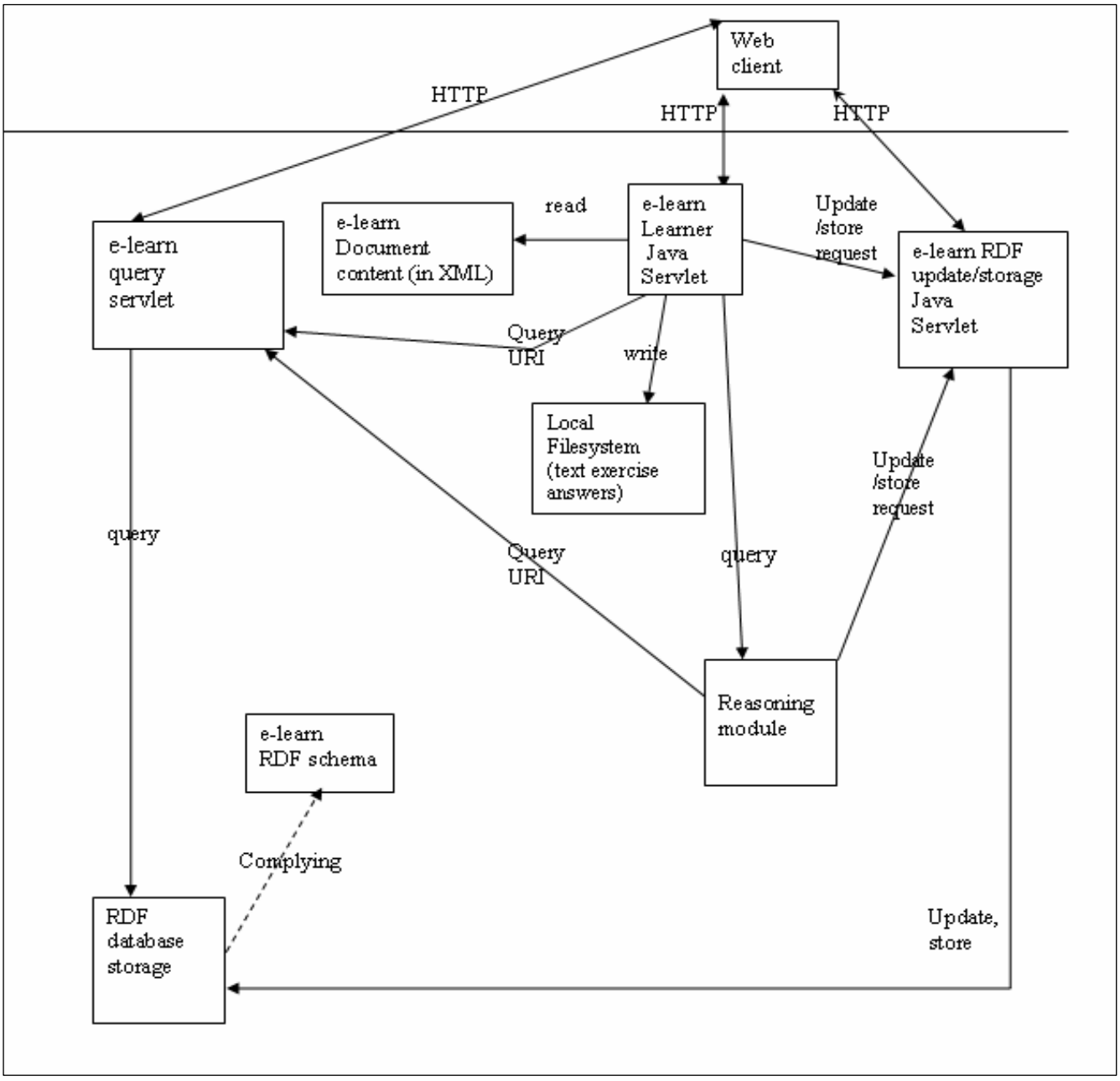

Figure 9: The Overall Architecture of our System

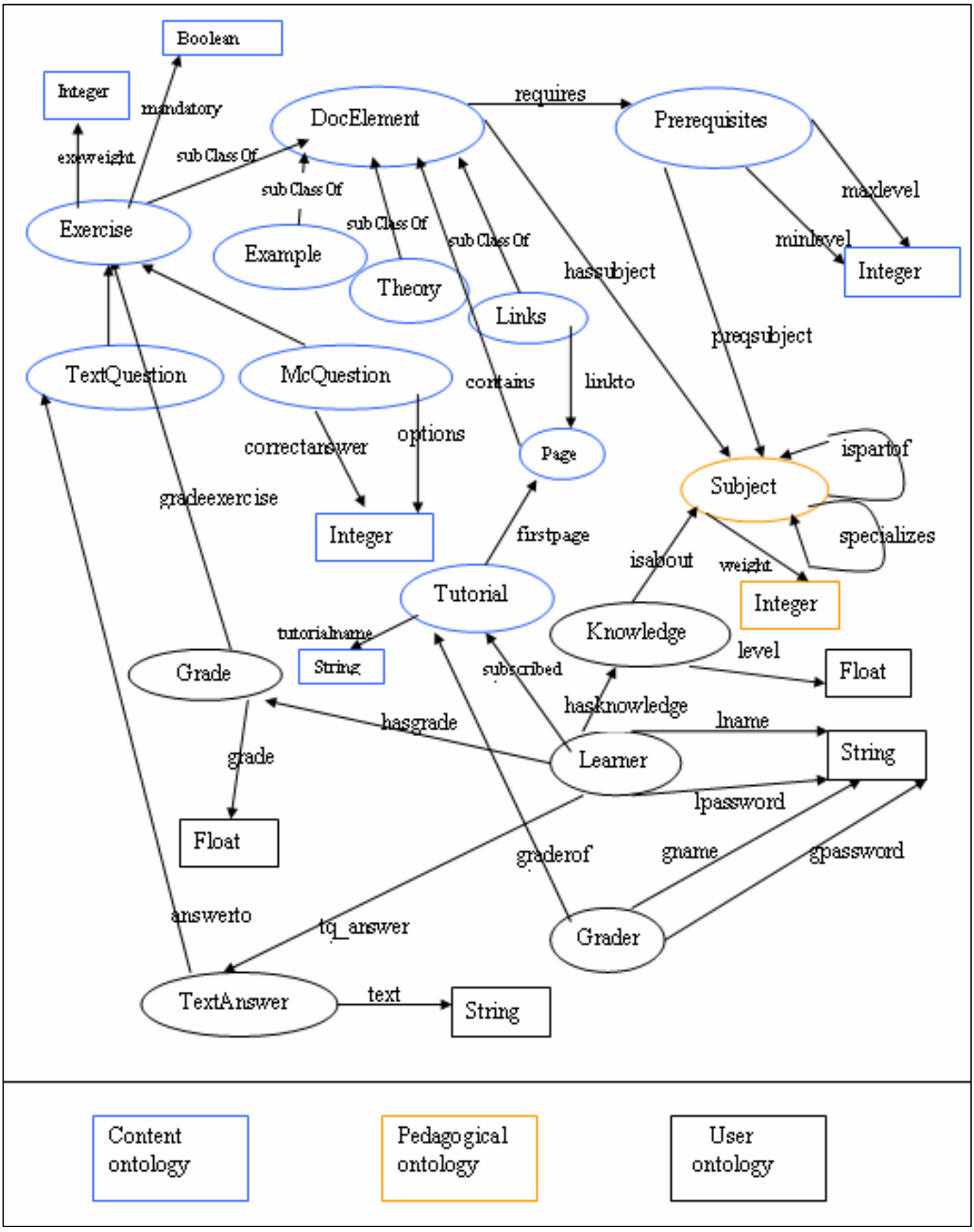

Figure 10: E-learn RDFS Schema

3.2.1 Pedagogical Ontology Examples

3.2.1.1 Programming

All elements in this figure are instances of "subject" class and all attributes are instances of "ispartof" Property of e-learn RDFS schema.

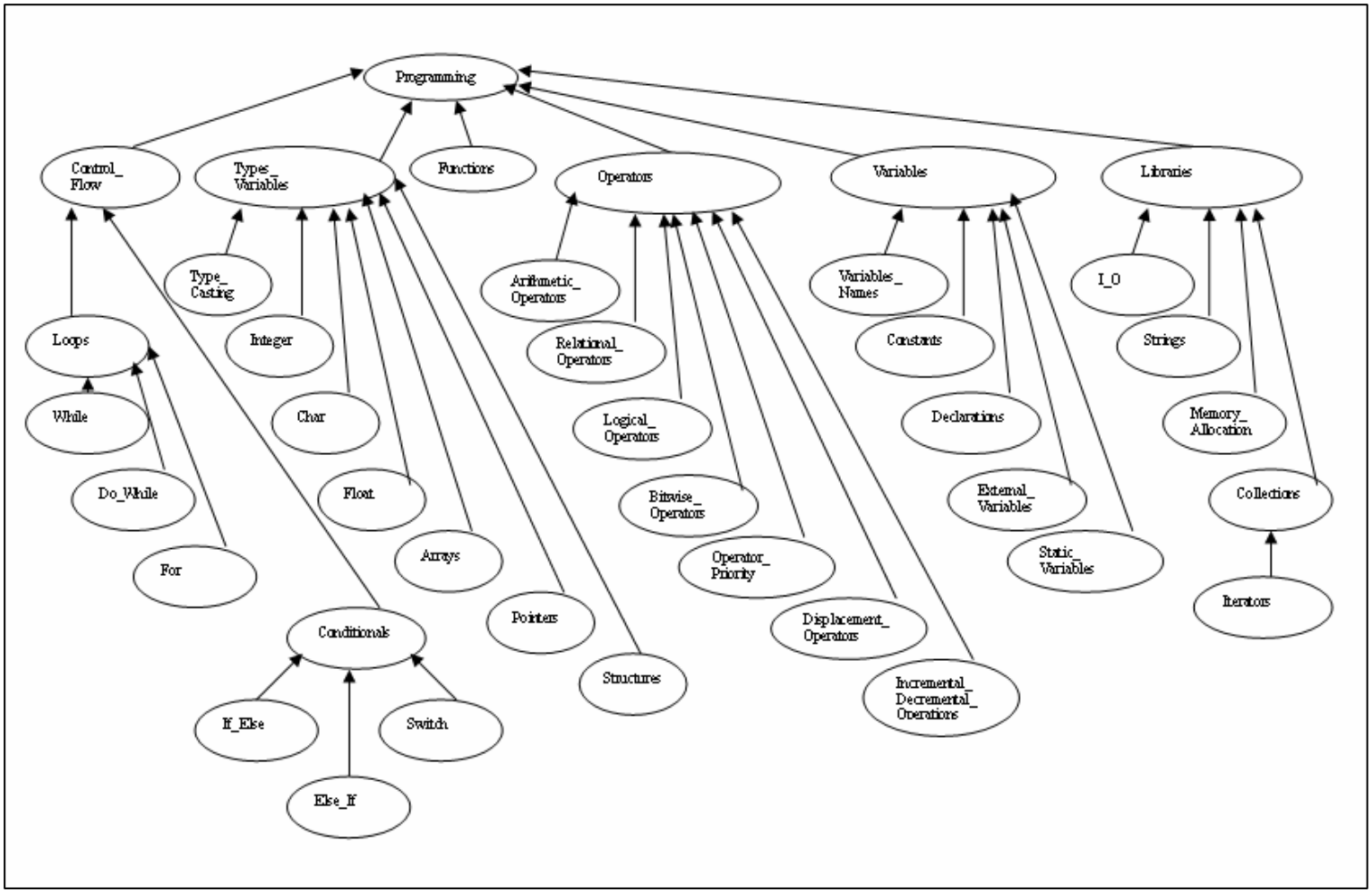

Figure 11: Programming

3.2.1.2 OOP Programming

All elements in this figure are instances of "Subject" class and all attributes are instances of "ispartof" Property of e-learn RDFS schema.

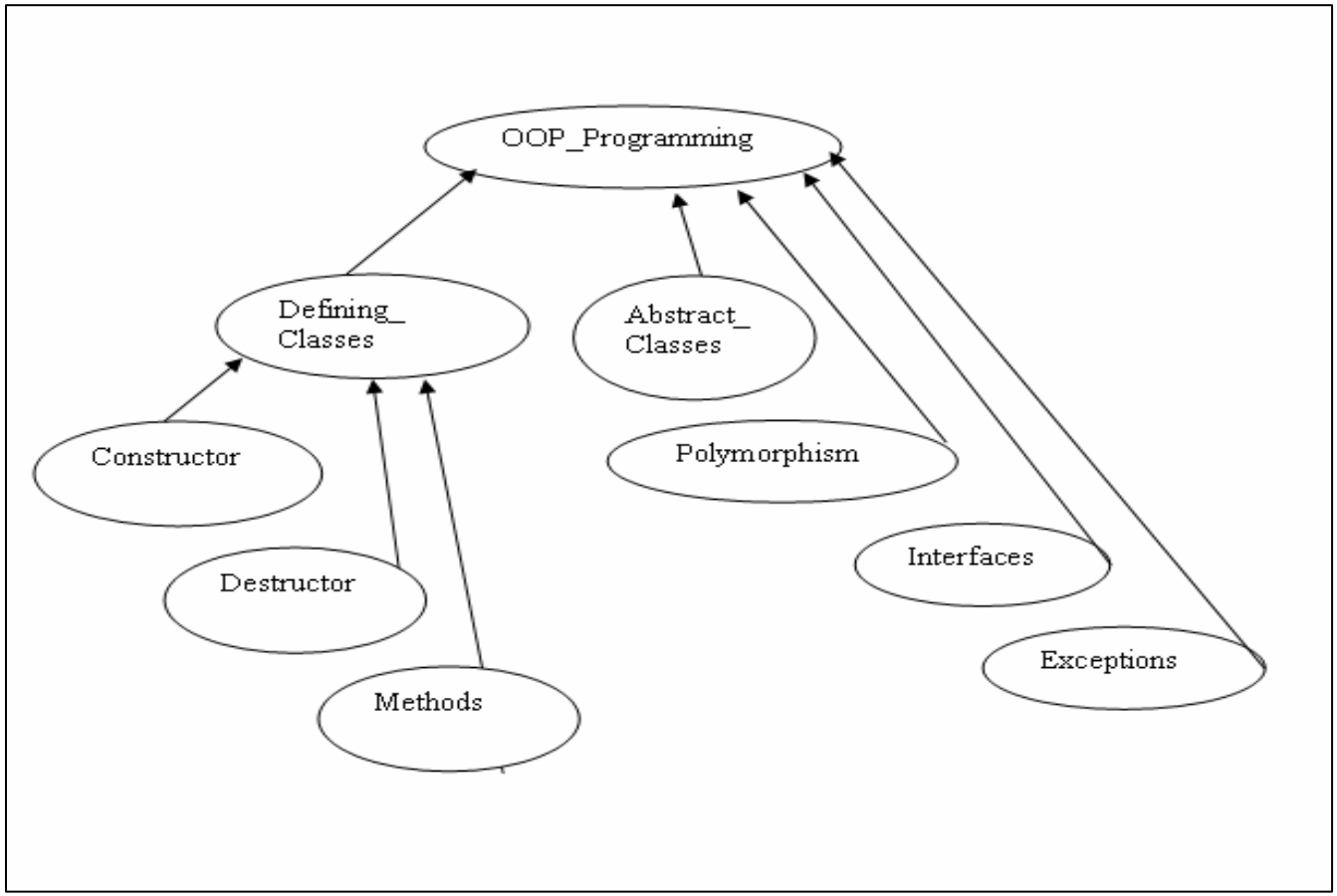

Figure 12: OOP Programming

3.2.1.3 C Programming

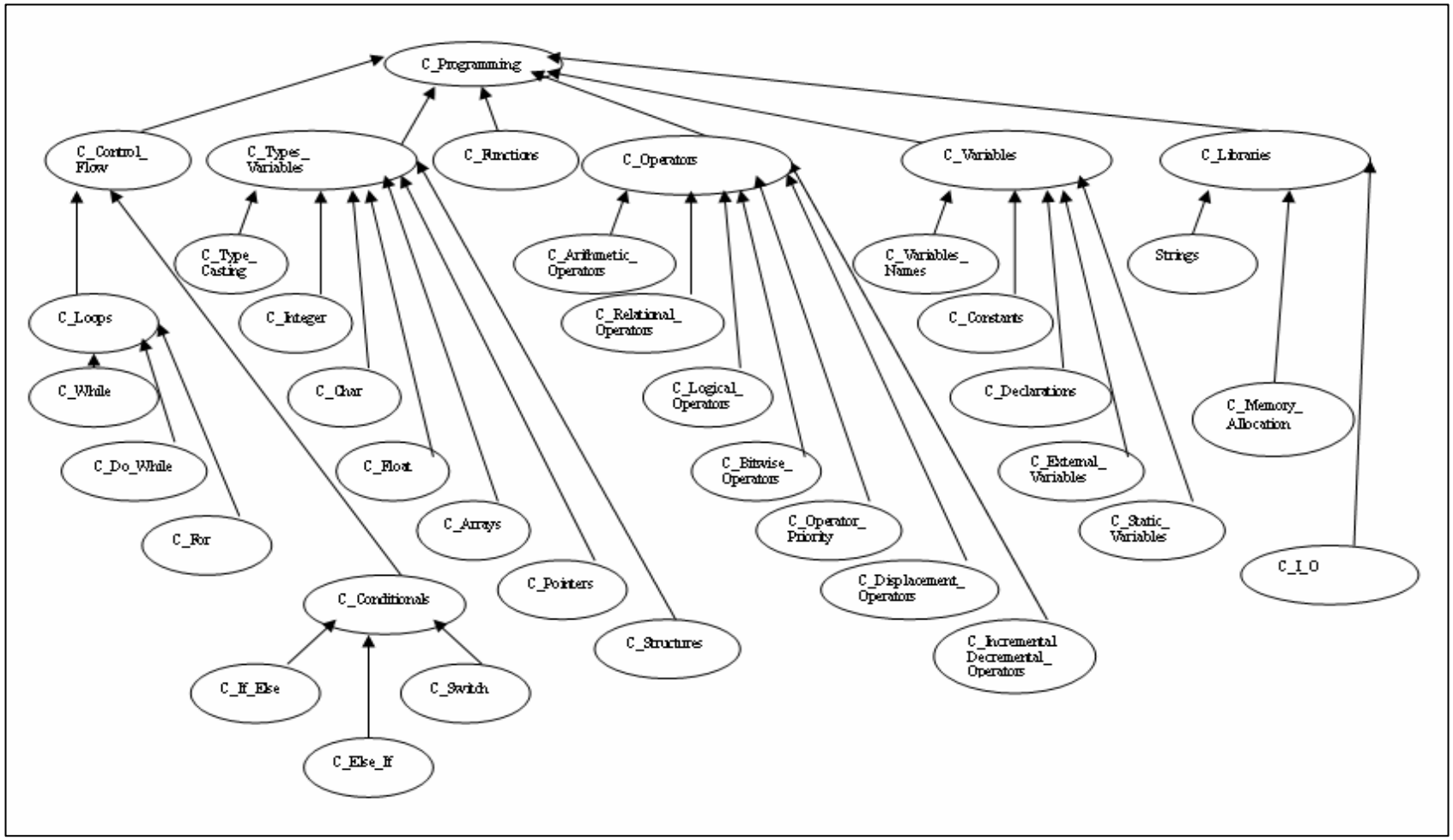

Figure 13: C Programming

All elements parts of “C_Programming” have an instance of the Property “specializes” connected to the corresponding element of “Programming» and are instances of “Subject” class of RDFS-schema

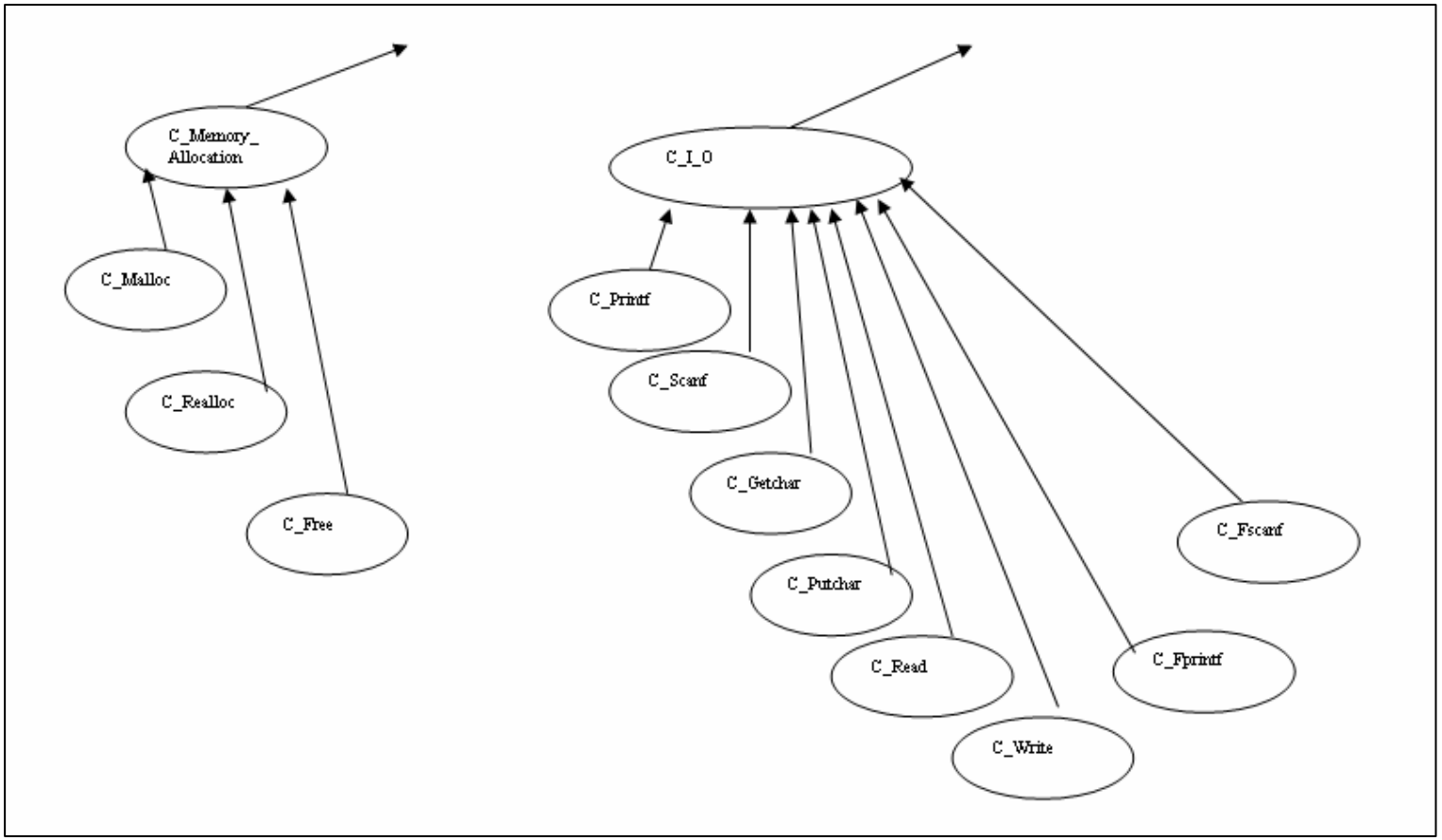

Figure 14: C Programming

3.2.1.4 C++ Programming

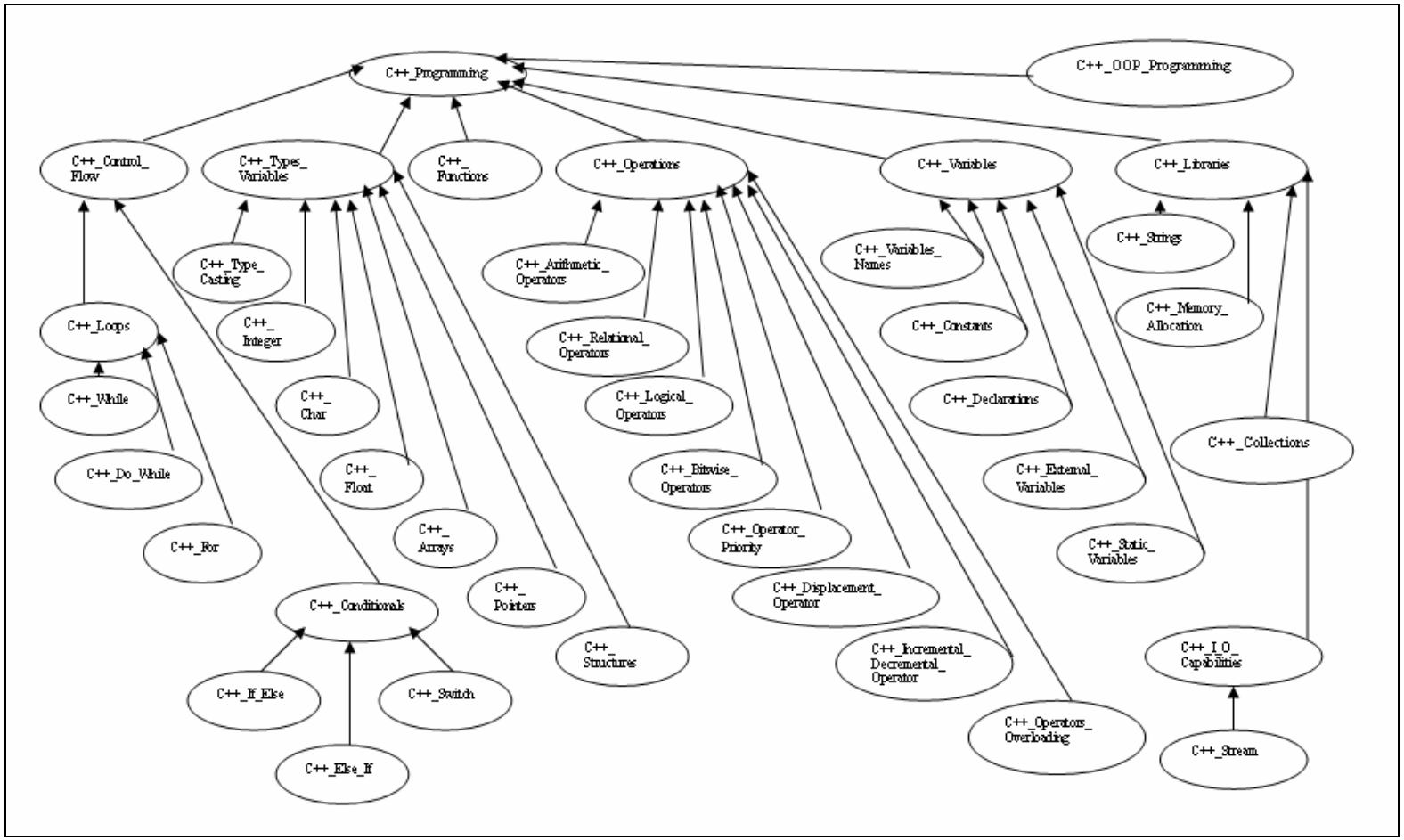

Figure 15: C++ Programming

All elements parts of “C++_Programming” have the Property “specializes” connected to the corresponding element of “Programming” or “OOP_Programming” and are instances of “Subject” class of RDFS-schema.

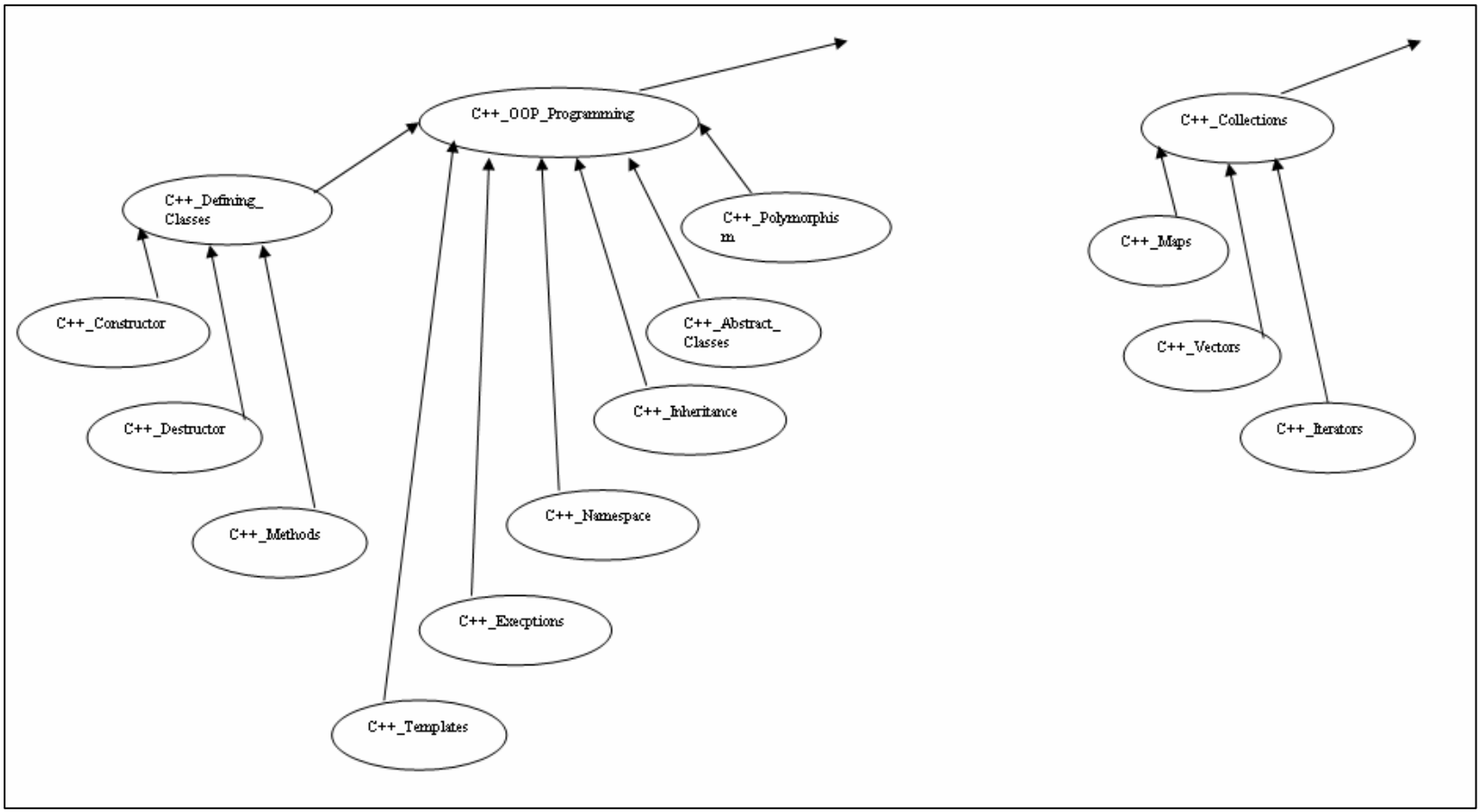

Figure 16: C++ Programming

3.2.1.5 Java Programming

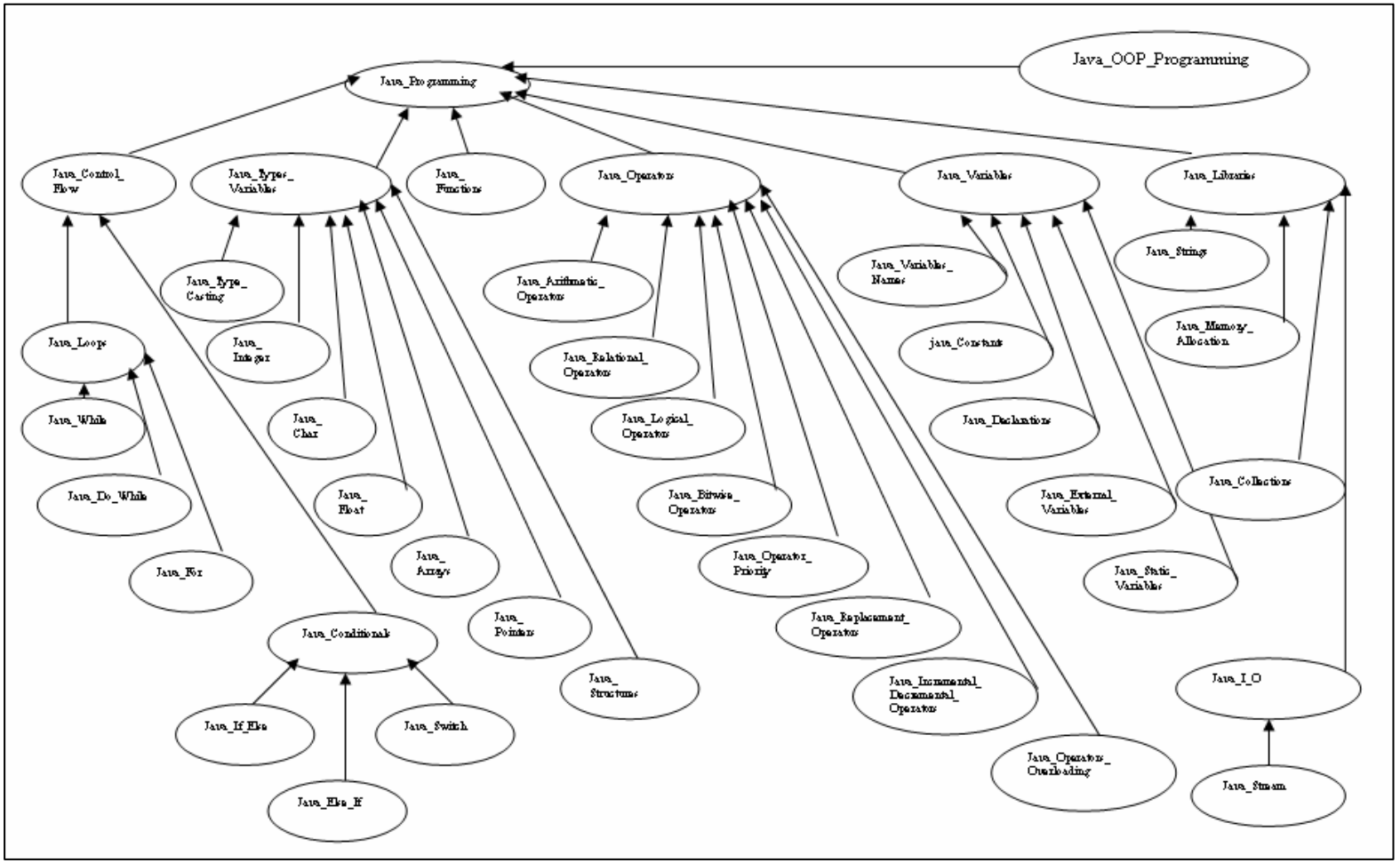

Figure 17: Java Programming

Some subjects are instances of subjects appeared in Programming schema, other are instances of subjects appeared in OOP schema, the rest are instances of “Subject” class of RDFS-schema.

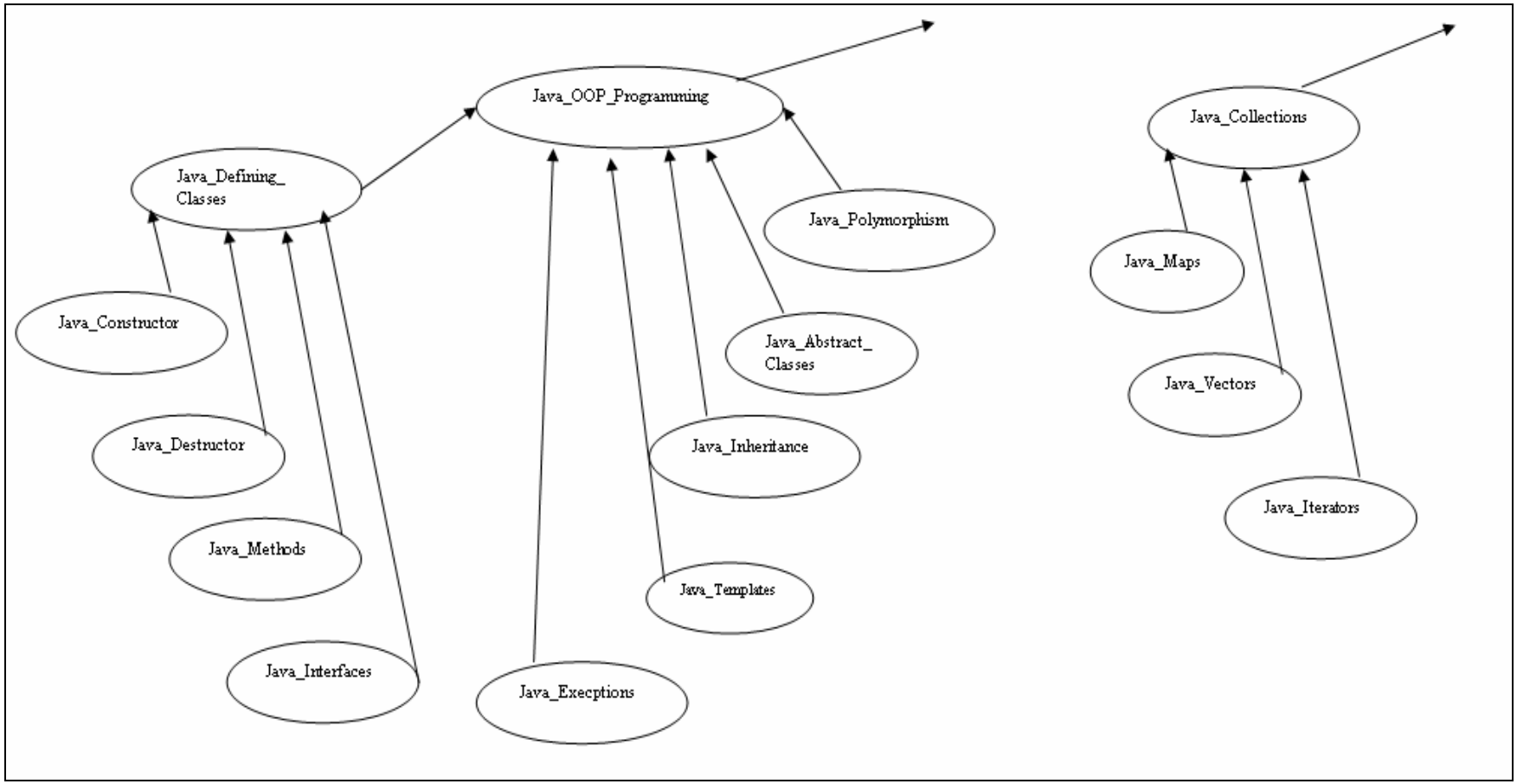

Figure 18: Java Programming

3.3 E-learn Document Content in XML

3.3.1 Pages

A *tutorial* consists of xml files like this one

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<page>
<toplinks>
        <toplink>
                <name>index.xml</name>
                <value>Java Tutorial</value>
        </toplink>
        <toplink>
                <name>language_basics.xml</name>
                <value>Language Basics</value>
        </toplink>
        <toplink>
                <name>operators.xml</name>
                <value>Operators</value>
        </toplink>
</toplinks>
<label>Arithmetic Operators</label>
<docelements>
        <theory>the_34</theory>
        <theory>the_93</theory>
        <example>exa_37</example>
        <example>exa_38</example>
        <example>exa_39</example>
        <tquestion>tq_95</tquestion>
        <tquestion>tq_96</tquestion>
        <mcquestion>mc_45</mcquestion>
        <mcquestion>mc_46</mcquestion>
          <link>lin_1</link>
          <link>lin_2</link>
</docelements>
</page>
```

"Toplinks" are links appearing to the top left of the screen showing the path from the first tutorial page ("index.xml") to current page. User can go back to any of these pages by clicking the link.

Every page contains document element ids. These ids correspond to document elements. Every document element has an RDF description and an xml file describing its content. RDF descriptions are stored to the RDF storage.

For every tutorial there is an “index.xml” file which is the first page visible to the user. User can load other pages, following the links described as <link> document elements.

3.3.2 Document Elements

3.3.2.1 Link XML File Example

```
<link>
        <id>lin_1</id>
        <linkto>Object_Oriented.xml</linkto>
        <name>Object-Oriented Programming Concepts</name>
        <preview>
                Preview HTML content
        </preview>
</link>
```

<name> tag marks the text appearing as the link, and <linkto> the xml file containing the page of the link, as describes in 2.2.1 <preview> contains a short description displayed under the link.

3.3.2.2 Theory XML File Example

```
<theory>
        <id>the_103</id>
        <text>
                HTML question content
        </text>
</theory>
```

3.3.2.3 Example XML File Example

```
<example>
        <id>exa_100</id>
        <text>
                HTML example content
        </text>
</example>
```

3.3.2.4 Multiple Choice question XML File Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mcquestion>
<id>mc_101</id>
<mcqlabel>Multiple Choice Question 1</mcqlabel>
        <preview>
```

```
            Preview HTML content
        </preview>
        <text> Question text question text question text.Correct answer 3. </text>
<toplinks>
        <toplink>
            <name>index.xml</name>
            <value>Java Tutorial</value>
        </toplink>
        <toplink>
            <name>language_basics.xml</name>
            <value>Language Basics</value>
        </toplink>
        <toplink>
            <name>operators.xml</name>
            <value>Operators</value>
        </toplink>

        <toplink>
            <name>rel_log_operators.xml</name>
            <value>Relational and Logical operators</value>
        </toplink>

</toplinks>

<type>single</type>
<options>
<option>answer1</option>
<option>answer2</option>
<option>answer3</option>
<option>answer4</option>
<option>answer5</option>
</options>
</mcquestion>
```

<type> can have value “, single” if user is restricted to select only a single option as answer, or “multiple” if multiple answers are accepted together.

3.3.2.5 Text Question XML File Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tquestion>
<id>tq_40</id>
<tqlabel>Text Question 1</tqlabel>
        <preview>
            Preview HTML content
        </preview>
```

```
        <text>
                Question text question text question text <p/>
                Question text question text question text <p/>
                Question text question text question text <p/>
                Question text question text question text <p/>
                Question text question text question text <p/>
        </text>
<toplinks>
        <toplink>
                <name>index.xml</name>
                <value>Java Tutorial</value>
        </toplink>
        <toplink>
                <name>language_basics.xml</name>
                <value>Language Basics</value>
        </toplink>
        <toplink>
                <name>operators.xml</name>
                <value>Operators</value>
        </toplink>

        <toplink>
                <name>arithmetic_operators.xml</name>
                <value>Arithmetic operators</value>
        </toplink>
</toplinks>
</tquestion>
```

3.4 RDF Database Storage

Knowledge domain RDF descriptions (instances of *Subject* and *Prerequisites* RDF classes), educative material descriptions (instances of *Docelement* class) and learner specific descriptions, are described in RDF the documents complying to *e-learn RDFS schema*. These documents can be stored to RDF database. In order to

describe a knowledge domain in an ontology model, it's possible to use an existing Semantic Web resource to extend a knowledge model, to combine parts of existing models, or construct a new model. Any RDF online document using e-learn RDFS schema is compatible with our e-learning system and can be directly used.

RDF descriptions about learners, like exercise grades, Learner knowledge, e.t.c. is stored to an RDF database. For this purpose we have used Jena framework.

3.5 E-learn Learner Java Servlet

E-learn Learner Java servlet processes e-learn document requests from the Web client, and transforms them to personalized Web pages, displaying only document elements associated to learner's knowledge level. Using this servlet, users can answer multiple choice questions and system grades the automatically. Learner also can submit files containing answers to text questions, graded by *Graders.*

3.6 E-learn Grader Java Servlet

This servlet is used by graders to store or update learner grades on text questions. Update and storage operations are done using a Java client class providing an API to E-learn RDF Storage Java Servlet.

3.7 E-learn RDF Update/Storage Java Servlet

In Jena implementation, we developed the following API, based on RDF triples.

A client can connect to E-learn RDF Storage Java Servlet requesting a set of the following operations:

add(Resource_namespace#Resource_ID,Property_namespace#Property#ID, Value)

Adds the specified triple

If the triple already exists, it does nothing

add(Resource_namespace#Resource_ID)

Creates the new Resource under namespace Resource_namespace using Resource_ID

If Resource already exists, it does nothing

del(Resource_namespace#Resource_ID,Property_namespace#Property#ID, Value)

Deletes the specified triple

If the triple does not exist, it does nothing

del(Resource_namespace#Resource_ID,Property_namespace#Property#ID)

Deletes the Property Property_namespace#Property#ID from Resource Resource_namespace#Resource_ID

If such Property does not exist, does nothing

del(Resource_namespace#Resource_ID)

Deletes the Resource Resource_namespace#Resource_ID and all its Properties, If resource does not exist, it does nothing.

For easier Java client connection to this servlet, we developed a Java client class *elearnupdateexecutor* having constructor

elearnupdateexecutor(String url)

and providing the method

boolean executeupdatequery(String query)

where String *query* represents a set of the above update operations

3.8 E-learn Query Servlet

Learner specific content stored in database is published to Semantic Web using *E-learn query servlet.* So authorized users can retrieve learner specific information. The output of the local servlet can be used by any other remote installations of our system, or from the local *reasoning module.* This feature provides to any authorized educational unit using our system, the ability to retrieve information about students from other educational units.

In Jena implementation, servlet respond to SPARQL queries, giving as output an XML SPARQL result set for SELECT queries, a Boolean value for ASK queries and an RDF model for DESCRIBE or CONSTRUCT query.

For easier Java client connection to this servlet, we developed a Java client class *elearnqueryexecutor* having constructor

elearnqueryexecutor(String url)

and the following methods

ResultSet executeselectquery(String query)

boolean executeaskquery(String query)

Model executeconstructquery(String query)

Model executedescribequery(String query)

This servlet was implemented using *Joseki,* a query server for *Jena* supporting SPARQL

3.9 Reasoning Module

The reasoning module queries RDF descriptions to get information about each document element subjects and prerequisites, related subjects, and user knowledge on them.

The *reasoning module* provides the predicate cansee(L,D) returning true or false to the question if learner L can see document element D. This predicate is true if and only if, for every prerequisite triple P (Subject,minlevel,maxlevel) associated with the document element D, the calculated user knowledge K on this Subject is between minlevel and maxlevel. It also provides recommendation rules. For any visible page, system recommends to the user some of the contained document elements.

3.9.1 Calculating user knowledge

User knowledge is calculated using a rule based algorithm. Different types of user knowledge can be calculated for every subject. These types are

1. Stored knowledge(U,S).

   It's the value of the knowledge of user U on subject S calculated on a previous algorithm execution and stored to the database This value can be -1, representing that a previous algorithm execution calculated that no knowledge value can be calculated using this algorithm. If not value is stored, it fails

2. Part knowledge(U,S).

   Calculates user knowledge on each subject that "ispartof" subject S (e.g. While, Do_While, For are parts of Loops) and finds a mean value for these

using subject weights. If there is not any "ispartof" subject where knowledge value can be calculated, it fails

3. Specialization knowledge(U,S).

   Calculates user knowledge on each subject that is connected through a "specialization" with subject S (e.g. C_Programming, Java_Programming, C++ Programmng are specializations of Programming) and find a mean value for these using subject weights. If there is not any "part-of" subject where knowledge value can be calculated, it fails.

4. Exercise knowledge(U,S)

   Calculates user knowledge, using exercise grades on this subject. The value is calculated using all mandatory exercises and only the optional exercises answered form this user on this subject. If user has not answered a mandatory exercise, use grade 0 on this.

5. Ancestor part knowledge(U,S)

   If there is a stored knowledge for user U on subject S, returns this value.

   Else find the direct ancestor of this subject using "ispartof" relation with stored value of user knowledge on this subject, and return this value.

6. Ancestor specialization knowledge(U,S)

   If there is a stored knowledge for user U on subject S, returns this value.

   Else find the direct ancestor of this subject using "ispartof" relation with stored value of user knowledge on this subject.

If A1, A2,...An the sequence of "ispartof" anscestors of S where An is the first ancestor of S where there is stored knowledge for user U and $Ai_1, Ai_2 \ldots Ai_N$, is the children of Ai, the "ancestor specialization knowledge" of U on S is

knowledge(S) =weight(S) /(weight($A1_1$)+weight($A1_2$)+…weight($A1_N$)) *

weight(S) /(weight($A2_1$)+weight($A2_2$)+…weight($A2_N$)) *

weight(S) /(weight(AN_1)+weight(AN_2)+…weight(AN_N)) *

User knowledge on a specific subject is calculated using the following algorithm, which have been implemented using Prolog rules.

A1. If the special value "-1" is stored, it is calculated previously that no value can be calculated on the following steps. So return failure calculating knowledge on this subject. Else go to step 2

A2. If there is a stored positive value about user knowledge on this Subject, return stored value. If there is not any stored value, goto step 3.

A3. If it's possible to calculate exercise knowledge and part knowledge on this subject, return (exercise knowledge * 2 + part knowledge) / 3 and store value for later use. Else go to step 4.

A4. If it's possible to calculate exercise knowledge and specialization knowledge on this subject, return (exercise knowledge + specialization knowledge) / 2 and store value for later use. Else go to step 5.

A5. Try calculating exercise knowledge, and store it for later use. If couldn't calculate any value using this rule, go to step 6.

A6. Try calculating specialization knowledge and store it for later use. If couldn't calculate any value using this rule, go to step 7.

A7. Try calculating part knowledge and store it for later use. If couldn't calculate any value using this rule, go to step 8.

A8. Try calculating ancestor part knowledge (without storing it). If couldn't calculate any value using this rule, go to step 9.

A9. Try calculating ancestor specification knowledge (without storing it). If couldn't calculate any value using this rule, go to step 10.

A10. Store special value "-1" as user knowledge for later use , meaning that no user knowledge can be calculated on this subject using these rules.

B. When a new grade is stored to any exercise, find all subjects of the exercise and delete the stored knowledge value for its predecessor on "ispartof" or "specializes" Property.

C. Schedule a process to run every specific time duration calculating and store knowledge values for every learner, using rules A1-A6. Precomputing user knowledge values reduces system response time significantly.

3.9.2 Recommendations

For any visible page, system recommends to the user some of the contained document elements (theory, exercises, links) using the following rules

3.9.2.1. Rules for recommendations

Theory

a. Recommend a theory that does not prerequisite knowledge.

b. Recommend the theory having the subject of the prerequisite knowledge, for which the user has the maximum knowledge level.

c. Provided that a theory has subject S where the user KL is greater than 2, refute the first rule for each theory having the same subject S except for those, for which has the subject in their prerequisite knowledge with the maximum user knowledge level.

d. Provided that the user KL for some of the subjects of the Theory is adequately big, e.g. > 8, do not recommend this theory by refuting the aforementioned rules.

Exercises

a. Recommend the mandatory exercises.

b. Recommend the exercises that have a subject of the prerequisite knowledge that is included in the subjects of the prerequisite knowledge of the theory that we have recommended.

c. Provided that there exists a theory that the user can not see and has as prerequisite knowledge the same subject, recommend exercises of the same subject.

d. Provided that the user KL for one of the subjects of an exercise is adequately big (e.g. >8), do not recommend the exercise by refuting the aforementioned rules.

Examples

a. Recommend examples that have a subject of the prerequisite knowledge that is included in the subjects of the prerequisite knowledge of the theory that we have recommended.

b. Recommend examples that have a subject of the prerequisite knowledge that is included in the subjects of the prerequisite knowledge of the exercises that we have recommended.

c. Provided that KL>8 for the subjects of an example, do not recommend it, by refuting the aforementioned rules.

Links

a. Recommend the links for which the user has for some of their subjects less knowledge than for all the links of the same page

b. Provided that KL>8 for some subject of a link, do not recommend it.

3.9.2.2. Rules in Dr-Prolog

The following rules need to be inserted in the metaprogram and are used for the implementation of the inequality relationships in the defeasible theories.

definitely(greater(X,Y)):- X>Y.

definitely(less(X,Y)):- X<Y.

definitely(equal(X,Y)):- X=Y.

definitely(greater_or_equal(X,Y)):- X>=Y.

definitely(less_or_equal(X,Y)):- X<=Y.

Defeasible rules

r1: cansee(User, DocEl), typeof(DocEl, Theory) =>

recom_theory(User, DocEl).

r2: cansee(User, DocEl), typeof(DocEl, Theory), (requires(DocEl, Pre)) =>

~recom_theory(User, DocEl).

r2 > r1.

r3: cansee(User, DocEl1), typeof(DocEl1, theory), hassubject(DocEl1, S1),

cansee(User, DocEl2), typeof(DocEl2, theory), hassubject(DocEl2, S2),

hasknowledge(User,S1,KL1), hasknowledge(User,S2,KL2),less(KL1, KL2) =>

~(recom_theory(User, DocEl1)).

r3 > r1.

r4: cansee(User, DocEl1), typeof(DocEl1, theory), hassubject (DocEl1, S), hasknowledge(User, S, KL), greater(KL, 2), requires(DocEl1, Pre1), preqsubject(Pre1,PS1), hasknowledge(User, PS1, KL1), cansee(User, DocEl2) typeof(DocEl2, theory), hassubject (DocEl2, S), requires(DocEl2, Pre2), preqsubject(Pre2, PS2), hasknowledge(User, PS2, KL2),

less(KL1,KL2) => ~recom_theory(User,DocEl1).

r4 > r1.

r5: cansee(User, DocEl), typeof(DocEl, theory), hassubject(DocEl, S), hasknowledge(User, S, KL), greater(KL,7) => ~(recom_theory(User, DocEl)).

r5 > r1.

r6: cansee(User, DocEl), typeof(DocEl, exercise), hassubject(exercise, S), mandatory(exercise, true) => recom_exercise(User, DocEl).

r7: recom_theory(User, DocEl), requires(DocEl, Pre1), preqsubject(Pre1, S) cansee(User,DocEl2), typeof(DocEl2, example), requires(DocEl2, Pre2), preqsubject(Pre1, S)

=>recom_exercise(User, DocEl2).

r8: cansee(User, DocEl1), typeof(DocEl1, theory), hassubject(DocEl1, S), ~(cansee(User, DocEl2)), typeof(DocEl2, theory), hassubject(DocEl2, S), requires(DocEl2, Pre), preqsubject(Pre, S), cansee(User, DocEl3),

typeof(DocEl3, exercise), hassubject(exercise, S)

=> recom_exercise(User, DocEl3).

r9: cansee(User, DocEl), typeof(DocEl, exercise), hassubject(DocEl, S), hasknowledge(User, S, KL), greater(KL,7) => ~(recom_exercise(User, DocEl)).

r9 > r6.

r9 > r7.

r9 > r8.

r10: recom_theory(User, DocEl), requires(DocEl, Pre1), preqsubject(Pre1, S) cansee(User,DocEl2), typeof(DocEl2, example), requires(DocEl2, Pre2), preqsubject(Pre1, S)

=>recom_example(User, DocEl2).

r11: recom_exercise(User, DocEl), requires(DocEl, Pre1), preqsubject(Pre1, S) cansee(User,DocEl2), typeof(DocEl2, example), requires(DocEl2, Pre2), preqsubject(Pre1, S)

=>recom_example(User, DocEl2).

r12: cansee(User, DocEl), typeof(DocEl, example), hassubject(DocEl, S),

hasknowledge(User, S, KL), greater(KL,6) => ~(recom_example(User, DocEl)).

r12 > r10.

r12 > r11.

r13: cansee(User, DocEl), typeof(DocEl, link), hassubject(DocEl, S) => recom_link(User, DocEl).

r14: cansee(User, DocEl1), typeof(DocEl1, link), hassubject(DocEl1, S1), cansee(User, DocEl2), typeof(DocEl2, link), hassubject(DocEl2, S2),

hasknowledge(User,S1,KL1), hasknowledge(User,S2,KL2),greater(KL1, KL2) =>

~(recom_link(User, DocEl1)).

r14 > r13.

r15: cansee(User, DocEl), typeof(DocEl, link), hassubject(DocEl, S), hasknowledge(User, S, KL), greater(KL,7) => ~(recom_link(User, DocEl)).

r15 > r13.

4 A Concrete Usage Example

Learner_1 is a new user of the system with no prior knowledge to any subject in programming.

Learner_3 is a new user of the system, with prior knowledge to all subjects related to C, C++ and Java Programming.

Since Learner_1 has not any prior knowledge to any subject, the value "-1" is stored in the knowledge base as his knowledge level in any subject.

Learner_3 has the following stored values in the knowledge base about his knowledge level:

Subject	**Knowledge Level**
C_Programming	5.8
C_Control_Flow	6
C_Loops	6
C_While	6
C_Do_While	6
C_For	6
C_Conditionals	6
C_If_Else	6
C_Else_If	5
C_Switch	7
C_Types_Variables	6
C_Type_Casting	4
C_Integer	7
C_Char	7
C_Float	6
C_Arrays	6
C_Pointers	5
C_Structures	4
C_Functions	5
C_Operators	5
C_Arithmetic_Operators	6
C_Relational_Operators	5
C_Logical_Operators	4
C_Bitwise_Operators	4
C_Operator_Priority	6

C_Displacement_Operators	5
C_Incremental_Decremental_Operators	4
C_Variables	6
C_Variables_Names	6
C_Constants	7
C_Declarations	5
C_External_Variables	6
C_Static_Variables	6
C_Libraries	7
C_Strings	7
C_Memeory_Allocation	5
C_Malloc	4
C_Realloc	4
C_Free	4
C_I_O_Capabilities	7
C_Printf	7
C_Scanf	8
C_Getchar	9
C_Putchar	9
C_Read	7
C_Write	6
C_Fprintf	6
C_Fscanf	6
Cplusplus_Programming	6
Cplusplus_Control_Flow	6
Cplusplus_Loops	7
Cplusplus_While	6
Cplusplus_Do_While	7
Cplusplus_For	5
Cplusplus_Conditionals	6
Cplusplus_If_Else	6
Cplusplus_Else_If	5
Cplusplus_Switch	7
Cplusplus_Types_Variables	4

Cplusplus_Type_Casting	5
Cplusplus_Integer	7
Cplusplus_Char	7
Cplusplus_Float	7
Cplusplus_Arrays	7
Cplusplus_Pointers	9
Cplusplus_Structures	6
Cplusplus_Functions	6
Cplusplus_Operators	6
Cplusplus_Arithmetic_Operators	6
Cplusplus_Relational_Operators	6
Cplusplus_Logical_Operators	5
Cplusplus_Bitwise_Operators	6
Cplusplus_Operator_Priority	6
Cplusplus_Displacement_Operators	6
Cplusplus_Incremental_Decremental_Operators	7
Cplusplus_Operators_Overloading	6
Cplusplus_Variables	7
Cplusplus_Variables_Names	6
Cplusplus_Constants	7
Cplusplus_Declarations	6
Cplusplus_External_Variables	7
Cplusplus_Static_Variables	7
Cplusplus_Libraries	7
Cplusplus_Strings	7
Cplusplus_Memeory_Allocation	8
Cplusplus_I_O_Capabilities	7
Cplusplus_Stream	7
Cplusplus_Collections	5
Cplusplus_Maps	6
Cplusplus_Vectors	5
Cplusplus_Iterators	4
Cplusplus_Defining_Classes	6
Cplusplus_Constructor	5
Cplusplus_Destructor	5

Cplusplus_Methods	7
Cplusplus_Polymorphism	5
Cplusplus_Abstract_Classes	4
Cplusplus_Inheritance	4
Cplusplus_Namespaces	5
Cplusplus_Execptions	6
Cplusplus_Templates	5
Cplusplus_OOP_Programming	5
Java_Programming	6.7
Java _Control_Flow	8
Java _Loops	8
Java _While	8
Java _Do_While	8
Java _For	8
Java _Conditionals	8
Java _If_Else	8
Java _Else_If	7
Java _Switch	8
Java _Types_Variables	7
Java _Type_Casting	6
Java _Integer	7
Java _Char	8
Java _Float	8
Java _Arrays	9
Java _Pointers	9
Java _Structures	6
Java _Functions	6
Java _Operators	7.5
Java _Arithmetic_Operators	7.5
Java _Logical_Operators	7.5
Java _Bitwise_Operators	7.5
Java _Operator_Priority	6.5
Java _Displacement_Operators	7.5
Java _Incremental_Decremental_Operators	8.5

Java_Operators_Overloading	8.5
Java _Variables	7
Java _Variables_Names	7
Java _Constants	7
Java _Declarations	7
Java _External_Variables	7
Java _Static_Variables	7
Java _Libraries	7
Java _Strings	8
Java _Memory_Allocation	7
Java _I_O_Capabilities	7
Java _Stream	7
Java _Collections	5
Java _Maps	5
Java _Vectors	5
Java _Iterators	5
Java _Defining_Classes	8
Java _Constructor	6
Java _Destructor	6
Java _Methods	7
Java _Polymorphism	8
Java _Abstract_Classes	7
Java _Inheritance	7
Java _Execptions	7
Java _OOP_Programming	7.4
Java_Interfaces	7

Table 1: Prior knowledge of Learner_3

The schedule process, precompute knowledge level for Learner_3 to all subjects, using rules A1-A6, and stores them in the knowledge base. Learner_3 has already stored knowledge to all subjects, except parts of subject “Programming”. The knowledge for Learner_3 on this subject (parts of “Programming”) is precomputed using rule A6, for “Specialization knowledge”. This rule calculates user knowledge on each subject that is “specialization” of subject S (e.g. C_Programming, Java_Programming, C++_Programming are specializations of Programming) and find a mean value for these using subject weights. If there is not any “part-of” subject where knowledge value can be calculated, fails.

The scheduled process precomputes the following knowledge values for Leaner_3 and stores them to knowledge base:

Subject	**Knowledge Level**
Programming	6.1667
Variables	6.3000
Types_Variables	5.6667
Functions	5.6667
Operators	6.1667
Control_Flow	6.6667
Libraries	7.0000
I_O_Capabilities	6.7500
Strings	7.3333
Memory_Allocation	7.5000
Collections	5.0000
Iterators	4.5000
Variable_Names	6.3333
Constants	7.0000
Declarations	6.0000
External_Variables	6.6667
Static_Variables	6.6667
Type_Casting	5.0000
Integer	7.0000
Char	7.3333
Float	7.0000
Arrays	7.5000

Pointers	7.6667
Structure	5.6667
Arithmetic_Operators	6.7500
Relational_Operators	6.1667
Logical_Operators	4.0000
Bitwise_Operators	5.8333
Incremental_Decremental_Operators	6.5000
Displacement_Operators	6.1667
Operator_Priority	6.1667
Conditionals	6.5000
Loops	7.0000
If_Else	6.0000
Else_If	5.6667
Switch	7.3333
While	6.6667
For	6.3333
Do_While	6.2000
OOP_Programming	6.2000
Defining_Classes	2.0000
Exceptions	7.0000
Interfaces	3.5000
Inheritance	2.7500
Polymorphism	6.5000
Abstract_Classes	5.5000
Constructor	5.5000
Destructor	5.5000
Methods	7.0000

Table 2: Recomputed knowledge of Learner_3

The first page of the tutorial contains the following document elements

Figure 20: First Page of the Java Tutorial

Learner_1 views the following content on the first screen of the tutorial

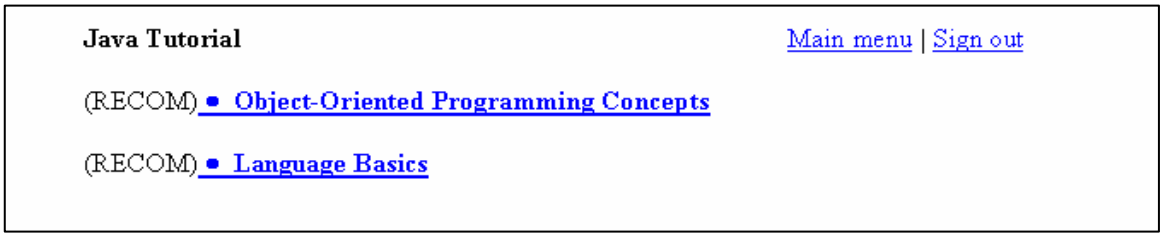

Figure 21: First Screen of the Tutorial for Learner_1

Learner_3 views the following content on the first screen of the tutorial

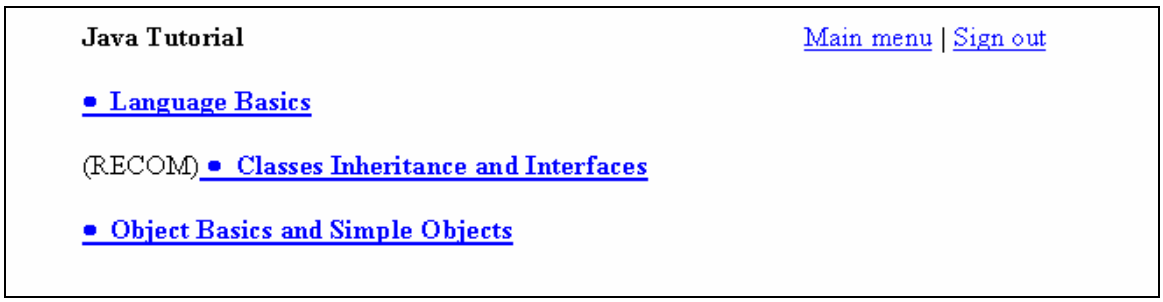

Figure 22: First Screen of the Tutorial for Learner_3

Explanation

Learner_1 views "Object-Oriented Programming Concepts" link because it requires Knowledge level in OOP_Programming lower than 6, and Learner_1 has not any knowledge in OOP_Programming

Learner_1 can view "Language Basics" link because it has not any prerequisites.

Learner_1 can't view "Classes Inheritance and Interfaces" and "Object Basics and Simple Objects" because he has not any knowledge to required subjects (e.g. Classes Inheritance and Interfaces requires knowledge level to "OOP_Programming" greater than 4 and to "Java_OOP_Programming" greater than 3).

System uses "rule a" for Links, to recommend to Learner_1 both visible links in this page.

"Rule a" for Links is "Recommend the links for which the user has for some of their subjects less knowledge than for all the links of the same page". Learner_1 has not any knowledge to any subject of the visible links, so he has the minimum knowledge for both links in the page.

Learner_3 can't view "Object-Oriented Programming Concepts" link because it requires: Knowledge level in OOP_Programming lower than 6, and Learner_3 has knowledge level 6.2 in OOP_Programming.

Learner_3 can view “Language Basics” link because it has not any prerequisites.

Learner_3 views “Classes Inheritance and Interfaces” link because it requires the following knowledge levels:

OOP_Programming: 4-10 (Learner_3 knowledge level is 6.2)

Java_Variables:5-10 (Learner_3 knowledge level is 7)

Java_Operators: 5 -10 (Learner_3 knowledge level is 7.5)

Java_Control_Flow: 5-10 (Learner_3 knowledge level is 8)

System uses “rule a” for Links, to recommend to Learner_3 “Classes Inheritance and Interfaces” “Rule a “for Links is “Recommend the links for which the user has for some of their subjects less knowledge than for all the links of the same page”. Learner_3 has the following knowledge level to link subjects:

Link “Language Basics” has subjects

Java_Variables (Learner_3 level is 7)

Java_Operators (Learner_3 level is 7.5)

Java_Control_Flow (Learner_3 level is 8)

Link “Classes Inheritance and Interfaces” has subjects

Java_Constructor (Learner_3 level is 6)

Java_Destructor (Learner_3 level is 6)

Java_Methods (Learner_3 level is 7)

Java_Interfaces (Learner_3 level is 7)

Java_Inheritance (Learner_3 level is 7)

Link “Object Basics and Simple Objects” has subject

Java_Defining_Classes (Learner_3 level is 8)

The subjects with the lower Learner_3 knowledge level are Java_Constructor and Java_Destructor in link "Classes Inheritance and Interfaces". So the system recommends this link to Learner_3.

Learner_1 selects the link "Language Basics"

The selected page of the tutorial contains the following document elements

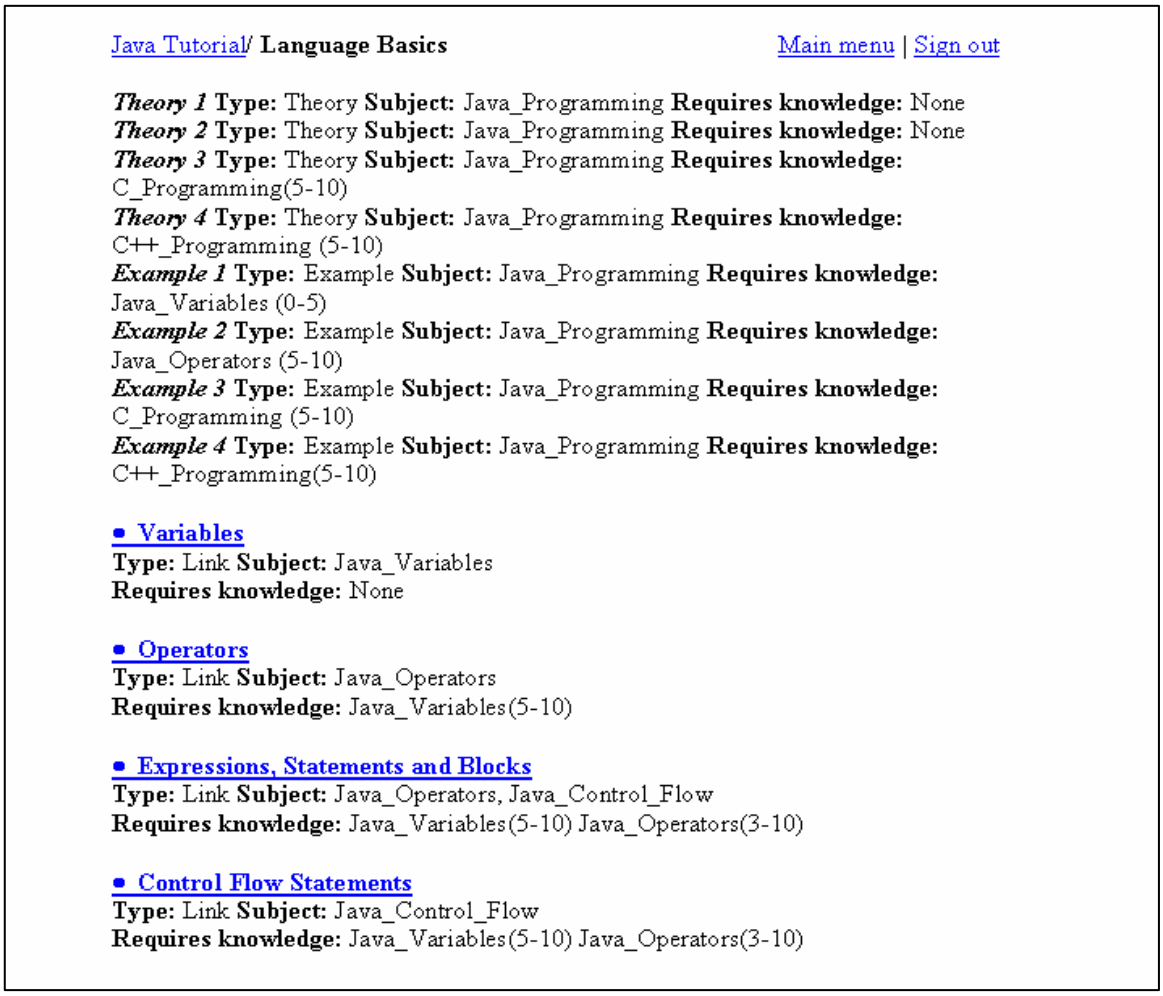

Figure 23: Language Basics

Learner_1 views the following content on this screen

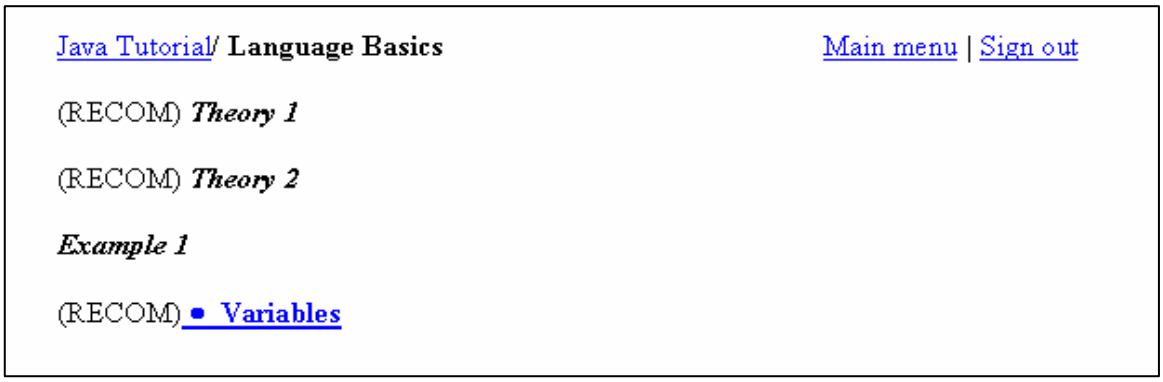

Figure 24: Java Language Basics for Learner_1

Explanation

Learner_1 can view "Theory 1" and "Theory 2" and link "Variables" because they have not any prerequisites.

Learner_1 can't view "Theory 3" because he has not the required knowledge level (5) in C_Programming.

Learner_1 can't view "Theory 4" because he has not the required knowledge level (5) in C++_Programming.

Learner_1 can view "Example 1" because it requires knowledge to Java_Variables lower than 5 and Learner_1 has not any knowledge to Java_Variables System recommends "Theory 1" and "Theory 2", using "rule a" for Theory. "Recommend a theory that does not prerequisite knowledge" This rule is not defeated by other rules.

System recommends "Variables" using "rule a" for Links: "Recommend the links for which the user has for some of their subjects less knowledge than for all the links of the same page".

Learner_1 has not any knowledge for Java_Variables subject.

Learner_1 selects the link “Variables”

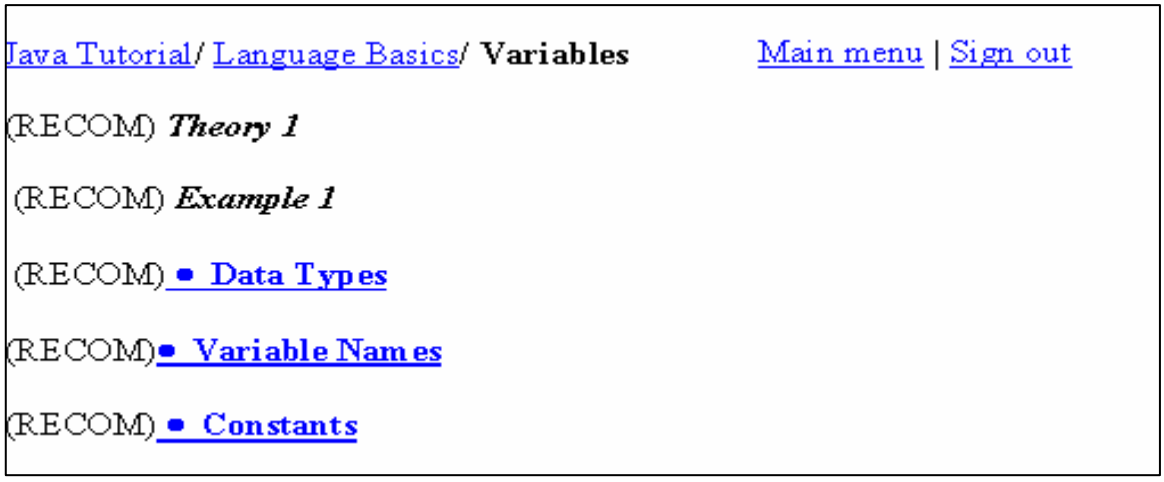

Figure 25: Java Variables for Learner_1

Explanation

Learner_1 can view “Theory 1”, “Example 1” and links “Data Types”, “Variable Names” and “Constants” because he has appropriate knowledge level. The system recommends all the above according to the rules for recommendation (chapter 3.9.2.1).

Learner_1 selects the link “Variable Names”

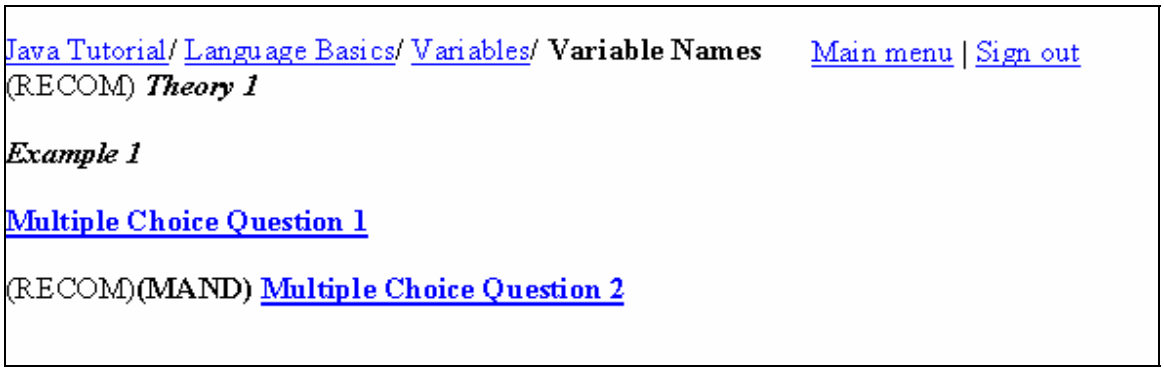

Figure 26: Java Variable Names for Learner_1

Learner_1 can view “Theory 1”, “Example 1” and links “Multiple Choice Question 1” and “Multiple Choice Question 2” because he has appropriate knowledge level. The system recommends “Theory 1” and “Multiple Choice Question 2” according to the rules for recommendation (chapter 3.9.2.1). “Multiple Choice

Question 2"is mandatory. Mandatory questions are used at the calculation of user knowledge. The value is calculated using all mandatory exercises that can be viewed by the user, and only the optional exercises answered by this user on this subject. If user has not answered a mandatory exercise, use grade 0 on this.

Learner_1 selects the link "Multiple Choice Question 2"

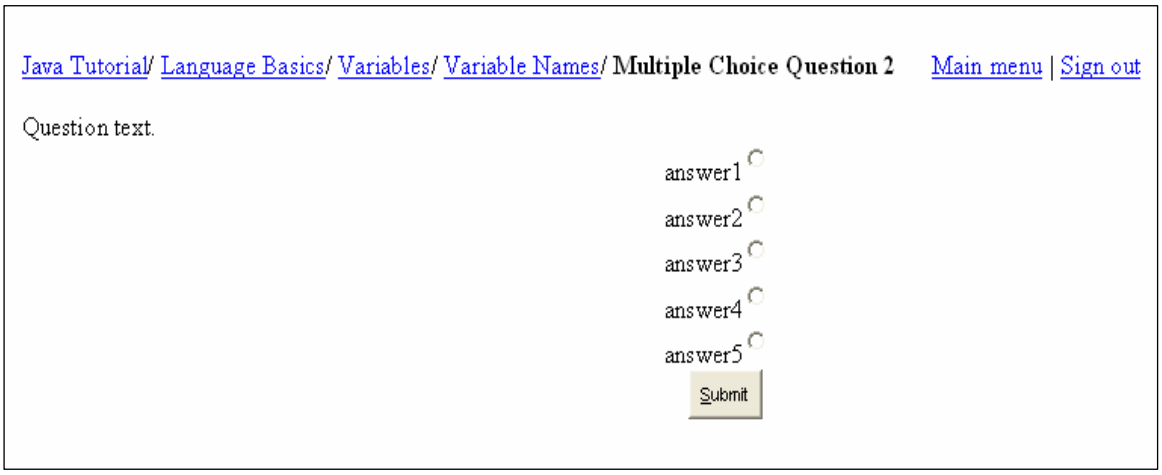

Figure 27: Multiple Choice Questions 2 for Learner_1

Learner_1 views the question and the corresponding answers and must select the answer that he believes to be correct.

Learner_1 answered the "Multiple Choice Question 2 Answer"

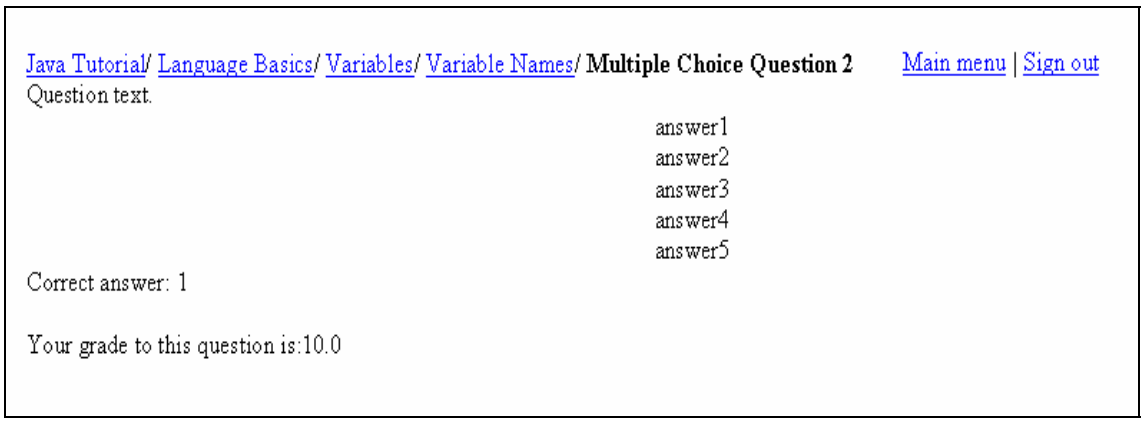

Figure 28: Multiple Choice Question 2 Answer for Learner_1

After Learner_1 answers the question the system returns the grade to the question and the correct answer. The same question can not be answered again.

Learner_1 view the "Variable Names" after answering "Multiple Choice Question 2"

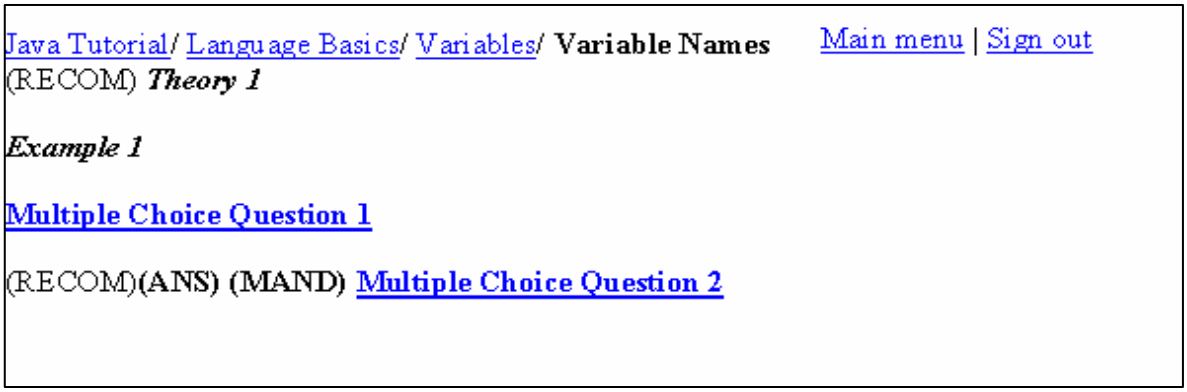

Figure 29: Multiple Choice Question 2 Answer for Learner_1

Learner_1 views "Multiple Choice Question 1" and "Multiple Choice Question 2" but this time "Multiple Choice Question 2" is marked as "Answered".

Learner_1 view the "Variables" after

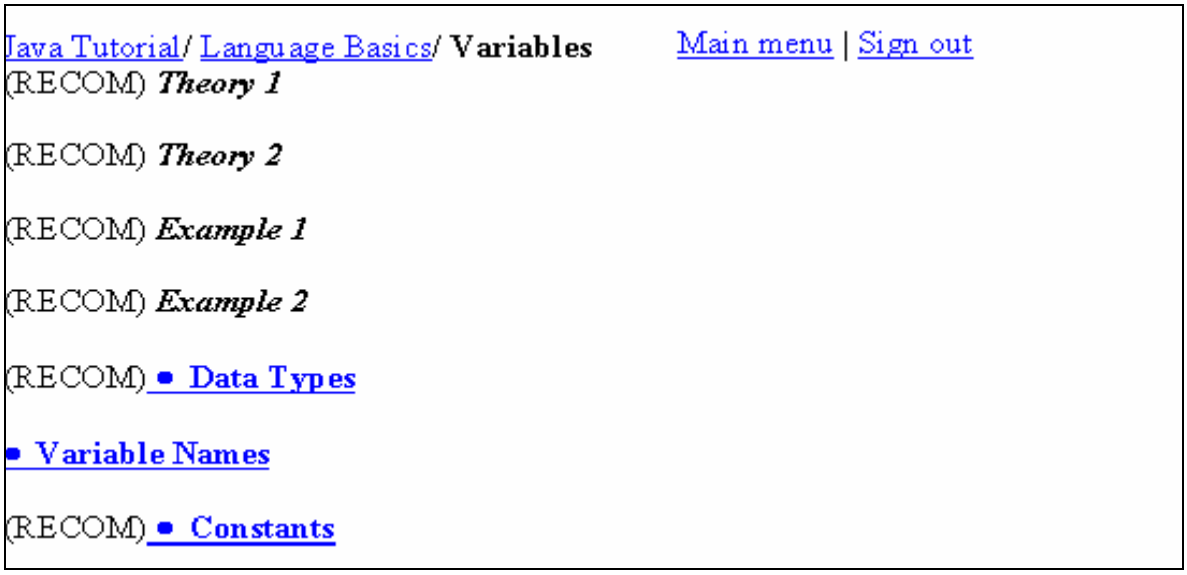

Figure 30: Java Variables for Learner_1

Learner_1 after answering correct "Multiple Choice Question 2" has gained knowledge on subject "Variables Names", according to rules in 3.9.1 the subject "Variables" will get a grade too. As a result he can see also "Theory 2" "Example 2"

and the link “Variable Names” is not any more recommended according to the recommendation rules.

With the same way Learner_1 can view the links “Data Types” and “Constants”, solve the corresponding exercises and gain knowledge at these subjects. Then he can go back to “Language Basics”.

Learner_1 view the “Language Basics” after finishing the “Variables”

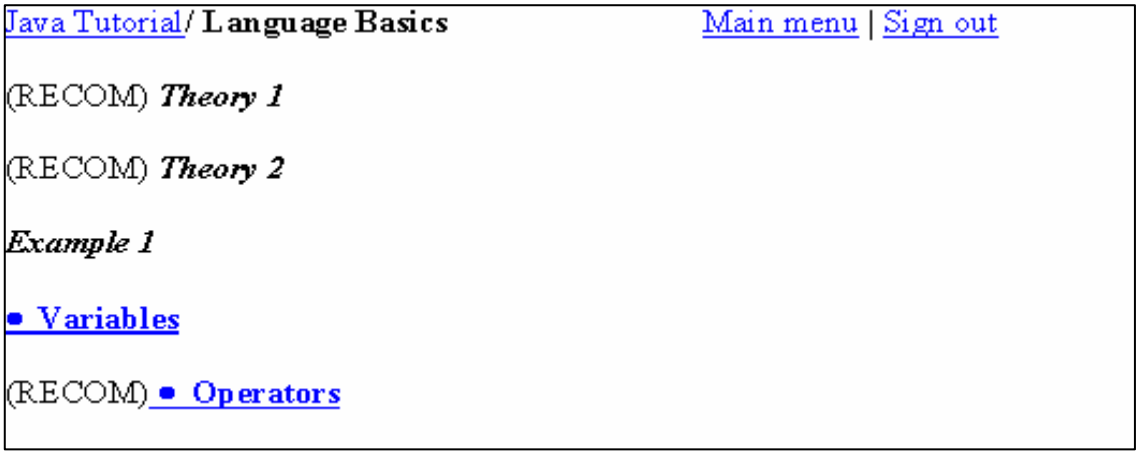

Figure 31: Language Basics after the Variables for Learner_1

Learner_1 returns to “Language Basics” after finishing the “Variables” subject successfully. Now he can see the link “Operators” because “Variables” is prerequisite and the learner has appropriate knowledge level.

With the same way Learner_1 can continue to the tutorial. The system will provide personalized information and exercises to the learner according to exercise he succeeded and the knowledge he gained.

5 Conclusions and Future Work

5.1 General Conclusions

In this report, we presented the design and implementation of a personalized rule- based e-learning system using Semantic Web technologies. The user of an e-learning application based on our system, can navigate between personalized Web pages, view links, theory, examples and exercises according to his knowledge level to their subjects, their prerequisites, or related subjects. The learner knowledge level for each subject is deduced by the reasoning module. This module uses logic over online RDF descriptions, to conclude or guess the user knowledge level, depending on learner answers to exercises on related subjects. The reasoning module makes also recommendation to the learner recommending the most appropriate content to focus his attendance.

Descriptions of knowledge domains and educative material in RDF and XML, supports the sharing of them between multiple educational centers, and description of learner attributes gives the ability to a learner attending lectures to multiple learning sources simultaneously, to share a common personalized user profile between them. Any education center can use its own educative material while using or extend parts of the material from other educational centers.

The use of defeasible logic for reasoning has the advantage of reasoning with inconsistent or incomplete information, a common phenomenon in these cases. The non monotonic behavior of defeasible logic supports easy revision of system hypothesis about user knowledge on specific subjects when data is considered, without having inconsistencies. Defeasible rules was also used to describe make recommendation rules

Our system is Semantic Web compliant, so remote agents can connect and query remote system resources. Our system can be extended to support intelligent agent communication and/or automatic ontology merging between different resource descriptions.

The system has a distributive architecture, so it can be easily extended with applications for handling, updating and processing the knowledge domain, educative material and learner descriptions. Changes to reasoning module are easy, so it's easy

to add new rules for recommendations or update or replace the algorithm estimating the user knowledge.

Communication between system components is based on defeasible logic and SPARQL standards, so the system can use any future compliant Semantic Web toolkits without any modification needed, increasing it's speed, security and reliability.

5.2 Future Work

In the future we plan to:

- Extend our system by considering a negotiating agent-based approach to support the functionality needed for different universities using our system simultaneously to automatically exchange information. In an agent-based scenario, different universities contribute to the knowledge represented by our system, including knowledge sources as well as information about students

- Support ontology merging: when ontologies from different authors and/or sources are merged, contradictions arise naturally. Defeasible logic can be very useful for conflict resolution, because it does not allow for contradictory conclusions to be drawn.

- Support *navigational* recommendations. When a user has not the required knowledge level to view a document element, the system recommends links to pages containing document elements that can help him advance his knowledge level on the relative subject.

- Develop applications with a user friendly graphical interface, supporting inserting and updating data to our knowledge base including teaching material and information about students.

- Consider more advanced techniques for recommendations and to calculate user knowledge on a subject, such as collaborative filtering, which takes into account information about a group of users in order to make recommendations to another, by assuming a similarity measure among them.

6 References

[1] T. Dietinger, “Aspects of E-learning Environments”, PhD thesis, Graz University of Technology, 2003.

[2] S.Tsai, P/ Machado, “Essay: Elearning, online learning, web-based learning, or distance learning: unveiling the ambiguity in current terminology”, *eLearn,* vol. 2002, no. 7 p.p. 3, 2002.

[3] A. Trifinova, M. Ronchetti, “A General Architecture to Support Mobility in Learning”, *Proceedings of the IEEE International Conference on Advanced Learning Technologies,* 2004.

[4] P. Brusilovsky and M. T. Maybury, “From adaptive hypermedia to the adaptive web”, *Communications of the ACM,* vol. 45, no. 5 p.p. 30–33, 2002.

[5] F. Mödritscher, V. M. Garcia-Barrios, Christian Gütl. “The Past, the Present and the future of adaptive E-Learning”, *In Proceedings of the International Conference Interactive Computer Aided Learning (ICL2004)*, 2004.

[6] S. Weibelzahl, “Evaluation of Adaptive Systems”, PhD thesis, University of Trier, 2003.

[7] S. Stoyanov, P. Kirschner, “Expert Concept Mapping Method for Defining the Characteristics of Adaptive E-Learning: ALFANET Project Case”, *Educational Technology, Research & Developement*, vol. 52, no. 2 p.p. 41–56, 2004.

[8] O. Park, J. Lee, “Handbook of Research for Educational Communications and Technology”, P.p. 651–660. Association for Educational Communications and Technology, 2003.

[9] L. Corno and R.E. Snow, “Adapting teaching to individual differences among learners”, *Handbook of research on teaching*, 1986.

[10] A. L. Soller, “Supporting Social Interaction in an Intelligent Collaborative Learning System”, *International Journal of Artificial Intelligence in Education*, vol. 12, p.p. 40–62, 2001.

[11] V. J. Shute, J. Psotka, “Handbook of Research on Educational Communications and Technology”, P.p. 1–99. Scholastic Publications, 1996.

[12] P. Brusilovsky, “The Construction and Application of Student Models in Intelligent Tutoring Systems”, *Journal of Computer and System Sciences International*, vol. 32, no. 1 p.p. 70–89, 1994.

[13] P. Brusilovsky, “Methods and Techniques of Adaptive Hypermedia”, *User Modeling and User-Adapted Interaction*, vol. 6, no. 2–3 p.p. 87–129, 1996.

[14] P. Brusilovsky, “Adaptive Hypermedia”, *User Modeling and User-Adapted Interaction*, vol. 11, no. 1–2 p.p. 87–110, 2001.

[15] Kinshuk, T. Lin. “User Exploration Based Adaptation in Adaptive Learning Systems”, *International Journal of Information Systems in Education*, vol. 1, no. 1 p.p. 22–31, 2003.

[16] P. De Bra, “Pros and Cons of Adaptive Hypermedia in Web-Based Education”, *Journal on CyberPsychology and Behavior*, vol. 3, p.p. 71–77, 2000.

[17] N. Henze , W. Nejdl. “Logically Characterizing Adaptive Educational Hypermedia Systems”, *In Proceedings of International Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems (AH’03)*, P.p. 15–29. AH2003, 2003.

[18] T. Berners-Lee, J. Hendler, O. Lassila. “The Semantic Web”. *Scientific American* 284(5), p. 34-43, 2001.

[19] T. Bray, J. Paoli, C.M. Sperberg-McQueen, and E. Maler (eds.). Extensive Markup Language (XML) 1.0 (Second Edition), 2000.

[20] S. Thompson, D. Beech, M. Maloney, and N. Mendelson (eds.). XML Schema Part 1: Structures, 2001.

[21] T. Bray, D, Hollander, and A. Layman (eds.). Namespaces in XML, 1999.

[22] T. Berners-Lee, R. Fielding, and L. Masinter (eds.). IETF (Internet Engineering Task Force) RFC 2396: Uniform Resource Identifiers (URI):Generic Syntax, 1998.

[23] O. Lassila and R. Swick (eds.). Resource Description Framework (RDF) Model and Syntax Specification 1, 1999.

[24] D. Brickley and R.V. Guha (eds.). RDF Vocabulary Description Language 1.0: RDF Schema, 2003.

[25] P.F. Patel-Schneider, P. Hayes, and I. Horrocks (eds.). OWL Web Ontology Language Semantics and Abstract Syntax, 2003.

[26] P.F. Patel-Schneider and D. Fensel. Layering the Semantic Web: Problems and Directions. In I. Horrocks and J. Hendler, editor, *The Semantic Web – ISWC 2002*, LNCS 2342, p. 16-29, Springer-Verlag, 2002.

[27] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Patel-Schneider (eds.). *The Description Logic Handbook*. Cambridge University Press, 2002.

[28] RuleML. *The Rule Markup Language Initiative*. www.ruleml.org

[29] B. N. Grosof, I. Horrocks, R. Volz and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic". In: *Proc. 12th Intl. Conf. on the World Wide Web (WWW-2003)*, ACM Press, 2003.

[30] A. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence* 104, 1-2:165 -209, 1998.

[31] J. Alferes and L. Pereira. Updates plus Preferences. In M. Aciego, I. de Guzman. G. Brewka, and L. Pereira, editors, *Proc. 7th European Workshop on Logics in Artificial Intelligence*, volume 1919 of *Lecture Notes in Computer Science*, p. 345-360. Springer, 2000.

[32] W. Marek and M. Truszczynski. *Nonmonotonic Logics; Context Dependent Reasoning*. Springer Verlag, 1993.

[33] Antoniou, D. Billington and M.J. Maher. On the analysis of regulations using defeasible rules. In *Proc. 32nd Hawaii International Conference on Systems Science*, 1999.

[34] R. Ashri, T. Payne, D. Marvin, M. Surridge and S. Taylor. Towards a Semantic Web Security Infrastructure. In *Proc. of Semantic Web Services 2004 Spring Symposium Series*, Stanford University, California, 2004.

[35] N. Li, B. N. Grosof and J. Feigenbaum. Delegation Logic: A Logic-based Approach to Distributed Authorization. In: *ACM Transactions on Information Systems Security* 6,1, 2003.

[36] G. Antoniou and M. Arief. Executable Declarative Business rules and their use in Electronic Commerce. In *Proc. ACM Symposium on Applied Computing*, 2002.

[37] G. Governatori, M. Dumas, A. ter Hofstede and P. Oaks. A formal approach to legal negotiation. In *Proc. ICAIL 2001*, 168-177, 2001.

[38] B. N. Grosof and T. C. Poon. SweetDeal: representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In *Proc. 12th International Conference on World Wide Web*. ACM Press, 340 – 349, 2003.

[39] D.Jonassen, T. Mayes & R. McAleese: A Manifesto for a Constructivist Approach to Uses of Technology in Higher Education. In Duffy, T.M., Lowyck, J. & Jonassen, D.H. (eds.): *Designing Environments for Constructive Learning Springer-Verlag, Berlin Heidelberg New York* (1993)

[40] Boud, D.: Moving towards autonomy. In Boud, D. (ed.): *Developing student autonomy in learnin, Second Edition*, Kogan Page, London (1988)

[41]G.Antoniou, D. Billington, G. Governatori and M.J. Maher (2001). *"Representation Results for Defeasible Logic". ACM Transactions on Computational Logic 2, 2 (2001): 255 – 287.*

[42] M.J. Maher (2002). *"A Model-Theoretic Semantics for Defeasible Logic", In Proceedings of Workshop on Paraconsistent Computational Logic , 67 - 80, 2002.*

[43] G. Governatori and M.J. Maher (2000). *"An Argumentation-theoretic Characterization of Defeasible Logic". In Proceedings of the 14th European Conference on Artificial Intelligence, Amsterdam, 2000. IOS Press.*

[44] H. Raiffa (1982). "The Art and Science of Negotiation". Harvard University Press.

[45] M. Maher, *Propositional Defeasible Logic has Linear Complexity, Theory and Practice of Logic Programming, 1 (6), 691-711, 2001.*

[46] A. Bikakis and G. Antoniou: "DR-Prolog: A System for Reasoning with Rules and Ontologies on the Semantic Web", 2005, Proc. *25th American National Conference on Artificial Intelligence* (AAAI-2005)

[47] G. Antoniou, D. Billington, G. Governatori \& M. Maher (2006):" Embedding Defeasible Logic into Logic Programming", *Theory and Practice of Logic Programming, to appear*

[48] A. van Gelder, K. Ross and J. Schlipf (1991): "The well-founded semantics for general logic programs". *Journal of the ACM* 38 (1991): 620—650

[49] B. N. Grosof, M. D. Gandhe and T. W. Finin: SweetJess: "Translating DAMLRuleML to JESS.RuleML" 2002. In: Proc. *International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*

[50] M. J. Maher, A. Rock, G. Antoniou, D. Billington and T. Miller (2001): "Efficient Defeasible Reasoning Systems". *International Journal of Tools with Artificial Intelligence* 10,4 (2001): 483—501

[51] Millard, D. E., Moreau, L., Davis, H. C., and Reich, S. (2000):"FOHM: a fundamental open hypertext model for investigating interoperability between hypertext domains". *In 11th ACM Conference on Hypertext and Hypermedia*, pages 93-102, San Antonio, Texas, USA.

[52] Gronbaek, K., Sloth, L., and Bouvin, N. O. (2000):"Open hypermedia as user controlled meta data for the web". *In Ninth International World Wide Web Conference*, pages 554-566, Amsterdam, The Netherlands.

[53] Kampa, S., Miles-Board, T., Carr, L., and Hall, W. (2001):"Linking with meaning: Ontological hypertext for scholars". *Technical report, University of Southampton*. citeseer.nj.nec.com/kampa01linking.html.

[54] Weal, M. J., Hughes, G. V., Millard, D. E., and Moreau, L. (2001):"Open hypermedia as a navigational interface to ontological information spaces". *In Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 227-236. ACM Press.

[55] Bechhofer, S., Carr, L., Goble, C., and Hall, W. (2001):"Conceptual open hypermedia = the semantic web?". *In Second International Workshop on the Semantic Web*, Hong Kong, China.

[56] Carr, L., Bechhofer, S., Goble, C., and Hall, W. (2001): "Conceptual linking: Ontology-based open hypermedia". *In Proceedings of the Tenth International World Wide Web Conference*, Hongkong.

[57] Brusilovsky, P. (2001):"Adaptive hypermedia". *User Modeling and User-Adapted Interaction*, 11(1-2):87-100.

[58] Henze, N. and Nejdl, W. (2001):"Adaptation in open corpus hypermedia". *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, 12.

[59] Bailey, C., Hall, W., Millard, D., and Weal, M. (2002):"Towards open adaptive hypermedia". *In Proccedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (AH 2002), Malaga, Spain.

[60] Henze, N. and Nejdl, W. (2003):"Logically characterizing adaptive educational hypermedia systems". *In International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems* (AH 2003), Budapest, Hungary.

[61] Peter Dolog and Wolfgang Nejdl: "Personalisation in Elena: How to cope with personalisation in distributed eLearning Networks". In Proc. of *International Conference on Worldwide Coherent Workforce, Satisfied Users - New Services For Scientific Information*, Oldenburg, Germany, September 2003.

[62] Nicola Henze, Peter Dolog, and Wolfgang Nejdl: "Reasoning and Ontologies for Personalized E-Learning.*Educational Technology & Society". Special Issue on Ontologies and the Semantic Web for E-learning*, 7(4):70-81,October 2004

[63] Bry, F. and Schaffert, S. (2002): "A gentle introduction into xcerpt, a rule-based query and transformation language for xml". *In International Workshop on Rule Markup Languages for Buisiness Rules on the Semantic Web*, Sardinia, Italy.

[64] Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003): "Description logic programs: Combining logic programs with description logic". *In Twelth International World Wide Web Conference*, Budapest, Hungary.

[65] M. Blochl, H. Rumershofer, and W. Wob: "Individualized e-learning systems enabled by a semantically determined adaptation of learning fragments". In *Proceeding of the 14th international workshop on Database and Expert Systems Applications*, pages 640–645, 2003.

[66] SPERO: Tele-Informatics System for Continuous Collection, Processing, Diffusion of Material for Teacher Training in Special Education. http://www.image.ntua.gr/spero.

[67] Yannis Tzitzikas, Vassilis Christophides, Giorgos Flouris, D. Kotzinos, H. Markkanen, Dimitris Plexousakis, Nicolas Spyratos**:** "Trialogical E-Learning and Emergent Knowledge Artifacts". 2006, *In the Poster Session of the 1st European Conference on Technology Enhanced Learning* (ECTEL-06),

[68] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle. "The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases". 2*nd International Workshop on the Semantic Web (SemWeb'01), in conjunction with Tenth International World Wide Web Conference (WWW10)*, pp. 1-13, Hong Kong. May 1, 2001.

[69] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis. "On Storing Voluminous RDF Descriptions: The case of Web Portal Catalogs". *In Proceedings of the 4th International Workshop on the Web and Databases (WebDB'01)- In conjunction with ACM SIGMOD/PODS,* Santa Barbara, CA. May 24-25, 2001.

[70] J. Broekstra, A. Kampman. "Query Language Definition". *On-To-Knowledge project deliverable 9.* March 2001.

[71] A. Kampman, F. van Harmelen. "Sesame's Interpretation of RDF Schema". *Aidministrator Nederland bv*. Version 1.2. April 24, 2001.

[72] J. Broekstra, A. Kampman, F. van Harmelen. "Sesame: a Generic Architecture for Storing and Querying RDF and RDF Schema". *To appear in the 1st International Semantic Web Conference (ISWC2002),* June 9-12, 2002. Sardinia, Italy.

[73] B. McBride. "Jena: Implementing the RDF Model and Syntax Specification". *In: Steffen Staab et al (eds.): Proceedings of the Second International Workshop on the Semantic Web- SemWeb2001*. May 2001

[74] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl. "RQL: A Declarative Query Language for RDF". WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA. ACM 1-58116-449-5/02/0005

[75] M. Sintek, S. Decker. "TRIPLE-An RDF Query, Inference, and Transformation Language". *In Proceedings of the Deductive Databases and Knowledge Management Workshop (DDLP' 2001).* Japan, October 2001.

[76] E. Prud'hommeaux, A. Seaborne, "SPARQL query language for RDF", *W3c Working draft* Otober 2006.

[77] D. Nute (1994). "Defeasible Logic". *In Handbook of logic in artificial intelligence and logic programming (vol. 3): nonmonotonic reasoning and uncertain reasoning.* Oxford University Press.

[78] M. Dumas, G. Governatori, A. Ter Hofstede, P Oaks/ “A formal approach to negotiating agents development” *Electronic Commerece Research and Applications*, 1,2 (2002)

[79] G. Governatori, M.J. Maher, “An argumentation-theoretic characterization of defeasible logic, in: W. Horn (Ed.), ECAI 2000. *Proceedings of the 14th European Conference on Artificial Intelligence*, Amsterdam, IOS Press, 2000.

[80] G. Antoniou, M.J. Maher, D. Billington, “Defeasible versus logic programming without negation as failure”, *Journal of Logic Programming* 42 (1) (2000) 47–57.