Anneli Edman

Combining Knowledge Systems and Hypermedia for User Co-operation and Learning
Combining Knowledge Systems and Hypermedia for User Co-operation and Learning

ANNELI EDMAN
Division of Computer Science
Department of Information Science
ABSTRACT


Hypermedia systems and knowledge systems can be viewed as flip sides of the same coin. The former are designed to convey information and the latter to solve problems; developments beyond the basic techniques of each system type require techniques from the other type. Both system types are frequently used in learning environments, and to a different extent utilise user co-operation.

A knowledge system consists of a formal representation of a domain theory enabling automated reasoning to take place within the domain. Since a formalisation cannot generally reproduce all relevant knowledge, the user’s co-operation is needed to obtain a well functioning system. To perform well in this co-operation, the knowledge in the system must be accessible and transparent to the user. Transparency can be achieved by means of explanations. In a learning environment transparency and co-operation are vital because the user needs to be active whilst the reasoning is being carried out - to be able to learn how to perform the problem solving.

To achieve transparency we introduce the notions of inferential context and conceptual context. These allow explanations to be composed at various levels of abstraction and from different perspectives and not only exploit a formalisation, but also informal descriptions of the domain knowledge. This facilitates the user’s learning of the domain knowledge and thus his/her ability to co-operate with the system in the problem solving.

We integrate techniques from knowledge systems and hypermedia in a system architecture. The architecture deals with formal and informal knowledge. The formal knowledge is used for the formal reasoning, which is based on knowledge systems techniques; the informal knowledge is exploited in this reasoning to generate explanations in different media. The relations between the formal and the informal theory are administered by a metatheory. The metatheory carries out the reasoning in the system and the communication with the user, i.e. the presentation of the explanations and the integration of the user's contribution in the reasoning. The system architecture is transparent, modular and promotes clarity, maintainability and reusability.

Key words: Knowledge-based systems, hypermedia, context, explanations, user learning, co-operation.

Anneli Edman, Division of Computer Science, Department of Information Science, Uppsala University, Box 513, SE-751 20 Uppsala, Sweden

© Anneli Edman 2001

ISBN 91-506-1526-2

Printed in Sweden by Uppsala University, Tryck & Medier, Uppsala 2001
Distributor: Department of Information Science, Uppsala University
To my family
This thesis is based on the following articles:

Article 1

Article 2

Article 3

Article 4

Article 5

Article 6

The articles are reprinted with permission of the publishers.
Acknowledgements

First, and most important, I would like to thank Andreas Hamfelt, my supervisor, without whose constant support I would not have been able to finish my thesis. Working with Andreas has been very rewarding and I have really appreciated our discussions.

Thanks are due to Sten-Åke Tämlund who first made me interested in research and employed me in his research group, UPMAIL. He also wrote a conference paper with me.

Anna-Lena Johansson and Agneta Eriksson-Granskog have been very important for my research training. They very early included me in their research group and have supported me throughout. We wrote a conference paper together, my first, and later we jointly wrote a book. Recently, Anna-Lena and I have worked in the same project, which has been very valuable to me.

Other co-authors of papers I have had much support from and enjoyed working with, are (in the chronological order in which I have co-operated with them): Olle Olsson, Lena Sundling, Åsa Lindman, Andréa Bender-Öberg, Anne Håkansson, and Sabine Koch. Especially Lena has been very important for the early work, which the thesis is based on.

The experts I have worked with in developing systems have been important to me and for my research results: Stig Ledin at the Swedish University of Agricultural Sciences in Uppsala, Göran Rosén of the Swedish Environmental Protection Agency, Gunnar Wassén, Uppsala University, Inge Bruce, Alunda Vårdcentral (Alunda Health Centre) and Mikael Olsson, Gothenburg University.

Moreover, I wish to thank my sponsors since I began my Ph.D. education: STU via UPMAIL, the International Energy Agency via a project with the Swedish University of Agricultural Sciences, the National Board of Education where my contact persons, Göran Nydahl, Kersti Hjertqvist and Lena Nydahl were valuable discussion partners. I am also grateful to Stockholm Vatten by way of Mikael Olsson, Gothenburg University.

I am indebted to my mentors outside the department for being such good role models: Ragnhild Lundström, Department of Economic History and Christina Gustafsson, Department of Education. My sincere thanks go to everyone who has read and commented on articles I have written or my thesis. Among them Keith Clark, Åke Hansson, Mats Cedvall, Sabine Koch, Tore Risch, Anne
Håkansson and Marianne Ahrne deserve special recognition. I am grateful to Suzy Lindström for correcting my English.

I am very grateful to Ingrid Klint and Torsten Palm, my longstanding colleagues at the department, for doing all they could to make my work easier. Tomas Andersson and Torsten Jonsson have been a great help in taking over my teaching load at the end of my thesis work and Gunilla Klaar for helping me with the practical problems around the dissertation. Ann Gunnarsson, Anna-Lena Kåberg, Mats Nordström, Lars Oestreicher, and Lars-Göran Svensk have also been a great support during the years. But I am indebted to all my colleagues at Computer Science as well as from others in the Department of Information Science, at the university, at Jönköping International Business School, and recently, at the Swedish National Defence College. My sincere apologies to anyone I have missed.

Thank you also to my former Masters students, especially Madeleine Kylberg, Jakob Lindström, Narin Mayiwar, Annika Widmark and Josef Öhlmér, who have investigated areas related to my thesis in their Masters’ theses.

Furthermore, support and encouragement from my friends have been important and I particularly want to thank Åsa Löwen and Ulla-Britt Strömberg who have been my discussion partners on research and writing. Also thanks to Veronika Granath for returning from Nicaragua in time.

Finally, I want to thank my large family. Their infinite patience and constant support have been enormously important. My husband Eric Hardenfeldt has always respected me and encouraged me in my work although it has affected our life for many years and limited the time we have spent together. My daughter Katja Edman has always believed in me and eagerly waited for me to finish my Ph.D. thesis. To my great joy Katja has not been discouraged by the number of years it has taken me to complete my thesis and has just started her own research education. My mother and father have always encouraged me to continue my education and been proud of me. My mother has really tried to make me concentrate on my research. My father has set a good example in seeing the practical applications of my research results. My brothers have also been good supporters, Inge Bruce who helped me with examples for one of my articles, and Owe Olsson who discussed the outline of my thesis. My mother-in-law Wailet Hardenfeldt and her sister Inez Lagerqvist have always been pleased with my successes. Everyone else in the family has also backed me. My thanks go out to all of you!
Contents

- Thesis overview .................................................................................................................. 9
  1.1 Summary...................................................................................................................... 10

2 Knowledge systems and learning.................................................................................... 14
  2.1 Knowledge systems ..................................................................................................... 14
  2.2 Knowledge systems in a learning environment............................................................ 18
  2.3 Co-operation and transparency ................................................................................... 22
    2.3.1 Categories of explanations.................................................................................... 23
    2.3.2 Classification of knowledge in model-based explanations....................................... 28

3 Hypermedia systems and learning................................................................................. 30
  3.1 Hypermedia systems ................................................................................................... 30
  3.2 Hypermedia systems in a learning environment.......................................................... 34
  3.3 Co-operation and transparency ................................................................................... 39
    3.3.1 The usability of the system engine ........................................................................ 40
    3.3.2 The usability of the content ............................................................................... 43
    3.3.3 The usability of the system’s structure ............................................................... 44

4 Programming methodology............................................................................................. 46
  4.1 Logic programming ..................................................................................................... 46
  4.2 Metalogic programming ................................................................................................ 50

5 Survey of papers ............................................................................................................. 52
  5.1 A case study concerning explanations .......................................................................... 52
    5.1.1 From a knowledge system shell to an educational expert system ......................... 52
    5.1.2 Article 1: Design issues concerning explanations in an educational expert system - a case study ........................................................................................................ 58
    5.1.3 Article 1: Experiences ......................................................................................... 62
  5.2 Informal domain context and hypermedia................................................................. 65
    5.2.1 Reproducing domain knowledge ......................................................................... 65
    5.2.2 Article 2: A basis for a system development methodology for user co-operative systems ........................................................................................................... 66
    5.2.3 Article 2: Analysing the domain context knowledge ........................................... 71
    5.2.4 Article 3: A system architecture for knowledge-based hypermedia .......... 75
    5.2.5 Article 3: Discussion ......................................................................................... 81
5.3 Case studies concerning learning environments................................. 84
  5.3.1 Article 4: A combined knowledge and hypermedia system to
attain educational objectives......................................................... 85
  5.3.2 Article 5: DLW – a learning environment for lake water diagnosis...... 87
  5.3.3 Article 6: Pedagogical issues for DLW – an interactive
learning system.............................................................................. 89
  5.3.4 Discussion.............................................................................. 94

6 Scientific contribution and related work.............................................. 96
  6.1 Scientific contribution.................................................................... 96
  6.2 Related work................................................................................ 100
    6.2.1 Dealing with domain context..................................................... 100
    6.2.2 Separating knowledge in different theories.................................. 102
    6.2.3 Co-operation between user and system...................................... 104
    6.2.4 System transparency................................................................ 105
    6.2.5 Learning environments............................................................ 108
    6.2.6 Support methodologies for program development...................... 113
    6.2.7 Tools for developing knowledge systems and hypermedia systems... 114

7 Future work...................................................................................... 116
References......................................................................................... 117
1 Thesis overview

Hypermedia systems and knowledge systems (also called knowledge-based systems), can be viewed as flip sides of the same coin (Rada & Barlow, 1989b). The former systems are designed to convey information and the latter to solve problems; developments beyond the basic techniques of either system type require techniques from the other system. Both system types are frequently used in learning environments and utilise user co-operation to a different extent.

A knowledge system consists of a formal representation of a domain theory enabling automated reasoning within the domain (a subject or a field). Generally a formalisation cannot reproduce all relevant knowledge in a real life domain and, consequently, it is only partial with respect to the domain theory. Therefore, a well functioning system often requires that knowledge be furnished from outside, in practise from the user. To enable the user to perform well in this co-operation, the knowledge in the system must be accessible, which does not tend to be the case in knowledge systems today. The user should, consequently, be able to examine the knowledge and therefore the system must be transparent and interactive. This is particularly important if the system’s knowledge is to be used for learning. The possibility of co-operation in problem solving is also vital in learning environments since the user usually needs to be active during the reasoning to be able to learn how to perform the problem solving.

Transparency can be achieved by means of explanations. In most attempts the base for the explanations is a symbolic formal representation of the knowledge in the system. Such a representation will lack the domain knowledge that cannot be symbolically represented, and also the knowledge not needed for the system’s reasoning, both necessary for a more thorough understanding of the domain. The knowledge missing can be understood as the contextual knowledge for the formalisation, which often resists formalisation attempts. If so, the knowledge needs to be presented in its natural and informal form for the user to be able to interpret it.

Knowledge systems are good at solving problems by means of formalised knowledge but not designed to present informal domain knowledge. Hypermedia systems, on the other hand, are designed to present information but not to solve problems. The information in a
hypermedia system can easily be expressed at various media; i.e., it can reproduce informal knowledge. In order to obtain a transparent system that can co-operate with the user in the problem solving, and from which the user may learn, one may combine the two techniques.

1.1 Summary

Within this thesis a system architecture combining techniques from knowledge and hypermedia systems is proposed. For knowledge systems, the combination of knowledge and hypermedia systems should offer an improved user interaction, reducing the weakness caused by excluded knowledge, and for hypermedia systems, improved navigation by means of problem solving reducing the weakness associated with static links. The intention with the proposed system architecture is that it should facilitate a user’s learning of domain knowledge and thus his/her ability to co-operate with the system when conducting the problem solving and vice versa. Furthermore, the system should be transparent and modular, and it should promote clarity, maintainability and reusability.

The proposed architecture reproduces a more complete domain theory than in ordinary knowledge systems or hypermedia systems. Part of the domain knowledge is represented as a formalisation using knowledge system techniques. This knowledge can be utilised to reason with to generate new knowledge within the domain. Knowledge needed to understand the formalisation and knowledge that cannot be formalised, the contextual knowledge, is reproduced utilising hypermedia techniques. Furthermore, the system can include the user’s interpretations of the domain knowledge, thereby enlarging the system’s knowledge.

The system architecture is divided in three theories. The formal knowledge is represented in a formal theory, e.g., as rules or objects. The contextual knowledge is reproduced in an informal theory, in is most natural form as text, pictures, sounds, animations, etc. The formal theory is used for the formal reasoning; the informal theory is exploited in this reasoning to generate explanations for the user. The relations between the formal and the informal theories are administered by a metatheory. This metatheory carries out the reasoning in the system, which comprises the problem solving, and the communication with the user, i.e., the presentation of the explanations and the integration of the user’s contribution in the reasoning. One advantage with this division of the knowledge into three distinct theories is that the system attains a high degree of modularity. Furthermore, the
respective modules are expressed in their natural way, which promotes clarity. This is vital for maintainability, i.e., facilitating updating and for making alterations to the system.

To promote reusability in system development programmable schemata for the metatheory, the formal theory and the informal theory are specified. These schemata are independent of the application domain and defined at the system structure level.

The user and the system co-operate in the problem solving. When the system needs the user’s interpretation of the informal domain theory the user is asked for a contribution. Naturally, the user’s contribution can be to provide some data, which is a common way for a user to interact with a system. But the system architecture allows a more advanced interaction. The user can contribute by giving a truth value to a statement, which the system has not succeeded in proving. Another alternative is that the user can decide whether a statement is equivalent to another statement, which the user can define or the system has knowledge about. The contributions are interpreted by the system and included in the system’s knowledge. Thus, the architecture facilitates incremental knowledge acquisition.

If the user is to be able to perform well in the co-operation it is vital that the system is transparent. To this end, advanced explanations are needed. To provide these explanations the notions of inferential context and conceptual context are introduced. The inferential context mirrors the problem solving through a continuous printout of what the system is doing at that moment. Furthermore, the domain’s inferential context can be presented in different abstraction levels by using a context tree, which is related to the problem solving within the domain. The information displayed is based on selected domain properties. We argue that these properties could quite coherently reproduce the domain knowledge needed for communicating the system’s knowledge to a user. These properties have been shown to be suitable for diagnosis and classification problems. Moreover, such a classification of properties is a support in the knowledge acquisition phase. The conceptual context is presented through different kinds of figures showing relations between objects and conclusions and conceptualisations of the domain. Consequently, the various explanations are composed at various levels of abstraction and from different perspectives and do not only exploit a formalisation but also informal descriptions of the domain knowledge.
Some important issues when developing a system supporting learning are that the system can depict a coherent domain theory, that the knowledge can be presented in different ways, and that the knowledge can be easily accessed and preferably tailored to the current user. Furthermore, it is vital that the user is active and has a goal to reach. The formal and informal theories in the system architecture may reproduce a coherent domain theory. Through the explanations of the inferential and conceptual context the knowledge can be displayed in different ways as various overviews of the domain. The links between the chunks of information are not static, but rather, are dynamically computed. This means that it is possible to adjust the explanations to the current user. Furthermore, the system can handle mixed-initiative dialogues; i.e., both the user and the system can control the interaction. It is essential that a mixed-initiative dialogue is provided in a learning environment. Such dialogues facilitate access to the system’s knowledge. It is obvious that the user is active, since the user and the system co-operate in the reasoning. The purpose of this collaboration is to solve a problem; thus the user has a goal.

The thesis is organised as follows: First the reader is introduced to knowledge systems and hypermedia systems, respectively, in connection with learning, transparency and co-operation, in Sections 2 and 3. Since metalogic programming will be used for the implementation of the system architecture a brief description of both logic programming and metalogic programming is given in Section 4. After these introductory sections a survey is made of the articles in Section 5. In 5.1 a discussion regarding design issues for co-operative systems that are suitable for learning is started. In article 1 a knowledge system designed for supporting learning which has improved explanation capabilities compared to ordinary knowledge systems is presented. The discussion ends with arguments for including informal domain knowledge in the design of a knowledge system. Informal domain context and hypermedia are investigated in 5.2. In article 2 a system architecture is described for co-operative systems that are also suitable in a learning environment. This system deals with informal and formal knowledge in the reasoning and presentation of the system’s domain knowledge to the user and it co-operates with the user in the reasoning. This system architecture can be used for implementing knowledge-based hypermedia systems, a topic which is discussed in article 3. The result from combining knowledge systems and hypermedia systems is an intelligent hypermedia system where the shortcomings of both system types are reduced. In Section 5.3 three case studies, which have been described in articles 4-6 are presented, all of which demonstrate the need for informal domain knowledge in systems supporting
learning. Section 6 consists of a discussion concerning the scientific contribution and how the work is related to other research. In Section 7 ideas regarding further work are presented.

Six articles are included in the thesis. For Articles 1, 2 and 3 the first author has provided the greatest input to the articles and has been most responsible for the direction taken by the work and the design and writing of the paper. For Articles 4 and 5 the contributions of the co-authors have been equal. For Article 6 Bender-Öberg is the first author. Both authors have equally contributed to the pedagogical ideas presented in the article, and decided the focus in the article.

Article 1

Article 2

Article 3

Article 4

Article 5

Article 6
2 Knowledge systems and learning

In this section knowledge systems are described and some experiences from utilising this type of system in learning environments are presented. Knowledge systems are investigated in relation to co-operation and transparency, and important issues for systems supporting learning are then discussed.

2.1 Knowledge systems

A knowledge system (often called an expert system) consists principally of a knowledge base, an inference engine, an explanation mechanism, and a user interface (see Figure 2.1). In the knowledge base domain knowledge may be represented as facts, heuristic rules for reasoning within the domain and metarules, i.e., rules about rules, structured objects such as frames, decision tables (Lucardie, 1994) and models of the domain (see, e.g., Andersson, 2000). During the problem solving the inference engine uses the knowledge base and input data through the user interface to reach a conclusion, which may, for instance, be the diagnosis of a malfunction or the classification of a substance (see further, e.g., Hayes-Roth, Waterman & Lenat, 1983; Durkin, 1994). The input and the conclusions reached for a consultation are stored in the dynamic knowledge base and the general domain knowledge is stored in the static knowledge base. Usually, a knowledge system is able to explain its conclusions. The explanation mechanism generates explanations based upon both the general and the case-based dynamic knowledge in the knowledge base.

![Figure 2.1. A knowledge system’s architecture.](image-url)
The knowledge base itself cannot be domain independent, but several parts of the knowledge system might be. In that case the system could serve as a *knowledge system shell* and be utilised as a tool when developing knowledge systems. In principle, only the knowledge base is implemented when developing a new system utilising a shell.

Let’s study a domain comprised of descriptions of two different types of fruits, which could be used for a classification, see Figure 2.2.

The fruits mandarins, *Citrus reticuláta*, and bitter orange also called Seville orange, *Citrus aurántium*, are citrus fruits. Both mandarins and bitter oranges are small, for citrus fruits, and their colour, once ripe, is orange. Their shape is round but they are somewhat flattened. What differ between the two fruits are the peel and the flavour. The peel on the mandarin is thin and peels easily. In contrast, the peel on the bitter orange is medium thick and not so easily peeled. The flavour of the mandarin is quite sweet in contrast to the bitter orange, which is bitter as the name suggests. You can eat the mandarin fresh, and also preserved. The bitter orange is mainly used preserved as marmalade and you do not eat it fresh.

Figure 2.2. Description of two types of citrus fruits.

The description in Figure 2.2 is presented in an informal way. If this domain knowledge should be captured in a knowledge base the knowledge has to be represented in a formal way. One way to represent it is in the form of rules. A rule consists of a conclusion and premises; the conclusion is true if the premises are fulfilled. First order logic may be used for a formalisation in the form of rules. The following symbols may be used in first order logic formulas:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>∀X</td>
<td>for all X</td>
</tr>
<tr>
<td>∃X</td>
<td>there exists at least one X</td>
</tr>
<tr>
<td>↔</td>
<td>if and only if</td>
</tr>
<tr>
<td>←</td>
<td>if</td>
</tr>
<tr>
<td>&amp;</td>
<td>and</td>
</tr>
<tr>
<td>v</td>
<td>or</td>
</tr>
<tr>
<td>¬</td>
<td>not</td>
</tr>
</tbody>
</table>

Let’s now formalise some of this knowledge in such a way that the formalisation can be used for a categorisation of fruits, see Figure 2.3.
Rule | Formalisation | Interpretation
--- | --- | ---
1 | ∀X(fruit(X, mandarin) ← (citrus_fruit(X) & appearance(X, mandarin) & flavour(X, sweet))). | For all X, X is the fruit mandarin if X is a citrus fruit, with the appearance of a mandarin and the flavour is sweet.
2 | ∀X(fruit(X, bitter_orange) ← (citrus_fruit(X) & appearance(X, bitter_orange) & flavour(X, bitter))). | For all X, X is the fruit bitter orange if X is a citrus fruit, with the appearance of a bitter orange and the flavour is bitter.
3 | ∀X(appearance(X, mandarin) ← (colour(X, orange) & shape(X, 'round and flattened') & peel(X, thin) & size(X, small))). | For all X, X has the appearance of a mandarin if X’s colour is orange, its shape is round and flattened, X’s peel is thin and it is small.
4 | ∀X(appearance(X, bitter_orange) ← (colour(X, orange) & shape(X, 'round and flattened') & peel(X, 'medium thick') & size(X, small))). | For all X, X has the appearance as a bitter orange if X’s colour is orange, its shape is round and flattened, X’s peel is medium thick and it is small.

Figure 2.3. Formalisation and interpretation of a description of two citrus fruits.

The inference engine could ask the user about a fruit and then use the formalisation to determine whether it corresponds to one of the fruits. Traditional inference approaches are to perform forward reasoning from the input to get an answer or backward reasoning trying to verify a hypothesis. When reasoning forwards, for instance from the information referring to fruit1, that the colour of the fruit is orange, the shape is round and flattened, the peel is medium thick, the size is small, the flavour is bitter and it is a citrus fruit, the system can conclude that the fruit is a bitter orange, see Figure 2.4. When, on the other hand, the system uses backward reasoning it chooses a hypothesis, in this case the fruit is a mandarin, and investigates whether the necessary conditions are fulfilled. Figure 2.5 illustrates this kind of reasoning for fruit2, based on the observations that the fruit is a citrus fruit, the colour is orange, the peel is thin, it is round, flattened and small. Forward reasoning is suitable when the number of relationships between the input and the data is limited and when it is necessary to get quick responses to changes in the input data, e.g., in a system for monitoring. Backward reasoning is appropriate when there are more input data than possible conclusions (Gonzalez & Dankel, 1993).
The ability to present the general knowledge in the system and the problem solving for an actual case is an important feature in a knowledge system. This presentation can be done from explanations made when reasoning why a question is asked and after the problem solving by asking how a conclusion was reached (Scott, Clancey, Davis & Shortliffe, 1977). According to the formalisation in Figure 2.3 the user could get a presentation of the rules for determining whether a fruit is a mandarin or a bitter orange if the user wants to know why the system is asking if the fruit is a citrus fruit. If the system concludes that
the fruit is a bitter orange, the user can ask how this answer was reached and get a printout of all the rules that have been used to arrive at this result, in the example, the rules (2) and (4).

2.2 Knowledge systems in a learning environment

It has been claimed that knowledge systems offer an ideal basis on which to implement tutorial programs. The reason is that they contain domain knowledge, the knowledge is often represented in a declarative way and separated from the interpreter who uses this knowledge and, moreover, the systems can offer explanations (Wenger, 1987). The idea of using a knowledge system for education was tested within the MYCIN project, which is one of the earliest and best-known knowledge system projects. MYCIN diagnoses infectious diseases in the blood and recommends appropriate therapy (van Melle, 1978; Buchanan & Shortliffe, 1984). In the GUIDON project Clancey (1979) studied the possibility of transferring the knowledge of MYCIN-like systems to students. The GUIDON system was developed with the intention of tutoring medical students. In GUIDON the domain knowledge was furnished by the MYCIN system and the teaching expertise was provided as a knowledge system on its own. The teaching expertise was independent of the knowledge base content.

During a session, GUIDON selects a case and describes it to the student. Thereafter the student’s role is to act as a diagnostician, asking questions to gather information and proposing hypothesis. The system intervenes when the student asks for help or when the system estimates that the student is not following the right track, according to MYCIN’s knowledge.

MYCIN’s rules were used for forming tests, guiding the dialogue with the student, summarising results and modelling the student’s understanding of the domain (Clancey, Shortliffe, Buchanan, 1979). Including meta-knowledge about the representations facilitated the flexible use of the rules (Davis & Buchanan, 1984).

When investigating whether the expertise in MYCIN could be transferred to the students it was found that they had difficulty understanding and remembering the rules. Furthermore, if a student performed a diagnosis in a different way from MYCIN the system indicated that the diagnosis was incorrect, regardless of whether the hypothesis
was reasonable. The reasons for this was that the experts’ diagnostic knowledge was not represented in the system and the rules in the system represented compiled expertise, lacking the knowledge needed to understand them to comprehend the rules (Clancey, 1983). In the NEOMYCIN system these shortcomings were addressed. Clancey and his team found that it was necessary to separate the strategic diagnostic knowledge from the domain facts and rules (Clancey & Letsinger, 1981). Therefore, they implemented two different subsystems, one for the strategy knowledge and one for the domain knowledge.

The reasoning strategies for medical diagnosis were hierarchically organised as metastrategies, which were used to control another knowledge system in the object domain. The knowledge base containing the domain rules was altered, e.g., the strategic information was taken away and control information concerning the order of rules or the order of premises in the rules was explicitly described.

The user’s part is important in NEOMYCIN since the main purpose of the system is that the user himself should solve problems, i.e., make diagnoses, according to the domain knowledge in the system. This problem solving does not have to be performed in the same way as in the MYCIN system. Therefore, it is necessary to offer the user explanations, both during and after a session.

To conclude, the experiences from these two projects showed that it is important to separate the domain knowledge from the knowledge of problem solving and also from the pedagogical knowledge. Moreover, the user’s learning can be facilitated by giving access to different kinds of explanations regarding the knowledge of both the domain and the actual problem solving.

There are a lot of examples of knowledge systems constructed as learning environments offering different kinds of aid to facilitate understanding and to tutor the subject, and where the students are actively working with the subject in a similar manner to in NEOMYCIN. Within mathematics, for example, there is a system APLUSIX supporting students at solving polynomial factorisation using the technique “learning by doing” (Nguyen-Xuan, Joly, Nicaud & Gelis, 1993) and a system assisting in performing deductions needed to solve mathematical problems of a symbolic nature (Forcheri & Molfino, 1993). SEPIA allows the students to investigate and explicate the role that qualitative reasoning plays in quantitative problem solving in sciences such as physics (Plötzner, 1993). Moreover, an expert system has been implemented to diagnose students’ misconceptions of science/engineering (Abdullah & Wild, 1995). In a research project
that aims to develop an intelligent computer-based learning environment for industrial
application the system JONAS has been implemented (Borges & Baranauskas, 1997).
The system enables shop-floor workers to test and put into practice new philosophies of
work in the context of manufacturing. In dental education RaPiD is used, a knowledge-
based assistant for the design of partial dentures (Davenport, Fitzpatrick, Randell,
Hammond & de Mattos, 1995). The InforMed Professor is a clinical instruction system
of breast disease diagnosis and management (Rahilly, Saroyan, Greer, Lajoie, Breuleux,
Azevedo & Fleiszer, 1996), which supports the integration of the declarative and
procedural knowledge needed in skilled clinical performance. For students in computer
science a knowledge-based help system has been implemented for a UNIX operating
system that assists students in accomplishing a given task and, at the same time, tutoring
the student (Fernandez-Manjon, Gomez-Hidalgo, Fernandez-Chamizo & Fernandez-
Valmayor, 1997).

Several tools have been implemented to support teachers in constructing course material.
An expert tutoring system for teaching computer programming languages has been
implemented through the World Wide Web as a tool for teachers and students (El-
Khouly, Far & Koono, 2000). The teachers can co-operate to put the learning material
together for one or more programming languages and then the student can use it as a
learning environment. A tool for automatically generating course material has been
implemented by Nussbaum et al (Nussbaum, Rosas, Peirano & Cárdenas, 2001). The
teacher makes use of stored knowledge to choose the relevant content and then the system
generates exercises from this knowledge. A simulator, controlled by a knowledge system,
interacts with the student during the exercises, and adjusts to the pupil’s needs. The
system has been implemented for pre-school children within mathematics. The REDEEM
system (Ainsworth, Grimshaw & Underwood, 1999) is an authoring environment for
intelligent tutoring systems. REDEEM allows teachers to utilise existing computer-based
material as a domain model and then combine this with their teaching expertise. The
system has underlying teaching knowledge that is overlaid by authored teaching strategies.
A domain-independent exploratory environment, called KREEK, has been developed into
which different knowledge bases may be loaded for perusal, manipulation and direct
inquiry (Purchase, 1993). It is even possible for the user to create and change knowledge
bases in the environment.

Moreover, knowledge system techniques, such as blackboard models, cased-based
reasoning and simulation of models, are utilised in tutoring systems. Blackboard models,
where the domain knowledge is represented in different knowledge sources, have been used for constructing training tutors in, for instance, second language learning (Dimitrova & Dicheva, 1997) and dynamic instructional planning (Gutierrez, Fernandez-Castro, Diaz-Ilarraza & Elorriaga, 1993). In case-based tutoring the system augments the user’s memory by providing analogical cases to use in solving a problem, which the user can utilise as guidelines (Namatame, Tsukamoto & Kotani, 1993). Case-based strategies for teaching have been implemented in several systems within different subjects such as natural science, business and jurisprudence (Schult, 1993). Gilligan et al (Gilligan, Shankararaman, Hinton & May, 1998) found that a case-based approach was appropriate within the veterinary medical domain and could have some value as a teaching aid. A simulation system models a dynamic system and can be used to study the behaviour of the model by altering the input parameters and studying its output. In some systems the user can alter even the model. The user’s task is typically to discover the rules, which govern the behaviour through scientific investigation (Baranauskas & de Oliveira, 1995). Scientific discovery is a rather difficult process and puts a large part of the responsibility for the knowledge acquisition process on the learner. The learner needs to have sufficient prior domain knowledge and be able to organise the learning process, and must have the capacity to choose and abstract from the quantities of information generated by the system (Goodyear, Njoo, Hijne & van Berkum, 1991). Experiments have shown that, when students are given an assignment to accomplish whilst learning in this manner, the exploratory environment is beneficial (de Jong, Härtel, Swaak & van Joolingen, 1996). The potential in developing simulation based learning material has been examined in the DELTA programme, within the EC, as a subproject, SIMULATE (de Jong, 1991). Special tools for implementing pedagogical simulations can be found, e.g., MELISA (Pernin, Guéraud & Coudret, 1996). Simulation models have been used to teach, for instance, the economics of developing countries (Kinney & Adams, 1995), photoelasticity by simulating experiments (Soares & de Andrade, 1996), transmission lines within physics (de Jong, Härtel, Swaak & van Joolingen, 1996), troubleshooting of simple electronic circuits (White & Fredriksen, 1990) and troubleshooting of a complex radar device (Kurland, 1989).

More recent presentations of tutoring systems do not tend to categorise the systems. Earlier a system was often introduced as a knowledge system, knowledge-based system, expert system or intelligent tutoring system, etc. Frequently knowledge system technology is used in one way or another, but this is not clearly pointed out. Interest has shifted towards what kind of domain knowledge can be or has to be included; the kind of
knowledge representation form that may be suitable; how the knowledge can be extracted by the user; how it can best be presented to the user, whether the user, the system or both should be in charge of the dialogue; how the system could supervise the user and when the system should intervene; whether the system, the user or both of them acting in cooperation should perform the problem solving; if it is suitable to utilise a game design; whether some tests should be included, etc.

2.3 Co-operation and transparency

Let’s return to the example about fruit. The informal domain knowledge in Figure 2.2 has a greater information content than its formal counterpart in the Figure 2.3. This is understandable, since a transformation from a rich language, such as natural language, to a restricted one, such as logic, means that some information will be lost.

As mentioned earlier, the formalisation in Figure 2.3 could be part of a knowledge base and function well in the system’s reasoning. It would be possible to formalise some more of the informal description of the fruits, but then the formalisation might become too detailed and, consequently, the reasoning become ineffective. But it is not possible to fully formalise a complete domain theory within the system, with the exception of the most trivial ones. It is, for instance, impossible to formalise, in any meaningful way, that a flavour is bitter.

Even if the domain knowledge in itself is formalised, the reasoning may benefit from cooperation with the user. For example, a system was implemented to debug logic programs (Edman & Tärnlund, 1983). The system utilised a specification for the program, formalised in first order calculus. To check whether a program had computed the right result, the system formally derived the correct output from the specification and a given input. We found that the user often had to restrict the specification of the program, since it was possible to derive a class of programs from the first specification. A design where the system could utilise the user’s knowledge about the program that was being debugged, during the derivation, and restrict the specification for the program when needed, should be better than trying to mechanise the debugging completely.

Usually it is not possible to formalise the whole domain theory and therefore it is necessary that the user co-operate in the problem solving to get a well functioning system.
The goal of a knowledge system should not even be to automate the problem solving but to optimise the performance of the joint system of user and knowledge system at problem solving (Stolze, 1991).

If the user is to be able to co-operate in the problem solving, the system must be transparent. Then the user will be aware of what kind of knowledge is included in the system, when the system lacks the knowledge it needs or is reasoning beyond its knowledge. Furthermore, acceptance of a transparent system is readily obtained because one can understand how the conclusion was reached and what it is based on.

It is of course particularly important that the system is transparent when implementing knowledge systems for learning. Learning and understanding are closely linked together and according to Schank (1986) “explanation is critical to the understanding process”. This is in agreement with the opinion, mentioned in Section 1, that transparency in knowledge systems design relies on explanations.

2.3.1 Categories of explanations

One can see three different methods used for explanations:
• Explanations in the form of canned text
• Rule-based explanations
• Explanations in second generation expert systems, which will be referred to here as model-based explanations.

_Canned text_ is a presentation of the domain theory, or a part of it, to the user in the form of natural language. Once the formal domain theory has been represented in the system, e.g., as rules, the canned text is associated with each part of the knowledge base or even each rule, explaining what the relevant part or rule is doing. When the user wants to know what the system is reasoning about, the system merely displays the text associated with what it is doing at the moment. In the example above, canned text explanations could be based on the interpretation related to every rule, see Figure 2.3.

The advantage with canned text is its simplicity. No problems will occur when generating text for presenting information to the user, because the text can be prepared carefully and in advance, and immediately displayed when the user so wishes. There are, though, several
objections to the canned text approach. All questions and answers must be anticipated in advance and all these answers have to be provided in the system, (see, e.g., Swartout, 1981). For large systems this is an almost impossible task. Furthermore, it is difficult to guarantee the concordance between what the system does and what it claims to do, since the formal knowledge in the system and the canned text associated with this knowledge can be changed independently. A third problem is to keep the context coherent when updating the text strings and another is that the system has no conceptual model of what it is saying. Thus it is difficult to use this approach if the system provides more advanced sorts of explanations.

*Rule-based explanations* were described briefly in Section 2.1. This form of explanation may be based on the static knowledge in the knowledge base or on dynamic knowledge for a special consultation. The explanation could be simply a printout of a rule or, more commonly, the content of the rule is rewritten and displayed in restricted natural language, quite like the interpretation in Figure 2.3.

Static knowledge can be presented if the user wants to know why a special question is posed by the system when the reasoning is being made. Then the explanation is based on the rule the system is trying to execute at that moment, or every rule in the knowledge base where the answer is a premise. It may also be possible to get a printout of the rules where these rules in turn are used as a premise. Let us illustrate this using the example in Figure 2.3. If the user asks why the system needs to know the size of the fruit, the system finds out that it is a premise in both rules (3) and (4) and these rules should be presented. If the user wants to know more, the user has to decide which conclusion to elaborate further, the appearance for the case of the mandarin or bitter orange. The rules where the chosen parameter is used are then presented, in this case, rule (1), if the user had decided to learn more about the appearance of mandarin, see Figure 2.6.

The dynamic knowledge presented is that obtained in a special session, after the conclusions have been reached. The user can get an explanation of how the system reached a special conclusion, i.e., a printout of the proof tree, see Figures 2.4 and 2.5. It is possible that the system has made several conclusions and then the user will decide which proof tree is to be displayed.
What size has the fruit?

Why “size”? 
The fruit has the appearance of a mandarin IF  
  the fruit’s colour is orange AND the shape is round and flattened  
  AND the peel is thin AND the size is small  
The fruit has the appearance of a bitter orange IF  
  the fruit’s colour is orange AND the shape is round and flattened  
  AND the peel is medium AND the size is small  

Why “appearance of a mandarin”? 
The fruit is a mandarin IF  
  the fruit is a citrus fruit AND the appearance is that of a mandarin  
  AND the flavour is sweet  

Figure 2.6. “Why” questions.

Rule-based explanations are easy to implement and of course they are consistent with the knowledge base and the system’s reasoning. If the rules are rewritten there may be some difficulty in getting syntactically correct sentences for all rules, but this can be solved. What is more serious is that presentations of this kind are criticised for being too extensive and too detailed. A possible way to solve this problem is to transform the proof tree so it gets more condensed, without losing important information (Eriksson & Johansson, 1985). But the problem that the rules contain a mixture of domain knowledge and control information still remains. The user may not grasp the system’s knowledge through this kind of presentation.

*Model-based explanations* are a completely different way of dealing with explanations, and one which has impact on the whole knowledge system design. According to Chandrasekaran and Swartout, “knowledge systems based on explicit representations of knowledge and methods with information about how and from what their knowledge was obtained, are the foundations for producing good explanations” (Chandrasekaran & Swartout, 1991). They argue that explanations can be as important as the conclusions themselves, which ought to be the case for knowledge systems in a learning environment. The general idea in their research is that the more explicitly the knowledge underlying a system’s design is represented the better explanations the system can give.

Model-based explanations are one of the issues in second generation expert systems. The idea behind second generation expert systems is to represent both deep and shallow knowledge and explicitly represent the interactions between both knowledge types (Steels,
Then the knowledge system has a model of the domain, which can be causal, functional, structural, etc. (Steels, 1990). Two different approaches for model-based explanations can be seen, the first representing the knowledge used for the explanations in a more abstract way than in early systems, and the second utilising a different kind of knowledge to explain the conclusions (David, Krivine & Simmons, 1993). The last approach is often called providing reconstructive explanations.

An example of the first of these types of explanation is NEOMYCIN, which gives an abstract representation of the knowledge that is the base for the explanations. Swartout and his associates are also working from this point of departure. In his work with the XPLAIN system (Swartout, 1981; Moore & Swartout, 1988) and the Explainable Expert System, EES, a further development of XPLAIN (Swartout & Smoliar, 1987; Swartout & Moore, 1993), Swartout has tried to find a way to capture the knowledge that was used by the programmer to write the program to improve the system’s explanations. The problem solving knowledge is explicitly represented and separated from the domain knowledge in XPLAIN. XPLAIN is a digitalis therapy advisor, adjusting digitalis dosing in cardiac patients. The domain model represents facts in the domain, which, for instance, may be states and causal relations. The system’s problem solving knowledge is represented by domain principles. These consist of three parts, namely a goal, a prototype method, which is an abstract method that describes how a goal can be achieved, and a domain rationale, which at a general level indicates the cases when the domain principle is to be applied. The XPLAIN system has a program writer, which is an automatic programmer. The program writer creates an expert system by generating a refinement structure, which is comprised of successive refinements of goals into prototype methods using the domain model and the domain principles. When an explanation is asked for, this is generated by an examination of the refinement structure and the step currently being executed. The EES approach is to have a dialogue with the user and employ feedback from the user to guide subsequent explanations utilising knowledge in XPLAIN (Swartout, Paris & Moore, 1991).

Work with reconstructive explanations, the second approach, is done, for instance, by the research groups in which Chandrasekaran and Wick belong. An interesting result from Chandrasekaran and his associates is a generic task methodology used for building knowledge systems to enhance explanations and consequently also transparency. The central idea in the generic task methodology is that there are generic tasks in knowledge-
based problem solving and that each task is characterised by the following (Chandrasekaran, Tanner & Josephson, 1988):
- A task specification in the form of generic types of input and output information.
- Specific organisation of the knowledge particular to the task.
- A family of control regimes that is appropriate to the task.

In their work they have identified four generic tasks for problem solving and these can be characterised as specified above. The tasks are classification, state abstraction, knowledge-directed information passing, and design by plan selection and refinement. Chandrasekaran et al claim that such a typology is very useful in explaining the control strategy for a system’s problem solving. The main idea is a conceptual decomposition of the problem-solving knowledge into agents. These agents combine knowledge with ways of using it and they are responsible for explaining the decisions they make. Justifications of the system’s knowledge can be represented separately as a causal story of the reasoning (Tanner & Keuneke, 1991). The historical development of the generic task methodology is well described in (Chandrasekaran & Johnson, 1993).

Wick and Thompson (1992) have stated the necessity to divide a knowledge system into two parts. One part comprises the knowledge used for the problem solving and the other the description of this activity, which they argue, is a complex problem-solving activity that depends on both the actual line of reasoning and additional knowledge of the domain. They have implemented a system called REX (reconstructive explainer) for generating reconstructive explanations. The system is divided into two different parts, one is the system performing the reasoning and the other one is the knowledge-based explanation system. The explanations generated by the explanation system can involve a complete reconstruction of how the expert system reasoned to reach a conclusion, with, for instance, new associations and the introduction of new objects not in the actual line of reasoning. In REX there is an interface between the knowledge system and the explanation system. This interface is defined by a knowledge specification, which is represented as a graph of potential solutions or hypotheses along with information about possible transitions between these hypotheses. To find a path through the knowledge specification when constructing an explanation, the A* algorithm is used. Since the reconstructive explanations are built on more information than the trace of the execution, they offer more flexibility than, e.g., rule-based explanations. Furthermore, the explanations can be tailored to a particular user (Wick, 1993).
Implementing model-based explanations leads to a different approach to the knowledge represented than those earlier knowledge systems and to the form in which it is represented and even how the system reasons with it. The results are that more and deeper domain knowledge is included in the system, that the explanations can be tailored to different users more easily and that they can be presented in a distinct way. Drawbacks may be that the system architecture gets more complicated and the knowledge needed in the system can be difficult to elicit leading to a protracted knowledge-acquisition phase.

2.3.2 Classification of knowledge in model-based explanations

What kind of knowledge is to be displayed through the explanations? Within model-based explanations there are some suggestions concerning the knowledge that the system ought to present to the user.

Clancey (Clancey, 1983) characterises knowledge needed for explanations in MYCIN - which he argues is applicable to other knowledge-based systems - in three categories:
(Cl I) strategy, which refers to a plan according to which goals and hypotheses are ordered in problem solving
(Cl II) structural knowledge, which consists of abstractions that are used to index the domain knowledge
(Cl III) support knowledge, justifying the causality between the problem features and the diagnosis, which may be somewhat redundant to the diagnostic associations.

Swartout (Swartout, 1981) found, through a series of informal trials, that the questions a user would like to pose to a knowledge system, in this case Digitalis Advisor (mentioned in Section 2.3.1), were the following:
(Sw I) questions about the methods the program employed
(Sw II) justification of the program’s actions
(Sw III) questions involving confusion about the meaning of terms.

Chandrasekaran, Tanner and Josephson (1988) state that the explanations relevant to the problem solving are the main issue. They categorise these explanations into three types:
(Ch I) how well the data match the local goals, which describes how certain decisions are made and what piece of data is used to arrive at a specific conclusion
(Ch II) justification of knowledge, which involves justifying fragments of domain knowledge by explaining a certain part of the knowledge base, generally not based on a special case.

(Ch III) explanation of control strategy, which clarifies the behaviour of the problem solver and the control strategy used in a particular situation.

There are resemblances between Clancey, Swartout, and Chandresekarar et al in terms of the categories of knowledge needed for explanations. These categories can be combined into four groups:

- problem solving strategy, including (Cl I), (Sw I) and (Ch III)
- justification of the system’s domain knowledge, including (Cl III), (Sw II) and (Ch II)
- local problem solving, including (Cl II) and (Ch I)
- term descriptions, including (Sw III)

These four groups will be elaborated upon further in Section 5.2.3.
3 Hypermedia systems and learning

This section starts with an introduction to hypermedia systems. Some of the advantages and disadvantages of using hypermedia systems in learning environments are presented. Then hypermedia systems are discussed, particularly in terms of co-operation and transparency.

3.1 Hypermedia systems

Hypermedia is a technique for presenting information in different media. A hypermedia system is a system that presents chunks of information, stored in nodes, in a non-linear way. This is illustrated in Figure 3.1 where the nodes contain information about citrus fruits. Different types of media, e.g., text, pictures, animations, and digitised speech, can be used in such a system. Hypermedia and hypertext systems are alike in the sense that the nodes in the systems are linked, but in hypertext systems the information consists of text only. A link can be seen as a relation between components, e.g., cards, frames, documents or articles (Halasz & Schwartz, 1994). Concept maps can also be considered as hypermedia components (Gaines & Shaw, 1995). In multimedia systems different media are used, but the link associations are not necessary.

Shneiderman (1989) has proposed “three golden rules” to determine whether hypertext/media is suitable for an application. This technique is adequate if a large body of information is organised into a number of fragments, if these fragments relate to each other, and if the user only needs a small fraction of information at any time.

Simplified, a hypertext system can be seen to consist of three levels (Campbell & Goodman, 1988), see Figure 3.2. The first one is the presentation level, which is the user interface. A hypertext abstract machine (HAM) containing the links and the nodes is the second one. The third is the database level where storage, shared data and network access are taken care of. The database is an ordinary one and not of interest here.
The mandarin is a citrus fruit. It is a small fruit and once ripe it is orange in colour. Its shape is round, but somewhat flattened. The peel is thin and is easily peeled. Mandarins are edible. There are similarities between mandarins and bitter oranges.

Citrus fruits can be utilized in many different ways. Lemon can be used as a culinary spice and for the production of citric acid. Orange is mainly eaten fresh, in segments and it is drunk as juice. Clementine, mandarin and grapefruit can be eaten fresh or preserved. Bitter oranges are mainli used preserved as marmelade.

The bitter orange, also called Seville orange, is a citrus fruit. It is a small fruit and once ripe it is orange in colour. Its shape is round, but somewhat flattened. The peel is medium thick and is not so easily peeled. Bitter oranges are edible. There are similarities between bitter oranges and mandarins.

Figure 3.1. A small hypertext structure.

Figure 3.2. A three level architecture for hypermedia systems.
In the HAM the nodes and links are administered using knowledge of the form of the nodes and links, and the attributes related to them. Usually a node consists of fixed information written by a person (the author), but there are systems where the information can be computed. In some systems the nodes are typed to differentiate the various structural forms.

Nodes, or parts of nodes, are connected by links in several different ways. With the exception of connecting a document reference to the document itself, a link can connect an annotation or a comment to any information in a node, see Figure 3.3. An anchor is a special kind of link associating some position or region within the node, or in another node, with a link. The use of an anchor link is demonstrated in Figure 3.1, where there is such a link from “edible” in the node with information about mandarins to a special part of the node presenting how to utilise citrus fruits, in this case “Clementine, mandarin and grapefruit can be eaten fresh or preserved”. Furthermore, among other properties, links can provide organisational information.

![Figure 3.3. An annotation connected to the remedy for scorbutus.](image)

Just like nodes, links can have names and types. In some systems the links can be turned on or off. Links may be one or two way. When having a two way link, for instance, between the nodes A and B, it is possible to follow a link from A to B, and also from B to A. There are systems offering multiway links. Some systems even have super-links to connect several groups of many nodes (Nielsen, 1995).
The interface level, in the three level structure, deals with the presentation of information in the HAM. Interesting issues connected to this task are, for instance, what commands should be available to the user, how the nodes and links should be presented and whether there should be some kind of overview diagram describing the user’s whereabouts in the network at any moment.

A hypermedia network can be browsed in three ways (Conklin, 1987). The user can follow the links and open nodes successively. Second, the system can offer a search for some keyword, string, etc. Third, the user can navigate around in the hypermedia system using a browser that displays the network graphically. Unfortunately the terms browsing and navigating are often used synonymously (Waterworth, 1992). In contrast, a more appropriate way of phrasing this would be: Navigation is towards a specific or general location and is used when one seeks a single known fact, something that fulfils a certain criterion, whereas browsing is conducted in the hope of discovering something interesting (Woodhead, 1991).

The Dexter Group, consisting of many of the designers of the early hypertext systems, has worked with defining hypertext interchange and standards. This work has led to the Dexter Hypertext Reference Model (Halasz & Schwartz, 1994), which gives a basis for comparing systems as well as for developing interchange and interoperability standards. The model provides a standard hypertext terminology coupled with a model of the important abstractions commonly found in hypertext systems.

The work with the Dexter model has resulted in a more detailed architecture model than the model presented above. The Dexter model divides a hypertext system into three layers, the run-time level, the storage level and the within-component level, as shown in Figure 3.4. The storage layer models the basic network structure of nodes and links. This layer describes a database that is composed of a hierarchy of components interconnected by relational links. A component consists of a content specification, a set of attributes, a presentation specification and a set of anchors (Grønbæk & Trigg, 1994). The within-component layer can be compared with the database level in the model given above and this layer can be considered to be outside the hypertext model. The run-time layer manages the interface offering the user the possibility to access, view, and manipulate the network structures.
### 3.2 Hypermedia systems in a learning environment

When hypermedia was introduced the potential for using this technology for learning was soon discovered. There are several tests showing that for some certain aspects of teaching hypermedia “works” (Whalley, 1993). In university education hypermedia plays an increasing role (Pohl & Purgathofer, 2000). Some are very enthusiastic, stating that hypermedia is “… the hottest thing to happen to education since the arrival of the microcomputer” (Moore, 1994). But there are also several reservations that need to be added to the first enthusiastic expectations of the technology. Some will be mentioned below. The advantages with using hypermedia in a learning environment are related to the information that can be reproduced in such systems and to the different ways to attain this information.

The technology facilitates the storage of large amounts of information. This information may well represent different views of a domain (Conklin, 1987). The student no longer has to accept just one author’s view, which is common when using a single textbook, but has the possibility to establish an opinion of his/her own from critically studying the information (Nielsen, 1995). Moreover, it is easy to extend the information. According to Whalley (1993) the most important pedagogical feature with hypertext is its malleability, where the possibility to let the information change over time is one important aspect.

---

<table>
<thead>
<tr>
<th>Runtime layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation specifications</td>
</tr>
<tr>
<td>Storage layer</td>
</tr>
<tr>
<td>Anchoring</td>
</tr>
<tr>
<td>Within-component layer</td>
</tr>
</tbody>
</table>

Figure 3.4. The Dexter Reference Model.
Hypermedia is a technique for presenting information in different media. It is obvious that a student needs to interact with information in different media (Waterworth, 1992) and therefore one can imagine that hypermedia systems are appropriate tools for promoting thought rather than straightforward learning by rote.

The user can easily access the information in the hypermedia system by navigating or browsing, and the initiative to reach the information is the user’s. By browsing, the user may get a global view of the material offered and after it has been presented it is possible to trace this material backwards (Conklin, 1987). Moreover, it is easy for the user to find different kinds of associations by tracing the links. Navigation, on the other hand, can support the search for specific information.

As stated above, one advantage a hypermedia system has for learning is the potential it has to offer a great quantity of information. At the same time, however, this can be a disadvantage, because of the fragmented nature of the information. The fragmentation may lead to difficulties perceiving the intended structure of the information (Whalley, 1993), which is necessary if the user is to be able to learn from the information. Grouping the material into clusters, though, promotes an abstract view (Woodhead, 1991). In general, hypermedia systems fail to support some of the important principles for understanding and memorisation and need to be supplemented with, e.g., learner guidance (Hammond, 1993).

In a hypertext system, for instance, the simple pointer and hierarchical structures may be more limiting than the implicit relationships in ordinary material, such as textbooks (Whalley, 1993). A textbook presents, e.g., an overview of the material through a table of contents and an index. With a hypertext system learners may encounter difficulties if the system does not offer an overview of the information available and explain the organisation of the material. Most of today’s hypertext documents offer an overview or provide a table of contents (Pohl & Purgathofer, 2000).

Regarding the navigation and browsing, it is the user who governs the search for information. It is often seen as important whether the system, the user or both control the interaction. When both the system and the user direct the interaction, it is said to be a mixed-initiative dialog. Such dialogs are directed towards an exchange of questions and answers between the user and the system, where the system must be able to respond to the student and eventually try to understand what the student is trying to do or what he/she
knows (Wenger, 1987). In a learning environment it is common to use mixed-initiative dialogues since they seem to best imitate the interplay between a tutor and a student. Ordinary hypermedia systems do not offer this kind of dialogues.

Furthermore, there is a danger that if students control their own search they might visit different nodes rather randomly, choosing what happens to attract their attention at a particular moment. This may be fun and an easy approach to adopt, but in terms of what is learnt, this strategy is not necessarily advantageous (Hammond, 1993). When evaluating the use of a hypermedia information system called Perseus, it was realised that different kinds of learners reacted quite differently (Yang, 2000). Mindful novice learners could and did take control of their own learning and actively worked with the hypermedia application, whereas some students were not prepared to accept the responsibility for learning on their own and they were unwilling to make the necessary investment of time and effort.

Even if the user searches in a more thorough way, the links offered may not be suitable to the task the student is supposed to solve, or to the individual in question. Independent of strategy there is the danger that one will get lost in hyperspace (Conklin, 1987). Furthermore, if the user knows what he/she is looking for, following links is not an effective strategy, and if the user does not know exactly what he/she is looking for, it is more efficient to follow linear sequences than to jump around (Waterworth, 1992).

An important observation is that hypermedia systems have not been designed as pedagogical tools. Indeed, the design has been focused on information retrieval rather than learning (Whalley, 1993) and learning is not the same as retrieving information (Hammond, 1993). The system offers material for viewing, but there is no guarantee that the user will adopt a productive strategy for understanding and learning from this material (ibid.).

When implementing a hypermedia system as a learning environment some important issues should be considered. Since such a system is passive, i.e., all initiative must come from the user; it is necessary to engage the user in searching for information to facilitate effective learning. Therefore, it is important that the student has some kind of goal to reach and can use the hypermedia material in this task (Jonassen, 1993). “Problem solving occurs when there is a goal to be reached, when the method for reaching it is not yet known, and when attempts to reach the goal are being made” (Wærn, 1989). When trying to solve a problem the student has to choose the sequencing when searching for the
relevant material and organise the problem solving activities, both are viewed as metacognitive skills (Hammond, 1993). A student’s metacognition during learning can even lead to his/her adoption of a new learning strategy (Wærn & Rabenius, 1985).

One problem is that the learner’s goal may not be the same as that of the author of the hypermedia system. It is often necessary to support the user with explicit guidance and perhaps some restrictions so the user can focus more easily on the relevant domain information. This guidance can take different forms, such as guided tours, recommended sequences or programmed learning where every step is defined. To get the user’s acceptance it is important that the system’s supervision is relevant to the actual problem, the domain and maybe even the user’s character.

Hypermedia technology seems particularly suitable in a learning environment where there is some flexibility in both the presentation of the material and the learning activity (Hammond, 1993). Hypermedia supports a ‘hands-on’ style of learning emphasising experimentation, discovery and synthesis, but not analysis and absorption (Woodhead, 1991). In such an environment, where exploratory tasks are facilitated, a user may learn implicit knowledge, i.e., procedural knowledge that cannot be described in a declarative way (Hammond, 1993). In general, learning-by-doing through, e.g., the performance of actions linked to a task seems to support retention of information (ibid.).

Duffy and Knuth state that hypermedia systems are very suitable when emphasising a constructivistic view of learning, if the users have the possibility to create nodes and links themselves (Cunningham, Duffy & Knuth, 1993). According to Akhras and Self (Ramberg & Karlgren, 1997) constructivists often assume that it is not possible to define knowledge in an objective way and that a student individually constructs personal knowledge from experiences. People tend to more easily remember material that they have generated for themselves than comparable material provided by someone else (Hammond, 1993). Experiences from creative hypermedia authoring, where the students developed hypermedia material themselves, have been encouraging (Wells, 1995). The students studying information technology in a Master of Arts course learnt more from implementing their own hypermedia material than from simply using preproduced examples.

Hypermedia is an appropriate technology for open learning activities, but may not be suitable for a system of the ‘drill-and-practice’ type (Nielsen, 1995), which repeats similar
actions a number of times. On the other hand, this kind of system has a rather limited usability.

Several hypermedia applications have been implemented as learning environments. One example is CESAR, which teaches hearing impaired children sign and written language (Diaz, Aedo, Torra, Miranda & Martin, 1997). The system utilises text, graphics and video when presenting stories and in exercises related to these stories. HIPERGRAFE is a hyperdocument aiming to help to teach Portuguese spelling rules (Strube de Lima & Beiler, 1997). The learner is the author, who composes multimedia pieces from an initial hyperdocument, and the system is supposed to facilitate the user’s construction of personal knowledge. Another example is a hypermedia tool designed for students of numerical analysis (Gazzaniga & Scarafiotti, 1997). The tool consists mainly of a collection of exercises and the system offers helps, hints and plain solutions to the student. The exercises are of three different types: quizzes, step by step exercises and complex exercises.

Tools for developing hypermedia systems in a learning environment are often called *authoring systems* or *authoring tools*. Authoring tools should, according to Shneiderman (1998), at least support the possibility to import articles or nodes, edit links, export collections of articles or nodes, print webs of links, and search the entire hypertext. The Microcosm system is an authoring system with an open hypermedia platform where teachers can create customised teaching materials based on sets of resources (Hall, Hutchings & White, 1995). These resources can be different thirdparty applications, produced locally or at other institutions. The datafiles, i.e., the resources, remain in the ordinary format and all link information is held in link databases. Microcosm has been utilised to develop material for several courses such as a locomotor course about joints of the human body and a course about properties of materials for engineering students. Another authoring tool supports developing and managing hypermedia courseware (Hendrikx, Duval & Olivié, 1995). Dynamic structures, queering and structural constraints can be included in the system. The tool has been used for students in several different disciplines for developing a self-study course in computer studies. HMLE is a hypermedia based learning environment for authoring courses in mathematics (Multisilta, Antchev & Pohjolainen, 1995). The system enables the teacher to use, for instance, mathematical text with formulas, mathematical tool programs, videos and interactive exercises.
It is important to consider aspects of cognitive psychology when designing programs that facilitate learning (Mayiwar, 2001). The students’ interaction with hypermedia and the hypermedia’s possibilities to generate and display complex presentations are in tune with newer cognitive theories referring to learning, but there are several problems and the research related to this so far is insufficient (Elmeroth, 1999).

### 3.3 Co-operation and transparency

Hypermedia systems facilitate information retrieval but not problem solving, in contrast to knowledge systems. The initiative is entirely the user’s - it is the user who searches for information, interprets it and makes use of it. But the access to information can, of course, contribute to the user’s understanding of the domain in question. Nielsen (1995) takes the view that “a hypermedia system works in collaboration with the user, who has the intelligence to understand the semantic contents of various nodes and determine which of its outgoing links to follow”. Firstly, we think that there is no real co-operation between a hypermedia system and a user since the system cannot take an active part in the problem solving, and can only present information in a passive way. Since a hypermedia system offers a number of information units, and links to these, the user has to maintain several tasks at the same time if he/she is to be able to grasp the domain information, which costs additional effort and requires concentration (Conklin, 1987) instead of reducing the effort required for the user. Secondly, an obvious demand for a co-operative system should be that only the relevant information is presented to the user to facilitate understanding and possibly problem solving.

In terms of transparency, there is also a major difference between knowledge systems and hypermedia systems: In a knowledge system it is important that the system’s knowledge and the problem solving are opaque to the user. In a hypermedia system, on the other hand, it is the ability to find information that matter and that the information in the system is organised so that the user can grasp it. In a knowledge system the explanation facility is able to select relevant knowledge from the system and present it in a suitable way. This is not the case in a hypermedia system, where the user chooses which node to visit and the information in the node is directly displayed to the user.

A hypermedia system’s transparency could be discussed in terms of the usability of the system. The usability of a hypermedia system can be seen as a combination of the
usability of the underlying system engine and the usability of the contents and the structure of the information in the system, as well as by how the contents and the structure fit together (Nielsen, 1995). This view of studying a hypermedia system corresponds to the problem stated by Elm and Woods as the “navigational difficulty” - the users do not know how the information in the hypermedia system is organised, how to find the information they seek or, indeed, whether this information is available at all (Dillon, McKnight & Richardson, 1993).

3.3.1 The usability of the system engine

The usability of the system engine concerns the browsing and searching mechanisms. Transparency may be achieved for a search engine through the utilisation of tools whose functionality is easily comprehended. Furthermore, the application of the tools needs to be evident and, something of great import here, the tools should assist the user to avoid him/her getting lost in hyperspace. Unfortunately, there are navigational difficulties for users when they need to make decisions about their location in an electronic information space (Dillon, McKnight & Richardson, 1993). According to Conklin (1987), graphical browsers and query/search mechanisms are the major technical solutions available for coping with disorientation in hyperspace. Navigational maps could be used as browsing tools. The advantages and disadvantages of navigational maps are discussed in the literature (see, e.g., Stanton, Correira & Dias, 2000).

Naturally, comparisons are made between reading text in hypermedia systems and reading it in books. But the underlying structure of the two media is not of comparable transparency (McKnight, Dillon & Richardson, 1991). Using the access mechanisms in a hypertext system, to be able to read the material, probably occupies the user much more than finding the appropriate information with an index in a paper text, since paper documents usually have some organisational standards referring to content pages, indices, etc (Dillon, McKnight & Richardson, 1993). In an experiment it was found that users reading hypertext spent significantly more time in the index/contents sections of documents than the users reading text on paper or word processor files (McKnight, Dillon & Richardson, 1990).

The user’s backgrounds may influence the way that they navigate in a hypermedia application. Navigation in a hypermedia system is often referred to as non-linear. When
comparing users’ selection of a linear or non-linear reading style Reed et al (2000) found that the style differed. The users with most years of computer experience took a larger number of linear steps than the others. Moreover, the users with authoring language experience took more linear steps than those with less authoring experience. Users with database or hypermedia experience had negative relationships with the number of linear steps, i.e., those with the most experience of this kind took fewer linear steps. In this experiment learning styles were also investigated in relation to linear-versus-non-linear navigation. Kolb (1984) identified four separate learning styles:

- divergers – individuals who are able to assimilate disparate observations, are oriented toward feeling, and can see things from different perspectives
- convergers – individuals who have problem-solving and decision making capabilities, and are unemotional
- assimilators - individuals who have logic and model building skills, and are able to organise information
- accomodators - individuals who are action and results oriented, seek opportunities, and are pragmatic.

The findings showed no significant correlation for learning style groups and the percentage of non-linear steps, which was not the expected result (Reed, Oughton, Ayersman, Ervin, & Giessler, 2000).

Several suggestions have been made of ways in which one can implement browsers to assist users to attain relevant information comprising, e.g., guided tours, backtracking, history lists, bookmarks, and landmarks (Nielsen, 1995). In a guided tour the user can choose the next node to follow in the tour through the hyperspace. But it should be possible to leave the tour at any point. Through a backtracking facility the user can follow the nodes he/she has visited backwards. A history list is a more advanced support than backtracking; it allows users to choose a previously visited node directly, using pictures and/or text denoting these nodes. Bookmarks provide users with the opportunity to mark those nodes he/she may want to return to later. Nodes highlighted as landmarks make the user realise which nodes are especially important. A complement to history lists is a “trail & current position indicator” (Bender, 1992) which informs the user of the hierarchical structure for the branch taken, up to and including the current location. The use of these facilities may contribute to the transparency of a hypermedia system.

Search facilities too may contribute to system transparency. Searching is more appropriate than navigation in information spaces that are large and unfamiliar (Nielsen, 1995). A full
text search to find occurrences of words specified by the user is one way of searching. The result may be an immediate transfer to the first node hit but, preferably, may also be a set of hit nodes from which the user can choose what he/she would like to look at. It may be further help to get the number of occurrences of the search string in every hit node. To be able to search for an incomplete string is also a useful facility. Using Boolean operations to apply a combined keyword search is an applicable method from database search (Conklin, 1987). These different methods are quite transparent to the user; it is easy to understand the functionality and the results from the searches.

Another method of information retrieval is to rank the nodes and links according to some measure of their importance and use the ranking in the search. For instance, a node could have a high rank if it is linked to several other nodes and a lower one if only a few links are connected to it. The nodes and links can be seen as a Bayesian inference network, where the ranking of the nodes and links are probabilities. A search in such a network is a computation using Bayes’ rule (Kreysig, 1970), where the result is a probability that a user’s information need is satisfied given a particular object (Belkin & Croft, 1992). The result is a list of objects together with their rank. Although this method is flexible because it can handle text and other kinds of objects - it can hardly be considered to be transparent to the user. To track the search it is necessary to show the probabilities in information space. Unfortunately, as stated above, searching is most suitable for large information spaces where graphical browsers do not work. Furthermore, it would be difficult to follow those computations where a lot of nodes are involved.

It is difficult to incorporate appropriate tools in a hypermedia system, which will suit different users, tasks and content. Naive users employ quite different search strategies compared to the experienced ones (Woodhead, 1991). When users have had more practice, the relative frequency with which they use different access mechanisms changes (Nielsen, 1995). Furthermore, users prefer different search facilities depending on the task to be performed, e.g., reading an encyclopaedia or conference proceedings (Wright, 1993). As far as the content is concerned, it is not possible to grasp the information using a browser when the material is richly interconnected, even for a reasonable size and complexity (Dillon, McKnight & Richardson, 1993).

Spence (1999), among others, proposes a framework for navigation which is appropriate for different environments. This particular framework deals with activities such as the registration of content, integration into an internal model or a cognitive map, interpretation,
and the formulation of browsing strategy. These activities constitute the navigational process and this process is usually iterative.

3.3.2 The usability of the content

In relation to content, usability deals with reproducing domain information in the form of nodes and links. As stated earlier, it must be possible to organise the required information into related chunks. A node consists of a chunk of information. For transparency it is preferable to express a single concept or idea in a node, since most users prefer this (Conklin, 1987). When organising the material for a hypermedia system, it is best to start from scratch instead of transferring ordinary text (Nielsen, 1995). It is possible that the whole structure of the text has to be reorganised to suit the new medium.

To achieve transparency the nodes should have the most appropriate form for reproducing the domain information. Nodes can be fixed, or semistructured containing information together with labelled fields and spaces for field values, or they can be composite, i.e., an aggregation of several nodes presented as a single one (Conklin, 1987). Experiments indicate that it is better to have many small nodes than a small number of large ones (Nielsen, 1995).

When designing the information structures and the user interface it is important to take into consideration users’ reading habits. For instance, users often ignore information that the author had hoped they would read. In experiments by Black et al (Wright, 1993), it was found that readers of hypermedia often used a glossary when they found an unknown word in a general text. However, when the text was more specialised, dealing with an unfamiliar subject, they seldom used one. But when clickable words in the text were visually cued, the frequency of accessing the glossary returned to a high level. In the case where the access to the definitions was separated from the current text, but was still on the same page, the readers chose to access these significantly less often. One conclusion is that design features can have an important influence on users’ willingness to jump (ibid.). Furthermore, when investigating reading preferences users evaluated scrolling higher than changing screen pages (van Nimwegen, Pouw & van Oostendorp, 1999). The content’s usability can, consequently, be affected by the design.
Transparency often implies a need to present a lot of information but it may also involve displaying less than the system contains, i.e., reducing the problem of overflow. In the PUSH project the metaphor “black box in a glass box” is applied (Höök, 1996; Höök, Karlgren, Wäern, Dahlbäck, Jansson, Karlgren & Lemaire, 1996). The complex inferencing of the user’s goal is hidden in the black box and a simplified view of what is going on is displayed to the user in the glass box.

For transparency reasons it is important for the user to understand the meaning of the links in a hyperspace. Usually an author puts in a link because it makes sense in terms of the semantic content of two nodes and not because of some global view of the information (Nielsen, 1995). A link can, therefore, be perceived as an associative path, but it may also denote dependencies between nodes (Waterworth, 1992). Using associative links is very flexible but it may be difficult for the user to understand the author’s associations. The user may grasp these associations more readily if the links have names and types. In second-generation hypertext systems one tries to capture the intended semantics in the relationships through link types (Nanard & Nanard, 1995).

The analogy between a hyperspace and a semantic network is straightforward (Conklin, 1987; Rada, 1990; Wang & Rada, 1995). The hypermedia nodes can be considered to represent single concepts and the links to represent the interdependencies between them. A semantic network may support the user’s understanding of the domain structure and can be used as a graphical browser (Jonassen, 1993). Semantic networks were created as a tool for representing sentences in natural language (Quillian, 1967). They are now used for knowledge representation in knowledge systems and other AI-systems. According to Woods (1975) there are some problems with semantic networks, such as how different kinds of links should be understood and that some knowledge is difficult to represent in a clear way. Some of these problems may remain, even if the node content in a hypermedia system is more comprehensive than in a traditional semantic network. Since links may be defined by an open set of associations (Jonassen, 1993), the links’ semantics, in particular, may be unclear.

3.3.3 The usability of the system’s structure

Usability of a hypermedia system’s *structure* concerns the transparency of the organisation of the content. When using a browser to navigate, the user is responsible for
controlling the search process. Therefore it is vital that the user is aware of the information structure in the system, in contrast to a traditional information system where the system has this information (Waterworth, 1992).

Overview diagrams are powerful tools to support the understanding of the structure. Nielsen (1995) describes several facilities connected to such diagrams, e.g., diagrams at various levels, fish-eye views, “footprints”, multiple diagrams and contextual cues. Normally, the whole hyperspace is too large to be displayed. Diagrams showing various levels of detail are useful to grasp the structure of the information, and can also be an aid when browsing. Using a fish-eye view means that close to the current location more details are shown in the diagram than are shown for those parts that are further away, where the information is gradually diminishing. The user can follow his/her browsing through “footprints” in the diagram, which support the understanding of location in the information space. If the information can be structured in many ways, the user can benefit from having access to different overviews. Contextual cues can divide the diagram into different areas that are related to each other by, for instance, utilising special background colours or patterns for different areas.

In experiments by Wright et al (1993) it was found that a user looked at the diagram more often if the author suggested in the text that the reader should do so. Moreover, it was better to include the diagram in the next screen page than to have a special diagram button that the user could choose to utilise.

Overview diagrams can be used before reading a text to get an understanding of the information structure, after reading it as a kind of summary, and during the reading to understand how different parts of the information are connected. Because diagrams present the information in a different form than it is presented in the nodes, where the relations between expressions are included in the text, users can test their understanding of the domain through them (Wright, 1993).

Van Nimwegen et al (Van Nimwegen, Pouw & van Oostendorp, 1999) tested two types of structure in hypertext, hierarchical and hierarchical with partial linearity. The experiments showed that users preferred a purely hierarchical structure.
4 Programming methodology

Metalogic programming is used for implementing the proposed architecture in the thesis. Therefore, logic programming and metalogic programming are introduced briefly.

4.1 Logic programming

A logic programming formalism will be used for our system architecture. A logic program is a finite set of logic sentences (Kowalski, 1982). Since the logic sentences are restricted to Horn-clauses (Kowalski, 1974), only a subset of first-order calculus is used. A Horn-clause is a universally quantified implication with only one consequence. The consequence, denoted \( C \) below, is an atomic formula. The antecedent in the Horn-clause, the premise, is a conjunction of atomic formulas or negated atomic formulas, denoted \( A_i \) below:

\[
C \leftarrow A_1 \land \ldots \land A_n, \text{ where } n \geq 0
\]

An informal reading of the sentence is “\( C \) if \( A_1 \) and \( \ldots \) and \( A_n \)”. An atomic formula is written as \( P(t_1, \ldots, t_m) \) where \( m \geq 0 \) and \( P \) is a predicate denoting a relation between the terms \( t_i \). A term may be a variable, a constant or a compound term \( f(s_1, \ldots, s_k) \) where \( f \) is a functor and \( s_i \) are terms.

A program \( p \) takes the form of a finite collection of Horn-clauses. The formalisation in Figure 2.3 is represented as such clauses in Figure 4.1. These are listed together with some facts about two of the fruits discussed, (5) - (12), i.e., Horn-clauses without a premise. The universal quantifiers in the clauses are implicit and therefore omitted.

An execution is conducted as a logical inference process for verifying a goal clause in relation to the program \( p \).

\[
p \vdash \text{goal}
\]

The goal clause is an existentially quantified conjunction of atomic formulas. The computation of a proof provides instantiation of the variables when unifying atomic formulas, which is, in fact, the result of the computation.
The proof is usually conducted as a resolution proof (Robinson, 1965), which is a refutation proof. The goal, which is an existentially quantified hypothesis, is negated. Hence the variables become universally quantified. In each resolution step two formulas are resolved to each other, where one contains an atomic formula as an antecedent (the first time it is the goal) and the other a unifiable atomic formula as a consequence. The result from the resolution step is a new Horn-clause containing all atomic formulas except the two that were unified. Thus the proof has succeeded when the result clause is the empty clause.

Usually, the proof procedure applies a depth-first search if there are several clauses that can be part of a resolution step. In a conjunction it is common to choose the left-most atomic formula first.

![Figure 4.1. Horn clauses describing fruits and facts about two fruits, denoted fruit1 and fruit2.](image-url)
In Figure 4.2 a resolution proof using the logic program in Figure 4.1 is presented. The question is whether there is a fruit described in the program that is a mandarin. The hypothesis

$$\exists Y \text{fruit}(Y, \text{mandarin})$$

is negated, which is represented as a Horn clause without a consequence, see the top node on level 1 in the figure. The resolution proof will be performed using a depth-first, left to right strategy. In every resolution step unification between the two expressions involved is performed. The unification consists of two steps; first generating unique variable names and then finding the substitutions for the two formulas so they can be unified.

In the first resolution step the goal is used together with the Horn clause (1), which is the only one that is applicable. When unifying \( \text{fruit}(Y, \text{mandarin}) \) and the consequence in clause (1) \( \text{fruit}(X, \text{mandarin}) \), \( X \) will be denoted by a unique variablename, \( X_1 \). The substitution \( X_1 = Y \) makes the two formulas identical, which means that they can be unified, and the resolution step is executed. The antecedent in clause (1) is now the goal, see level 2 in Figure 4.2. Since the execution is performed from left to right, the first clause, \( \text{citrus}_\text{fruit}(Y) \), will be in focus in the next resolution step. The strategy for choosing clauses is to take the first clause that can be unified with the current clause. In this case it is clause (5). In the unification \( Y \) is substituted to \( \text{fruit}_1 \). In the next resolution step clause (3) is utilised. The antecedent in this clause is placed ahead of the remaining clause, \( \text{flavour}(\text{fruit}_1, \text{sweet}) \), because of the depth first strategy. When trying to resolve \( \text{colour}(\text{fruit}_1, \text{orange}) \) it is not possible to unify the sentence with a clause in the program. Therefore, the system backtracks trying to find the nearest node where another clause can be used than in the first resolution. There are no alternatives to \( \text{appearance}(\text{fruit}_1, \text{mandarin}) \), but on level 2 there is another option, clause (6) could be unified with \( \text{citrus}_\text{fruit}(Y) \), substituting \( Y \) to \( \text{fruit}_2 \). This is a fruitful step leading to a refutation on level 9, under the premise that \( Y = \text{fruit}_2 \).
Figure 4.2. A resolution proof in the form of a tree.

Prolog (Colmerauer, Kanoui, van Caneghem, Psero & Roussel, 1973) was the first more widely used logic programming language. For a thorough presentation of the language, see, e.g., (Bratko, 2001; Johansson, Eriksson-Granskog & Edman, 1989; Sterling & Shapiro, 1997). The logic programming language Prolog is suitable for implementing knowledge systems, see, e.g., (Awad, 1996; Johansson, Eriksson-Granskog & Edman, 1989; Lucardie, de Gelder & Helsper, 1997; Rowe, 1988; Walker, McCord, Sowa & Wilson, 1987). There are several reasons for this (Edman & Olsson, 1985). The first one is that Horn clauses, the base for a logic program, have the same form as heuristic rules in rule-based knowledge systems. Furthermore, the reasoning in a logic programming system is comparable to the problem solving in such a system. It is also easy to construct explanations. Compare the resolution proof in Figure 4.2 with the tree in Figure 2.3, which can be used for explaining a how question. The difference is that only the
successful branch in Figure 4.2, the right one, is presented and in Figure 2.3 it is displayed as a whole tree.

4.2 Metalogic programming

To generate explanations a logic program actually studies the current execution of a program and the program itself. Then the reasoning is carried out on a metalevel. Metalogic programming is a powerful tool. Another example of a facility, which can be realised through metalogic programming, besides explanations, is an altered inference strategy to, e.g., breadth-first. Moreover, it is possible to utilise metaknowledge and evaluate conclusions on the metalevel. Furthermore, integrating a user in the reasoning calls for metaprogramming possibilities.

In metalogic programming (Bowen & Kowalski, 1982) a theory in the form of a logic program \( p \) comprising clauses is encoded as a term \( [p] \). The provability relation \( \vdash \) is formalised as a predicate \( \text{proof} (\_ , \_ ) \). This predicate is defined by clauses \( p \text{ proof} \) constituting a logic program, often referred to as a metainterpreter.

Accordingly, \( P \vdash \text{goal iff} \ p \text{ proof} \vdash \text{proof}(\[p\], \[\text{goal}\]) \)

which is sometimes regarded as a pair of opposite reflection principles for passing between the object level and the metalevel. The ways of encoding a formula \( p \) into a corresponding term \( [p] \) (cf. Gödel encoding) are discussed in the metalogic programming literature (see, e.g., Eshghi, 1987).

Basically, there are two options: (1) ground-term representation in which variables of the object language formula become ground (i.e., variable-free) terms in the encoding and (2) non-ground encoding in which object language variables become variables of the metalanguage. The latter representation makes it easier to exploit a built-in object-level reasoning mechanism (including unification) at the risk of causing confusion between variables of the different language levels.
As already pointed out, the explicit availability of the proof (computation) predicate \texttt{proof} through the interpreter clause program \texttt{p proof} in metalogic programming makes possible to monitor and control the deduction process, as well as to “customise” it.

Thus, metaprograms treat other programs as data. It is particularly easy to write metaprograms in Prolog due to the fact that both programs and data are Prolog terms (Sterling & Shapiro, 1997) and because of Prolog’s symbolic-manipulation capabilities (Bratko, 2001). A metaprogram, actually a metainterpreter, that can execute a goal with respect to the program in Figure 4.1, can be seen in Figure 4.3. The first argument in \texttt{proof} is omitted since we presuppose that the program is included. The first sentence takes care of an antecedent (goal) that is true, as in \texttt{citrus_fruit(fruit1)}, which is equivalent to \texttt{citrus_fruit(fruit1) ← true}. The second clause deals with a goal compound with “and”, the last takes a consequent and finds the related antecedent. This antecedent has to be true if the clause is to be fulfilled.

\begin{verbatim}
proof(true).
proof((A1, A2)):-
    proof(A1),
    proof(A2),
proof(C):-
    C ← A,
    proof(A).
\end{verbatim}

Figure 4.3 A simplified metainterpreter for Horn clauses.

The metaprogram in Figure 4.3 could, for instance, conclude that \texttt{fruit(fruit2,mandarin)} is true. The question for the program is then \texttt{proof(fruit(fruit2,mandarin))}. This metainterpreter could easily be augmented with, for instance, an extra argument for a proof tree, which could serve as a simple explanation facility.
5 Survey of Papers

The articles included in this thesis are summarised in the section. Moreover, the relations between the articles are presented and reflections are made upon the solutions arrived at in the different papers.

5.1 A case study concerning explanations

By developing knowledge systems we have gained a lot of experiences regarding explanations. The early work conducted by the group centred upon the implementation of Klöver, a tool for developing knowledge systems. This tool was subsequently used to develop a system for energy forestry selecting clones and one for lake water analysis, Analyse More. The three systems will be presented below. Experiences from using Analyse More are presented and discussed.

5.1.1 From a knowledge system shell to an educational expert system

*Klöver*

Klöver (Edman & Sundling, 1989) is a knowledge system shell, implemented in Prolog. The system architecture is a further elaboration of the architecture presented in Figure 2.1 (Section 2.1), referring to the communication with the user.

In most knowledge systems the interpreter puts questions to the user, when input is needed in the problem solving. This strategy means that the questions can be rather dispersed and confusing to the user. A *question generator*, which performs an initial interrogation, can overcome this problem (Sundling, 1989). In Klöver the questioning is performed in a preplanned order governed by such a question generator. This order, though, is tailored to the user’s answers. When the user has answered a question the input is checked to see which subsequent questions should be displayed, and which questions should be eliminated. If, during the reasoning, the interpreter needs the answer to a question that the user has not answered beforehand, the question and questions related to the particular one are posed. A *question base* comprises the information about the questions, containing, e.g., the relations between questions and given answers. Thus, the
architecture in Klöver includes two more subsystems, a question generator and a question base, than the architecture presented in Figure 2.1, see Figure 5.1.

![System Architecture Diagram]

5.1. Klöver’s system architecture.

All parts in Klöver are domain independent except the static knowledge base and the question base, which means that when a new system is developed only these two parts are implemented. The shell has been used for developing systems in knowledge system and knowledge management courses at Uppsala University for about 10 years.

Klöver was used when EIA, an expert system for supporting construction of dam sites, was implemented (Håkansson & Öijer, 1993). The EIA system was developed together with Lennart Strömqvist, professor in Environmental Consequences and Analysis at Uppsala University. The system assesses the impact of dam-site on the surrounding environment. It supports the user in river development projects with advice concerning the aspects that should be considered when building dams. Moreover, the user is provided with different kinds of information that may support learning within the domain. The system handles different tropical environments and is used in South America, but may also be used in Africa. Both English and Portuguese are handled by the system. Later a system called KANAL was built on top of EIA (Håkansson & Widmark, 1996; Håkansson, Widmark & Edman, 2000). The KANAL system is a knowledge acquisition tool supporting a domain expert when constructing and maintaining the knowledge base.

When Klöver was implemented there were no knowledge system shells in the market that included a question generator. Indeed, we do not know of any others as yet, despite the importance of having a user interface that is adaptive.
A system for selecting clones

Klöver has been used to implement a knowledge system for energy forestry in collaboration with experts at the Swedish University of Agriculture Sciences in Uppsala (Edman, 1990a). The expert system was a part of a project funded by the International Energy Agency, IEA, in a research program including investigations of the possibilities to develop expert systems for energy forestry (Edman, 1990b). The system selects appropriate crops for the sites under consideration.

The expert, Stig Ledin, found several advantages with the expert system technique (Edman, 1991). The main advantage related to the property that new knowledge can be added during the development and the whole lifetime of the system, and is still based on the former knowledge. The reasoning in the domain did not have to be restated from the beginning as a result of new knowledge acquired, which had often been the case earlier in research discussions at the department, according to Stig Ledin. Another advantage was that the system development improved his knowledge. The expert had to analyse his knowledge and take part in organising it during the interviews in the knowledge acquisition phase. During the testing Stig Ledin had the opportunity to experiment with his knowledge and to test new theories by changing the knowledge in the system and studying the result. Furthermore, the future collection of new knowledge could be more organised since the development of the system indicated missing or uncertain knowledge within the domain.

The system is a prototype and has not been tested with real users, i.e., farmers. In particular, the development of a genuinely useful application requires an improvement to the explanations (Edman, Ledin & Ohlsson, 1990). The prototype offers only ordinary rule-based explanations.

Experiences from using the system made it clear that the rule-based explanations were inadequate. These were appreciated by the expert but could hardly be of genuine value for a farmer, mostly because the rules only capture fragments of the domain knowledge and one cannot expect the farmers to have the context knowledge needed to understand it. Furthermore, the compiled knowledge in the rules is difficult to grasp, which is a general problem in first generation expert systems.
Analyse More

In the next system developed, Analyse More, these problems were addressed and the explanations in the Klöver system, in particular, were developed further. The rule-based explanations were extended with static canned text describing the domain context from different perspectives. When formulating the rules special attention was paid to the fact that these rules should function as a base for rule explanations (Edman & Sundling, 1991b). The Analyse More project was a part of a larger program initiated by The National Board of Education for developing software for schools in the environment protection field (Edman, 1990c). The system was developed in co-operation with experts at the Swedish Environmental Protection Agency and teachers.

Analyse More aims at supporting students in upper secondary schools in sampling and evaluating the quality of water in lakes. The student should get an increased understanding of choice of method and interpretation of data (Edman & Sundling, 1991a). The investigations are coupled to the teaching in several subjects, such as biology, physics and chemistry. The multidisciplinary nature of the system is seen as a pedagogical advantage.

The students are supposed to first visit a lake to observe and measure biological, physical and chemical indicators. Usually a group of students is split up into subgroups and different groups perform different investigations. Afterwards, the system can evaluate the status in the lake from the students’ analyses. During the system’s reasoning, the error margins for the different tests is taken into consideration (Edman & Sundling, 1991b), errors introduced by the accuracy of the equipment and the techniques. If one does not succeed in doing all the tests that are related to the system’s inferencing, the system can still give evaluations based on the incomplete data. However, the confidence to the result is influenced. A conclusion can be shown with different degrees of certainty, ranging from high probability, probability, possibility, some certainty, cannot be excluded, probably not, to definitely not.

The presentation of the result includes both reports on the various kinds of analyses, so that each subgroup can learn what their tests have shown, and a result where all analyses are combined together in a weighed measure. It is pedagogically important to obtain this feedback for the individual groups’ measurements. Furthermore, the students have the possibility of changing an arbitrary number of input values and thereafter let the system do another evaluation. By doing this they may investigate the impact of different parameters.
Another pedagogical advantage is that Analyse More supports the students in the testing (Edman & Sundling, 1991c). In the dialogue, the unit to be used for the measurement and the interval in which the value may occur are presented when the user is asked for a numerical input, see Figure 5.2. Errors in the measurements may occur for several reasons, such as the use of defective instruments and lack of experience of sampling. Some of these can be detected automatically by the system thanks to a natural concordance among parameters. When the system comes across such an inconsistency in the input, this is pointed out to the student. The student may then voluntarily change one of the inconsistent values, to a new value or omit it. Moreover, short descriptions of Swedish Environmental Protection Agency’s recommendations for the practical carrying out of tests are included in the system and may be obtained by selecting the “Method” button, see Figure 5.2 and 5.3.

![Figure 5.2. An example of a question.](image)

When information is committed to human memory it is often accompanied by additional redundant information. These details facilitate recall by providing additional paths for associations to retrieve the information (Anderson, 1985). Hence, to facilitate learning, the system offers the possibility to store additional information in form of notes in the margin, see Figure 5.4, via the “Note” button displayed in Figure 5.2.
Figure 5.3. A description of the method for measuring conductivity.

Method

Swedish standard: SS028123

Measuring principle:
The conductivity constitutes the solutions electrolytic conductivity (but is measured as its resistance) and it is expressed as inverted resistance. It is dependent on the temperature and is stated at a standard temperature, 25°C.

Figure 5.4. A note in connection to the question about conductivity.

Notes

The conductivity is dependent upon the surroundings. If the surroundings in the rainfall area are oligotrophic they give only little salt and the conductivity will be low. In eutrophic water you get high values of the conductivity. The nearness to the sea has an influence on the salt distributed by the air, especially sodium and chlorides.
5.1.2 Article 1: Design Issues Concerning Explanations in an Educational Expert System - A Case Study

The purpose of the article was to gather information about knowledge systems in a learning environment. Therefore, the user interface is in focus, with especial attention being paid to the explanations provided by the system.

An investigation of the explanations in Analyse More is presented in the paper. This is based on a categorisation of knowledge for explaining the problem solving activity proposed by Chandrasekaran et al (Chandrasekaran, Tanner & Josephson, 1988), already presented in Section 2.3. The different knowledge types are:

Type 1: (Ch I) how well the data match the goals
Type 2: (Ch II) justification of knowledge
Type 3: (Ch III) explanation of control strategy

The explanations of Type 1 and 3 are dynamically generated in run-time. Type 2 explanations are generally not based on a special case. The three knowledge types will be related to the different kinds of explanations in Analyse More.

Explanations of Type 1, *how the data match goals*, are of two different kinds in Analyse More, presentations of answers to “why” and “how” questions, respectively. When displaying explanations the rules are slightly rewritten to make them easier to understand. A “why” question is answered through a printout of all the rules where the answer is a premise. It is also possible to get a presentation of the rules where these rules in turn are used as a premise, which Figure 2.6 (Section 2.1) illustrated. Through this facility the user can investigate the whole knowledge base.

A “how” question can be answered once the problem has been solved and shows how the current solution is reached. In the printout it is stated whether a truth-value of a premise is established through user supplied information or if the value is inferred by the system. How the system has concluded that the chemical and physical evaluation of a lake is eutrophic (nutritious) is presented in Figure 5.5.
It can be inferred that the chemical_physical_evaluation is eutrophic with high probability because:
- phosphorus > 20 is user supplied and
- water_transparency < 5 is user supplied and
- water_colour = green is user supplied and
- pH > 7 is user supplied and
- condition_of_oxygen is eutrophic is inferred (at least probable) and
- condition_of_chlorophyll is eutrophic is inferred (at least probable) and
- condition_of_conductivity is eutrophic is inferred (at least probable)

Figure 5.5. Presentation of a rule in response to a “how” question.

The quality of the rule-based explanations is highly dependent upon the knowledge base. Both the structuring of the rules and the design of the knowledge base are important. In Analyse More the ambition has been to formulate each rule so it describes a meaningful chunk of the domain knowledge and, at the same time, has a premise of limited complexity. The rules then serve as a suitable base for explanations, but the number of rules has remarkably increased. When designing the knowledge base the organisation of the rules is the main issue. The rules have been organised in order to reproduce the expert’s decision making provided that they are grouped together according to his reasoning strategies. Within each such group the rules are sorted in descending order of priority.

Explanations of Type 2, justification of knowledge, describe why the rules in the knowledge base are true. In Analyse More this is done through two kinds of canned text. In connection with “why” questions it is possible to get a kind of text definition, a supplement to the rules. The user can ask why a second time to reach the next level’s definition. In Figure 5.6, the first definition refers to the amount of reed. Then follows definitions for the reed zone, plants in general, and finally, the importance of different groups of parameters for the overall analysis.

The system’s domain knowledge can also be justified by providing references to relevant literature. The student may, after further reading, be able to examine the validity of the system’s knowledge. References to more comprehensive descriptions are included in the canned text presenting methods, e.g., Swedish Standard: SS028123 in Figure 5.3, reached via the “Method” button.
The lake is eutrophic if the reed, *Phragmites Communis*, is growing thickly.

In a eutrophic lake there is a remarkable amount of different species in the reed zone. There are only a few species in an oligotrophic lake.

The number of plants is dependent on the access of nutrient and also on the shape of the lake, whether the strand is steep or shallow. However, "Analyse More" does not deal with the shape of the lake.

The biological evaluation of a lake is primarily founded on the phytoplankton, periphyton algae, animals in the sediment and plants. Next in priority are animals on the waterside, zooplankton, fishes and finally birds.

Figure 5.6. Definitions connected to the “why” button.

Explanations of Type 3, *explanation of control strategy*, are performed in several different ways in Analyse More. For the problem solving the importance of different indicators and the order in which they are used are included, both for a special consultation and, more generally, for all the different consultations. Furthermore, overviews of the domain knowledge are offered.

Before or after a consultation it is possible to get information about the different groups of indicators in the knowledge base, and their relative importance. For instance, the importance of the eight categories involved in deciding the biological status of a lake (see Figure 5.6) is evaluated on a scale from 1 to 5, where 5 denotes the highest importance.

The importance of parameters is included in the rules and can be seen in presentations of “why” answers during the interrogation. After the problem solving, the importance appears when displaying how the result was inferred. This is not so obvious when only one rule is involved as in Figure 5.5, where all premises related to the current conclusion were true. But suppose that the conductivity of the water was not measured. Several other rules will then try to reach a conclusion. Every rule is connected to a certainty factor reflecting the importance of the parameters involved in the premise. The stepwise evaluation of the conclusion is presented in Figure 5.7.
It can be inferred that the chemical_physical_evaluation is eutrophic with some certainty because:
phosphorus > 20 is user supplied

It can be inferred that the chemical_physical_evaluation is eutrophic with possibility because:
optical_attributes eutrophic is inferred (at least probable)

It can be inferred that the chemical_physical_evaluation is eutrophic with possibility because:
pH > 7 is user supplied or
pH is close to > 7 is user supplied and
condition_of_oxygen eutrophic is inferred (at least probable)

It can be inferred that the chemical_physical_evaluation is eutrophic with probability because:
condition_of_chlorophyll eutrophic is inferred (at least probable)

Figure 5.7. Presentation of rules used to infer a conclusion.

Furthermore, an abstract overview of the parameters involved in concluding a specific outcome is included, see Figure 5.8. The boxes denote conclusions the system has inferred. Different outcomes utilise different sets of parameters. This kind of explanation is supposed to give the student a holistic view of the relations between conclusions and parameters.

Figure 5.8. The connection between parameters for the conclusion the lake is eutrophic.
Analyse More is a learning environment and the intention is that the system should support students with different levels of prior knowledge. The explanation given for the problem solving is one way of offering support. However, to be able to grasp a domain supplemen ting context knowledge is important. Therefore, the system offers additional knowledge in the form of descriptions of the methods used for measurement and sampling, see Figure 5.3. Notes in the margin provide another form of context knowledge, see Figure 5.4. Furthermore, the student can get explanations of domain specific terms used by the system, see Figure 5.9. The information included in the question, unit and interval for the current measurement, is also a kind of context knowledge, see Figure 5.2.

A water is oligotrophic when it is poor in nutritment. Eutrophic is the opposite term to oligotrophic.

Figure 5.9. Explanation of the term oligotrophic.

5.1.3 Article 1: Experiences

Compared to other knowledge systems used in learning environments at the time Analyse More was certainly competitive. The system was quite advanced with an interface that tailors the questions to the user’s answers and offers different kinds of explanation. Analyse More is reliable, it never fails to give solutions and is able to evaluate them. Furthermore, the inference engine can find discrepancies between observations, consider margins of error for different tests and reason with incomplete data. Analyse More is robust, in spite of the fact that the two subsystems, the chemical-physical one and the biological one, are very different from each other. The chemical-physical subsystem deals with few parameters and the relations between these parameters are rather well established, but the margins of error can be considerable. The biological subsystem, on the other hand, consists of a large number of parameters, the relations between which are not easily represented, and yet for which there are no error margins to include.

As described, Analyse More comprises rule-based explanations and additional context knowledge, chiefly in the form of canned text. The system was tested several times before and after its release, by teachers involved in the project and by us in the project team. In two other tests we observed the users when working with the system to get a picture of how they used the system; of special interest was their use of the explanations. In the first
one teachers of the Natural Resource Use Programme used the system. One day half of the group made the physical and chemical measurements and the other half the biological observations, then the next day this order was reversed. The first day the groups cooperated by using each others’ results to get a complete evaluation of the lake consulting Analyse More and the next they could use their own results. In the second test a class of students in year 2 in upper secondary school used the system. The class had previously made an excursion to a lake. In their report of the excursion, the evaluation of the lake utilising Analyse More should be included. The students had performed different kinds of measurements and observations and they exchanged results with each other to enable them to make an adequate evaluation since nobody had performed all tests.

The experiences from the two tests were the same, though the user groups were so different both in age and knowledge. All users were focused on getting the result from Analyse More and not so interested in the explanations provided during the interrogation. After the first session most of the users consulted Analyse More again, testing with different input values. They were then recommended to test the different buttons and study the domain knowledge provided. Hence these users utilised more of the system, but they only had enough patience to investigate the information in relation to a few of the indicators asked for. The canned text in the form of definitions, descriptions of methods, and notes was understandable and also appreciated. However, the rule-based “why” explanations were confusing and had to be explained. The “how” explanations were more understandable, but had still to be discussed verbally. The additional knowledge provided in the pull-down menus, e.g., the overviews was hardly used at all.

The system’s user manual describes the rule-based explanations provided. The users had access to manuals, but hardly any of them chose to use them, either before or during the sessions. Thus it was deduced that it was easy enough to use the system for the problem under consideration without reading the manual. The menus were mainly used to control the system’s behaviour, to start a consultation, for example, or store data on file and read stored data. Other facilities reached via the menu, e.g., providing information about the potential values of the indicators and overviews, were seldom utilised. Even the facility to list the database, which refers to solving the current problem, was rarely used.

Nevertheless, the users were satisfied with Analyse More. But one should observe that they used Analyse More mainly as a tool to get an evaluation of the status of a lake, and
not as a tool to learn how to perform such an evaluation. Furthermore, they had confidence in the system’s way of inferring the conclusions.

Moreover, the scope of the system entails co-operation between groups, e.g., by sharing results from the analysis and comparing results. Hence, we observed collaborative work and eager discussions about the results. Encouraging such dialogue is beneficial to effective learning, according to constructivists (Lim, 2001).

Analyse More works well for problem solving but not as a learning environment. The user interface may be improved; e.g., it may be better to move some of the facilities from the menus to buttons so the users would not have to break off from what they were doing during the consultation. It is also possible to control the user’s behaviour by giving them tasks that entail a thorough search for information, as discussed in Section 3.2. But essential problems will still remain. Firstly, the domain context provided by canned text is fragmented and it may be problematic for the user to develop knowledge of his/her own utilising the system (Whalley, 1993). Moreover, the canned text approach can be considered as a theoretically weak attempt to implement explanations. One of the reasons is that the system does not have access to a coherent informal description of the domain knowledge, only to isolated pieces. Furthermore, it is difficult to analyse the relation between the rules and the canned text. Since this kind of representation is not flexible, one problem is that it is not possible to use the information in different ways in a system, the information cannot be re-used in different contexts.

Secondly, the rule-based explanations are difficult to understand even though the rules in Analyse More were specially structured. One rule may be fairly understandable, but the problem is that a single “why” explanation can involve several tens of rules. The number of rules for a conclusion was also increased by the design of the rules. Information about the groups of rules has to be presented on a highly abstracted level (Edman, 1994), which could be combined with presentations of what the systems is doing at the moment.

To conclude, a system architecture more powerful than the one in Analyse More is needed. One requirement is that the domain knowledge can be reproduced more thoroughly than through rules and canned text. We think, though, that it is possible to improve the ideas using text, and other representations as pictures, sounds etc., connected to objects in the knowledge base to describe the domain knowledge. But the intention is not to improve the canned text approach by extending text with other media. The
improvement consists of reproducing a coherent informal description of the domain knowledge together with formal domain knowledge. The informal description should serve as a basis for generating explanations and the formal knowledge constitutes the basis for problem solving. A system architecture should include knowledge concerning these two knowledge types and how they relate to each other.

5.2 Informal domain context and hypermedia

In this section the reasons why it is important to reproduce informal domain knowledge and how it could be reproduced in a system are discussed. In Article 2, we present a system architecture for co-operative systems, which is also suitable for a learning environment. This system architecture may be used for implementing knowledge-based hypermedia systems, which is the subject in Article 3. A selection of important issues described in Sections 2 and 3 are discussed in relation to the proposed system architecture.

5.2.1 Reproducing domain knowledge

As stated, in a system architecture for knowledge systems it is vital that the domain knowledge can be reproduced to an extent that corresponds to the purpose of the system. All domain knowledge cannot formally be captured since the various forms of knowledge communicated by human beings are richer than formal representations. The question arises, therefore, how these forms could best be exploited in a knowledge system. A formalisation of domain knowledge in a system can be supplemented with informal descriptions of domain knowledge. This informal knowledge would relate to and complement the knowledge in the formalisation and thereby extend the knowledge in the system. Moreover, since a formalisation is unable to reproduce all knowledge in the formal theory a formalisation is, in general, only partial with respect to the domain theory. Therefore, a well functioning system requires that knowledge be furnished from the outside, in principle from the user. In the course of doing this, the system must be capable of adequately presenting the informal knowledge.

Thus, the system and the user co-operate in the problem solving. As stated in Section 2.3, optimising the problem solving capabilities of the system and the user together could be
the goal, rather than to fully automate the problem solving. To enable the user to do well in
the co-operation, a prerequisite is that the knowledge in the system is accessible. This
holds for both formal and informal knowledge descriptions. The user should, conse-
quently, be able to investigate the knowledge and, in a learning environment, get
information, which supports learning, i.e., the system should be transparent and
interactive.

The knowledge reproduction forms for the formal and the informal knowledge in the
system will differ since they describe domain knowledge in different ways. Transparency
can be enhanced if the knowledge representation and reproduction forms are chosen so
the knowledge can be represented in the system and be presented to the user in a natural
way. Naturally one must be able to reason with the knowledge.

5.2.2 Article 2: A basis for a system development methodology for user
cooperative systems

In the second article included here, we propose an architecture for user co-operative
systems, like knowledge systems. Important issues in such systems are to reproduce the
domain context, explain the context for the user, and utilise incremental knowledge
acquisition, i.e., to incorporate the user’s contributions.

A context can be considered to be “everything that is used to give some meaning to a
message” (Cahour & Karsenty, 1993). In a knowledge system formal knowledge is
utilised in the problem solving. The meaning of the formal representation is a certain part
of the informal domain knowledge. When formalising knowledge three separate and
distinct “theories” are involved in the process (Kleene, 1971). These theories are:
(a) the informal theory, IT, of which the formal system constitutes a formalisation
(b) the formal system or object theory, OT, and
(c) the metatheory, MT, in which the formal system is described and studied.

Two different perspectives can be taken in the metatheory. The metamathematical one,
where the informal theory is simply disregarded and only the formal properties of the
formal theory are investigated. Or, alternatively, the metatheory can examine the relation
between the formal and the informal theory. Thus, the informal theory is regarded as being
the intended interpretation of the incomplete formal theory. The first perspective
presupposes that the informal theory is so fully understood that it can be replaced altogether by a formal theory. For knowledge systems, this is generally not possible. The system can only have a partial axiomatisation of the formal object theory because the informal theory is only partially understood (Hamfelt, 1992). Therefore, the metatheory examines the relation between the formal and informal theory, and not only the formal theory.

![Diagram](image)

Figure 5.10. The relationship between the informal theory, IT, and the object theory, OT. The shadowed area is the difference set, IT – OT.

In OT the formal knowledge is represented, e.g., as rules. IT is composed of the knowledge complementing and corresponding to OT together with the user’s contributions, see Figure 5.10. All the relations between IT and OT are administered by the metatheory. MT carries out the reasoning in the system based upon inferences in OT together with the information the user supplies. Therefore MT has the task of conducting the communication between the user and the system. This communication includes an integration of the user’s contributions in IT. The user may want to inspect the knowledge in the system, either to learn from the system or to be able to co-operate with it whilst reasoning. Thus, it is vital that the system can present the system’s informal domain theory to the user. This presentation is generated and displayed by MT. MT also carries out automated reasoning about the informal knowledge stored in the system, e.g., giving the user access to relevant information and for enabling it to produce adequate presentations. Figure 5.11 gives an overview of the system architecture. The formalisation of the architecture is not presented here, but can be found in the article.
The user's interpretation of context identifies the implicit part of MT transferring from the informal to the formal.

MT's formal reasoning with the formal theory OT, known formulas are directly used, new formulas corresponding to the user's interpretation (the context) are identified.

MT's presentation of context comprises explanations of the informal domain knowledge and a display of the reasoning. The explanations can be obtained during the problem solving and refer to the concepts involved in the reasoning. A rough distinction can be made between these objects according to their participation in different phases of the problem solving. An object could be a goal object, i.e., in which case the system will try to conclude a value for the object, and, if successful, it will present the object and the value as one of the results from the reasoning. The user can provide values for some objects and these will be called ground objects. All objects between goal objects and ground objects are called intermediate objects. Values for these can be perceived as intermediate results. They can be used in the final conclusion or in concluding values related to other intermediate objects, which will be used later on in the reasoning of goal objects. Figure 5.12 shows the different kinds of objects in the form of an object tree.

The grouping of objects is used when building up the informal domain theory. Connected to each group of objects are several determined properties. Contextual knowledge coupled to each object and every property is reproduced in IT.
Suppose that a domain, childhood illnesses, is to be described and utilised for diagnosing such illnesses. In Figure 5.13 the object tree for measles is given. When the system asks the user for the knowledge required for the reasoning, e.g., ‘high fever’, the user may ask for an explanation in relation to the question. The explanation is a presentation of the context related to the branch ‘childhood illnesses measles’, ‘early symptoms measles’ and ‘high fever’. The explanation is built up from the informal knowledge about the root object, the intermediate object and the ground object. In Figure 5.14 the explanation is displayed. The different properties related to each object are given in upper-case letters.

Figure 5.12. Different object categories.

Figure 5.13. Objects related to the childhood illness measles.
In addition to the explanations offered during the reasoning the user gets a presentation of the actual problem solving. The system displays what it is trying to prove and the premises for the current goal. The user is asked for a contribution when the user’s interpretation of the informal domain theory is needed. The contribution may be some data, e.g., ‘hacking cough’. It may also be that the user is asked for a truth value for an intermediate goal that the system cannot prove, e.g., whether the patient has ‘late symptoms measles’, or whether the current goal is equivalent to another sentence which can be used in the reasoning. The system’s interpretations of the user’s answers are then used in the problem solving. The outcomes are continuously presented to the user.

**measles**
TOTALITY: To diagnose, in this case measles, you investigate the child’s early symptoms and if there are special symptoms later during the illness eventually even these, but also if the child is immune to the disease.
ORDER: The program will investigate the early symptoms first.
CONCORDANCE: Some diseases give immunity. Measles is such a disease, which give life longs immunity.
JUSTIFICATION: You get immunity if you have had measles or are vaccinated against the disease.
CLASS: The disease measles is a virus and it infects through drops and it is very infectious. The incubation period is 8 - 11 days.
GENERAL INFORMATION: Generally, try to keep the child in bed or in stillness until the child has no fever. The illness lasts for about 8 days. Possible complications are inflammation of the ear, pneumonia and bronchitis.

**early symptoms**
TOTALITY: Early symptoms for measles, are hacking cough and/or a cold together with fever.
ALTERNATIVE: When you diagnose a disease and the child has recently fallen ill the diagnosis can only be based on the early symptoms. If the child has been ill for some days the diagnosis may be based on both early and late symptoms.
JUSTIFICATION: For measles it is obvious that the illness has two phases, an early and a late one, where the symptoms differ a lot.

**fever**
OBSERVATION: Indication of fever is that it is high when the temperature is 39 degrees or more, it is moderate high between 38 and 39. Less than 38 degrees is not estimated as fever.
JUSTIFICATION: These levels are not exact since fever can differ between individuals. Young children, though, usually get higher fever than older children or adults.
METHOD: The temperature is measured with a thermometer. (An animation of how the measurement can be performed.)
INTERVAL: The temperature is in the interval 36 to 42 degrees Celsius.

Figure 5.14. An explanation related to fever and the disease measles.
The metatheory in the system has to be quite advanced, performing reasoning based on both OT and IT and integrating the user’s interpretations of the informal knowledge in IT. MT represents the relations between the different kinds of knowledge. The architecture is realised in first order logic, which is capable of expressing arbitrary relations between syntactical terms. The division of the knowledge into three distinct theories gives the system a high degree of modularity. Moreover, the respective modules are expressed in their natural way, which in turn promotes clarity, making the system easy to survey. This is vital for achieving maintainability, i.e., facilitating updating and alterations to the system. Since first order logic has the benefit of closeness to natural language, it also promotes comprehensibility.

5.2.3 Article 2: Analysing the domain context knowledge

Above in Figure 5.14 some domain properties coupled to different categories of objects were displayed. These properties were written in upper-case letters. Through these properties the domain context was reproduced. The properties are identified in a particular domain but have a more general applicability. The list of all the domain properties for the example domain childhood illnesses is listed in Figure 5.15.

<table>
<thead>
<tr>
<th>No.</th>
<th>Domain property</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Justification of domain knowledge</td>
<td>Goal, Intermediate, Ground</td>
</tr>
<tr>
<td>2</td>
<td>Evaluation of conclusions or objects</td>
<td>Goal, Intermediate, Ground</td>
</tr>
<tr>
<td>3</td>
<td>Term descriptions</td>
<td>Goal, Intermediate, Ground</td>
</tr>
<tr>
<td>4</td>
<td>Totality of the objects involved when inferring a conclusion</td>
<td>Goal, Intermediate</td>
</tr>
<tr>
<td>5</td>
<td>Concordance between objects</td>
<td>Goal</td>
</tr>
<tr>
<td>6</td>
<td>Class of objects</td>
<td>Goal</td>
</tr>
<tr>
<td>7</td>
<td>General information</td>
<td>Goal</td>
</tr>
<tr>
<td>8</td>
<td>Alternative ways to reason to reach a conclusion</td>
<td>Intermediate</td>
</tr>
<tr>
<td>9</td>
<td>Method for the observation/measurement of an object</td>
<td>Ground</td>
</tr>
<tr>
<td>10</td>
<td>Unit for the observation/measurement</td>
<td>Ground</td>
</tr>
<tr>
<td>11</td>
<td>Interval for the measurement</td>
<td>Ground</td>
</tr>
<tr>
<td>12</td>
<td>Observation/description of an object</td>
<td>Ground</td>
</tr>
</tbody>
</table>

Figure 5.15. A categorisation of the knowledge in the informal theory.

In Figure 5.15 every property can be found together with a number and the kind of objects that may have the property. The first one is justification of domain knowledge, which is important for the user’s confidence in the system’s domain knowledge. Referring to
childhood illnesses, a justification can be that these diseases are common among children but rather unusual among adults because they are very infectious and therefore you probably get the diseases as a child and, furthermore, several of them give life-long immunity. The second, evaluation of conclusions or objects, presents, e.g., that one can be more certain about the diagnosis measles if both the early and late symptoms show measles. If the child has only recently fallen ill, the late symptoms cannot be studied of course and the confidence in the conclusion is lower. An evaluation of an object could be that one symptom is more important than another when inferring a conclusion, e.g., high fever is more important than a cold when diagnosing measles. This information is especially valuable for the user when he/she is supposed to contribute with a truth-value, i.e., when the system cannot infer a conclusion. The third, term descriptions, can define the meanings of terms so the user and system are using the same definition. An example relevant to measles is a symptom called Koplik’s spots, which are defined as small white spots inside the cheeks, close to the back teeth. The forth property is the totality of the objects involved when inferring a conclusion. To infer measles, both the early and the late symptoms are of interest together with immunity. An example of the fifth, concordance between objects, is that the patient cannot have measles if he/she is immune to it even if the symptoms show measles. The sixth, class of objects, describes, e.g., that measles is a virus (called Parotit virus). The next one, general information, could be anything that is important to describe to the user, e.g., recommended treatment and possible complications. Number (8), alternative ways to reason to reach a conclusion, deals with the order the reasoning will be performed, e.g., that the early symptoms will be investigated first. The properties (9) – (12) deal with observations and measurements and how to perform them. (9) describes the method, e.g., how to measure the temperature; (10) the unit for the measurement, the system is using Celsius degrees for the temperature; (11) the interval for the temperature, this is 36-42 degrees C; (12) refers to observation/description of an object, e.g., pictures of different skin eruptions or the sound of different kinds of cough.

In Section 2.3, knowledge in model-based explanations was investigated. The knowledge was classified into four groups: problem solving strategy, justification of the system’s domain knowledge, local problem solving, and term description. Let’s relate our categorisation of domain knowledge, which is utilised for the explanations, to this classification.
(I) problem solving strategy
This kind of knowledge comprises information about the system’s reasoning and the strategy for this reasoning and can be a base for explanations about the methods the system employs. To be able to understand the problem solving strategy the user has to know which objects are involved, how they can be used to reach a conclusion, the importance of different groups of rules referring to different objects and in what order they are used.

In IT this knowledge corresponds to
(4) totality of the objects involved
(8) alternative ways to reach a conclusion
(2) evaluation of conclusions

(II) justification of the system’s domain knowledge
The user may trust the domain theory more easily if special connections between different objects in the domain are described, if class belongings for objects are made clear and if general information connected to an object is presented. All this information may help the understanding of the domain theory. Moreover, to get the user’s confidence the system has to verify that the theory implemented in the system resides on a sound domain theory and even that the system’s actions are justifiable. To be able to do this the system has to connect rule groups to the knowledge in the theory the rules are trying to capture. In IT this knowledge corresponds to
(1) justification of domain knowledge
(5) concordance between objects
(6) class of objects
(7) general information

(III) local problem solving
To grasp the method behind the system’s way of solving problems the user may want to understand how an object is used locally, i.e., how the object is used in the reasoning. Therefore, it is useful to know whether the knowledge can be structured in such a way that the reasoning can be based on special groups of rules. This supports the understanding of the problem solving on a local level. In IT this knowledge corresponds to
(4) totality of the objects involved in an intermediate conclusion
(8) alternative ways to reach an intermediate conclusion
(2) evaluation of conclusions or objects
(6) class of objects
(IV) term descriptions

It is, of course, important that the user understands the terms used by the system and that the system and the user apply these terms in the same way. Both descriptions of terms and general information may be useful. In IT this knowledge may be represented in

(3) term descriptions
(7) general information

After the comparison between our classification of knowledge in IT with the demands described from Clancey, Swartout, and Chandresekar et al referring to knowledge needed for explanations, it becomes apparent that some of the knowledge we have found important is not mentioned by them. The categories (9) - (12), referring to the methods for measurement, the unit and the interval for measurements, and the descriptions of objects, are not incorporated in the categories (I) - (IV) above. The actual information deals with how to obtain information needed in the problem solving and supports the user in finding this data. We find these categories of information important for the user, especially since the user is an active part of the reasoning process. The explanations in our system architecture are not supposed to describe how the reasoning was performed by the system, as in several other systems, but to communicate the system’s knowledge to enable it to cooperate with the user during the problem solving.

The way of categorising the objects and then connecting the different groups with properties is a practical procedure for reproducing domain context. The domain properties presented above have proved to be suitable for diagnosis and classifications, even if they may be coupled to different object categories within different domains. These properties have been used in three domains, in addition to childhood illnesses. These properties are tested for the domain in Analyse More. For this lake water diagnosis, based on chemical-physical and biological observations, the properties were suitable, even though these two types of domains are very different. In a decision support system for small firms, Small Business Manager (Öhlmér, 1998), the explanations regarding the cash flow behaviour in a firm were based on the categorisation. All the properties listed in 5.15, except three (evaluation of conclusions or objects, alternative ways to reason to reach a conclusion, and concordance between objects) were utilised.

The way to categorise knowledge coupled to the object tree has been used in two domains, where the properties were different. First the method was used for reproducing context knowledge in a matching facility (Johansson, 2000) implemented within an EC project,
easi-isae (Gerret, 1997). This facility is used by teachers and school leaders when searching for candidate companies for co-operation in the production of computer-aided learning material and by companies trying to find teachers for the role of content provider or intellectual author in the production of such material. The method was adopted by all six of the countries co-operating within the project, but they decided the properties used locally. Secondly, the method has been used in an explorative initial study examining the requirements of specified information for different target groups, in conjunction with major accidents in the society (Lindström, 1999). The study shows that the types of information differ between different target groups and between accident phases. Structuring the domain context as objects in an object tree facilitated the management of the information, both the acquisition and the presentations.

We have presented a way of classifying the knowledge needed in a system designed to be transparent. There are several advantages with such a detailed classification. First, we think the categories (1) - (12) could quite coherently reproduce the domain knowledge needed for communicating the knowledge to the user. Secondly, a detailed classification is a support in the knowledge acquisition phase, since the knowledge engineer can direct the interviews with these categories in mind. Furthermore, it is easier to update changes when the knowledge is divided into smaller parts, because only the actual category has to be updated.

The proposed architecture in Article 2 deals with informal knowledge reproduced in its most natural way, i.e., in different media. However, hypermedia systems are designed to present information in different media. The question arises, whether one can categorise the system architecture as a hypermedia system.

5.2.4 Article 3: A system architecture for knowledge-based hypermedia

In the third article we analyse various constellations of merged hypermedia and knowledge systems. Moreover, a programmable system schema for supporting the composition of intelligent hypermedia systems and choosing a suitable representation language is introduced.

Hypermedia systems and knowledge systems can be seen as a complement to each other (Rada & Barlow, 1989b). As remarked in Section 3.2, one problem with hypermedia
systems is that the user can easily get lost or disoriented when searching for information. To overcome this problem and to fully exploit the opportunities of hypermedia, knowledge-based computer reasoning can be utilised. Knowledge systems, on the other hand, are not designed to present information and hypermedia techniques can therefore be applicable.

Knowledge technology offers methodologies for program and system development for complex domains. Therefore, it has been a natural step to apply these techniques to hypermedia (see, e.g., Bench-Capon, Soper & Coenen, 1991; Hamfelt & Barklund, 1990; Rada & Barlow, 1989a; Soper & Bench-Capon, 1992, Woodhead, 1991) resulting in what may be termed knowledge-based hypermedia (Heath, Hall, Crowder, Pasha & Soper, 1994) or intelligent hypermedia.

Obviously, the notion of a link is important in hypermedia systems. A hyperlink is strictly at the metalevel to the semantic or content level (Woodhead, 1991). In second-generation hypermedia, link types are used as an attempt to capture the intended semantics in the relationships between the nodes, which is important for transparency (discussed in Section 3.3). Naming of links is a rather simple typing technique. Knowledge-based methods yield more powerful typing by allowing definitions of relations as production rules or logical predicates. These methods allow the definition of all kinds of links and, furthermore, the links can be computed at run-time and do not have to be defined in advance.

We call a hypermedia system intelligent if and only if knowledge-based computer reasoning sets up its links. This definition covers the notion of canned text, or canned information since other modalities than text are involved. The purpose of canned information is to present the informal domain knowledge to a user. Canned information can be seen as a first step towards intelligent hypermedia. Advantages and disadvantages of the canned text approach are discussed in Section 2.3.1. The major drawback is that canned information cannot represent a coherent informal domain theory, it can only represent it as fragments. Since it is not possible to use the information in different ways to adapt and reuse it in different contexts, the canned information can be viewed as static hypermedia. Intelligent hypermedia systems, on the other hand, must be able to dynamically compose pieces of informal knowledge to present to the user. It should also be capable of guiding the user to static informal knowledge, i.e., canned information.
An intelligent hypermedia system must perform computer reasoning about the informal presentation of the domain knowledge in contrast to knowledge systems that derive conclusions from a formalisation of domain knowledge. Hence, the knowledge bases of the respective system type comprise fundamentally different kinds of knowledge. Since the two types of knowledge bases might be combined, intelligent hypermedia can be divided into two categories. The first one will be called *unary intelligent hypermedia* systems, which have only one knowledge base representing knowledge about the links. The second category will be denoted *dual intelligent hypermedia* systems and these systems have in addition a knowledge base representing knowledge about the problem solving for the domain.

Unary intelligent hypermedia requires that the relations between pieces of informal knowledge are made programmable instead of hard-wired in the system. This makes it possible for the system to present a coherent informal theory. Dual intelligent hypermedia has an extra requirement that the relation between the formal domain knowledge, OT, and the informal theory, IT, is made explicit. Moreover, some notion of soundness has to be promoted between OT and IT, by, at the very least, arranging the two theories being updated simultaneously. These aspects should be handled at the level of the metatheory, MT.

To support program development, ready-made program structures are essential. Structured programming, where parameterised program schemata are supplied to facilitate programming (Backus, 1978), is an example of such programming support. At the system architecture level, this corresponds to providing a programmable schema for the whole system, parameterised with schemata for the modules of the system. In the case of intelligent hypermedia systems the system schema corresponds to MT and should be parameterised with two schemata of the modules IT and OT.

MT expresses relations between objects within the respective theories IT and OT or between them. Thus MT requires a representation language capable of expressing relations. Preferably, the language should have well-defined formal semantics and intelligible external semantics. Moreover, both IT and OT may have a complex inherent structure. MT should therefore have adequate data structures for modelling this structure. OT requires a language expressive enough for formally approximating the knowledge of the actual domain. Most domains require at least relations (see, e.g., Muggleton & De
Raedt, 1994). First-order logic is a relational language with well-defined formal semantics and external semantics close to natural language.

Depending on the expected needs of the modules, the schemata should support appropriate programming techniques, e.g., for computer reasoning, object-oriented programming, etc. In the article we conclude that a conceptual language would be suitable for MT, but as yet no well-established logic programming languages with concepts exist. However, first-order logic programming languages with ordinary terms are available. These languages allow restricted metaprogramming facilities, required here since MT is a metatheory in relation to IT and OT. Metalogic programming is a suitable representation language for the architecture and convenient for defining program development methodologies (see, e.g., Hamfelt & Nilsson, 1997) as well.

Therefore, the proposed architecture comprising MT, IT and OT (described in Article 2) is implemented in metalogic programming. Only IT and MT are involved in a unary intelligent hypermedia system where informal domain knowledge is reproduced as canned information in IT and knowledge about the relations between the information in IT is represented in MT. In a dual intelligent hypermedia system all tree modules are utilised. The problem solving knowledge is represented in OT. Now MT must also represent the deduction rules for OT and deal with the problem that OT is incomplete. That means that when OT is insufficient for deducing a conclusion, MT is to invoke IT. This is to see whether OT can be claimed to contain additional formulas, not identified so far, corresponding to the user’s interpretation of the informal knowledge in IT. This implies that MT must know the relations between the knowledge in OT and IT, and also be able to deal with the user’s contribution to the problem solving.

MT should at least promote *maintainability* (facilitating updating and altering of the system) and *usability* (through adequate user interfaces and documentation), which are attributes of well-engineered software (Sommerville, 1996). These ends are met since MT gives a high degree of modularity and allows the respective parts to be expressed in their natural way, thus promoting clarity and making it easy to survey the system.

From the viewpoint of the user transparency is vital. Without adequate transparency the user will not be able to co-operate in the problem solving. The user accesses the domain knowledge through presentations of the inferential context and the conceptual context.
The inferential context is partly presented as explanations related to the object tree, see Figure 5.13 and 5.14 in Section 5.2.2. The explanations are dynamically generated from the informal knowledge reproduced in IT. The inferential context is also shown through a continuous printout of what the system is doing during the problem solving. Moreover, the system structure affords the possibility of letting the metalevel monitor the reasoning along several diagnostic branches (differential diagnosis) so as to identify any termination branches to be excluded. This is claimed to model an important aspect of medical diagnosis (Schneider & Sandblad, 1979). If the user, for instance, has observed fever and skin eruption all illnesses having these symptoms are displayed, as demonstrated in Figure 5.16.

The conceptual context is presented through different kinds of figures showing relations between objects and conclusions and conceptualisations of the domain. Let the domain still be childhood illnesses. The relation between symptoms and illnesses is an example of the first type, see Figure 5.16. This figure is generated according to the symptoms observed so far. In the case that only fever had been established the figure would also included mumps. If, on the other hand, a cough had been present together with fever and skin eruption, the number of illnesses would have decreased to include just measles and German measles.

One conceptualisation of the domain would be to present the different causes of, e.g., infection. One cause is a virus and Figure 5.17 pictures medical knowledge in connection with the childhood illnesses caused by viral infections. This information can be cut down to one branch if, for example, the focus is only measles.
- no metabolism of it's own
- nucleus, called virion, of nucleic
- has a shell of protein
- propagates only in living cells
- diameter 10 to 300 nanometer
- can infect all living organisms

- RNA lays bare in the cytoplasm
- new virus are released from the dead cell
- life cycle 6 hours

- DNA lays bare in the nucleus
- new virus are stored in the dead cell
- life cycle 20 to 24 hours

Figure 5.17 German measles, measles, mumps and chicken pox are caused by viruses.
The system’s knowledge and reasoning is communicated to the user at various abstraction levels through these presentations of the conceptual and inferential contexts. This is essential for providing a comprehensive view of the diagnosis that includes the context and not only the details. Thus, the user’s ability to contribute to the reasoning in addition to providing data is improved.

To conclude, knowledge systems and hypermedia systems complement each other. Developments of each system type require techniques from the other type since both automated reasoning and unguided navigation have inherent shortcomings. Automated reasoning is incapable of dealing with full domain theories and therefore requires support from the user in interpreting the unformalised parts of the domain. To this end, the user needs access to informal descriptions of the domain knowledge and the system’s inferencing. These descriptions must be generated on at run-time since they reflect arbitrary states of reasoning. Hypermedia systems can offer access to static informal descriptions but, complemented with knowledge based techniques, such descriptions can also be computed dynamically. Moreover, in a hypermedia system the user is lost unless guidance is provided, which can be naturally implemented with knowledge techniques.

We have integrated an informal domain theory, a formal object theory axiomatising the informal theory and a metatheory analysing the properties and interrelations between them into a system architecture. This architecture is to serve as a programmable schema for supporting the composition of intelligent hypermedia systems. On the whole “programming in the large” is supported by the schema, which defines the overall system structure, whereas “programming in the small” is supported by knowledge modelling techniques.

5.2.5 Article 3: Discussion

In Section 2 and 3 we investigated knowledge systems and hypermedia systems as learning environments in relation to aspects such as co-operation and transparency. A number of the findings described in Section 2 and 3 will be discussed in relation to the proposed system architecture.
When using knowledge systems for learning some important issues are representing different kinds of knowledge in the system, performing the problem solving in an appropriate manner and constructing a system that is transparent. For hypermedia systems the usability in relation to the system engine, the content and the structure are important, together with special requirements that exist for learning environments, such as supporting a goal-directed search.

Let’s start with knowledge systems. It is vital that the knowledge in such systems consists of the relevant knowledge within the domain for the problem solving and for describing the context. A system’s problem solving must reflect the experts’ way of reaching the solution if the user is to be able to learn from it. Both the knowledge and the problem solving have to be transparent so the user can understand the system’s knowledge. Therefore, the explanations should include at least the kind of information represented by the groups labelled (I) - (IV), presented in Section 2.3. Furthermore, the context knowledge and the problem solving knowledge must be separated from each other.

In the proposed system architecture it is obvious that the different knowledge types have been kept separate. In OT, the problem solving knowledge is represented and, in IT, the domain context knowledge is reproduced. MT deals with the different knowledge types and can generate several kinds of explanations to present to the user, referring both to the inferential and conceptual context. When presenting the inferential context the explanations comprise more knowledge types than those specified in the four groups. Moreover, the current problem solving is displayed. When presenting the conceptual context, different views of the domain knowledge are generated. In this way the demand for transparency is met. If pedagogical knowledge and a user model are to be included, these could also be represented in OT and IT.

A system architecture cannot guarantee that the relevant knowledge is implemented in a system, only that the current knowledge can be reproduced in the schemata offered. The suggested form of the knowledge representation in OT nowadays is in rules, but could be other suitable forms. In IT a variety of informal knowledge can be included, in its most natural form. Moreover, the user’s contributions enlarge the system’s domain knowledge, and in a way can guarantee that relevant problem solving knowledge is included in the system. For the moment the architecture does not extend the context knowledge with user contributions.
Regarding hypermedia systems, the system engine must support the user to find the relevant information and deal with the problem that he/she can easily get lost in hyperspace, e.g., by offering or recommending guided tours to restrict the search. Unfortunately the links may not be the right ones for a student’s particular needs or for the current task. In relation to the content, the opportunity to mirror different views of the domain knowledge ought to be exploited. Several small nodes are considered preferable to a few extensive ones and, furthermore, the nodes should be of the right form in accordance with what is to be reproduced. It is also important that the user can understand the meaning of the links, since they denote a specific relation between the nodes. Moreover, one should be aware of the problem of information fragmentation. In addition, the structure of the hypermedia content is important and the organisation of the information is best presented through different overviews.

These observations are rather general. When dealing with hypermedia systems for learning some other issues are in focus too. Firstly, the user has to be active to be able to learn. Therefore, to facilitate learning through hypermedia material, the user should have a goal to solve. Secondly, it is recommended that one have a system that can handle mixed-initiative dialogues, where both the user and the system can take the initiative.

This was a summary of important findings related to hypermedia systems. These will now be discussed in relation to our system architecture presented in Article 2 and 3. In the proposed system architecture the user is guided to the relevant domain context through knowledge-based reasoning, thus the user will not get lost in the search for information. As a matter of fact the system offers a guided tour related to the problem solving at hand. The problem with improper links can be handled because the links are not static, but dynamically generated by MT. This generation of links can be tailored to the current user and to the goal. The knowledge in the system is re-used in different contexts. Moreover, various kinds of knowledge are included in IT to serve different users and to mirror different perspectives of the domain knowledge. Several views of the domain knowledge are offered through the presentations of the inferential and conceptual contexts. Some of these may also serve as overviews of the domain. Displaying the object trees is another way of presenting an overview of the domain. In OT and IT it is possible to reproduce the domain knowledge in a coherent manner, which is a way to deal with the fragmentation problem. Different media can be utilised to store the information in the nodes and therefore will be suitable for the knowledge. Naturally the size of the nodes can differ. Furthermore, the links are set up through knowledge-based reasoning. Links are presented...
through the object tree together with the headings in the related explanation, i.e., the categories such as totality and order; this is exemplified in Figure 5.14.

When studying the system as a learning environment it is obvious that the user is active, since the user and the system co-operate in the problem solving. The purpose of this collaboration is to solve a problem, thus the user has a goal. Furthermore, the user can choose between providing input, interpreting knowledge or asking different questions to get presentations of the inferential and the conceptual context. So there is a form of mixed-initiative between the user and the system.

To conclude, the system architecture promotes most of the important issues mentioned above. By combining knowledge technology with hypermedia technology we get a powerful architecture in which it is possible to reproduce a coherent domain theory and which can serve as a learning environment where the user co-operates with the system to conduct the problem solving.

5.3 Case studies

In this section three case studies are presented. All three describe further elaborations of the Analyse More system, addressing the problems described in Section 5.1.3. In the articles the necessity of including informal domain knowledge in systems supporting learning is pointed out. The section ends with a discussion related to all three articles.

In the fourth article Analyse More is combined with a hypermedia interface, which facilitates the students’ interaction with the system, in terms of communicating the input and tapping the system’s knowledge. The focus in the fifth and sixth articles is quite different, stressing the importance that the pupils themselves learn how to perform and carry out a diagnosis. The knowledge system part is used to find the system’s opinion for use as a comparison, which supports learning. A short presentation of the system is given in the fifth article and a more thorough one in the sixth paper. In the latter some pedagogical aspects of the system are discussed.
5.3.1 Article 4: A combined knowledge and hypermedia system to attain educational objectives

Knowledge systems can, as stated earlier, be a valuable educational tool. But there are problems for the user when understanding and learning from such a system arising from the quantity of problem solving knowledge included in the system.

From experiences using Analyse More we have concluded that there are four major problems. Firstly, the system does not enable the user to obtain control or take the initiative. However, a mixed initiative between the user and the system is important, see e.g. Shneiderman (1986). Secondly, the user has severe difficulty in understanding the system’s reasoning, the problem being both on a detailed rule level and on a higher strategic level. Thirdly, the two first factors in combination limit the user’s ability to acquire an overview and thereby to develop an understanding of the problem solving process. Finally, a problem pertaining to the general domain knowledge, relates to the difficulty in understanding the ecological system as a whole, which was strongly emphasised by the teachers. Students frequently make the error of solely regarding a part of the ecological system without understanding the effects and correlations to other parts.

The approach taken to tackle these problems was to redesign Analyse More and merge techniques from hypermedia and knowledge systems, enabling each system’s advantages to be incorporated. The combination of hypermedia and knowledge system technologies supports both types of knowledge we wish the system to convey, problem solving knowledge and general domain knowledge, see Figure 5.18. Parts of the general knowledge are not used in the problem solving, but can still contribute to the global understanding of the topic. The nature of this knowledge demands a flexible representation, best supported by hypermedia. It is preferable to represent the problem solving knowledge in a knowledge system.

The four problems have been addressed in the design in the following way. An important change made in relation to the student’s control and initiative is that the user decides the topics to work with, biological or chemical/physical, and which indicators in the problem solving he/she will study, e.g., algae. A form is used for communicating the input: the content of the form is dependent on which topic the user has chosen and on the information supplied so far. The question generator, described in Section 5.1.1, is utilised to set up the forms. Secondly, when the user so chooses, a diagnosis is performed by the
system. Then the knowledge system presents a conclusion, or points out that some important indicators are missing, which means the user has to supplement the data with more observations.

![Diagram](image)

Figure 5.18 Two types of knowledge conveyed by the system.

The four problems have been addressed in the design in the following way. An important change made in relation to the student’s control and initiative is that the user decides the topics to work with, biological or chemical/physical, and which indicators in the problem solving he/she will study, e.g., algae. A form is used for communicating the input: the content of the form is dependent on which topic the user has chosen and on the information supplied so far. The question generator, described in Section 5.1.1, is utilised to set up the forms. Secondly, when the user so chooses, a diagnosis is performed by the system. Then the knowledge system presents a conclusion, or points out that some important indicators are missing, which means the user has to supplement the data with more observations.

As far as the understanding of the reasoning process the system provides explanations on several levels, by presenting rule-based “why” and “how” explanations, and through metalevel explanations presenting rule groups instead of separate rules. An example of a rule group is all rules deciding the status of algae, which comprises many tens of rules. Moreover, the system’s presentation of intermediate results, e.g., the status of algae, which is part of the biological status of the lake, contributes to the understanding of the problem.
Another contribution comes from the possibility to obtain information concerning vital indicators, such as phytoplankton and algae, which are the most important indicators when concluding the biological status.

Regarding the overviews and understanding the complexity of the domain a different kind of support is given. First the division between topics opens up the possibility of studying the importance of one topic compared to another. Then more support comes from the forms, which are used for the input because these are ways of presenting the scope of the domain and giving an overview of it. Through a “Tell me more” button, different kinds of informal domain knowledge are presented with the aid of several media covering, e.g., methods of measurements and observation, information concerning indicators, and miscellaneous domain knowledge. This information can be reached throughout the session.

To conclude, the proposed merged system utilising a knowledge system can be taken as the point of departure and later amended by hypermedia techniques facilitating the user’s understanding of and learning about a domain, in this case lake water diagnosis.

5.3.2 Article 5: DLW – a learning environment for lake water diagnosis

The article presents a learning environment Diagnosing Lake Water, DLW. The main objective for the system is to support students when learning how to perform a lake water diagnosis, implying that the student must be helped to acquire his/her own problem solving method. The approach is based on merging techniques from hypermedia and knowledge systems.

Different components are utilised in the system. These are called indicators and parameters. Indicators should be observed or measured, e.g., the pH-value and alcalinity are measured (denoted ground objects in Section 5.2.2). The indicators affect the parameters (denoted intermediate objects in Section 5.2.2). Based on the pH-value and alcalinity, the parameter acidity can be determined to be, e.g., oligotrophic. These parameters in turn affect the final conclusion, in this case the chemical/physical status.

The student chooses a topic to work with. An overview of the domain is presented, stressing the domain’s hierarchical nature. This overview is presented in the grey zone in
Figure 5.19 Menu of parameters.

Figure 5.20 The initial hierarchical overview.
The current indicators are presented in a form when investigating a parameter. The user should fill in the relevant values for the measurements or observations in this form. Help is offered regarding how to measure or observe the indicators. Moreover, rules of thumb are presented to support the user in deciding the status of the parameter, see Figure 5.21. Later the user may determine the status of the lake, by combining the outcomes of different parameters. The system can then present a conclusion based on the student’s answers for the indicators and the student may compare his/her results with the system’s.

Supporting a student when learning how to perform a diagnosis implies not solely aiding the student in obtaining a conclusion, but also in understanding the domain knowledge, and applying this knowledge in various situations. This necessitates a profound comprehension, referred to as deep learning (Marton, 1984). Therefore, one of DLW’s main points of focus is enabling the student to draw the necessary conclusions by him/herself. Different features support this, e.g., clear presentations of the relevant components within the domain, access to various kinds of information concerning a certain component, including both factual and relational information, overviews over the hierarchical domain, and modes of manipulating the components.

5.3.3 Article 6: Pedagogical issues for DLW – an interactive learning system

In this last article we study the DLW system from a pedagogical view, by examining how some pedagogical methods are utilised and supported in the system’s design and how the design is related to Bloom’s taxonomy (Bloom, 1956).

The prime concern that was addressed in and therefore underlies DLW’s design was to support students in learning to diagnose lake water, i.e., enabling them to learn about the topic at large and perform the diagnosis themselves. It has been noted that problems of diagnosis are frequently perceived as difficult to comprehend by students, since they may deal with data that is incomplete, unreliable or erroneous (Hayes-Roth, Waterman & Lenat, 1983). A first step in attacking the problem is to divide the problem solving process into a number of phases. For a novice, we perceive the process of solving a problem to consist of four (non-sequential) phases: (i) identifying central objects, (ii) obtaining general information about the objects, (iii) acquiring structural knowledge concerning the interrelations of objects and (iv) the problem solving itself.
Pedagogical methods such as overviews, depth and width perspectives and a combination of the top-down and bottom-up procedures included in DLW’s design supporting students in acquiring the information required in the four phases will be discussed.

The value of overviews has often been emphasised, for instance by Woods (1994) who states that overviews are an important ingredient as a means of supporting deep learning. DLW offers several types of overviews. The initial partition of the domain in chemical-physical and biological spheres followed by the menu of parameters, see Figure 5.19, gives overviews of the domain and also supports the user’s identification of central objects. Through the menu of parameters an overview concerning the interrelations between objects is presented, in this case displaying the system’s structure as hierarchical, a suggested structure for problem solving (ibid.). Moreover, the hierarchical overview, see Figure 5.20, and the one that is built up in connection with the user’s investigations also present the interrelations between objects. The form presenting a parameter, see Figure 5.21, offers an overview at a deeper level in the hierarchy. Apart from offering identification of indicators relating to a certain parameter, the form offers opportunities for accessing general information concerning these indicators and it indicates effects and to some extent the application of the indicators under examination. Another example is the table, see Figure 5.22, which states the student’s opinion of what each parameter indicates about the status of the lake. The table presents a brief summary of the parameters and provides a basis for forming a final conclusion, which is part of the problem solving.

DLW supports both depth and width perspectives. As far as the depth perspective is concerned, it is possible to follow a branch by specifying a parameter and indicator of interest. The depth view offers an insight into specific detailed information. The width view, in contrast, presents a broad outlook denoting objects at the same hierarchical level, see e.g. Figures 5.19 and 5.22. Both these perspectives primarily reinforce the relationships between objects, acknowledging the third phase of the problem solving, but they do so from two different angles. The depth view inclines towards the interrelationships of objects and their effects, part of the problem solving processes’ fourth phase. The width view, on the other hand, leans towards the second phase, by offering an insight into parallel information streams. Together these perspectives provide the means of adopting a holistic as well as a serialistic approach, enabling the support of different learning styles.
Figure 5.21 The acidity parameter comprising the indicators pH-value and alcalinity.

Let’s turn to the information moduling and sequencing in DLW. Since a part of DLW consists of a hypermedia module, the opportunity arises to divide the domain’s information into suitable chunks, and to make a division both in respect of size as well as of logical units. As Woods proclaims, chunking facilitates mental processing (Woods, 1994). Moreover, chunking supports problem solving, when performing diagnoses, through a divide and conquer approach, suggested by Steels among others (Steels, 1987). In addition, chunking facilitates a flexible design, which offers the student the possibility of pursuing his/her preferred learning style, and obtains the information sequenced according to individual preference.
As several authors (Clancey & Letsinger, 1981; Stanton & Stammers, 1989; Wærn, 1989) have previously stated, complex learning demands both top-down and bottom-up procedures. This is reflected in DLW’s design. DLW commences with a top-down approach presenting information of a more general kind. As the student proceeds with the investigation, the information successively becomes more detailed, see Figure 5.23. At each level the student has access to general information concerning the current topic. When the user reaches the indicators it is time to give the value of these and than try to determine what effect these will have on the parameter. Then the bottom-up process begins, which is suitable for the actual problem solving process.
Let’s relate the features of DLW’s design to the learning stages described in Bloom’s pedagogical strategy (Bloom, 1956): knowledge, comprehension, application, analysis, synthesis and evaluation. First is the *knowledge stage*, in which the student concentrates on collecting basic information. DLW supports the student in this stage by providing information through the “Tell me more”-facility, and the rules of thumb and initial overviews provided. The next stage, the *comprehension* one, is upheld by the structuring of information through overviews, denoting hierarchical structure, chunking of knowledge and top-down procedures. The *application stage* can be related to determining the value of parameters. We suggest that the bottom-up procedures and to some extent the overviews aid the student in this respect. The forth stage, *analysis*, involves drawing a conclusion about the status of the water. Here both bottom-up and top-down procedures are of value. In addition, overviews and the “tips” supplied also support the student in this stage. When it comes to *synthesis*, we believe that the depth and width perspective and the emphasis on hierarchical structure, in combination with basic facts enable the student to process the information, so that he/she may produce a coherent picture. Finally, *evaluation* is supported by the possibility for the student to compare his/her results with those of the system.

To conclude, DLW’s design offers support for all six learning stages within Bloom’s pedagogical strategy, but the student gets more support in the first four stages.
5.3.4 Discussion

The case studies were directed towards making improvements to Analyse More to make it better serve as a learning environment. In the first of these case studies, Article 4, hypermedia was used mainly as an interface to the knowledge system Analyse More. The objective was to support learning by letting the user gain more control over the interaction with the system and to support it with both general domain knowledge and problem solving knowledge. More emphasis was put on explaining the general domain knowledge than in Analyse More. The presentations of the two knowledge types were also performed differently.

Articles 5 and 6 presented a system, DLW, where a user actually performs the problem solving him/herself with the support of the system. The student has to acquire his/her own problem solving method to learn how to make a diagnosis. We state that, for a novice, the problem solving process consists of identifying central objects, obtaining general information concerning the objects, acquiring structural knowledge concerning the interrelations of objects, and the problem solving itself. DLW’s design supports the student in working through these phases by offering access to context knowledge and problem solving knowledge, presented in several different ways and making use of hypermedia. When the user has made a diagnosis the system gives feed-back by presenting its own solution performed by the knowledge system part.

All the articles in this thesis stress the importance of reproducing problem solving knowledge and contextual knowledge in a system designed to support learning and cooperation. In the architecture proposed in Article 2 and 3 both these knowledge types are reproduced in OT and IT, respectively. The examples of how this knowledge is presented to the user differ somewhat in the articles. It is important to note that different styles of presenting OT and IT knowledge can be implemented in MT.

In Section 3.2 it was stated that hypermedia systems may support learning methods through learning-by-doing and constructivism. The examples given in Article 2 and 3 show a system that is responsible for planning the inferencing and therefore govern the user’s involvement in the reasoning. This also means that the user is not given a full-blown opportunity to learn-by-doing. In that case the user should perform all of the problem solving by him/herself and only get support from the system to perform the task. In Article 5 and 6, on the other hand, the user performs all of the problem solving. The
user makes the diagnosis by first deciding the value of different parameters, which are related to measurements and observations, and then performing an evaluation of these parameters together. MT can, naturally, handle both types of co-operation with the user. Moreover, since the user can actually add problem solving knowledge to the system, the constructivistic view of learning is supported.

There are both advantages and disadvantages related to the responsibility for the problem solving. When the system rules the execution, the user participates in the reasoning, but cannot take complete responsibility for it. One advantage of this is that the user does not get lost and only relevant knowledge is presented to the user, on the other hand, the user cannot practise performing all of the problem solving. When the user is in charge he/she may learn by actually performing the problem solving. The disadvantages inherent to this are that it is the user’s responsibility to determine a goal to direct his/her work and to search for information in a systematic way. In Section 3.2 it was mentioned that this might be a problem for some students.
6 Scientific contribution and related work

In this section the contribution made by the research presented here will be summarised and the system architecture will be discussed in relation to other research.

6.1 Scientific contribution

In this thesis we propose a system architecture for knowledge based hypermedia. Taking the respective weaknesses of knowledge systems and hypermedia systems as the point of departure, knowledge-based hypermedia strives to take advantage of their respective strengths to reduce these weaknesses.

We argue that pure knowledge systems are inherently weak in that they are restricted to formalisations of the domain. In general a formalisation cannot fully reproduce the domain knowledge. Often, however, the success of a knowledge system is dependent on this excluded knowledge. We term this kind of knowledge the context of the formalisation. A context can roughly be defined as “everything that is used to give some meaning to a message” (Cahour & Karsenty, 1993), such as, the problem solving situation, the participants involved, the mode of interaction through which the communication occurs, the discourse taking place and the external world (Mittal & Paris, 1993). The main focus of this thesis is how to deal with this context. The reason why this knowledge is excluded may range from difficulties in knowledge elicitation to intractability and even to the inherent restrictions concerning formalisation of knowledge as such.

Pure hypermedia systems, on the other hand, are inherently weak in the sense that they are designed only to convey information based on static links. They do not possess any knowledge with which to support navigation. Knowledge is characterised by the ability to understand and exploit information (Awad, 1996), e.g., to solve problems. A pure hypermedia system is dependent on the user’s own interpretation of the author’s presentations of the links to navigate to find further relevant information.
The strength of a knowledge system is its capacity for problem solving in a domain based on formalised knowledge. The strength of a hypermedia system is that it can readily make available associated information when studying some topic. For knowledge systems, combining the two system types allows, an improved user interaction, reducing the weakness with respect to excluded knowledge and for hypermedia systems, improved navigation by means of problem solving, reducing the weakness with static links.

A knowledge system makes inferences from a formal theory constituting a formalisation of some domain knowledge, whereas a hypermedia system conveys, information about some domain in an informal form. As a philosophical basis for our architecture for knowledge based hypermedia, we consider a formal theory representing some domain and formalising some informal theory describing the same domain. This view corresponds to Kleene’s separation into three distinct theories OT, IT and MT, where OT is the formal system or object theory, IT the informal theory of which OT constitutes a formalisation and MT is the metatheory in which OT is described and studied. In a metamathematical analysis MT would only consider OT as a syntactical object from which further syntactical strings can be extracted strictly and exclusively according to the syntactical manipulations admitted by a set of postulated inference rules. In such a study, the fact that OT formalises IT is intentionally and carefully disregarded. However, in the real life domains for which knowledge systems are intended, the OT does not formalise a mathematical IT, but rather a domain comprising a great variety of objects and attributes which cannot be decomposed to a minimal set of primitive objects. For the purpose of a knowledge system it would be futile only to undertake a metamathematical analysis of such an OT. Moving outside metamathematics, MT can also be used for studying the relation between OT and IT, and IT is taken as the interpretation of OT.

IT can be, and normally is, expressed in a richer language than OT. Since it constitutes an informal description of the domain knowledge, it is actually the external observer’s (the user’s) interpretations of and inferences from IT that determines what IT describes. Thus the aforementioned excluded knowledge – the context – can be furnished to the system. To be effective this interactive co-operation with the user must respect pedagogical principles.

In the architecture MT carries out problem solving with OT. MT relates OT to IT and whenever needed in the problem solving it consults the user for interpretations of the
related IT descriptions and merges these into OT. Moreover MT composes IT presentations depending on the state of problem solving (explanations).

The main contribution of the work included in this thesis is to have made a clear separation with a philosophical underpinning of the general concepts involved in a knowledge-based hypermedia system. We have shown how this structure can be used to reduce the weaknesses of the two system types involved. The knowledge system part is strengthened by the user’s interpretation of the informal context supplied by the hypermedia system. The hypermedia part is strengthened by knowledge-based navigation.

The proposed system architecture presupposes that the user and the system co-operate in the problem solving, thereby improving the quality of the solution. During the reasoning the system interacts with the user, accepts his/her contributions and integrates them into the system’s knowledge, tasks a co-operative computer should perform (Clarke & Smith, 1993). To succeed in the co-operation the user has to understand the system’s knowledge; thus the system has to be transparent, which is usually achieved by providing explanations. In the thesis we have defined a number of properties used for reproducing the domain context, that is used in the explanations of the system’s knowledge. The domain context in IT is structured as objects in a context tree, and objects at different levels in the tree are connected to special domain properties. Hence, the properties can describe the context with different levels of abstraction and from different perspectives. The system’s explanations comprise a presentation of both a conceptual and an inferential context of the domain. The conceptual context is based on the domain properties and different overviews. The inferential context mirrors the current problem solving and is a description of knowledge in OT.

In addition, we have illustrated how the system structure provides a basis for developing learning environments. Fundamentals of learning are (Woods, 1994): motivating the learner, setting up goals, giving the context, selecting knowledge that is not known by the user, ordering the presentation of this knowledge and planning this presentation, facilitating the learning by being active and aiding the recall of the learner. The system architecture supports several of the items mentioned through the co-operative problem solving and by offering different kinds of explanations (see Section 6.2.5 for a more thorough discussion). The system architecture is domain independent and therefore other desirable properties can be realised within the architecture.
Moreover, to promote reusability in system development we have specified programmable schemata for MT, OT and IT, which are independent of the application domain and defined at the system structure level. Eventually MT is to contain a library of various schemata for navigating in IT and for reasoning in OT. These schemata directly support the modeling of knowledge, as well as system and program development. The division of knowledge into the three distinct theories gives the system a high degree of modularity. Furthermore, the respective modules are expressed in a natural way, which in turn promotes clarity, making the system easy to survey. This is vital for achieving maintainability, that is, facilitating updating of and alterations to the system.

As described in Section 5.2.3, we have applied the proposed architecture to different domains where the task has been to perform a classification or a diagnosis. Two of the domains refer to a classification of lake water, based on biological and chemical-physical observations respectively. These two are quite different, although they deal with the same domain. There are several hundred biological parameters and only few of them are expected to be found in any one lake. The number of chemical-physical parameters, on the other hand, is limited to 15, but these parameters are more troublesome since the measurement of some of them is difficult to perform and therefore the observations may be uncertain. A third domain is diagnosing childhood illnesses. Here the number of symptoms is limited, but the same symptom may occur as an early and/or a late symptom of an illness. The symptoms may be difficult to determine through measurement, hence the user’s co-operation in the diagnosis is very important. In a forth domain, decision support regarding cash flow behaviour in small firms (Öhlmér, 1998), the context knowledge used for explanations was classified according to our properties.

Our way of categorising knowledge in different levels, in the form of a context tree, has been used in two domains, in which the properties were different. The method was used for reproducing context knowledge in a matching facility (Johansson, 2000) implemented within an EC project, eaisae (Gerret, 1997). Teachers and school leaders use the matching facility when searching for candidate companies for co-operation in the production of computer aided learning material. It is also used by companies trying to find teachers for the role of intellectual author in the production of such material. At the time we became involved in the project, the problem here was that the six countries co-operating within the project defined their own properties. Moreover, a single user can choose to see the other countries’ views of the domain. Our method of using a context tree to describe the domain was adopted by all six countries and worked well. The method has also been
used in an explorative initial study examining the requirements different target groups have for specified information in liaison with major accidents in society (Lindström, 1999). Defining the information was complicated, because the types of information differ both between target groups and for phases of accident. Structuring the domain context as objects in a context tree facilitated the management of the information, both in terms of its acquisition and the presentation.

It is important to note that our properties have been proven to be suitable for problems relating to diagnosis and classification. We have not tested them for other kinds of problems, e.g., for planning and design.

6.2 Related work

In this subsection we relate our system architecture to other research. The subsection is organised as follows. First we discuss our way of dealing with the domain context, and after that the separation of knowledge in three different theories, OT, IT and MT. The cooperation between the user and the system and a discussion regarding system transparency follow. Then we study learning environments. Support methodologies for program development are related to other work. Remarks regarding tools for developing knowledge systems and hypermedia systems end the subsection.

6.2.1 Dealing with domain context

An essential part of our work has been to deal with the context and present it in such a way that the user can understand the system’s knowledge content. “A crucial problem is to find a way for representing the contextual component of knowledge explicitly. The literature is not very helpful at this level because context is rarely implemented explicitly. “ (Brézillon, Pomerol & Saker, 1998).

We have argued that, to reproduce the domain context, it is insufficient to provide information about the object theory, i.e., to only provide the knowledge needed for the reasoning. Wenger expresses the same opinion: One way to articulate superficial knowledge in a knowledge base is to augment it with information that explicitly provides
the missing justifications for the knowledge (Wenger, 1987). Hence, the limitations of formalisation necessitate the inclusion of informal domain knowledge in the system.

There have been various proposals for formalising context. In a sense, we stand in opposition to them, since we take the context to be the informal meaning of the domain knowledge that is not captured in the formalisation, and that frequently cannot even be formalised with any known methods. Nevertheless, there are similarities between our approach to dealing with informal domain knowledge and the formal approaches to context. We carry out formal reasoning with pieces of informal knowledge, so as to focus the user’s attention on relevant information, for example. This is paralleled by Giunchiglia’s (1993) view that an agent never considers all the available knowledge, and therefore reasoning is local to a subset – a context – of the facts known by an individual. Moreover, we share the opinion that it is useful to have different contexts, which are theories of the same phenomenon, but describe it at different levels of approximation.

The importance of separating knowledge from the symbol level in a computer system is emphasised by Newell (1982). He proposes a knowledge level, defining the nature of knowledge, which lies just above the symbol level. The system at the knowledge level is an agent. This agent has certain goals, a set of actions and a body of knowledge, which consist of some kind of memory and knowledge relating goals and actions. In our system architecture the goals are defined in MT and the actions in OT. MT and the user together possess the body of knowledge where IT serves as a link between both parts. Moreover, our view of the user and the system as mutual parts deals with the problem that the knowledge level is, according to Newell, only an approximation and sometimes a rather incomplete one (ibid.).

Newell argues that advantages with the knowledge level theory are that a distinction between the knowledge and symbolic levels gives a comprehensive and consistent view of knowledge and knowledge representation and allocates a deserving role to mathematical logic.

Lucardie (1994) states that “Newell’s theory sets preconditions by providing the framework for developing and analysing theories of the nature of knowledge”. We agree with Lucardie that such a framework is important, especially within the area of knowledge systems.
6.2.2 Separating knowledge in different theories

Aiello and Levi (1988) propose that structuring knowledge in separate theories and relating them by means of metaknowledge is a way to increase the expressive power of knowledge representation languages. There is a need to represent explicitly various types of knowledge in a knowledge-based system (Benaroch, 1996). Many intelligent systems today combine several reproduction forms for knowledge modelling and knowledge representation (Devedzic´, 1999). The reason is, according to Devedzic´, that all techniques have strong points, but they also have some limitations and shortcomings. In our system architecture the informal domain knowledge is represented in IT in its most natural form, such as text, pictures, animations, sounds, etc. In OT the knowledge can be formalised as, for instance, rules and objects. MT’s main role is to represent the relations between the different kinds of knowledge and therefore a formal language capable of expressing arbitrary relations, such as first order logic, is used.

We decompose the knowledge into three theories. Although several authors consider them to be just two theories, for instance Wenger (1987) who claims that two forms of knowledge representation are needed in a knowledge communication system, the internal and the external representations. In an ideally “intelligent” knowledge communication system, the interface is strictly an external representation of the expertise that the system possesses internally. This model corresponds to our theories of IT and OT, but it lacks the metatheory for relating the two representations. In (Edman, Hamfelt, Koch, & Wagner, 2000) we exploited our architecture for outlining a web-based user adaptive learning environment where the web sites are expressed in controlled natural language (Fuchs, Schwertel & Schwitter, 1999a; Fuchs, Schwertel & Schwitter, 1999b). In such a language the external representation (IT) is directly mapped to the internal formalisation (OT) allowing MT to query the web-sites directly for implicit knowledge.

Clancey and his associates found that it was necessary to separate the strategic diagnostic knowledge from the domain facts and rules when using the knowledge in MYCIN as a tutoring system (Clancey & Letsinger, 1981) (introduced in Section 2.2). These two knowledge types can be compared to MT and OT respectively.

Wick and Thompson (1992) also stated that it was necessary to divide the knowledge in REX (introduced in Section 2.3.1) into two parts, one for the knowledge used for the problem solving and one for the description of this activity with an interface between. In
our system MT performs both sorts of reasoning, the problem solving required to reach a conclusion and the generation of the explanations. Therefore, a special interface between the two theories is not needed as it was in REX, which is advantageous as this was the weakest part of the system. The interface is a semantic network and it will only be useful if the number of nodes is restricted, both because of the type of the representation and of A*, the search algorithm utilised. In REX a schema of constraints is implemented in order to approach the consistency problem between the two subsystems. The constraints are of two kinds, i.e., constraints imposed on the problem and the solution. Problem constraints determine the kind of reasoning cues that should pass from the expert system to the explanation system. The solution constraints control the amount of freedom given to the explanation system when generating the explanation as opposed to the reasoning cues. In our system MT knows about the relation between the knowledge used in the problem solving and in the explanations and also about the structures of the theories, which we think is an important advantage.

Work can also be found in the literature that relies upon a decomposition of knowledge into three theories. In a system dealing with control on the subway line, Brézillon et al distinguish between the domain knowledge, the knowledge of how to use it and the knowledge required to communicate (Brézillon, Pomerol & Saker, 1998). This can be compared to OT, MT and IT respectively. Moreover, in an earlier article, Brézillon (1992) proposed an explainable knowledge system comprised of an application knowledge base (AKB), an explanatory knowledge base (EKB), and a domain-independent part called the Manager. The Manager performs the reasoning by using both knowledge bases and it synchronises the two lines of reasoning. Furthermore, it handles the interaction with the user. The division of the system into three parts resembles our approach, where OT corresponds to AKB, IT to EKB and MT to the Manager.

The Dexter Hypertext Reference Model (Halasz & Schwartz, 1994) (introduced in Section 3.1) provides a standard hypertext terminology coupled with a model of the important abstractions commonly found in hypertext systems. The model exhibits certain features reminiscent of our system architecture. The Dexter model divides a hypertext system into three layers, the run-time layer, the storage layer and the within-component layer. The storage layer models the basic node/link network structure. This layer describes a database that is composed of a hierarchy of components interconnected by relational links. This corresponds to our MT. The within-component layer has no direct counterpart in our framework and is treated by the Dexter model as being outside the hypertext model. The
storage layers treat hypertext as an essentially passive data structure, whereas our MT allows one to incorporate intentional rules offering the possibility to perform deduction. In Dexter, the functionality for the user to access, view and manipulate the network structures is captured by the run-time layer of the model which would correspond to the program that executes MT in our approach.

The Amsterdam Hypermedia Model (Hardman, Bulterman & Van Rossum, 1994) extends the Dexter model by adding the notions of time, high-level presentation attributes and link context to it. The temporal relationships between data are partitioned into two classes: those related to identification of the components which are to be presented together (called collection), and those related to the relative order in which these components are presented (called synchronisation). The principal difference from the Dexter model is that the composite components serve to build a presentation structure rather than simply to collect related components for navigational purposes. This resembles the possibility in our framework to perform a dynamic computation of the presentation of IT knowledge.

6.2.3 Co-operation between user and system

A knowledge system cannot initially include all knowledge that will be needed (Brézillon & Cases, 1995), which implies that knowledge must be acquired incrementally from the user when required. The reason for this incompleteness may, for instance be, that it is hard to cover the knowledge, there are gaps in the completeness of the knowledge, such as missing constraints and the knowledge is changing (Shadbolt, 1997). Furthermore, the knowledge may not be formalised. A solution to the problem, that we utilise, is that the user and the system co-operate in the problem solving and that the system is augmented with the user’s contributions. The demands on a co-operative computer, as defined by Clarke and Smith (1993), are addressed. These are that the computer must be able to recognise and accept the user’s goals together with the user in an interactive manner, work towards super-ordinate goals in solving complex tasks, and support the formation of new attitudes the user has towards the task.

The user may need support when contributing to the system’s knowledge. “Knowledge acquisition scripts” (Tallis & Gill, 1999; Blythe, Kim, Ramachandran & Gil, 2001) support users when modifying the knowledge in a knowledge system. The scripts provide a context for relating individual changes to different parts of the knowledge base and guide
the user in changing all related parts in a consistent way (Tallis, to be published). This approach is implemented in ETM, a knowledge acquisition tool. Our architecture accepts that the user augments the knowledge in the system, but not that the system’s knowledge is changed by the user, which ETM supports. Our focus, though, is to get a well functioning system that acts in co-operation with the user and not to develop a tool for knowledge acquisition.

According to Stolze (1991) the focus may not be to fully automate the problem solving, but to optimise the combination of user and knowledge system. Brézillon and Abu-Hakima (1994) state that problem solving is a co-operative task between a user and a system. Instead of describing how the system can solve a problem the goal is to figure out how the system and the user can work together, which is one of the main topics of “Intelligence Augmentation” (Borges & Baranauskas, 1997). When the user is supposed to learn from the system’s knowledge it is vital that the user’s role is an active one (Cunningham, Duffy & Knuth, 1993), which is an automatic side effect of a co-operative system. Furthermore, there is a hypothesis that interactivity supports learning because the user can control his/her learning process (Elmeroth, 1999).

Nielsen states that a user and a hypermedia system co-operate, since the user can understand the semantic contents of nodes and decide which link to follow to reach the next node (Nielsen, 1995). Our opinion is that there is no real collaboration since the system can only present information and not provide support in the problem solving. The user has to perform the problem solving outside the system. Furthermore, the user has to keep track of several tasks at the same time, such as searching for relevant links to appropriate nodes to obtain the domain knowledge, which costs effort and requires concentration (Conklin, 1987). Besides, Beishuzien et al (1996) argue that the cognitive demands on readers of hypertext are higher than on readers of ordinary texts, which include a table of contents and an index. We state that when there is a co-operation between the system and the user, the system has to support the user in finding the relevant information and maybe even in tailoring the information to the user’s current needs.

6.2.4 System transparency

If the user is to be able to co-operate in the problem solving the system has to be transparent. In knowledge systems transparency is closely related to explanations. In an
experiment Gregor has showed that in co-operative problem solving the need for explanations was higher (2001). An efficient co-operation between a human and a system depends on the system’s capabilities to explain, to incrementally acquire knowledge and to make explicit the context (Abu-Hakima & Brézillon, 1994; Brézillon & Cases, 1995; Brézillon, 1998).

Chandrasekaran and Swartout (1991) state that “knowledge systems based on explicit representations of knowledge and method with information about how and from what their knowledge was obtained are the foundations for producing good explanations”. They argue that explanations can be as important as the conclusions themselves. The general idea is that the more explicitly the knowledge underlying a system’s design is represented, the better the system’s explanations will be. We fully agree with this.

Explanations are of course important in a learning environment and may influence the users’ performance. For instance, offering users an explanation facility led to a better performance when using a statistical package (De Gref & Neerincx, 1995). Through tests it was observed that when the explanations were suitably designed both performance and learning were improved (Gregor & Benbasat, 1999; Gregor, 2001).

Model-based explanations are used in second generation expert systems. In these systems both deep and shallow knowledge and the interactions between these knowledge types are represented (Steels, 1985; 1987). This implies that the knowledge system has a structural model of the domain (Steels, 1990). We argue that our system together with the user’s interpretations can be considered as reproducing a domain model in the following way. IT mirrors the domain on different levels of abstraction and from different perspectives. OT describes relations and objects within the domain in a formalised way. MT utilises IT and OT to present the conceptual and inferential domain context to the user as a basis for the user’s interpretation of the domain model. Furthermore, we argue that the explanations generated by our system are model-based. In Section 2.3.2, a classification of knowledge in model-based explanations was presented. This classification was based upon Clancey’s, Swartout’s and Chandrasekaran’s work. Later, in Section 5.2.3, the properties coupled to different categories of objects in our system were displayed, properties used for reproducing the domain context. These properties are utilised in the explanations. When comparing the classification with our properties, we found that our system generates all the knowledge types suggested for the current model-based explanations and even displays some additional information.
There is a lot of research related to explanations. For instance Benaroch (1996) generates the explanations in four different modules. These modules define and justify the knowledge in the knowledge base, justify strategic principles, explain why the solution produced is good and how the task is solved. All these knowledge types can also be generated in our system.

The focus for the explanations in XPLAIN (introduced in Section 2.3.1) is to present how the result was attained once the reasoning has been conducted (Swartout, 1981). The knowledge in XPLAIN, referring to OT in our system, is the result of an automatic programming process using the domain model and principles. The domain principles in XPLAIN are part of our IT, but IT contains more knowledge. IT consists of knowledge not needed in the reasoning but needed for the understanding, to which Swartout has no real equivalence. Furthermore, the knowledge in IT can be reproduced in the most suitable form and does not have to be represented symbolically, as the domain principles have to be. The implementation of XPLAIN is a kind of hybrid system where the relationship between the knowledge needed for the reasoning and for the explanations respectively is not explicitly defined. How this process is performed is represented in the automatic programmer. In our system the relations between OT and IT are distinctly defined in MT, which we state is of great importance for the theoretical framework of knowledge systems. Updating the theories is facilitated by this construction, which may be complicated in a system using automatic programming. Furthermore, MT facilitates the reasoning and the user’s participation.

Swartout, and Chandrasekaran and his associates are, principally interested in describing the system’s problem solving methods to the user, i.e., what the system has done and how it was performed. The focus for our research is different, since our main aim is to reproduce so much of the domain knowledge that the user can understand what the system “knows”, and maybe learn from the system and co-operate with it in a meaningful way whilst conducting the reasoning. Moreover, for most realistic problem situations, giving explanations at the end of the problem solving is clearly inappropriate according to Cawsey et al. (1992).

Brézillon and Cases (1995) state that one of the failures with knowledge systems is that they cannot generate relevant explanations for users because the system has no knowledge about the user’s problem solving context. Therefore, the user and the system must co-
construct the explanation in the current context of problem solving (ibid.). In our system architecture the user interprets the informal knowledge and the system’s knowledge in OT may be augmented with the user’s contribution. We also utilise the possibility to let the user confirm a goal that cannot be proved, i.e., to change properties of MT by extending it with a theorem, in the same manner as, among others, e.g., Sergot (1983), Aiello & Levi (1988) and Hamfelt (1992). Incremental knowledge acquisition is consonant with monotonic classical logic as long as the formalisation is only extended with additional axioms, and no accepted formulas are excluded afterwards. It is also possible to let IT expand with information from the user, as Brézillon et al suggest (1995), but this has not yet been implemented.

So far, we have only discussed transparency in relation to knowledge systems. A hypermedia system’s transparency could be discussed according to the usability of the system. The usability of a hypermedia system can be seen as a combination of the usability of the underlying system engine and the usability of the contents and the structure of the information in the system, and by how well these two fit together (Nielsen, 1995). The system engine in a hypermedia system deals with browsing and searching mechanisms. These are not utilised in our system architecture for knowledge-based hypermedia because of the shortcomings of static links and the lack of support. In our system the links are generated by knowledge-based reasoning, which means that the information can be generated dynamically, and that it is composed on the basis of its relevance and the user’s needs. Moreover, the content can be reproduced as a coherent domain theory in IT, which is interpreted by the user, and the information is accessible through different kinds of overviews.

6.2.5 Learning environments

Like intelligent tutoring systems, or ITS as they are known, the knowledge-based hypermedia system can serve as a learning environment. ITS comprises domain expertise, pedagogical expertise, a student model and an interface (Wenger, 1997). The knowledge-based technique is useful in tutoring systems. Gobet and Wood state that without knowledge-based models of the learning process, the attempts to implement tutoring systems intended to help learners to construct links between their procedural knowledge and conceptual understanding have had limited success (Gobet & Wood, 1999). Kelly (1990) suggests that the domain component offers the most promising support to teachers
when comparing the four components of an ITS. Knowledge-based technology is of course useful for implementing this component. Moreover, knowledge-based interface design can be useful for the design of intelligent educational systems (Bourdeau & Borne, 1991). Callear (1990) chose to use a knowledge-based method when implementing the student model in a course-oriented ITS. Expert system shells have also been used by students to build knowledge systems as a means to learn the current domain concepts and relations (Nydahl, 1991). Hypermedia systems can also support pupils’ learning if they get the opportunity to create nodes and links themselves (Cunningham, Duffy & Knuth, 1993). An example of such a system is HIPERGRAFE which supports students learning Portuguese spelling rules by composing multimedia pieces from an initial hyperdocument (Strube de Lima & Beiler, 1997).

When constructing systems for tutoring, the fundamentals of learning should be considered. According to Woods (1994) these relate to motivating the learner, providing goals, giving the context, selecting knowledge that is not known by the user, ordering the presentation of this knowledge and planning the presentation, facilitating the learning by being active and aiding the learner to recall information that he/she might like to reinspect.

It may be problematic to deal with motivation in a system, if solving the current problem does not offer enough motivation to the user. Utilising ideas from computer games may be a solution. “Think of what it takes to learn a game compared to what has to be done in school. To play a game well requires the same kinds of learning, study, understanding and practice as are required of any educational activity. There is no reason why the learning and studying required in education should not be as captivating and enjoyable as studying of the game. It is remarkable how little scientific knowledge we have about the factors that underlie motivation, enjoyment and satisfaction” (Norman, 1994). And indeed, with this in mind, there is a knowledge system Mission: Water (Kylberg, 2001) that uses ideas from games. The system is related to Analyse More and DLW (introduced in Section 5.1.1 and 5.3.2 respectively). Klöver (introduced in Section 5.1.1) was partly used as a tool in the implementation of Mission: Water. The system encourages the pupils to perform the analysis themselves, after being assigned a mission (ibid.). To perform this the user needs to find relevant information and execute tests, under constraints of time, money and instruments.

The other demands made by Woods’ are dependent on the possibility to identify goals, organise the interaction with the user, reproduce the domain knowledge, present the
domain context from different perspectives and through the provision of different overviews, and support problem solving. All these tasks are addressed in our proposed system architecture and are handled by MT.

Woods’ fundamentals refer to the domain model, i.e., a reproduction of the domain. However, the content must be related to problem solving. According to Wittgenstein “The meaning of information is determined by its use” (Elmeroth, 1999). Domain information may be knowledge personal to the user provided that one knows how to use it, and problem solving facilitates its use. The domain model and problem solving within the domain can be seen as the kernel in a learning environment. When reproducing the domain knowledge the forms of knowledge representation are important. For instance, the representations of complex scientific and mathematical domains can substantially determine what is learnt and how easily (Cheng, 1999). In tests Wood (1999) found that different representations, such as diagrams and written instructions, presented different perspectives of a domain. Although different representations contained equivalent information, they supported various kinds of reasoning, insights and problem solving. Ainsworth (1999) has even described a functional taxonomy of multiple external representations, MERs, which are used to support cognitive processes in learning and problem solving.

Advanced knowledge representation techniques allow the presentation of both a conceptual and an inferential context, which is realised in our system architecture. The conceptual context is obtained by structuring the documents by concept-based techniques, which, e.g., allow the domain knowledge to be described at different levels of abstraction. In contrast, the inferential context is obtained as a description of the reasoning carried out to find a certain document. This is a way of partitioning the information, known as chunking; according to Woods (1994), chunking facilitates mental processing. One associated problem, which is typical for hypermedia systems, is fragmentation of the knowledge which may make difficulties for the user when trying to comprehend it (Whalley, 1993). We argue that the domain is reproduced in IT and OT in a coherent manner, thereby dealing with the fragmentation problem. The displayed conceptual and inferential context both offer overviews of the domain, which is important as a means for supporting deep learning (Woods, 1994). Moreover, they support top-down and bottom-up procedures, both of which are vital for complex learning (Clancey & Letsinger, 1981; Stanton & Stammers, 1989; Wærn, 1989). The problem solving starts in a top-down manner and when the user has contributed with his/her knowledge the process switches to a bottom-up
one. The display of the conceptual context can be presented top-down and bottom-up in relation to specific objects. Furthermore, the initiative is divided between the user and the system which is important (Shneiderman, 1986).

To support in-depth learning in a learning environment for schools it is vital that the system can be integrated with other media, e.g., textbooks and students’ guides (Lindh, 1997). Furthermore, the system has to be interactive so the pupils can get answers to questions and also so that the system can support the pupil’s work with system relating to the individual’s performance (ibid.).

CORE is a method for the acquisition of competence in complex domains, based on so-called “guided discovery” (Boyle, 1997). CORE is an acronym capturing the design principles of the approach; it stands for Context, Objects, Refinement, and Expression. A comparative evaluation has been performed of a multimedia learning environment for nuclear medicine, based on CORE, the comparison was made with lectures and an electronic book (Hogg, Boyle & Lawson, 1999). All three were based on the same curriculum. One found that the students appreciated the CORE approach and especially the possibility to learn in context with real examples that integrated diverse subject areas, encouraged engagement in the learning process, and learning from mistakes. In our system we could deal with these concepts but so far we have concentrated on developing a co-operation between the system and the user and we are not trying to implement a system that tutors the user like, e.g., a teacher.

A group of graduate students had the opportunity to use a knowledge-based hypermedia system in conjunction with a course to train teachers working with those with behavioural disorder (Kraus, Reed & Fitzgerald, 2001). In developing the system, the goal was to enhance the problem-solving skills of teachers who work with children with behavioural problems. The program provides a knowledge system in which the user can seek and explore information, get different experts’ views on a case and perform problem solving from multiple roles. It is interesting to note that users, i.e., the teachers benefited equally from using the program, regardless of their personal learning styles as categorised by Kolb (introduced in Section 3.1.1) (ibid.). Their interpretation was that the program provided a variety of learning opportunities for learners with different learning styles. Likewise, we adopt the believe that most users can benefit from an environment offering different kinds of knowledge and different ways of displaying this knowledge.
Diaz, Aedo, Torra, Miranda & Martin (1997) propose a hypermedia training system to support different styles of learning. There are two parts: the information element used in the exercise (e.g., texts, images, animation and videos) and the logic of resolution, also called strategy, which must be applied. The first can be compared with IT and the second with OT. Similar to our architecture, the system offers the modification of information elements without altering the strategy used to solve the problem and vice versa. The same knowledge can be the focus of attention from different perspectives since information elements can be tied to the different strategies used to solve problems. Moreover, the most appropriate strategy for each user can be selected since the same logic of resolution can be applied to distinct information elements. The relation between the strategy and the information objects is stored in the exercise base, which thus corresponds to MT.

A knowledge-based hypermedia system for the topic of dyeing textile materials was developed for use in secondary schools (Pivec & Rajkoviè, 1997). The system consists of a knowledge base implemented in Nexpert Object where the experts’ knowledge and decision making process are stored, a relational database containing e.g., data about dyes and technological procedures, explanation pages presenting basic knowledge and a user interface in hypermedia cards. The user interface deals with the user’s input, guides the session and provides the user with additional explanations of the system’s answer as well as providing domain knowledge. Compared to our system architecture, OT corresponds to the knowledge base and the relational database, IT to the explanations pages and MT to the manager of the user interface and the hypermedia cards.

The Protocol Assistant is a knowledge-based hypermedia system which gives the user access to abstracts of published papers, using the hypertext facilities of HTML (Simpson, Kingston & Molony, 1998). The system offers support in two different ways, either as some kind of wizard which guides the user through the decision making or as a hypertext manual which leads the user to the information relevant to the current decision. Our system architecture plays both roles. The system is a problem-solving wizard, but a wizard that is open to a user’s co-operation. At the same time the system generates relevant presentations of the domain context to the user so he/she can solve the current problem.
6.2.6 Support methodologies for program development

Software projects benefit from support methodologies for program and system development. An important component in a methodology is a system architecture providing reusable program and system module schemata. We have taken some steps towards creating such an architecture for knowledge-based hypermedia. It is important to relate to alternative or complementary approaches to support the development of knowledge-based systems and hypermedia systems.

KADS (KBS Analysis and Design Support) and CommonKADS are tools supporting programming in the large for knowledge systems (Wielinga, Van de Velde, Schreiber & Akkerman, 1993; Schreiber, Wielinga, de Hoog, Akkermans & Van de Velde, 1994). These tools however concentrate on knowledge modelling and do not prescribe any particular representation approach such as object-oriented or rule-based languages. CommonKADS promotes reusability in knowledge modelling by a library of so-called metamodels. In contrast to these tools, our approach is to promote reusability in knowledge representation by supplying the above program schemata for MT, OT and IT. These schemata are independent of the application domain but are defined at the system structure level for a particular family of systems (dual intelligent hypermedia systems). MT is to contain a library of various schemata for navigating in IT and for reasoning in OT. These schemata directly support knowledge modelling and also system and program development.

Our approach and KADS (Wielinga et al., 1993; Schreiber et al., 1994) are inspired by the same emphasis on structure in system development. KADS methods represent a structured systematic approach to the initial system development phases, i.e., to the analysis and the definition of requirements and to the design of the system (Tansley & Hayball, 1993). In contrast, our focus is on the subsequent phases of software design and implementation where we impose a certain system structure at a very general level.

When developing hypermedia systems one problem encountered when authoring hypermedia material is to adopt the facilities offered by the media and the current authoring tool. Tests show that novice authors created documents in a rather traditional way (Pohl & Purgathofer, 2000). Overview maps were the only innovative features utilised by the authors. Most of the nodes and links were created on the overview map and a lot of time was spent on editing this map. Thus, there is a need for guidance when designing
hypermedia material. A step towards such a support is Kemp and Buckner’s (1999) taxonomy of design guidance for hypermedia systems comprising design methods, models, principles and guidelines, standards and rules. However, Mendes and Hall (1999) found through interviews that there is no need for an authoring methodology to improve hypermedia applications. Authors do not seem to change their writing processes because they are using hypermedia. Another way of improving authoring is to offer an environment so the authors can implement the applications without making much effort, reuse the information and easily maintain the application (ibid.). The schemata in our proposed system architecture support the author by offering a structure for representing the informal knowledge and different kinds of overviews.

6.2.7 Tools for developing knowledge systems and hypermedia systems

Expert system shells and authoring systems are the tools used for developing knowledge systems and hypermedia systems, respectively. Expert system shells support the development of formalisations of domain knowledge corresponding to OT and provide inference machines for this object level. Second generation expert system shells also support the modelling of deep knowledge through the use of, for example, model-based reasoning (Steels, 1990). These shells may also support interface design but do not facilitate the presentation of a more coherent IT and have no metatheory for dynamically composing informal presentations reflecting the current stage of reasoning in the domain theory. Authoring systems, on the other hand, provide support when reproducing information in the system, when creating nodes and when deciding associations between these nodes. These systems cannot perform problem solving and facilitate the search for relevant information. Efficient access to the information content of a hypermedia system requires more advanced guidance (Waterworth, 1992). Second-generation hypermedia systems provide such facilities, mostly implemented by drawing on techniques from knowledge systems.

Nanard and Nanard (1995) propose a knowledge-based method for taking into account the context of the use of anchored texts. Factual knowledge about the document base is first extracted from the documents by a user-interactive linguistic analysis run under the control of a human expert. This knowledge is used to map a hypertext structure - characterising the semantics of the anchors - on the document base in order to easily
access semantically related information. In our terminology, this corresponds to a semi-
automic extraction of MT knowledge.

The Microcosm system (Hall, 1993) provides a sort of metahypermedia tool (Boyle, 1997) characterised by a separation of links from the information stored in the nodes in contrast to traditional hypertext systems where the links are embedded in the documents. In such an open hypermedia system the links may be processed separately as data. This corresponds to our separation of MT from IT.

Nørmark and Østerbye (1995) separate the internal hypertext representation from the external presentation in a manner similar to our MT and IT. This enables that one gets a conceptually clean internal representation as a common basis for a variety of tools for processing hypertext and an arbitrary number of different presentations of the same internal representation.

In adaptive hypermedia, support can be provided at two levels: through the selection and presentation of content at the node level, or by navigational guidance at the link level (Boyle, 1997). Both correspond to MT reasoning for presenting IT, but the former is a display of conceptual context, whereas the latter is inferential context.

To conclude, knowledge systems and hypermedia systems are flip sides of the same coin insofar as they complement each other (Rada & Barlow, 1989b). Developments of each system type require techniques from the other type since both automatised reasoning and unguided navigation have their inherent shortcomings. Automatised reasoning is incapable of dealing with full domain theories and, therefore, requires support from the user for interpreting the unformalisable parts of the domain. To this end, the user needs access to informal descriptions of the domain knowledge and the system’s inferencing. These descriptions must be generated in run-time since they reflect arbitrary states of the reasoning. Hypermedia systems offer access to static informal descriptions, but complemented with knowledge based techniques, such descriptions can also be computed dynamically. Moreover, in a large hypermedia system the user is lost without guidance, which, naturally, can be implemented with knowledge techniques.
7 Further work

Our architecture supports system development by providing the programmer with programmable schemata for the three system parts: IT, OT and MT. Visualisation of these schemata could further facilitate the development work. Our long-term objective is a system architecture where we can carry out piecewise compositions of intelligent hypermedia systems through diagrams giving graphical descriptions of the schemata of the architecture. In addition, this would support the knowledge-acquisition phase since the system would support structuring of domain knowledge, and even through its diagrams, display the knowledge for validation, see, e.g., the Wisdom system for knowledge modelling (Lucardie, De Gelder & Helsper, 1997).

To deal with IT we have identified a number of domain properties of relevance to the tasks of diagnosis and classification. It is impossible to account for all properties that might be relevant completely and the architecture should be kept open for extensions. However, we will continue to analyse and identify properties for other tasks such as design and planning, etc. It is important to identify such properties in advance, not only because they determine how to compose explanations, but also because they ease the knowledge acquisition phase. Empirical large-scale tests are needed for evaluating the proposed architecture. Therefore, in further work, we will carry out a fully-fledged implementation of a system.

Knowledge-based models of the learner (Gobet & Wood, 1999; Wood & Wood, 1999) can be used for adapting the explanations to the current user’s needs. One feature of intelligent interface technology is that a representation of the user is included. The representation describes facets of user behaviour, knowledge and aptitudes, and may have a greater or lesser degree of formality (Benyon & Murray, 2000). Such a user model (called the student model in tutoring systems) can be incorporated into the general schema of our architecture. However, adaptable explanations may violate the usability principle of predictability. The system will not always give the same response given the same input which may reduce the transparency of the system (Höök, 2000). Thus there is a trade-off between adaptability and the predictability of explanations. However, by enabling the user to choose to have alternative explanations presented may reduce this problem.
References


Edman, A. (1990a) The expert system vs. the information system and a prototype of an expert system for selecting appropriate *Salix* clones with respect to the site. *Proceedings of Expert System Workshop*. Swedish University of Agriculture Sciences, Uppsala, Sweden.


130


