KADS[II]

ESPRIT Project P5248 KADS-II

An Advanced and Comprehensive Methodology
for Integrated KBS Development

Machine Learning Issues in CommonKADS

Walter Van de Velde
Agnar Aamodt

Document Id.	KADS-II/TII.4.3/TR/VUB/002/3.0
CEC Reference	D2.11
Date	January 31, 1992
Distribution	Consortium, CEC

KADS-II Consortium

This document is part of a research project partially funded by the ESPRIT Programme of the Commission of the European Communities as project number 5248. The partners in this project are Gap Gemini Innovation (F), Cap Gemini Programator (S), Netherlands Energy Research Foundation ECN (NL), ENTEL SA (ESP), IBM France (F), Lloyd's Register (UK), Swedisch Institute of Computer Science (S), Siemens AG (D), Touche Ross MC (UK), University of Amsterdam (NL) and Free University of Brussels (B).

Cap Gemini Innovation
Paris Research Centre
86/90, rue Thiers
92513 - Boulogne
France
Contactperson: M. Georges
Phone +33.1.49.10.53.98 Fax: 33.1.49.10.06.15
E-mail: mari@capsogeti.fr

Cap Gemini Programator
Norra Ringv. 36 Box 7
S-721 03 Västerås
Sweden
Phone: +46 21 14 44 50 ; Fax: +46 21 11 41 00
Contactperson: W. Menezes
E-mail: wm@industri.vs.cgl.se

Netherlands Energy Research Foundation ECN
Software Engineering & Research Department
P.O. Box 1, NL 1755 ZG Petten
The Netherlands
Phone: +31 22 46 44 58; Fax: +31 22 46 18 64
Contactperson: Dr. J.M. Akkermans
E-mail: akkermans@ecn.nl

ENTEL SA
P. de la Castellana 141
28046 Madrid
Spain
Phone: +34 1 348 12 23 ; Fax: +34 1 279 10 74
Contactperson: S. Gala
E-mail: sgala@ic.eritel.es

IBM France
Rue de la vieille Poste
BP 1021, 34006 Montpellier
France
Phone: +33 67 34 40 43; Fax: +33 67 34 48 96
Contact person: G.Rondot
E-mail: rondot@mopprofs.vnet.ibm.com

Lloyd's Register
29, Wellesley Road
Croydon CRO 2AJ, Surrey
United Kingdom
Phone: +44 816 814 040 ; Fax: +44 816 816 814
Contactperson: R. Martil
E-mail: rob@lloydreg.uucp

Swedish Institute for Computer Science
KBS Department
Isafjordsgatan, 16428 Kista
Sweden
Phone: +46 87 52 15 20 ; Fax: 46 87 51 72 30
Contactperson: R. Gustavsson
E-mail: rune@sics.se

Siemens AG ZFE BT SE 1
Otto Hahn Ring 6
D-81739 München
Germany
Phone: +49 89 636 40798 ; Fax: 49 89 636 42284
Contactperson: C. Bauer
E-mail: cb@zfe.siemens.de

Touche Ross Management Consultants
The Knowledge Based Systems Centre
Peterborough Court, 133 Fleet Street
London EC4A 2TR, United Kingdom
Phone: +44 71 936 3000 ; Fax: +44 71 583 1198
Contactperson: L. Land
E-mail:

University of Amsterdam
Department of Social Science Informatics
Roetersstraat 15, NL-1018 WB Amsterdam
The Netherlands
Tel: +31 20 525 6789; Fax: +31 20 525 6896
Contactperson: B.J. Wielinga
Electronic mail: wielinga@swi.psy.uva.nl

Free University of Brussels
Artificial Intelligence Laboratory
Pleinlaan 2, B-1050 Brussels
Belgium
Phone: +32 2 641 641 3700 ; Fax: +32 2 641 3729
Contactperson: W. Van de Velde
E-mail: walter@arti.vub.ac.be

This document was produced using the latex wordprocessor

Machine Learning Issues in CommonKADS

Walter Van de Velde
Agnar Aamodt

Synopsis
This report represents the intermediate results of Task II.4.2 "Machine Learning Issues". The goal of this task is to analyze the opportunities for applying machine learning methods in the various phases of the KBS life cycle. It is studied how machine learning techniques can contribute to the construction, modification, implementation, and maintenance of KADS-II knowledge level models. The purpose of this document is to describe how machine learning can be integrated in a principled way into the CommonKADS modeling framework and life-cycle approach. Four distinct roles for machine learning are proposed. Classification schemata for machine learning methods are described and a specific one for CommonKADS is proposed. The document also provides a general discussion of the different roles for machine learning within CommonKADS from a top-down perspective and sets out the lines for further work on specific issues.

Document Id.	KADS-II/TII.4.3/TR/VUB/002/3.0
CEC Reference	D2.11
Date	January 31, 1992
Distribution	Consortium, CEC

Revision Information

Revision date: January 31, 1991

Modifications: This deliverable was delivered before the review in November 1991 (document ID: KADS-II/TII.4.3/TR/VUB/001/3.0., October 31, 1991). No changes to the content of the report have been suggested. The present document is the final version of the D2.11 deliverable. No major changes have been made to the previous version, but some adjustments have been made and a few paragraphs were extended, in order to improve clarity.

Contents

1 Introduction

The document is the result of the work to date on Task II.4.3 “Machine Learning Issues”. The KADS-II project aims at developing a ’standard’ and state-of-the-art methodology and a set of support tools for the specification, development, and maintenance of knowledge-based systems. Task II.4.3 studies machine learning issues related to this. In the context of this task, machine learning must be understood in a broad sense, including techniques for concept formation, knowledge compilation, change of representation, as well as highly automated methods for knowledge acquisition.

The purpose of this document is to describe how machine learning can be integrated in a principled way into the CommonKads modeling framework and life-cycle approach.

1.1 Objectives of the task

Large scale application of knowledge intensive AI systems stands no chance of success if these systems have to be build and maintained by hand only. Ingenious programming environments are not at all sufficient to increasingly adapt programs to new problems and needs. Intelligent systems need to be able to acquire knowledge and experience more easily, and largely on their own. This is the long term research objective of machine learning ([Michalski et al., 1983, Michalski et al., 1986, Kodratoff and Michalski, 1990]).

The overall goal of this task is to perform some of the scientific groundwork that is required to bring machine learning (ML) up to the level of the state of the art in knowledge based systems. We acknowledge that research in machine learning has produced an important stock of techniques and insights. Nevertheless, we will argue that the actual needs, arising from current practice in knowledge-based systems have not been sufficiently addressed yet. Machine learning has not been studied within frameworks like the one provided by CommonKads although this must clearly be a target if the field is to become of practical relevance. This gap is the one that this task attempts to bridge.

In particular, Task II.4.3 has three major objectives:

1. To explore the opportunities for applying machine learning techniques within a CommonKads project. This concerns both the usage of machine learning as a tool in the project life-cycle, as well as a tool for learning within the product KBS.

2. To apply the modeling perspective of CommonKads to machine learning itself,

resulting in intermediate models of machine learning that can be integrated with models of problem solving in a CommonKads product.

3. To analyze the implications for project management by formulating guidance for the insertion of machine learning tasks and activities in the configurable CommonKads life-cycle.

This document addresses the first two objectives.

1.2 General approach

Our approach to the problem of integrating machine learning within the CommonKads framework and methodology can be characterized as "top-down". We have chosen not to start from a survey of state-of-the-art machine learning and knowledge acquisition techniques and to investigate how these can be used within CommonKads. Instead, we will consider the general framework of CommonKads as it has been developed with a focus on problem solving, and investigate how it also can be usefully applied to learning. From the perspective of problem solving this is an original approach, since it is an attempt to really integrate learning and problem solving at the same level instead of treating learning as an "add-on" to an existing problem solving framework.

The approach also reflects an important shift in machine learning research. Instead of looking at learning in isolation we investigate learning in the context of modeling knowledge based problem solvers. It is this context which ultimately determines the role, the goal, and the success or failure of learning. A proper understanding of this context is a primary requirement for a theoretically and methodologically well-founded approach to Machine Learning. Studying this context is exactly one of the goals of KADS-II. In this task we will elaborate the theory for learning instead of for problem solving. The result will be a methodology for configuring the life-cycle of a learning task and for managing its proper execution. There is currently no such methodology of machine learning. There is a collection of techniques and tools, but no "simple" way to use them. Such a methodology is different from a clear understanding of machine learning techniques, just like an effective method for driving a car is different from detailed knowledge of a car's workings. So far, Machine Learning research has been concentrating on building good cars, but insufficient effort has been spent on learning to drive them well. Recent machine learning-books [Michalski and Kodratoff, 1990, Shavlik and Dietterich, 1990] are, though invaluable sources of reference for the active machine learning-researcher, not at the level of the potential machine learning-user i.e., the knowledge engineer.

Machine learning is a broad field, with several views within it as to the purpose and role of machine learning-methods within KBS development and maintenance.

For example, one purpose may be to study to what extent a computing system is able to form general concepts based on a series of examples, another to improve the performance speed of a system, and a third to gradually extend and refine a system's problem solving ability as the system is being used. Given the various purposes and motivations for incorporating machine learning methods into KBS development and maintenance, there are also a number of different views on what type of method to use in order to fulfill a certain learning purpose. The methods may be variations of the same basic idea, but they may also reflect fundamentally different views to learning in machines. Examples are different views to the utilization of existing knowledge in the learning process, to what degree generalizations should be attempted, to the importance of logical consistency of learned knowledge vs. its pragmatic usability for some target problems, etc.

Hence, we can envision a space of machine learning methods, where the *purpose* - or high level goal - of learning is one important dimension, and some characteristics of the actual *learning method* is another. A characterization of the CommonKads methodology also forms a description space, where the descriptive dimensions will have to be chosen according to the perspective and purpose of the description[1]. In this study we take a *task oriented* approach, by focusing on development tasks performed during construction and maintenance of a system developed according to the CommonKads methodology.

So, at the outset we are given two spaces. One is the space of machine learning methods, the other is the 'current' space of CommonKads descriptors - i.e. descriptors of the CommonKads methodology and models as viewed without specifically taking ML methods into account. This is illustrated in figure 1. The aim of this report is to identify possible and useful mappings between these two spaces.

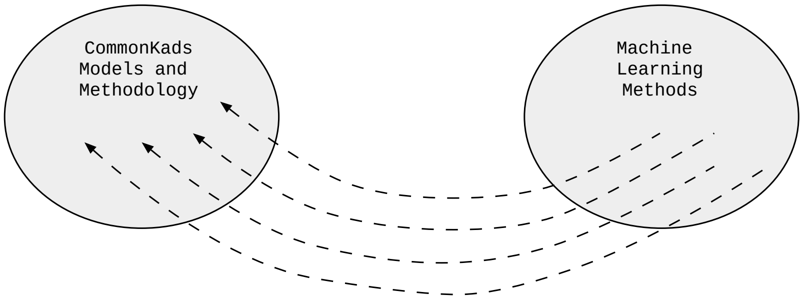

Figure 1: Finding a mapping between the spaces of CommonKads methodology and model components, and machine learning techniques

The approach which will be taken to address this problem is first to define an appropriate way to structure the two spaces, and then attempt to 'merge' parts

[1] For example, a description from a work-package/project-task/project-subtask perspective would be different from one that applies an end-product/component/subcomponent perspective

of them by identifying useful mappings. What we should end up with, eventually, is a task structure of CommonKads development tasks - described at some level of abstraction, the models associated with these tasks, and a mapping of useful machine learning methods onto this structure.

This leads to the following three-step process:

- First, the main processes and model components of CommonKads are briefly reviewed, based on the Life Cycle model[2], and the various models in the CommonKads model set[3]. The structure and the entities which come out of this will constitute a description of the CommonKads space at a particular (high) level of abstraction. This will give the basis for a top-level categorization of roles for machine learning within CommonKADS.
- Secondly, a classification of current methods and approaches within machine learning is made, partly based on classification schemes taken from the existing machine learning literature. Within this classification possible types of methods for use within CommonKads can be identified and compared.
- Finally, the possible application areas for machine learning methods are matched with characteristics of the methods available, leading to a suggestion of what role machine learning methods should have in KADS-II. Possible areas where machine learning methods could be fruitfully used must be identified, and the ways in which this may be achieved must be investigated.

1.3 Scope and structure of this report

In line with the top-down approach taken in this task, this report highlights the various roles we see for learning within the CommonKads methodology, describe the kind of analysis required to realize these goals, and compare it with the way machine learning has traditionally tried to work toward a methodology. It is not our goal to provide an all-encompassing analysis of learning within this framework and our investigations will raise a lot of questions that can not be answered within the scope of this task or project. Subsequent work on the task will fill in parts of the general scheme, in particular those related to project management and integrated uses of machine learning. This will be the subject of subsequent documents ("Management of Machine Learning Tasks within the CommonKads Life Cycle model" and "Knowledge Level Descriptions of Machine Learning Methods"). This document on the other hand lays out our approach toward the problem and sets up the framework within which subsequent work can be situated and related to the state of the art.

[2] Defined in document KADS-II/TR/T2.1/LR/0020/0.2, Framework Life-Cycle Model
[3] Defined in KADS-II/WPI-II/RR/UvA/017/2.0, Models in the KADS II Life Cycle Approach

The following two chapters correspond to the first and second step, respectively, of the three step process outlined in the previous section. A first pass on the third step is covered by chapters 4 and 5. Chapter 4 discusses the role of machine learning methods in developing and maintaining the CommonKads models, and suggests a set of mappings. Chapter 5 discusses what role machine learning methods could play within the problem solving methods, taking the approach of viewing problem solving as construction of a case-specific model. Chapter 6 sums up the results of this study.

2 Four Roles for Learning in CommonKADS

In this section we put forward four distinct roles for machine learning techniques in the CommonKads KBS methodology. We describe these in two role classes, emphasizing the distinction between process (life-cycle) and product.

- Machine Learning within the CommonKads Life Cycle
  1. Learning as a modeling operator: learning techniques can be used to push models through states.
  2. Learning about modeling: learning can be used to improve the control over the modeling process itself.
- Machine Learning within the CommonKads product
  1. Learning as part of problem solving: learning methods can be used within the problem solving process itself.
  2. Integrated, sustained learning: During normal use of a system, learning techniques can be used to acquire domain knowledge from the environment in which the application is being used.

The CommonKADS framework and methodology brings together two distinct views on modeling. The first pertains to the process of KBS development, while the second one is more fundamental and refers to the nature of problem solving. These views are related to the first and second of the above classes, respectively. The first two sections of this chapter give a motivation for the above four roles, and introduce some important issues. This serves as a background for the more elaborate discussions in chapters 4 and 5, where the roles also are related to the state of the art in machine learning methods. Section three explains the rationale for a knowledge level view to machine learning, while the last section describes ways to focus the role machine learning in CommonKads.

2.1 Machine learning within the CommonKADS life cycle

In CommonKads KBS construction is viewed as a modeling activity. The target KBS is a computational model of some desired behavior in a task reality. In a CommonKads project various other models are being constructed that highlight different aspects of this product. The current set of CommonKads models is the following: the Organizational Model, the Applications Model, the Task Model, the Agent Model, the Model of Expertise, the Communication Model, the Design Model and the System Model. These models are, of course, not independent. For example, the design model must describe a design that implements a behavior as it is predicted from the model of expertise. The entire project is thus the construction of a set of models that are in a relation of mutual consistency to one another. One role of machine learning in CommonKads is to help in the development of these models.

2.1.1 Learning as a modeling operator

The modeling process can be conceptualized as the application of a series of *modeling operators*. A modeling operator constructs part of a model by, for example, combining or refining other models. For example, it may apply a method to decompose a task in the model of expertise, or it may augment a design model by choosing a representation for a particular domain entity. Some modeling operators will be only informally defined, like translating from descriptions in books to a domain knowledge in the model of expertise. But others will be more explicitly described, like the application of a problem solving method to decompose a task. Yet others will be supported by software tools, like the selection of an appropriate design model.

In our exploration of the role of machine learning in CommonKads we view machine learning techniques as modeling operators. Within the entity-relationship definition of a model[4] this means that machine learning methods are viewed as *techniques* for the acquisition and manipulation of the model's *content*. This simple re-statement immediately sets the scene for the analysis in this paper: the whole scheme for applying modeling operators and for managing this process should be demonstrated to be applicable and to be useful for machine learning operators. Only this will ensure a smooth and fundamental integration of learning techniques in the knowledge engineer's tool set.

> *One role of machine learning is to serve as a technique for the construction of the various models of a task reality in the process of KBS development.*

The CommonKADS life-cycle approach[5] is based on the notions of *model states* and *risks*: the purpose of project work is to evolve the product model through a series of meaningful states. The purpose of project management is to identify areas of high risk and to set up the proper mechanisms to control these risks. As previously stated, this document will focus on the role of machine learning in project work.

2.1.2 Learning about modeling

The CommonKADS product model is configured on a case-by-case basis and can be more or less complex. Similarly the management of the model development pro-

[4] Defined in KADS-II/WPI-II/RR/UvA/017/2.0, Models in the KADS II Life Cycle Approach
[5] Defined in document KADS-II/TR/T2.1/LR/0020/0.2, Framework Life-Cycle Model

cess will be different from project to project. In order to support the modeling process CommonKads provides various types of support. For example, the CommonKads model set supplies the developer with a set of initial or *template* models as well as the relationships between these models. In addition, libraries of generic modeling components - for example of task decompositions, inference structures or implementation modules - provide support in filling in the templates.

Much of this methodological support is embodied in a series of software tools that together constitute the CommonKADS workbench. Some of these are instrumental in executing the modeling activity, for example for the selection and application of modeling operators that operate over the various models. An important role for machine learning is therefore the accumulation of experience within these tools. One long-term goal of the CommonKADS workbench could be to learn about supporting the knowledge engineer in his activity.

> *A role of machine learning is to gradually enhance control over the modeling process by accumulation of experience in the CommonKADS workbench tools.*

One could experiment with learning apprentice systems [Mitchell, 1983], learning from experience [Van de Velde, 1988], or even autonomous exploration of modeling possibilities. One could also incorporate case-based reasoning methods [Slade, 1991, Aamodt, 1990], enabling re-use of previous, concrete modeling experience. An example of the latter approach, is a case-based assistant for system development made by Arthur Andersen [Foley and Orttung, 1991].

Instead of making demonstrator systems of automatic learning methods as such, we would advocate here a strong user-oriented perspective. Careful consideration of the way in which knowledge engineers would use the CommonKADS workbench must lead to the identification of the support needed, the identification of opportunities for learning, and to evolutionary design of a useful set of tool features.

2.2 Machine learning within the CommonKADS product

The previous section has outlined the role of machine learning in the model construction process. A second role is in the incorporation of learning capabilities in the product itself.

2.2.1 Learning as part of problem solving

Apart from the CommonKADS view that KBS development is a modeling activity, there is another view on modeling which reaches into the the heart of the expert system notion. It is the view that problem solving itself is a process of constructing a case-specific model of a system. This is different from other views in which problem solving is viewed for instance as logical inference [Genesereth and Nilsson, 1987], search in a problem space [Newell, 1990] or the execution of a number of activities [McDermott, 1988]. In CommonKads, the solution of a problem is not simply a value of a property or a message on a screen. Rather it is the result of a process of constructing a certain type of model of a system. This model is justified by the general knowledge about it and the case specific information which was obtained about it.[6] For example, a medical doctor may prescribe a certain drug because he has, during a diagnostic session, constructed a certain model of the patient[7]. This model may be based on a detailed understanding of the processes involved or just on experiential knowledge or highly approximate and qualitative laws. This model captures ones understanding of the situation [Clancey, 1989].

This view on modeling leads to a different role for learning:

> *Machine learning techniques can be used to construct part of the case specific model of a task instance.*

This view has a number of consequences which will be further explored in this task. In particular the traditionally strong distinction between problem solving and learning is weakened or disappears. Of course we do not mean to say that learning and problem solving have the same role to play in intelligent activity. What we mean is that these two activities can be described using the same machinery. A whole research scheme appears for analyzing learning from the different perspectives that CommonKADS proposes for analyzing problem solving (organization, task, agent, knowledge, design,...). Such an analyses of machine learning methods will be the main contribution of our work on *knowledge level modeling* of ML methods, as described in a subsequent document.

2.2.2 Integrated, sustained learning

"What we want is a machine that can learn from experience". Alan Turing's (1947) wish has still not been satisfied. Nevertheless, knowledge systems that learn can have several advantages. For example, they can improve performance over time,

[6]The model oriented view of problem solving should not be equated with model-based reasoning. In model based reasoning the idea is that problem solving can achieved by reasoning on a deep model of the system. Such a model is a model in terms of basic laws, first principles or models of physical processes underlying the behavior of a system. Here we mean a much wider class of models, namely any qualitative model.

[7]In fact, prescribing a drug is part of the process of constructing (confirming) the model, depending on the patient's reaction to the drug. Therefore the result of problem solving is actually a side-effect of the model construction process.

learn from a knowledgeable user, learn to adapt their behavior to the type of user they have before them, learn to provide adequate explanations, learn to approach a problem in different ways, and so forth. Basically they provide a chance for building systems that are more flexible, faster, more knowledgeable, or better adapted to the environment they are working in. In terms of life cycle one can think of parts of the development being delegated to the maintenance of the knowledge system.

The problems of actively maintaining a KBS have until now received surprisingly little attention in AI, although there is a strong awareness of this problem - as illustrated, e.g., by the maintenance difficulties experienced by the XCON group [McDermott, 1986]. Improving manual maintenance methods is one of the motivations behind a structured modeling approach to knowledge acquisition such as CommonKADS. Sustained learning is especially important in domains where the scope and-or life span of situation specific knowledge is small. In these cases it is hard or impossible to freeze all relevant knowledge in a system at design time. Sustained learning is also important whenever adequate problem solving requires knowledge which can only be obtained "in the field", like knowledge about frequency of problem occurrences or knowledge dependent on the problem history (non-Markov problem-solution sequences). Also, fine tuning of knowledge and parameters to the particular environment that the system finds itself into may improve the adequacy of problem solving behavior.

It is even not hard to argue that a system that does not learn is only working hard to make itself obsolete. In fact, once a knowledge system is being used it is clear that there will be processes of learning going on. First, the users will learn to adapt their working habits to maximally profit from the systems competence. Often they will not only learn to use the system but, moreover, they will learn about their task as well. If the system has sufficiently "knowledge level" explanation capabilities then its behavior will become more and more predictable since the user should be able to understand why the system did what it did.[8] This has, however, a dramatic implication, namely that the knowledge system - by the mere fact of being used - is working hard to make itself less and less useful as an intelligent assistant. Thus, sustained learning is often a condition for the long-lived success of a knowledge system.

> *A role of machine learning is to allow for systems that learn from experience through practice.*

A goal of machine learning research is to enable self-learning in computer systems. This goal is, in general, approached by developing theories and methods for processing and retaining observations (facts that are input to the system) in a way that

[8]This is only the case provided the system is sufficiently intelligent, i.e. approximates the behavior implied by the knowledge level explanations it provides

enables them to be used in future problem solving. A fundamental principle for sustained learning is that each problem solving experience is a powerful source of learning. A primary objective of the learning is to avoid mistakes that were made before. A second objective is to retain experience in a form that enable the solving of new problems, i.e. problems that are different from the ones experienced so far. Sustained learning methods integrate problem solving and learning by adding a learning step to each problem solving session. As further described in chapter 3, the development trend for these methods is towards knowledge-intensive learning, where existing knowledge is actively used within the learning method itself. Such methods may learn by storing and re-using case-specific knowledge (case-based reasoning), or by modifying generalized knowledge (explanation-based learning).

Sustained learning in CommonKADS requires to be able to model learning at the same level as problem solving. In terms of the entity-relationship diagrams of the CommonKADS models, modeling techniques must be described so that their input and output become part of the contents of the models. But in addition the CommonKADS modeling framework must be extended: it must be able to represent persistent side-effects, such as those generated by sustained learning. With sustained learning, the CommonKADS product model continuously changes as the system is being used. At least, the system itself will change during use, and to what degree - and how frequently - these changes should be reflected in the models of the model set, is a project management issue.

2.3 A knowledge level view on machine learning

In the previous section we have formulated and motivated four distinct roles for learning within the CommonKADS framework and methodology. Our approach to studying the various roles is top-down. We start from the general framework of CommonKads as it is being developed for problem solving, and apply it to learning. The advantage of this approach is that it sets the scene for all further work: formulation of learning as modeling operators, formulation of modeling guidelines[9] that use these operators, using the operators to develop and maintain CommonKads models, and so forth. All of the experience gained in applying this machinery to problem solving can be put to use on the problem of learning. The CommonKads framework is "the" framework for integration.

Traditionally, the roles and effects of machine learning methods, e.g. their descriptions in terms of what input they take and what output they produce, has almost exclusively been viewed at the symbol level. We will - as indicated above - argue that the knowledge level is the right level for a possible integration of machine learn-

[9] Modeling guidelines are pointers from features of the task reality to modeling operators, and will be elaborated in Task I.5 documents.

ing methods into CommonKads.[10] The motivation for this choice, from a machine learning perspective, is as follows:

- The knowledge level is the level at which the knowledge engineer specifies expertise during knowledge acquisition. This facilitates interaction between the systems and the knowledge engineer, which supports mixed initiative scenarios. This is important since the CommonKads methodology is decidedly user-oriented.
- Because knowledge level languages are based on theories of problem solving and because machine learning is strongly related to problem solving, the knowledge level is a good candidate to obtain a uniform descriptive framework. This also enables advances related to problem solving, such as new models or modeling support tools, to be related to learning issues.
- From a more pragmatic standpoint, knowledge level characterization of learning can be expected to solve much of the same problems that the approach is currently dealing with in problem solving: selection of problem solving methods for tasks, combination of methods, robustness and quality of explanation, and so forth.

Though it is hard to deny the success of the knowledge level approach as it is now being applied to knowledge based systems, there have been some critiques with respect to whether the knowledge level notion can be usefully applied to learning. This goes back to an early attempt by Dietterich [Dietterich, 1986] to analyze learning at the knowledge level. His conclusion was that (1) important forms of learning, such as explanation based generalization, could not be described at the knowledge level, and (2) that an encompassing knowledge level analysis of learning systems is far beyond the state of the art and may even be impossible. In our opinion Dietterich's paper was somewhat misconceived because it used a very narrow view of learning - the acquisition of new knowledge - and studied the effect of learning on a performing agent, rather than the learning process itself. It is now clear that learning can have other purposes than acquiring new knowledge, with an equally observable effect on behavior (e.g., operationalization). Also, learning methods use various forms of biases that determine the various quality criteria of what is learned. Finally, the "why" of learning itself is subject to analyses at the knowledge level.

We think it is worthwhile to take up this line of work again for the following reasons. First, the notion of knowledge level itself has been evolving over the last years. Knowledge level models are now being stated in terms like problem solving "method", "task decomposition" or "domain model" [Wielinga et al., 1991,

[10] One may also expect interesting results from modeling the organizational aspects of machine learning but we do not pursue that idea here.

Steels, 1990]. This allows for a more fine-grained epistemological analyses in terms of knowledge categories and the roles played by each of them, and constitutes no significant departure from the original knowledge ideas. Moreover, descriptions in these terms are much easier to understand and manipulate than the "flat" knowledge level description that Newell proposed. A second reason why we think the knowledge level approach is now feasible is that considerable experience has been gained through the application of the idea to problem solving.Finally, we think that the knowledge level approach to machine learning is promising because it may solve a number of problems analogous to those that are addressed in advanced problem solving research:

- They make explicit the knowledge which is implicit in learning methods
- They allow for selection of learning method based on task features
- They allow for the rational reconstruction of learning methods
- They allow for the modification and combination of learning methods
- They allow for the comparison of different methods
- They allow for the construction of new methods from re-usable generic components.

2.4 Focusing the role of machine learning in CommonKADS

Although we emphasize a unified view of machine learning, problem solving, and modeling, we may want - initially at least - to focus the actual application of machine learning methods within the CommonKads methodology. The starting points for focusing should be at a level which encompasses the CommonKads development *process* as well as the major *models* being developed and maintained within this process. This is achieved by starting with the CommonKads model set, as described in section 2.1. Overlaying these models, however, are the more general views on modeling within CommonKads, which gives some overall focusing guidelines, independent of the specific models in the model set.

The following is a set of suggested constraints for focusing the discussion of the role of machine learning methods within CommonKads:

- Only learning with respect to the following three modeling activities are of interest:
  - Model construction, i.e. use of machine learning to help generate partial conceptual models, or other CommonKads models.

- Model instantiation, i.e. use of machine learning to help generate the case model during problem solving.
- Model maintenance, i.e. use of machine learning to help in continuously maintaining and adapting the knowledge during normal use of the system.

Note that the three roles of machine learning are related to the modeling activities which take place *before* problem solving takes place, *during* problem solving, and *after* a problem has been solved, respectively.

- Only learning within particular phases of these three modeling activities may be of interest. This focus represents a further refinement of the three modeling activities listed in the previous point. This refinement should be determined by the set of useful machine learning methods available.
- Only learning with respect to a subset of the CommonKads models may be relevant. A minimum set of models should include: The Model of Expertise (learning methods in developing and maintaining a conceptual model), and the Communication/Agent models (learning as a means of gradually transferring more tasks from a human agent to the computer system).

To sum up, this chapter has described four roles for machine learning methods in CommonKads, grouped within two top level classes:

1. Learning with respect to modeling.
   - Learning about modeling.
   - Learning as a type of modeling operator.
2. Learning within the KBS product.
   - Learning as problem solving
   - Learning as continuous knowledge maintenance

These classes may be further specialized for the various CommonKads models. An example is given in chapter 4, where a specialization of learning roles for the Model of Expertise is discussed, relating machine learning methods to the various model development phases:

- Learning applied to elicitation and initial modeling
- Learning applied to model development
- Learning applied to model validation and modification

- Learning applied to model maintenance during runtime

In the next chapter, characteristics of current machine learning methods is described. The two subsequent chapters (4 and 5) then discuss how these methods can be used within the two top classes of learning roles listed above.

3 Machine Learning Methods

This chapter provides an overview of current machine learning methods by describing them within a classification scheme adapted to our purpose. Several ways of classifying machine learning methods have been presented in the literature but no classification scheme has been generally agreed upon. All schemes are somewhat biased by the author's view of the role of machine learning and of what type of methods he/she favors. Our perspective within KADS-II also puts a bias as to how the machine learning space should be dimensionalized, of course. In order to have some confidence that our classification scheme has both a solid basis within existing research, as well as the ability of being useful for our purpose, we first give a brief presentation of three classification schemes. On this basis we define a machine learning classification scheme for CommonKads.

3.1 Three classification schemes for machine learning methods

At a high level, it is generally agreed that machine learning methods may be classified into two broad classes: symbolic and non-symbolic ones. The class of symbolic learning methods contains methods that learn explicit, symbolic concept descriptions, where a concept is described by a set of explicitly identified features. The goal of these methods is to learn useful concept descriptions for improving system performance[11], represented in a way that also make the learned knowledge inspectable by humans. This is the major class of learning techniques from a knowledge-based systems point of view. It covers data-intensive, empirical concept formation from examples, knowledge-intensive methods of concept formation, knowledge operationalization, knowledge compilation, as well as case-based and analogy-based methods. Learning methods in the second class, the non-symbolic ones, aim at learning of improved performance within more indirect, implicit or distributed representation forms. These methods are often refereed to as non-symbolic - or sub-symbolic - methods, since they do not necessarily represent concepts as distinct knowledge units inspectable by humans. Parameter tuning, neural network learning and genetic algorithms are grouped within this class.

Symbol learning methods constitute the primary class of methods addressed by this study. However, the other method class is also included in this review, for two reasons: First, to serve in a comparison with the symbolic methods - thereby increasing the understanding of their potentials and limitations. Second, to indicate possible method components for combined symbolic and non-symbolic learning techniques.

[11] Performance in a wide sense, including the quality of the result of a problem solving process as well as the speed of the process

The distinction between the two classes of methods is not as clear-cut as what may be implied from the distinctions made above. Although most of the methods can be classified as member of one of the classes, there are also methods which it is hard to classify within only one class (e.g. connectionist methods where the network nodes are - or gradually become - explicit concepts). Further, there is a significant - and increasing - amount of work addressing the integration of method components from the two main classes.

The three classification schemes briefly reviewed are those suggested by Jaime Carbonell [Carbonell, 1989], Michalski and Kodratoff [Michalski and Kodratoff, 1990], and the one developed in the ACKnowledge project [Van Someren et al., 1990].

3.1.1 Carbonell's scheme

Carbonell's scheme contains four "paradigms", each with a set of "dimensions" representing issues of focus - or approaches - of methods within a paradigm.

A strength of this scheme is that it covers all the major types of learning. A weakness is that it is very general: It is a two-level scheme where all learning types are described within the four classes (paradigms). Two of the classes cover symbol learning methods - i.e. learning of explicitly represented symbolic concepts, and two covering non-symbolic/sub-symbolic methods. Within each of the classes of symbolic learning techniques, a set of feature types for more fine-grained discrimination is suggested.

I. The Inductive Paradigm

This paradigm covers empirical learning from many examples (instances), using little or no domain knowledge. Aims at building classes which correctly classifies positive examples without including negative ones. This is the paradigm for 'classical' machine learning methods such as the Version space method [Mitchell, 1982], AQ [Michalski et al., 1983], ID3 [Quinlan, 1986], etc. Description types (i.e. sub-dimensions for discriminating between methods) within this paradigm are:

- Concept type.
  Discriminant concepts vs. characteristic concepts. Learning of discriminant concepts strive for a concept description which is able to accurately determine whether an example is an instance of the concept or not. An example may not be an instance of more than one concepts. A characteristic concept description addresses learning of concepts which may not have a classical definition in terms of a set of necessary and sufficient features. Prototypical and probabilistic definitions belong to this sub-class.

- Description language.

The language for expressing input instances and output definitions of concepts. Attribute-Value lists, propositions, first-order expressions, object-descriptions. Most methods represent inputs and outputs in the same language.

- Noise and uncertainty handling.
  Learning in real world environment often involves noise and uncertainties in input descriptions. Earlier methods disregarded this problem, while more recent methods attempt to take them seriously.
- Source of instances. Most methods assume that the learning instances are supplied by a teacher, while some methods address learning based on input from other sources.
- Incremental versus one-shot induction. One of the fundamental properties of a learning method is whether it learns by comparing a lot of instances at the same time, or whether it is able to incrementally modify an existing concept definition when new examples are presented.

II. The Analytic Paradigm

Learning by analyzing one or a few examples by use of domain knowledge. This knowledge may constitute a strong theory - assuming a closed domain - as in by classical explanation-based methods[12] (e.g. EBL [Mitchell et al., 1986], EBG [DeJong and Mooney, 1986]), or a weaker theory - addressing open real-world domains - as assumed by extended explanation-based methods (e.g. KI [Murray and Porter, 1988], CONCLAVE [Van de Velde, 1988]), and analogical and case-based[13] methods (e.g. CASEY [Koton, 1989], PROTOS [Porter et al., 1990], SWALE [Schank and Leake, 1989], CREEK [Aamodt, 1991]). Discriminating descriptor types of this paradigm are:

- Degree of generalization.
  Some methods are 'eager' generalizers, while other take a more conservative approach. At the lower end of the generalization spectrum are the case-based methods, where the basic idea is to store parts of concrete problem solving episodes, not generalizations.
- Representation of instances.
  A learning instance for an analytic learning method may be just a problem description and a solution, but it may also contain a trace of how the solution was arrived at, and even justification (explanations) of steps in this path.

[12] An overview is given in [DeJong, 1988]

[13] Carbonell often uses the term *analogical learning methods* to cover both analogical and case-based methods. A distinction often made elsewhere (e.g. [Burstein, 1989]) is that case-based methods make use of previous problem solving episodes within the same domain as the current problem domain (intra-domain mapping), while analogical methods get their previous episodes from another domain (inter-domain mapping).

- Learning from success or failure.
  Classical EBL learns from success, while other explanation-based methods and case-based methods also learns from failure.

- Open vs. closed loop learning.
  Open loop learning is a one-pass learning process, while a closed loop implies iterative learning in order to improve system competence or performance based on feedback from its behavior (possibly through a user).

III. The Genetic Paradigm

Genetic algorithms [Holland, 1987] takes empirical learning to its extreme by applying a 'genetic evolution' approach to learning and thus requiring a large amount of training instances. These algorithms learn by generating 'mutants', recombining these, and test them for 'fitness' against a selection criterion. The most fitted will survive and be re-used in subsequent mutation-recombination cycles. Genetic algorithms are often associated with 'classifier systems' which learn classification rules as bit vectors. Genetic algorithms are a special class of *selectionist* systems [Edelman, 1987] which take their inspiration from biology and evolution theories.

IV. The Connectionist Paradigm

These methods learn by adjusting weights within a network of nodes and links [Rumelhart et al., 1987]. The network structure is fixed for each learning task. This approach also relies on a large amount of input data. Connectionist approaches are non-symbolic or sub-symbolic forms of learning. Connectionism is sometimes used synonymously to 'neural networks', sometimes as a superclass of neural networks, where the latter is reserved for purely non-symbolic, distributed approaches to connectionism. Connectionist systems are a special class of *dynamic systems* in which large sets of interconnected entities evolve through an internal dynamics to an equilibrium state.

In summary, Carbonell's scheme emphasizes a distinction between learning in data-rich vs. knowledge-rich environments. Learning from many examples using little knowledge is represented by inductive concept learning, connectionist and genetic learning, while knowledge-intensive methods are represented by analytic learning methods. In [Carbonell, 1989] the scheme is used in an attempt to map characterizations of learning tasks to learning methods. We will come back to this in a later chapter.

3.1.2 Michalski and Kodratoff's scheme

While Carbonell's four groups are chosen from a *learning paradigm* point of view, the Michalski/Kodratoff scheme basically takes what may be called an *inference method* approach. Actually, the two top level classes of this scheme discriminates according to a kind of learning purpose, i.e. whether the learning aims at *synthesizing* new concepts, or *analyzing* existing knowledge for operationalization or specialization purposes. The synthetic purpose maps completely to the class of *inductive* methods, while the analytic purpose maps to the class of *deductive* methods. These two categories also map rather closely to Carbonell's inductive and analytic categories, as far as symbol learning methods are concerned. The major difference is that the two categories in the Michalski/Kodratoff scheme is intended to cover all learning methods, also non-symbolic ones. This is a strength in a sense that is provides a more general scheme, and a weakness since it fails to make a distinction - at a sufficiently high level - between these two types of learning.

The Michalski/Kodratoff scheme can be viewed as a four-level scheme. The first level is described above. Below this, is a level of classification according to *input type*. Descending from the level of inference type (the upper level) is also a classification according to the *role of prior knowledge*. The whole scheme is listed below:

I. Synthetic (inductive) methods

- Sub-grouping according to type of input
  - Learning from examples
  - Learning from observation
- Sub-grouping according to role of prior knowledge
  - Empirical
    - Empirical generalization
    - Qualitative discovery
    - Conceptual clustering
    - Neural nets
    - Genetic algorithms
  - Constructive Induction
    - Abduction
    - Constructive generalization
  - Multistrategy
    - Integrated empirical and explanation-based learning
    - Multistrategy constructive learning

II. Analytic (deductive) methods

- Sub-grouping according to type of input

  - Example-guided
  - Specification-guided

- Sub-grouping according to role of prior knowledge
  - Axiomatic
    - Classical explanation-based learning
    - Axiomatic program synthesis
  - Constructive deduction
    - Abstraction
    - Deductive generalization
  - Multistrategy
    - Integrated empirical and explanation-based learning
    - Multistrategy constructive learning

III. Analogical (combined inductive and deductive) methods

- Multistrategy
  - Integrated empirical and explanation-based learning
  - Multistrategy constructive learning

As seen, analogical methods are viewed as combinations of inductive and deductive inference methods. It should be noted that the interpretation of deduction in this scheme goes beyond purely logical deduction (which in the scheme is called axiomatic deduction), in that it also incorporates plausible inference steps. The terms 'constructive' and 'explanation-based' indicates knowledge-intensive methods.

A peculiarity of this scheme is that case-based learning is viewed as a type of inductive learning, not as analogy learning. This is motivated by the fact that although case-based methods store knowledge as units of specialized - non-induced - knowledge, it performs a generalization step when matching a problem with a previous case during problem solving. However, this view stretches the definition of induction very far, to say the least.

In summary, this scheme emphasizes a classification of machine learning methods according to the underlying inference method, i.e. inductive, deductive or a combination. Inductive methods map to synthetic learning tasks, deductive methods map to analytic learning tasks, while combined methods map to combined synthetic and analytic tasks. A major difference of this scheme with respect to the previous is that it views inductive learning independent from knowledge-poor methods. Hence, the scheme also cover knowledge-intensive inductive methods, that use an underlying domain model to guide and justify inductive steps (which in a sense unifies induction and abduction).

3.1.3 The ACKnowledge scheme

The final scheme discussed is the one developed for assessing machine learning methods within the ACKnowledge project ([Van Someren et al., 1990]). The purpose of this scheme was to provide classes for description of machine learning methods in order to integrate them into the ACKnowledge workbench (the KEW). The task of defining a classification scheme in ACKnowledge was different from the one in KADS-II, since the role of machine learning methods was already specified for ACKnowledge. machine learning methods should serve as a type of *tool within in the KEW*, for giving the expert/knowledge engineer assistance in knowledge Elicitation and modeling.

This classification scheme therefore focuses on criteria for determining what the functional properties of a method is - in terms of input/output descriptions - and to what degree a machine learning program can be integrated - at the symbol level - to other tools. The following five main categories are used to discriminate relevant properties of machine learning methods[14] within ACKnowledge:

- Textual description of the method.
  This is a mere informal description of the learning method, with no structured or specified vocabulary.
- Function of the method
  This dimension captures what type of learning effect the method has, and is used to select learning methods on the basis of the intended purpose of the learning task in the development process. Six different functional categories are specified: Elicitation, Modeling, Evaluation, Refinement, Restructuring, Translation. The first five functions relates to improving certain aspects of a knowledge model, using a single representation language, while the Translation function is a translation between representation languages.
- Representation language.
  Describes the representation language of a learning method for the purpose of interfacing learning tools with each other - and with other tools. It distinguishes between the *knowledge type* represented (i.e. control knowledge, object knowledge, deeper models, heuristics, uncertainty, etc.), the *syntax* of the language, its semantics (related to its interpreters), and its pragmatics (related to its use).
- Effect on the quality of knowledge.
  This dimension categorizes the learning in terms of what it 'does' to the knowledge. Descriptor types are in terms of improvements to the operationality of

[14]There are also a set of secondary descriptors not related to learning *per se*, for example additional functions such as information management, visualization, code generation,

the knowledge, its explanatory power, modularity, structural simplification, user inspectability.

- Communication properties.
  Characterizes the degree and type of involvement by the user in the learning process, and describes the type and form of interaction (knowledge entry, visualization, knowledge manipulation).

This scheme differs from the others in that it emphasizes the usefulness of a learning method for a certain purpose or function. It is not a scheme for classification of learning methods with respect to what type of learning approach or paradigm they subscribe to. From a CommonKads point of view, its strength is that it addresses the purpose of learning methods within a domain similar to ours. A weakness is that most of its dimensions apply to symbol level criteria.

Potentially useful perspectives from our points of view - and after they have been given a knowledge level interpretation - are the dimensions of *learning functionality*, *knowledge types learned*, and *user interaction*.

3.2 A classification scheme for machine learning methods within CommonKADS

The three schemes just reviewed basically give us a classification according to learning method (data-intensive vs. knowledge-intensive symbol learning methods, neural networks, genetic learning), a combination of learning type and inference method (synthetic and inductive learning vis. analytic and deductive learning), and a combination of learning functionality and knowledge types learned, respectively. Together they point to important relationships between *what is to be learned*, *what knowledge to start from*, and *the learning method* to be used in each of these learning situations.

Our description space for organizing machine learning methods has the following three dimensions:

- Purpose of the learning
- Knowledge intensiveness
- User cooperation

These dimensions will be used to evaluate methods with respect to their usefulness in CommonKads by relating them to specific learning tasks within the CommonKADS space of project and product activities.

I. Purpose of the learning

Describes what type of learning task a learning method attempts to solve. A learning task is, for example, to improve performance speed, to form new concepts from examples, to learn improved problem solving behaviour from experience, to increase the breadth (the cover) or the depth (the detail level) of a knowledge base, or a combination of such goals.

The learning purpose is described in terms of the *learning goal* (exemplified above), the *input* to the learning process, and the resulting *output*. The input description characterizes the type of input (e.g. single examples, multiple examples, no examples, general relationships), and the descriptors of the input (e.g. what type of information describes an example). The output - i.e. what the learning method is able to learn - is elaborated in the paragraphs below.

The output of a learning method determines the type of knowledge which is learned by the method. Some sub-categories are:

- General concept descriptions
- Specific experiences (cases)
- Axiomatic knowledge (assuming 'strong theory' domains)
- Open-domain knowledge (assuming 'weak theory' domains)
- Problem solving heuristics
- Control knowledge
- Explanations[15]
- Problem solving performance

Currently machine learning techniques are defined in terms of single, uniform problem solving tasks (usually classification) and evaluated in terms of only a few criteria (usually classification accuracy). Differences between performance tasks can be used to select appropriate machine learning techniques, to configure machine learning techniques if a complex knowledge base must be learned and to decide how the available knowledge and data are to be used. More generally goals of learning will be described by use of the categories of the 'CommonKads space' as indicated in chapter 1. Here a lot of work is to be done.

Different knowledge types are further characterized by different *quality criteria*. Quality criteria can be viewed as a generalization of the operationality notion

[15] It should be noted that here explanations is the *learning target*. It should therefore not be confused with explanation-based learning, where explanations are used as part of the *learning method*.

[Mitchell et al., 1986]. They express the pragmatic aspects of knowledge use: level of abstraction, correctness, effectiveness, noise-sensitivity, robustness, etc. These quality criteria allow one to distinguish between different methods for, for example, concept learning that learn qualitatively different descriptions and concept types[16]. They capture at an abstract level, a type of *bias* [Utgoff, 1984] implicit in a learning method i.e., its preference for one kind of generalization or another. The appropriate bias must be related to characteristics of the environment in which the problem solving activity, to which a learning goal relates, occurs: its criticality, the diversity of problems that occur, the degree of inspectability, etc [Van de Velde, 1988]. Here the full CommonKADS task feature work[17] can be used.

II. Knowledge intensiveness

This dimension describes a learning method's ability to utilize an existing body of knowledge in its learning. This criteria has become increasingly important due to the swing towards learning methods that utilize the existing knowledge base. All methods can be said to utilize knowledge of the learning domain to some degree, but in some methods this knowledge is an implicit part of the learning method itself. The "knowledge intensiveness" dimension discriminates between these type of methods and methods that take advantage of the currently existing, explicitly represented, domain model. The latter methods will improve their learning effect over time, due to improvements in the knowledge model. Distinct sub-categories of this dimension are:

- Not utilizing (explicit) domain knowledge
- Utilizing some knowledge types
- Utilizing the entire knowledge model

In the latter sub-category, a learning method has the same knowledge at its disposal as a regular problem solving method.

These sub-categories map relatively well to the distinction between classical inductive methods (ID3, AQ, etc.), some explanation-based methods (EBL [Mitchell et al., 1986], EBG [DeJong and Mooney, 1986], CONCLAVE [Van de Velde, 1988]), and other explanation-based and case-based methods (e.g. KI [Murray and Porter, 1988], CHEF [Hammond, 1987], CASEY[Koton, 1989], CREEK [Aamodt, 1991]), respectively. In addition to these categories it should be observed that different methods make different assumptions about the knowledge they use. Refined distinctions of knowledge intensiveness thus take into account

[16]This issue is somewhat elaborated in chapter 5, in relation to learning of 'open concepts' by integrated learning from experience (sustained learning)

[17]Described in KADS-II/T1.5/WP/VUB/002/1.0 "Analysis of Task Features"

the different qualities of the knowledge, like correctness, operationality, level of abstraction, effectiveness, and so forth [Van de Velde, 1988]. Note that these criteria are also used to characterize the purpose of learning, but here they apply to the knowledge being used for learning itself.

The knowledge intensiveness dimension will help in identifying possible machine learning methods for the different phases of knowledge model development. Advantages of knowledge-poor methods are that they can be used early in the model building process, since they do not assume an existing knowledge model. Advantages of knowledge-richer methods are their ability to learn within an existing knowledge model, and, for some of them, also to improve their learning ability (as well as their problem solving ability) over time.

III. User cooperativity

This dimension describes the ability of a learning method to pull the user into the learning process, e.g. by observing a user's behavior or by actively interacting with him in order to get some missing information, a confirmation or rejection of a weakly supported hypotheses, etc. Within a general knowledge acquisition and modeling environment, the strength of a machine learning method is not assessed by its degree of autonomous learning, but by its contribution to the total development and maintenance process - including the ability to get and utilize help from the user.

Interaction between system and user in this sense assumes that the user has some knowledge about the application domain, and that he is able to - and allowed to - contribute this knowledge to the system. The user may be a competent end user - as, e.g., when the system is used as a decision support tool - or a domain expert training a system which will later be put into use (i.e. the expert takes on the role of an "oracle" as defined in [Valiant, 1984]).

Sub-categories are:

- No user interaction
- Utilizing information provided by the user
- Actively interrogating the user

This dimension will help in identifying possible machine learning methods for integration into interactive knowledge acquisition or problem solving methods. The different types of "oracle" lead to refined distinctions on this dimension. For example, certain methods expect only yes or no answers from the users, others also require an explanation.

In this subchapter, three main dimensions has been defined for describing machine learning methods with respect to their relevance and usability within CommonKads.

These dimensions span the space of machine learning methods by defining three axis along which relevant properties of the methods can be compared and related to the methods' potential use. Data points in this space will have various degrees of granularity, ranging from full-fledged machine learning programs, via complete methods, to sub-methods. The structuring of method properties as described here, is an initial suggestion. It will need to be refined and extended in the course of being applied. In the next two chapters, the role of machine learning methods will be discussed with respect to the process of developing CommonKads models, and with respect to problem solving and knowledge maintenance within the KBS product, respectively. A more extensive analysis of machine learning methods and method components - including applications and refinements of the descriptive framework - will be given in a later document ("Knowledge Level Description of Learning Methods").

4 Machine Learning within the CommonKads Life Cycle

In this section we address the question of how to integrate machine learning within the CommonKADS Life Cycle Approach.

Models are central in CommonKADS. The product of applying the CommonKADS methodology is a set of interconnected models that describe a desired problem solving organization and behavior from different perspectives. In the 'model document' - "Models in the KADSII Life Cycle Approach" (KADS-II/WPI-II/RR/UvA/017/2.0) - models are being described in terms of *content* and *techniques* (the *project work* aspects), of different *states*, state *dependencies* and *quality criteria* (the *project control* aspects) and finally, a meta-level description. In a CommonKADS project models a selection of these models are pushed through a sequence of states until they satisfy desired criteria (see also "Framework Life-Cycle Model" KADS-II/TR/T2.1/LR/0020/0.2).

There are at least two ways to integrate machine learning into this life cycle approach. The first one is to integrate it as a tool for project work, the second one as a tool for project control.

- Integrating machine learning in project work is done by viewing machine learning as a modeling technique associated with some of the models in the CommonKADS model set. This is the view of machine learning for moving models from state to state. The first model that comes to mind in the context of machine learning is of course the Model of expertise, but in principle learning can be related to any model.
- Integration of machine learning in project control can be done by accumulation of experience in the CommonKADS project control tools. The goal of project control is the selection of models, model forms and techniques so as to move the product model as quickly as possible into its target state. Various forms of learning, in particular learning from experience can be envisaged to be useful here.

The model document[18] describes some of these models as well as the model construction process. This consists of basically two conceptual phases which we will refer to as *configuration* and *instantiation*. Configuration is the selection of models from the CommonKADS model set, the definition of their form and of their interrelationships. This gives the skeletal product model, a structure to be filled in as the project progresses i.e., during the instantiation phase. It is in this phase that the techniques associated with the different types of models are used.

[18](KADS-II/WPI-II/RR/UvA/017/2.0)

The logical way to proceed from here would be to go over the different models and to identify for each of them the machine learning techniques that can be used to construct them. Instead, we picture the filling in as follows. First a model is selected to become part of the product model. For instance, one chooses to include one or several task models. Next one chooses a form for the model. This form is dictated partly by its definition, partly by the particular formalism one uses to express it in. For example, the conceptual model may be expressed in natural language, a task feature language, a conceptual modeling language, or the CommonKADS formal modeling language. Once the language has been chosen, a technique can be selected to fill in that particular form.

```
              Configuration                 Instantiation

WORK:   select model  -->  select form     fill in content
                              \               /
                               \             /
CONTROL:                 eval state -> select technique -> eval state
```

Note that a model may contain references to other models in the CommonKADS model set. So the application of a technique also covers the management of the interdependencies between the models.

4.1 Learning and acquisition of knowledge level models

Currently machine learning techniques are defined in terms of single, uniform problem solving tasks (usually classification) and evaluated in terms of only a few criteria (usually classification accuracy). Differences between performance tasks can be used to select appropriate machine learning techniques, to configure machine learning techniques if a complex knowledge base must be learned and to decide how the available knowledge and data are to be used. Because this requires explicit representations of the knowledge level model, this raises the additional problem of learning the knowledge level model.

These issues are all related to acquiring or refining an actual knowledge-based system and its knowledge level model. We do not think that fully automated learning of one from the other is possible. Rather, knowledge level and symbol level models form a pair that needs to be kept in a consistent rationalization/operationalization relation to each other, as illustrated in figure 2.

In the process of learning about this “knowledge level/symbol level pair”, information from one level can be used as a *bias* in learning about the other level. More specifically, the following issues should be addressed:

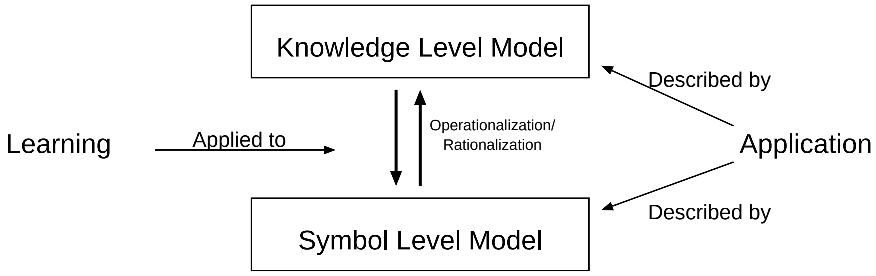

Figure 2: Learning applied to transformations between knowledge level and symbol level, and vice versa; named operationalization and rationalization, respectively.

- Functional validation of a knowledge base on the basis of a knowledge level model and subsequent refinement. The validation serves as a starting point to refine both the knowledge base and the knowledge level model (considering the validation findings), using machine learning techniques such as explanation-based learning, inductive and abductive learning. Depending on the model (for example, "deep" or "shallow"), the machine learning techniques used to revise the knowledge will be different.
- Using knowledge level models as bias in learning and knowledge acquisition. In KADS-II several knowledge level models for specific types of problem solving will be produced. It should be investigated to what degree these models can be used to structure and direct the learning process. For example, if associations will be used in a hypothesis driven heuristic classification scheme a learning method should be able to exploit this e.g., by focusing on learning good *discriminative* properties as opposed to rules that only describe *characterize* in terms of symptoms.
- Learning of and from explanations. Experts and end–users require a system to give adequate explanations of its problem solving behavior. Therefore explanations are a major goal in knowledge engineering. Swartout and his colleagues (e.g. [Neches et al., 1985, Swartout, 1983]) take explanations (and requirements for explanations) as the prime input to the knowledge acquisition process. Porter and Murray [Murray and Porter, 1988] view a system's attempt to produce a sufficiently strong explanation for a new piece of information as the basic method for learning by knowledge integration.[19]
- Learning and refining control and strategic knowledge. In the rationalize/operationalize relation, strategic knowledge at the knowledge level is related to control decisions at the symbol level. For example control decisions can be rationalized in terms of partial orders for describing different preference criteria for inference and action. In CommonKads we will need tech-

[19] Although learning of explanations is a different issue from learning from - or by use of - explanations, methods for learning of explanations must be able to utilize the full capability of an existing knowledge base, i.e. they are typically explanation-based methods.

niques to diagnose missing strategic knowledge in a knowledge level model and to acquire/learn it (use of explanations and examples; of cost analysis and decision analysis of the model; of predefined general strategies etc...). An issue of focus should be to learn the appropriate strategies to be used by means of observing how a teacher solves example problems, and to subsequently learn from the failure and success of ones own problem solving behavior [Gruber, 1988, Lopez and PLaza, 1990].

4.2 A set of machine learning methods useful for model development in CommonKads

In this chapter we have, so far, attempted to show how an integration of machine learning techniques into the CommonKads methodology could be done in a principled way, and at an abstract level. We have emphasized the value of describing this integration at the knowledge level, and outlined an approach for exploring these ideas.

In this section, we will be more concrete, and take up again the question of how to map development phases and models of CommonKads - as described in chapter 2 - to the typology of machine learning methods - as outlined in chapter 3.

4.2.1 The role of task features in selection of learning methods

A task reality is characterized by a set of *task features*. In task TI.5 the role of task features in selecting CommonKADS Library components is studied. Hence, a possible incorporation of machine learning methods into the CommonKads library, should also identify a set of feature types for determining whether machine learning methods could be used to develop and/or maintain a model, and - if so - which method(s) to use.

In [Carbonell-90] a suggested list of domain problem characteristics which favor the selection of a particular machine learning paradigm is presented:

- Signal-Symbol mapping, i.e. mapping from analogous signals to discrete symbols. Suggested approach: Connectionism.
- Continuous pattern recognition, i.e. going from analog signals to a small discrete set of signal classes. Suggested approach: Connectionism.
- Discrete pattern recognition, i.e. from collections of features to membership in one of a predefined set of classes. Suggested approach: Inductive learning of decision trees.

- Acquiring new concept descriptions, i.e. from examples to general descriptions. Suggested approach: Induction of characteristic concept descriptions.
- Acquiring rules for expert systems, i.e. getting from behavioral traces to general rules. Suggested approach: If a knowledge model exists, explanation-based or analogical methods. If not, inductive or genetic approaches.
- Enhancing the efficiency of rule-based systems[20], i.e. going from search guided by weak methods to domain-dependent, focused behavior. Suggested approach: Analytic techniques ranging from chunking of macro-operators to explanation-based, case-based and analogical learning methods.
- Instruction and symbiotic reasoning, i.e. from stand-alone systems to collaborative problem solvers. Suggested approach: Inductive learning of characteristic concept descriptions, or case-based/analogical methods.
- Integrated reasoning architectures, i.e. from general principles to focused behavior within selected domains. Suggested approach: All should apply (depending on the domain), although analytical methods have been most successful so far.

4.2.2 From single methods to combined methods

Real world tasks are always complex tasks. The role of machine learning methods with respect to the various tasks of a real world knowledge-based system can be as follows: Initial task-related domain models can be developed using connectionist methods at the sensor interface (or any interface for collecting a large amount of data from the environment). Inductive methods can be used for generating (or suggesting) empirical rules for classification into symbolic concepts. Analytic methods can be used to guide the inductive process (combining empirical and explanation-based methods), improve the problem solving performance - both in speed and in pragmatic relevance of solutions, and enable a system being used to adapt to an ever changing environment (combining case-based and explanation-based methods).

4.2.3 From single methods to partial methods

So far, we have basically viewed the notion of a learning method as a complete mechanism for learning in some sense. In order to assess the possible contributions of machine learning to CommonKads, we should also look inside these full methods to see whether they contain subcomponents useful in CommonKads modeling

[20] machine learning methods has traditionally been focused towards rule learning. Fortunately, the community is moving away from viewing a knowledge base solely as a rule base. The term rule-based system should here be interpreted in a wider sense, i.e. as *knowledge-based system.*

tasks. Some problems in machine learning have been subject to particularly intensive studies, since they are problems which need to be addressed by almost any learning technique. This has resulted in a set of general methods aimed at tackling these problems. Core methods of this type, which are relevant for modeling in CommonKads, include methods related to the following learning tasks:

- *Credit assignment.* This is the task of assigning the credit (or blame) to parts of a knowledge model (including problem solving steps) in order to identify the part responsible for a failure.
- *Abstraction*
  This is the task of searching for a more general description of a set of facts or concepts.
- *Concept matching.* This is the task of identifying a concept description similar to a given concept description. Methods for similarity assessment covers the whole range from syntactic feature matching to semantic matching based on an explanation of similarity.
- *Explanation generation.* This is the task of trying to produce an explanation for an observation, in order to confirm/disconfirm that the observation is coherent with the existing knowledge model.

Clearly these sub-tasks are not only related to learning, they also occur within intelligent behavior one would normally not refer to as learning (e.g. diagnosis). This again highlights the usefulness of a unified view - at some level of abstraction - of learning and problem solving. A knowledge level analyses is likely to reveal closer resemblances than might be expected at first.

4.3 An example: developing and maintaining a model of expertise

So far, the role of machine learning techniques has been viewed within the perspective of the the entire set of CommonKads models. In order to relate machine learning techniques to the development of a single model, we need to explicate the development process of this model. The corresponding tasks may be different for different models. In the LCM document[21], such a decomposition was to some extent done for the Model of Cooperation. In the following section, we do a decomposition of the development phases for the Model of Expertise, since this is one of the major models for possible application of learning techniques[22]. We will also relate the

[21] KADS-II/TR/T2.1/LR/0020/0.2

[22] This does not exclude other single models from application of machine learning methods. However, it seems reasonable to assume that if machine learning methods may be applied to any

model of expertise to its 'subsequent' models: The Design Model and the System Model.

Learning is about acquiring knowledge and problem solving ability for future improved behavior. The notion of *knowledge maintenance* captures the periodical or continuous revision and incremental adjustments to knowledge. Maintaining the Model of Expertise, e.g., can not be described (in sufficient detail for our purpose) without relating it to the Design Model and the System Model. The Design Model, in this sense, is an intermediate between the Model of Expertise, representing a knowledge level (knowledge-use level) model, and the computational model of the running system. Hence, updates to the Model of Expertise may be done by externally accessing this model directly (propagating the changes through the Design Model to the System model), or by getting feedback from the System Model. This feedback may be in the form of failures for which the Model of Expertise will be used to find the cause and repair, which in turn will be propagated to the System Model. It may also be in the form of changes already made within the System Model (by methods for learning from experience, say), which then leads to updates in the conceptual model in order to keep the coherence of the two models - for the purpose of subsequent maintenance.

The model document[23] outlines an inference structure for the model building process at the top level, i.e. managing the development of the CommonKads *model set*. So far, no schemes has been suggested for the development of the separate models. The inference structure for the total model building process may, however, be used as a template for structuring the model-specific development processes. Below, a high level task structure for the Model of Expertise is suggested, on the basis of this inference structure and general knowledge acquisition theory.

The developing process of the Model of Expertise may be broken down into five subtasks. Each of these tasks takes as input a certain state of the Expertise model, and produces as output a model state which represents an improvement of the input model. The Model of Expertise (also referred to as the Expertise model), in this example contains subtasks representing Elicitation, Analysis, Model building, Validation, and Modification. In Figure 3, these tasks are interconnected into a task structure, where the relations (arrows) are input/output dependencies.

For example, the *Build the Expertise model* task depends on input from the *Analyse and Suggest initial structures* task and the *Modify the Expertise model* task. Dependencies between the development of the Model of Expertise and other CommonKads models are illustrated by dashed lines.

Below, a possible mapping between learning methods and the development phases

model - or combinations of models - *at least* the model of expertise will be included in this set of models.

[23](KADS-II/WPI-II/RR/UvA/017/2.0)

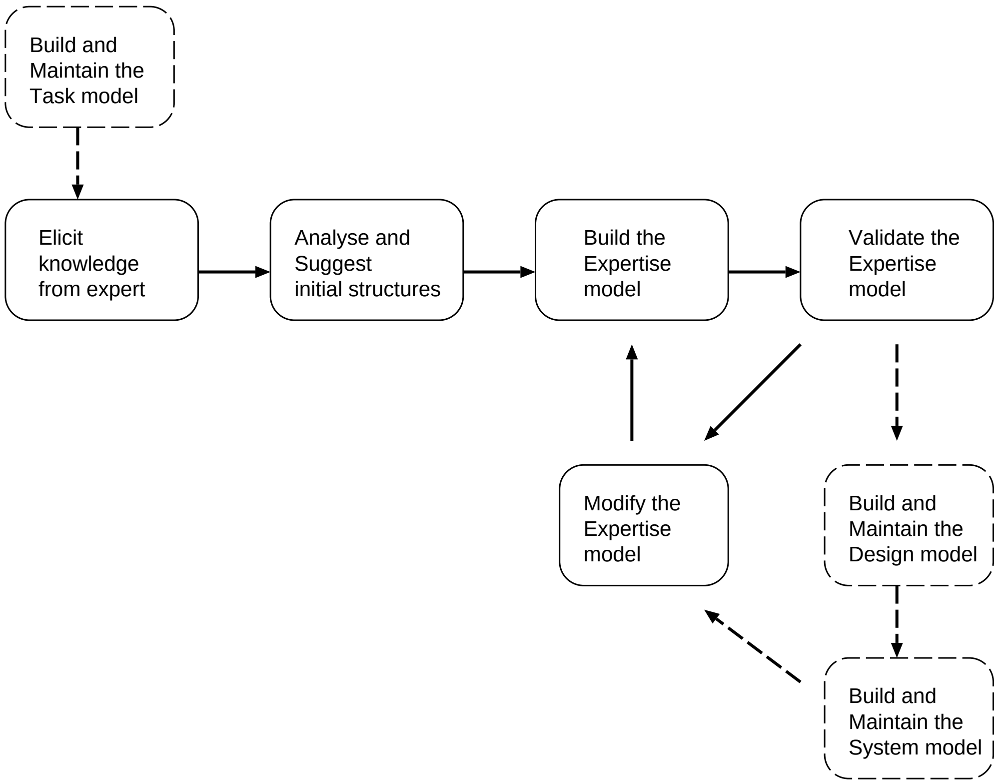

Figure 3: Subtasks in developing the Model of Expertise. Relations are input/output dependencies. Dashed arrows are dependencies to other CommonKads models, shown as dashed boxes. The five boxes drawn with unbroken lines is an expansion of a similar dashed box entitled 'Build and Maintain the Expertise model'

described in this example is listed[24] The methods represent various levels of granularity

Mapping machine learning methods to development tasks:

- Elicit knowledge from expert.
  - Conceptual clustering. Indicating possibly useful concepts.
  - Concept induction. Data rich, knowledge poor environments.
- Analyse and suggest initial structures.
  - Concept induction. Generating possibly useful concept hierarchies. Learning by use of background knowledge, in later stage.
  - Case-based learning, knowledge-poor methods for bottom-up acquisition of specific and general knowledge (e.g. the PROTOS approach).

[24]The characteristics of the machine learning methods are described in chapter 3 and in previous section of the present chapter.

- Build the Expertise model.
  - Explanation-based learning.
  - Knowledge integration.
  - Case-related learning.
- Validate the Expertise model.
  - Credit assignment methods.
- Modify the Expertise model.
  - Learning from experience. Generalization-based sustained learning, Case-based reasoning.
  - Knowledge integration.

5 Machine Learning within the CommonKads product

In this document up to now we have studied the roles of learning in the CommonKADS Life Cycle. Here we turn our attention to *integrated* uses of machine learning i.e., where the CommonKADS product itself contains machine learning components.

Any aspect relevant to the behavior of a CommonKADS knowledge-based system must be reflected in the product model. So must learning. Learning can be *represented* in any model of the CommonKADS model set. In this part of the document, however, we focus on two of these models: the Model of expertise and the Design model. We are aware that interesting insights may be gained from modeling the organizational aspects, or the impact of learning on the agents related to it but will not take up this topic for the present discussion. The following section starts the analysis of learning at the knowledge level, the level of description in the model of expertise. This is followed by a more concrete description of what we mean by 'learning as problem solving'. The third subsection outlines a useful way in which learning can be integrated with problem solving, and suggests topics for further investigation. As a suggestion on how to focus this work we provide in the fourth subsection an summary of the state of the art in learning techniques for sustained learning. In the last subsection we discuss the mapping between the Model of expertise and Design model in relation to machine learning.

5.1 Describing machine learning methods at the knowledge level

In CommonKADS, learning processes (machine learning as well as other knowledge acquisition processes) should be described at the knowledge level. Whereas symbol level models describe the "how" of learning, a knowledge level model emphasizes the "what". Understanding what a learning method does is a crucial step towards being able to use it rationally (the "why" of learning). In a way analogical to the description of an application, we can now view a learning process as being described by a knowledge level/symbol level model pair in a consistent rationalization/operationalization relation to each other (see Figure 4 and compare with Figure 2).

A strategy for exploring this relationship within CommonKADS is analogous to the analyses of problem solving methods and must proceed as follows:

- Analyze learning methods within the CommonKads knowledge level framework. This analysis identifies the knowledge level components, the global

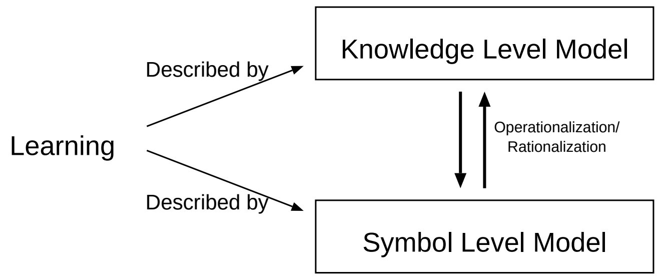

Figure 4: Learning is described at the knowledge level as well as at the symbol level. A desired learning behavior is rationalized by its knowledge level model, and operationalized by its symbol level model. The two models must be in a mutually consistent relation to each other.

structure of each method, the control scheme and the knowledge used by it. Residues of this analysis that re-occur end up in the CommonKADS library of generic modeling components.

- Use the knowledge level descriptions for the rational re-construction of prototype methods, for learning within the Model of Expertise. Components of this reconstruction end up in the CommonKADS library of second generation building blocks.

The results of the analyses and implementation efforts can be used to experiment the refinement and combination of methods, and - possibly - the construction of new ones. For example one could envisage combination of similarity based learning and explanation based learning to build an acquisition cycle of concepts based on a method of generalize, test and explain.

A reasonable way to focus this effort could be to start working within particular classes of learning techniques: Explanation-based learning methods, in particular explanation-based generalization [Mitchell et al., 1986, DeJong and Mooney, 1986], learning through progressive refinement[Van de Velde, 1988]; similarity-based learning techniques, in particular the whole range of decision-tree induction algorithms (TDIDT and ID3 [Quinlan, 1986], ID5 and ID5R [Utgoff, 1989], IDL [Van de Velde, 1990]), and generalized version space methods [Hirsch 89]; various case-based, memory-based and instance-based learning methods [Porter et al., 1990, Rissland et al., 1989, Aamodt, 1991, Lopez and Plaza, 1991]. Current research is indicating significant advantages of combining several learning paradigms, e.g. similarity-based and explanation-based [Pazzani, 1989, Kodratoff and Michalski, 1990], explanation-based and case-based [Hammond, 1987, Schank and Leake, 1989]. The study of single learning techniques should therefore be followed by investigating promising combinations.

The results of our investigations on this will be the topic of a subsequent document ("Knowledge level models of machine learning methods").

5.2 Learning as problem solving

The approach taken here to the role of machine learning in CommonKads, is based on a unified view - a problem solving view - on all the tasks a system will have to solve. Basically, a system pursues two types of goals: Application goals, i.e. problems set up by the real world application (e.g. "find a diagnosis"), and learning goals (e.g. "do not repeat this mistake"). Learning goals relate to learning in a wide sense, and can be broken down into knowledge acquisition problems or machine learning problems. However, we claim that these can all be viewed as problem solving goals, as illustrated in Figure 5.

```
ACQUISITION AND LEARNING          APPLICATION PROBLEM SOLVING

     learning goal                      application goal
     /          \                              |
 ka-goal      ml-goal                       aps-goal
    |            |                             |
 ps-goal      ps-goal                        ps-goal
```

Figure 5: All goals can be viewed as problem solving goals. The lines, read downwards, indicate decomposition relations (and 'aps' indicates application problem solving).

This unified view enables a knowledge level description of knowledge acquisition and learning problems, in the same basic language as for application problems. The analyses of problem solving studies the dependencies between goal, task, problem solving method and domain knowledge. A first important issue to be studied is therefore to understand learning goal, learning task, learning method and knowledge used by these, in the same way as for application problem solving:

```
ml-goal                aps-goal
    ml-task                aps-task
    ml-method              aps-method
    domain knowledge       domain knowledge
```

Figure 6: Unified description of machine learning and application problem solving.

It is our hypothesis that such an analyses of learning can be made and that it is useful to do so. We assume that it can be made since it can be made for problem solving. We argue that it is useful because it highlights the different roles of activities of basically the same nature.

Within this view, the *methods* that enable the learning is the main point of focus. The learning methods exploit domain knowledge to accomplish learning tasks that contribute to learning goals. Important questions that need to be answered:

- Specification of learning tasks from goals.
- Selection of a method for a task.
- Identification of domain knowledge to be used by the method.

An additional question is "where do the learning goals come from". Here we envisage different possible answers depending on the different quality criteria of behavior. Often, however, the learning goals will be frozen into the system at design time. The origination of learning goals will be discussed briefly in the next subsection.

5.3 Learning and problem solving as subgoals of each other

The framework for knowledge level modeling outlined in the previous two chapters allows for a unified description of the processes underlying different functionalities of a system: learning, problem solving, knowledge acquisition. These goals may be externally given, or may be generated by the system itself as in problem solving. Examples of such learning goals are to acquire the knowledge needed to solve a reference set of problems in an application domain, to solve problems within a given time limit or to learn to produce an adequate explanation of a solution (given an external description of adequate explanations). A major issue is therefore to study the dynamics of this process: where do the learning goals come from? How do they arise from other activities (e.g. problem solving)? How are learning tasks set up and methods selected? How can the different tasks be properly coordinated to result in maximal goal satisfaction?

Figure 7 illustrates a process which starts with an application problem being given to a system. This sets up an application problem solving goal, which activates/generates a task structure, which in turn leads to a problem solving method being selected.

This problem solving method will have some knowledge requirements. Let us assume that these requirements are only partially fulfilled by the existing domain knowledge model. An example could be that the system has a model consisting of causal, functional and component-structural knowledge, which enables it to solve problems by searching for plausible inference paths in this knowledge structure. Its overall learning goal is, say, to learn more direct associations between problem descriptors and solutions, by reducing the length of solution paths. The system sets up a learning goal (l-goal) in order to attempt to acquire knowledge of this type. This learning goal may be achieved through machine learning or knowledge acquisition tasks, and in this case a machine learning task is selected (figure 7). The underlying machine learning method relies on a certain type of domain knowledge but this

```
input problem
       \
      aps-goal
        \
       aps-task  l-goal
         \       /    \
        aps-method    ml-task   ka-goal
           \            \       /    \
            domain       ml-method     ka-task
            knowledge      \            \
                            domain      ka-method
                            knowledge     \
                                           domain
                                           knowledge
```

Figure 7: Generating learning and knowledge acquisition goals from problem solving activity. The figure should be read in a top down, left to right manner. The lines indicate control flow.

knowledge turns out not to be complete enough to give sufficient support to the machine learning method. So, a new learning goal is set up, in this case triggering a knowledge acquisition task which - through its method - acquires the necessary knowledge (e.g. by posing one or more questions to the user/expert).

This example is just one way of interaction between problem solving and learning tasks (through goals and methods). Other levels and structures of integrating problem solving, knowledge acquisition and machine learning may be just as appropriate. A thorough study of this topic requires at least the following [Van de Velde, 1991]:

- Develop constructs to express the interdependencies among different goals and tasks and the dynamics over them.
- Study the mechanisms for goal generation and goal decomposition, in particular the emergence of learning goals from problem solving activity.
- Study the integration schemata that can be used to control activities related to different tasks to obtain maximal goal satisfaction.

In the following subsection we provide an overview of state of the art methods for sustained learning that can be used to focus this work.

5.4 Sustained learning techniques

An increasing amount of machine learning research is now concentrating on incremental methods that enable learning while solving real problems. This includes approaches for learning of generalized knowledge (concepts, heuristics, relationships), of specialized knowledge (specific facts, entire cases), as well as combinations of them (case index structures, generalized cases). These activities have shown promising results that should encourage more intensive research into sustained learning methods for real world knowledge-based systems applications. Support for this statement is given by the results achieved within the following three research areas: Knowledge-intensive learning, apprenticeship learning, and case-based reasoning.

Knowledge-intensive learning represents a major paradigm shift that has taken place within the machine learning field during the last few years. Earlier machine learning methods addressed concept learning by induction, based on superficial similarities of features rather than on their meaning within a knowledge-rich model. Most of the research on knowledge intensive learning methods are explanation-based learning methods.

The notion of learning apprentice systems was introduced in [Mitchell et al., 1985] as "interactive knowledge-based consultants that directly assimilate new knowledge by observing and analyzing the problem solving steps contributed by their users through their normal use of the system". This does not represent a particular learning method or set of methods, rather an approach to sustained learning where new knowledge is continually acquired through observation and analysis. This approach is also well-suited to the application of semi-automatic learning methods, since an apprentice must be allowed to ask questions in order to increase its understanding[25].

Case-based reasoning represents an integrated approach to learning and reasoning. By retaining problem solving cases - as explicit descriptions containing problem descriptors, the solution arrived at, parts of the derivation path, etc. - previous cases can be used to solve new problems. Case-based reasoning represents the major approach to sustained learning in today's machine learning research. Learning becomes a process of extracting relevant information from a problem solving experience, and indexing this case in the system's knowledge structure in a way that facilitates retrieval when a similar problem is encountered later. [26].

A dimension for classifying learning methods - as described at the end of chapter 3 - is the 'openness' of knowledge learned (a subcategory of 'Type of knowledge'). A property of the three types of learning methods described above are that they all address learning of open concepts. Many machine learning techniques[27] assume a classical view of concept definition. This view defines a concept as a set of attributes that are singly necessary and jointly sufficient

[25]See [Murray and Porter, 1988] for an overview of a knowledge intensive learning apprentice system developed within the CYC environment

[26]Earlier work of importance to this field includes work by Schank and Kolodner on memory structures for learning and reasoning [Schank, 1982, Kolodner, 1983] and the work on transformational and derivational analogy by Carbonell [Carbonel, 1986]

[27]Notable examples are classical induction methods like Version Space [Mitchell, 1982], AQ [Michalski et al., 1983] and ID3 [Quinlan, 1986]

to classify examples of a concept. Such definitions may be adequate for mathematical and other strictly formal concepts, but as shown by several authors [Wittgenstein, 1953, Smith and Medin, 1981, Schank et al., 1986] it is extremely difficult to present classical definitions of naturally occurring concepts like 'game', 'bridge', 'chair', 'car', 'bird', etc. Smith and Medin [Smith and Medin, 1981] distinguish between the 'classical', the 'probabilistic' (e.g. degree of prototypicality) and the 'exemplar' (purely extensional) view of concept definitions. Non-classical concepts are defined intensionally by a set of typical properties - i.e. a set of default properties - and/or extensionally as the set of individuals, or exemplars, that are classified by the concept. Methods that try to learn non-classical concept definitions need support from an extensive domain model in order to explain why an exemplar is within the range of a concept, why some features are more relevant than others, etc.

Given that learning in open domains is a common feature of all sustained learning methods, the two major approaches to this type of learning differ in the generality of their learned knowledge:

As mentioned in the introduction, one approach aims at improving its body of general knowledge, by using a problem just solved as a check and a source for updating its knowledge [Van de Velde, 1988, Tecuci, 1988, Wilkins, 1989]. The learning method is usually some variant of explanation-based learning. Learning typically takes place only if the system fails to solve a problem. The problem solution - with possible annotations - are then entered by the user, who also may contribute to the learning process through a dialog with the system. The type of knowledge learned also determines the type of reasoning that the system is able to do. This approach is therefore useful for systems that solve problems based on generalized knowledge, such as heuristics or deeper (e.g. causal) relationships.

The other major approach aims at the learning of case-specific knowledge [Rissland et al., 1989, Porter et al., 1990, Aamodt, 1990]. This knowledge is used in the case-based problem solving approach characteristic of these systems: A problem is solved by retrieving a previously solved problem similar to the current one, and using the previous solution in solving the new problem. The past solution may be directly applied to the new problem, or modified - depending on the degree of similarity between the two cases. Case-based reasoning systems also have a body of general knowledge, which is used to support the retrieval and matching of cases (similarity assessment), modification of a past solution (case adaption), constructing a new case (extracting relevant features), and updating the case base (case indexing). These systems may learn both from a failure - by generating a new case, as well as from a success - e.g. by strengthening the remindings from abstracted problem features to a successfully used past case. Failures may also relieve faults or holes in the general domain knowledge, and thereby provide pointers to manual or semi-automated refinement methods.

Although some work is conducted on integrating case-based and generalization-based problem solving, there is yet no significant activity in integrating the two corresponding learning methods. Integrated problem solving methods tend to adopt a case-based method of learning (e.g. [Rissland et al., 1989, Branting, 1989, Veloso and Carbonell, 1991, Aamodt, 1991]).

5.5 Mapping knowledge level descriptions onto the symbol level

Whereas the terms of analysis for the knowledge level become gradually agreed upon, the best way for mapping a knowledge level model to the symbol level is still a matter of intensive research. Several approaches are being explored. In CommonKads the "structure preserving mapping" of models from the knowledge level to the implementation level, guided by additional design decisions is attempted. The experiments on implementing Components of Expertise [Steels, 1990] by Vanwelkenhuysen and Rademakers [Vanwelkenhuysen and Rademakers, 1990] had a similar aim. Chandrasekaran (unpublished work) proposes a general architecture like SOAR [Newell, 1990] as a basis for flexible implementations. Yet another approach is based on the direct interpretation of knowledge level models by a general interpreter which would be the equivalent of applying an ML2[28] interpreter to a formal knowledge level model of learning.

In relation to these issues the following three complementary approaches are possible:

- Course grained mapping from models to learning and acquisition tools. This is close to the generic task approach which CommonKads rejects as a general solution. Nevertheless it allows for the rapid integration of existing implementations, which is of great practical value. In this way results from MLT and ACKnowledge can be integrated in the project.
- Fine grained mapping from models to implementation. Here methods are implemented in such a way that the components identified at the knowledge level have a corresponding computational representation. This is the most flexible solution, and the one advocated by the CommonKADS design philosophy. It remains to be seen to what extend this can be applied for effective implementation of large-scale industrial systems.
- Direct interpretation of knowledge level models of learning and acquisition methods. This approach builds on previous work done in the REFLECT project and continued by work on the KADS-II formal modeling language (ML)2. Here highly formalized knowledge level models are interpreted by a

[28] Described in KADS-II/T1.2/TR/ECN/006/1.0 "(ML)2: A formal language for KADS models of expertise"

general (not method specific) interpreter. This has already shown promising results when applied to some types of problem solving and in CommonKads it could be applied to integrated learning and knowledge acquisition as well. This approach is valuable when one wants to have fast feedback from knowledge level modeling decisions. Since the knowledge to be interpreted is at a meta level, weaknesses of this approach when interpreting domain models of the real world (i.e. to preserve the intended semantical contents of knowledge-level expressions), should not cause problems in our case.

As work on design in CommonKADS progresses these ideas may be tried out on some of the learning methods that will be analyzed and implemented.

6 Conclusions and Future Work

In this document we have provided an initial analyses of how to integrate machine learning within the CommonKADS modeling framework and KBS life cycle approach. Our strategy has been a top-down one in which we start from the CommonKADS framework which is focussed on problem solving. Based on an analysis of where - and to what degree - CommonKads could benefit from incorporating automated learning methods, and a review of relevant parts of the machine learning field, we have suggested core points where learning naturally fits in. An important message from this study is that machine learning - as well as application problem solving - is most fruitfully analyzed at the knowledge level. This also enables a goal- and task-oriented integration of machine learning and problem solving.

Two broad avenues of work have been explored: learning with the CommonKADS life-cycle and learning within the CommonKADS product itself. More specifically this has lead to four roles:

- Learning within the CommonKADS life cycle
    1. Learning as a modeling operator: learning techniques can be used to push models through states.
    2. Learning about modeling: learning can be used to improve the control over the modeling process itself.
- Learning within the CommonKADS product
    1. Learning as problem solving: learning can be used in the problem solving process itself.
    2. Integrated learning: learning techniques can be used to acquire specific domain knowledge from the environment in which the application is being used.

The top-down integration of machine learning allows us to fall back on the existing CommonKADS framework. We have laid out an approach that stays as close as possible to the style of analyses for modeling problem solving. Rather than coming up with an all-encompassing analysis this has resulted in a research agenda too long to be tackled within the scope of this task or project. Some of the issues that need to be investigated:

- How can learning be described within the various intermediate models?
- How can learning be used as a technique associated with a model?
- How can learning be integrated within the modeling tools?

- What are the modeling guidelines for learning?
- How can learning techniques be analyzed to understand the structure of the space of learning of methods?
- How can a learning task be characterized to allow for the selection of an appropriate learning technique?
- How can a process of problem solving and integrated learning be controlled?

In this document we have also suggested work toward answering these questions. Ongoing work on this task will provide some of these answers. This will be important both for CommonKADS and for the field of Machine Learning. CommonKADS will benefit from this analyses because it opens up a variety of possibilities to integrate advanced techniques from Artificial Intelligence, both as a tool for KBS development and as an integrated part of the target KBS. Learning is recognized to be a cornerstone of intelligence and a promising tool for the construction of large and complex systems. There will also be spin-off benefits to machine learning research from this analyses, since it will make contact with the real needs that arise from current practice in state-of-the-art KBS work as exemplified by the CommonKADS approach.

7 Acknowledgment

The authors thank Enric Plaza (Institut de Investigation en Intelligencia Artificial, Centre de Estudias Avancads, Blanes, Catalunya, Spain), and Maarten Van Someren (SWI, University of Amsterdam) for discussions and valuable insights on the topics treated here.

References

[Aamodt, 1990] Aamodt, A. (1990). Knowledge intensive case-based reasoning and sustained learning. In *ECAI-90; Proceedings of the ninth European Conference on Artificial Intelligence,*, 1–6. ECCAI. Stockholm, August, 1990.

[Aamodt, 1991] Aamodt, A. (1991). *A knowledge-intensive approach to problem solving and sustained learning.* PhD thesis, University of Trondheim, Norwegian Institue of Technology, Trondheim, Norway.

[Branting, 1989] Branting, K. (1989). Integrating generalizations with exemplar-based reasoning. In *Proceedings from the Case-Based Reasoning Workshop, Pensacola Beach, Florida*, 224–229. DARPA, Morgan Kaufmann.

[Burstein, 1989] Burstein, M. (1989). Analogy vs. cbr; the purpose of mapping. In *Proceedings from the Case-Based Reasoning Workshop*, 133–136. Sponsored by DARPA, Morgan Kaufmann.

[Carbonel, 1986] Carbonel, J. (1986). Derivational analogy. In Michalskiad, R., J.G.Carbonell, and Mitchell, T. (Eds.). , *Machine Learning - An artficial Intelligence Approach*, volume II, 371–392. Morgan Kaufmann.

[Carbonell, 1989] Carbonell, J. (1989). Paradigms for machine learning. *Artificial Intelligence*, 40(1-3), 1–9.

[Clancey, 1989] Clancey, W. (1989). Viewing knowledge bases as qualitative models. *IEEE Expert*, 4(2), 9–23.

[DeJong, 1988] DeJong, G. (1988). An introduction to explanation based learning. In *Exploring artificial intelligence; papers from the 6th and 7th National Conferences on Artificial Intelligence*, 45–81. Morgan Kaufmann Publ.

[DeJong and Mooney, 1986] DeJong, G. and Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning Journal*, 1(2).

[Dietterich, 1986] Dietterich, T. (1986). Learning at the knowledge level. *Machine Learning*, 1, 287–316.

[Edelman, 1987] Edelman, G. (1987). *Neural Darwinism.* Basic Books, NY.

[Foley and Orttung, 1991] Foley, C. and Orttung, M. (1991). Ebirt: Extending computer-aided software engineering by harnessing corporate experience. In *Fourth International Symposium on AI, Mexico*, 430–438.

[Genesereth and Nilsson, 1987] Genesereth, M. and Nilsson, N. (1987). *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Los Altos, CA.

[Gruber, 1988] Gruber, T. (1988). A method for acquiring strategic knowledge. In *Proceedings of the 3rd Knolwedge Acquisition for Knowledge-based Systems Workshop*, 21 pgs, Banff, Canada.

[Hammond, 1987] Hammond, K. (1987). Explaining and repairing plans that fail. In *Proceeding of IJCAI-87*, 109–114. Morgan Kaufmann.

[Holland, 1987] Holland, J. (1987). Genetic algorithms and classifier systems: Foundations and future directions. In *Proceedings 2nd International Conference on Genetic Algorithms*, 82–89, Cambridge, MA.

[Kodratoff and Michalski, 1990] Kodratoff, Y. and Michalski, R. (1990). *Machine learning; An artificial intelligence approach*, volume III. Morgan Kaufmann.

[Kolodner, 1983] Kolodner, J. (1983). Reconstructive memory, a computer model. *Cognitive Science*, 7, 281–328.

[Koton, 1989] Koton, P. (1989). *Using experience in learning and problem solving.* PhD thesis, Massachusets Institute of Technology, Laboratory of Computer Science. MIT/LCS/TR-441.

[Lopez and PLaza, 1990] Lopez, B. and PLaza, E. (1990). Case-based learning of strategic knowledge. Report de Recerca GRIAL 90/14, Centre d'Estudis Avancats de Blanes, CSI, Blanes, Spain.

[Lopez and Plaza, 1991] Lopez, B. and Plaza, E. (1991). Case-based learning of strategic knowledge. In Kodratoff, Y. (Ed.). , *Machine Learning - EWSL-91; European Workin Session on Learning, Porto, Portugal*, 398–411. Springer Verlag.

[McDermott, 1986] McDermott, J. (1986). Making expert systems explicit. In *Proceedings of the 10th IFIP Congress.* Dublin, Ireland.

[McDermott, 1988] McDermott, J. (1988). Toward a taxonomoy of problem solving methods. In Marcus, S. (Ed.). , *Automating Knowledge Acquisition for Expert Systems.* Kluwer Academic Publishers.

[Michalski et al., 1983] Michalski, R., Carbonell, J., and Mitchell, T. (1983). *Machine Learning - An Artificial Intelligence Approach.* Tioga Publ. Co, Palo Alto, CA.

[Michalski et al., 1986] Michalski, R., Carbonell, J., and Mitchell, T. (1986). *Machine Learning - An Artificial Intelligence Approach.* Tioga Publ. Co, Palo Alto, CA.

[Michalski and Kodratoff, 1990] Michalski, R. and Kodratoff, Y. (1990). Reserach in machine learning; recent progress, classification of methods, and future directions. In Kodratoff, Y. and Michalski, R. (Eds.). , *Machine learning; An artificial intelligence approach*, volume III, chapter 1, pp. 3–30. Morgan Kaufmann.

[Mitchell, 1982] Mitchell, T. (1982). Generalization as search. *Artificial Intelligence*, 18(2).

[Mitchell, 1983] Mitchell, T. (1983). Generalization as search. *Artificial Intelligence*, 18(2), 203–226.

[Mitchell et al., 1986] Mitchell, T., Keller, R., and Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning Journal*, 1(1), 47–80.

[Mitchell et al., 1985] Mitchell, T., Steinberg, L., and Shulman, J. (1985). A knowledge-based approache to design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(5), 502–510.

[Murray and Porter, 1988] Murray, K. and Porter, B. (1988). Developing a tool for knowledge integration; initial results. In *Third Knowledge Acquisition for Knowledge-Based Systems Workshop*, Canada. Banff. 13 pgs.

[Neches et al., 1985] Neches, R., Swartout, W., and Moore, J. (1985). Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Trans. Softw. Eng.*, 11, 1337–1351.

[Newell, 1990] Newell, A. (1990). *Unified Theories of Cognition.* Harvard University Press.

[Pazzani, 1989] Pazzani, M. (1989). Creating high level knowledge stuctures from simple elements. In Morik, K. (Ed.). , *Knowledge representation and organization in machine learning*, 258–288. Springer Verlag.

[Porter et al., 1990] Porter, B., Bareiss, R., and Holte, R. (1990). Concept learning and heuristic classification in weak theory domains. *Artificial Intelligence*, 45(1-2), 229–263.

[Quinlan, 1986] Quinlan, J. (1986). Induction of decision trees. *Machine Learning Journal*, 1(1), 81–106.

[Rissland et al., 1989] Rissland, E., Kolodner, J., and Waltz, D. (1989). Case-based reasoning, from darpa machine learning program plan. In Hammond, K. (Ed.). , *Proceedings from the Case-Based Reasoning Workshop*, 1–13. DARPA, Morgan Kaufmann.

[Rumelhart et al., 1987] Rumelhart, D., McClelland, J., and et. al. (1987). *Parallel distributed processing; explorations in the microstructure of cognition*, volume 1-2. MIT Press.

[Schank, 1982] Schank, R. (Ed.). (1982). *Dynamic memory.* Cambridge University Press, Cambridge.

[Schank et al., 1986] Schank, R., Collins, G., and Hunter, L. (1986). Transcending inductive category formation in learning. *Behavorial and Brain Sciences*, 9, 639– 650. Pages 651-686 contains commentaries by several authors.

[Schank and Leake, 1989] Schank, R. and Leake, D. (1989). Creativity and learning in a case-based explainer. *Artificial Intelligence*, 40(1-3), 353–385.

[Shavlik and Dietterich, 1990] Shavlik, J. and Dietterich, T. (1990). *Readings in machine learning.* Morgan Kaufmann, San Mateo, CA.

[Slade, 1991] Slade, S. (1991). Case-based reasoning: A research paradigm. *AI Magazine*, 12(1), 42–55.

[Smith and Medin, 1981] Smith, E. and Medin, D. (1981). *Categories and concepts.* Harvard University Press, Cambridge, MA.

[Steels, 1990] Steels, L. (1990). Components of expertise. *AI Magazine*, 11(2 (Summer)), 29–49.

[Swartout, 1983] Swartout, W. (1983). XPLAIN: A system for creating and explaining expert consulting systems. *Artificial Intelligence*, 20, 285–325.

[Tecuci, 1988] Tecuci, G. (1988). *DISCIPLE; a theory, methodology and system for learning expert knowledge.* PhD thesis, Universite de Paris-Sud, Centre d'Orsay.

[Utgoff, 1984] Utgoff, P. (1984). *Shift of Bias for Inductive Concept Learning.* PhD thesis, Rutgers University, Rutgers University.

[Utgoff, 1989] Utgoff, P. (1989). Incremental induction of decision trees. *Machine Learning Journal*, 4(2), 161–186.

[Valiant, 1984] Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27, 1134–1142.

[Van de Velde, 1988] Van de Velde, W. (1988). *Learning from experience.* PhD thesis, Vrije Universiteit Brussel, Brussel.

[Van de Velde, 1990] Van de Velde, W. (1990). Idl: or taming the multiplexer. In *Proceedings of the European Working Session on Learning.*

[Van de Velde, 1991] Van de Velde, W. (1991). Reasoning, behaviour, and learning: A knowledge level perspective. In *Proceedings of AISB 91.* Society for the study of Artificial Intelligence and the simulation of behaviour.

[Van Someren et al., 1990] Van Someren, M., Aamodt, A., Chippington-Derrick, T., Doize, M., Jullien, C., Needham, D., Post, W., Rugg, G., Terpstra, P., and Whitehouse, S. (1990). A classification of techniqies for knowledge acquisition - part 1 and part 2. Technical Report ACK-UvA-T1.2-DL-004-D, The Acknowledge Consortium, University of Amsterdam.

[Vanwelkenhuysen and Rademakers, 1990] Vanwelkenhuysen, J. and Rademakers, P. (1990). Mapping a knowledge level analysis onto a computational framework. In *ECAI 90*, 661–666.

[Veloso and Carbonell, 1991] Veloso, M. and Carbonell, J. (1991). Learning by analogical replay in prodigy: First results. In Kodratoff, Y. (Ed.). , *Machine Learning - EWSL-91; European Workin Session on Learning, Porto, Portugal*, 375–390. Springer Verlag.

[Wielinga et al., 1991] Wielinga, B., Schreiber, A., and Breuker, J. (1991). Kads: A modelling approach to knowledge engineering. KADS-II Paper KADS-II/T1.1/PP/UvA/008/1.0, University of Amsterdam.

[Wilkins, 1989] Wilkins, D. (1989). Knowledge base refinement using learning apprenticeship learning techniques. In Morik, K. (Ed.). , *Knowledge representation and organization in machine learning*, 247–257. Springer Verlag.

[Wittgenstein, 1953] Wittgenstein, L. (1953). *Philosophical investigations.* Blackwel. pp. 31-34.