Published in *Data and Knowledge Engineering* 16 (1995) 191-222, Elsevier.

Different roles and mutual dependencies of data, information, and knowledge - an AI perspective on their integration

Agnar Aamodt[a,*], Mads Nygård[b]

[a]*University of Trondheim, Department of Informatics, N-7055 Dragvoll, Norway*
[b]*Sintef Delab, Database Technology Group, N-7034 Trondheim, Norway*

Abstract

The unclear distinction between data, information, and knowledge has impaired their combination and utilization for the development of *integrated systems*. There is need for a *unified definitional model* of data, information, and knowledge based on their roles in computational and cognitive information processing. An attempt to clarify these basic notions is made, and a conceptual framework for integration is suggested by focusing on their different *roles* and *frames of reference* within a decision-making process. On this basis, ways of integrating the functionalities of databases, information systems and knowledge-based systems are discussed by taking a *knowledge level* perspective to the analysis and modeling of systems behaviour. Motivated by recent work in the area of *case-based reasoning* related to decision support systems, it is further shown that a specific problem solving episode, or case, may be viewed as data, information, or knowledge, depending on its role in decision making and learning from experience. An outline of a case-based system architecture is presented, and used to show that a focus on the retaining and reuse of past cases facilitates a gradual and evolutionary transition from an information system to a knowledge-based system.

Keywords. Integrated systems, knowledge modeling, machine learning, case-based reasoning.

1. Introduction

There is a continuously growing interest within the fields of databases, information systems, and knowledge-based systems toward *integrated systems*. By integrated systems we will in this paper understand systems that combine the functionality and technical properties of a knowledge-based system with that of a database and/or information system. The respective scientific communities are to an increasing degree focusing their research on principles and methods targeted at such an integration. Intersection areas such as deductive databases [21, 35], knowledge base systems [26, 63], knowledge-based information retrieval [58], and intelligent decision support systems [12] are

[*]Corresponding author. Email: agnar.aamodt@ifi.ntnu.no

examples of this trend. Recent methodologies for knowledge acquisition and development of knowledge-based systems also follow this general path, by turning attention to methods and tools that relate knowledge-based system components to other parts of an integrated system [53, 71].

In order to develop successful methods for this type of integration, a characterization of the things to integrate is needed: How should data, information, and knowledge be characterized so that their differences, and other relationships relevant for systems integration, are identified? Unfortunately, the research directed toward integrated systems is most often based on a vague and unclear view of what the relevant differences are. In particular, the relation between information and knowledge is a source of much confusion and misunderstanding, as pointed out by several authors (e.g. [13], pages 70-94 and [59], pages 365-372). This calls for a unified view to what the three concepts stand for, and what their relevant similarities and differences are.

The objective of the work reported in this paper has been to gain a clearer understanding of the fundamental issues related to systems integration in the above sense, and to point out a direction for design of integrated systems on that basis. In that sense, the paper should be regarded partly as a paper that presents a theoretical framework - a high level computational model - of the processing of data, information and knowledge, and partly as a position paper that on this basis points out and argues for a new approach to integrated systems. In particular, we will show how recent improvements in AI, and particularly in the two subareas of *knowledge modeling* and *case-based reasoning*, can lead to a novel and promising architecture for integrated systems. Given the increased complexity of tasks and domains that future decision support systems will have to address, an important motivation for the work presented here is the need for systems that are able to adapt to a changing environment, i.e. to continually learn from their decision support experiences. Our research is based partly on previous theoretical studies (e.g. [1, 4, 7, 41]), partly on experience from projects on integrated systems development (e.g. [25, 42, 58, 71]).

In the next section we introduce the basic problems related to the distinction between data, information, and knowledge, and define the context for the type of integrated systems discussed throughout the paper. In the following two sections the theoretical framework that defines the characteristics of data, information and knowledge, is presented: Section 3 describes a unified definitional model, and discusses it from the perspective of *roles* in a decision making process, while section 4 discusses the *frame of reference* issue. The latter is the issue of whether the referent (the "owner") of the knowledge in a knowledge-based system, or the information in an information system, is the computer system or a human user. This is followed in section 5 by describing a knowledge level account of system behaviour, as a suitable framework for knowledge and information modeling. In section 6 a novel approach to integrated system design is suggested, centred around a combined case-based and generalization-based view of data handling, information management, and knowledge-based problem solving. Section 7 summarizes and concludes the paper.

2. Fundamental issues of systems integration

2.1. The need for a definitional model

The development of a unified and coherent model that defines data, information, and knowledge is far from a straightforward task. Attempts to resolve this issue in the general case, e.g. to answer questions such as "What is knowledge?" and "What is information?", has been a major problem of philosophers and scientists since ancient times. Seen on this basis, it is clearly beyond the scope of computer science as a discipline to provide a general definition of these terms. What computer science - leaning on its subarea of artificial intelligence - may provide is an answer to the more limited question arising from addressing this problem from a particular *perspective*, namely that of a *computational information processing system*. This limits the task to a characterization of data, information, and knowledge from the perspective of development and operation of this type of systems. Adopting this particular perspective, however, does not mean that we completely de-couple from a more general, and common sense oriented, understanding of terms. It only means that we are operating within a specialized context subsumed by the more general one. Our definition of terms should therefore be coherent (e.g. in the sense of [62]) with a global definition, but its scope or range of cover will be less. That is, there may be some usage of the terms data, information, and knowledge that will not be covered by our model. It should also be noted that our notion of a computational information processing system includes symbol-processing computer systems as well as information processing models of the human mind. We will not initially make an explicit distinction between artificial and natural systems of this kind. Our focus is on computer systems, although our general discussions will apply to some cognitive models of the human mind as well. In fact, our framework is inspired by influential work in cognitive science ([46], [57]) as well as computer science research.

The distinction between data and information has been discussed within the database and information systems communities for many years, without having resulted in a final conclusion. A possible reason is that several perspectives easily get mixed in discussions about definitions of concepts that are *polymorphic*. A polymorphic concept is a concept which can not be defined by a classical definition, i.e. as a set of necessary and sufficient features that are universally valid [50, 72]. Typical examples of polymorphic concepts are car, chair, orange, bird. Such concepts have very complex definitions, or - more to the point - they have several definitions depending on the context of interpretation. A car, for example, is not the same concept for a mechanical engineer as it is for a logistics planner or for an environment protection activist. Complex abstract concepts, such as information and knowledge, are clearly also of this kind. Mathematical concepts are counter examples. Geometrical objects such as circles and triangles, for example, have precise classical definitions. In order to get the meaning of a polymorphic (non-classical) concept, it has to be understood within a particular context [19], i.e. related to some purpose or intended use, and seen from a certain perspective. Hence, it is not surprising that the definition of information varies depending on whether it originated within, e.g., electrical signal theory, database theory, library science, or pragmatic decision making.

The primary type of systems we address here is *decision support systems* in the wide sense, i.e. including databases, information systems, and knowledge-based systems. An objective of our research is to gain the necessary understanding, within the context of a decision-making process, of the issues related to integration of data, information, and knowledge. We will show that by viewing data and information within a scope that also includes knowledge, it becomes clearer what type of meaningful distinctions that can be identified between the two. This is fed back into a clarification of how data and information differ from knowledge, and what implications this may have for the future development of integrated systems.

There is, in general, no known way to distinguish knowledge from information or data on a purely representational basis. That is, when viewed solely as represented items or structures in a machine, or on paper for that matter, they all "look" the same. Although knowledge structures typically are more complex and more tightly inter-related than other structures, this does not always have to be the case. Attempts to make distinctions based on size or complexity are therefore likely to fail. Another option - and the one chosen here - is to identify how and for what purpose the structures are used, i.e. what the various *roles* of data, information, and knowledge are in a computational decision-making process. Hence, in addition to the represented structures themselves, their *interpretation* within the contexts they are applied, and by whom they are interpreted and applied, becomes important. The latter aspect leads to the *frame of reference* problem of data, information, and knowledge [16], in which interpretation processes and the agents (humans or machines) performing the interpretations are interrelated. By the term agent we mean a system with the capability of reasoning and of taking actions on the basis of its reasoning. The crucial question here is which agent a particular body of knowledge or information should be assigned to, or whether knowledge and information can be regarded as objective and independent of a particular interpreter. For example: Is the knowledge of a knowledge-based system actually that system's knowledge? Or does it have another frame of reference, such as the system developer or user? What is the reference of information in an information system, or in an integrated system? Is there something called objective knowledge, independent of a local interpreter? And how does knowledge arise and get updated in a system, i.e. how does a system learn? Although we will not go into a deep philosophical discussion of these problems, we shall see that the answers to these questions have important consequences for development methods and modes of operations of future integrated systems.

2.2. A modeling level

What we have motivated and briefly introduced so far, is the need for an integrated view of data, information, and knowledge, and a model of their interdependencies based on their roles and frames of reference in a computational decision making process. What is further needed, in order to describe and analyse their relationships, is an expressive description language at a suitably abstract level. In order to describe properties of systems in an appropriate way, a knowledge level view [40, 65] is adopted. At the knowledge level, a system is described as an agent having *goals*, an *environment* of action and interaction, and *knowledge* of how to achieve goals. Further, a *principle of rationality* is assumed which says that an agent will use its knowledge in such a way that its goals are achieved, provided it has the knowledge needed. We shall see how data and information, too, can be meaningfully described from a knowledge level point of view. Recent advances within the areas of knowledge acquisition and knowledge modeling have made the knowledge level notion more concrete and directly applicable for systems analysis and design, as exemplified by methodologies such as KADS [70], Generic Tasks [15], and Components of Expertise [52]. The latter is the most flexible one in the sense of being least biased by a particular system development view, and it is this methodology that has been most influential to our work.

2.3. A role for specific cases

Until a few years ago, the domain knowledge explicitly encoded in knowledge-based systems was almost exclusively of a general type. Examples are conceptual and relational models such as semantic networks, frame systems or rules holding general concepts and their interrelations and mutual dependencies. General domain knowledge may be shallow and associational - as the knowledge normally contained in heuristic rules, or it may be deeper and more principled knowledge models - such as a causal model in a semantic network. A different and more recent AI paradigm for problem solving and learning is *case-based reasoning* [47, 29, 7], where knowledge is represented and utilized primarily in the form of specific and non-generalized experiences - called cases.

A case-based reasoning process is a kind of analogical reasoning, where the system tries to find a previous case in its case base similar to the current problem situation. A problem is solved by reusing the solution of the previous case, directly or after some modification. A problem case just solved is retained as a new case, or merged with another similar case, thus enabling the system to learn from each problem solving experience. A case may take various forms. It may be a simple feature list containing just a few items, or a large and rather complex structure. Our thesis is that a type of knowledge-based system that uses cases as its major type of knowledge, favours an easier and more natural integration with database and information system parts than systems based solely on generalized forms of knowledge. In a case-based system, the *knowledge is in a form that is normally associated with data and information,* namely specific registrations of observations related to particular events, episodes, or situations.

We will later discuss how the capturing of knowledge as concrete cases, and not only as generalized expressions, may facilitate a gradual transition between a passive decision support system, such as a type of database or information system, and a more active decision support system, such as a knowledge-based system. The distinction between active and passive systems here refers to the degree of active decision making support that a system is able to provide, such as suggesting solutions to problems, critiquing a user's choices, pointing out useful information for a given situation, identifying mistakes made, notifying important consequences, etc.

3. Data, information, knowledge

In addition to providing a platform for the subsequent discussions on integration issues, the purpose of this section is to contribute to a clarification of the confusion regarding the use of the terms data, information, and knowledge within the computer science community in general. The model presented is clearly an abstraction, in that it leaves out many details. It should be regarded as a first approximation, and a basis for further discussions of possible extensions and refinements.

3.1. A perspective of general decision making

The perspective taken is that of a general decision-making process, irrespective of the type of decision that is made. A multi-agent environment is assumed where a decision-making agent, also referred to as a reasoning agent, receives input from and returns output to an environment external to it. A reasoning agent may be a human being or a machine. Figure 1 illustrates the basic context. The figure illustrates a decision step, which is performed by a reasoning agent in interaction with its external environment. The environment typically contains other agents that supply the input and receive the output. In a simple set-up, the external agent is a terminal user, and the primary agent (grey area in the figure) is a terminal-based computer system.

In the following, a unified model of data, information, and knowledge is presented. The three concepts are viewed from two complementary and interdependent perspectives: The roles they take in a decision-making step, and their frame of reference. We will discuss the roles within a simple set-up, involving a computer system and a terminal user (see Figure 1). A multi-agent process is viewed as multiple single-agent processes, where each agent views the other reasoning agents as parts of its environment.

3.2. A definitional framework

In this subsection, the concepts of data, information, and knowledge are discussed within the simplest possible context: That of a single agent decision-making process. The following three points summarize the essential differences between the three concepts:

- Data are syntactic entities
  - data are patterns with no meaning; they are input to an interpretation process, i.e. to the initial step of decision making.
- Information is interpreted data
  - information is data with meaning; it is the output from data interpretation as well as the input to, and output from, the knowledge-based process of decision making.
- Knowledge is learned information
  - knowledge is information incorporated in an agent's reasoning resources, and

made ready for active use within a decision process; it is the output of a learning process.

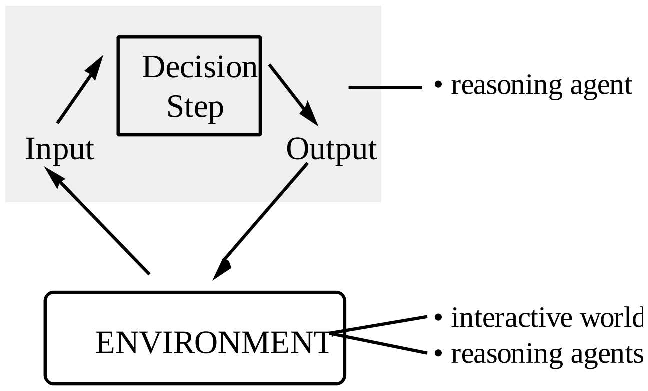

Figure 1: A decision-making process.

The distinction between data and information seems to be in accordance with several authors who discuss this relationship. Silver [49], for example, although taking the perspective of a production process where data is the raw material and information the product, ends up with a distinction that maps well to ours. A difference is that we place more emphasis on the syntax vs. semantics distinction. This makes the notion of *interpretation* central, since it is through an interpretation process that a syntactic structure is transformed into a semantic, meaningful entity.

Our definition is also consistent with the version saying that "Information is data which is used in decision-making", but goes beyond this 'standard' definition since it links the use of data to the underlying interpretation process that enables this use. Knowledge, then, is what is needed in order to perform the interpretation, and what gets learned from new information.

The role of knowledge, in general, is therefore to play the active part in the processes of transforming data into information, deriving other information, and acquiring new knowledge - i.e. to learn. This leads to the following summary of knowledge roles:

I	To transform data into information	- referred to as *data interpretation*
II	To derive new information from existing	- referred to as *elaboration*
III	To acquire new knowledge	- referred to as *learning*

Note that the term knowledge is used here in a very general sense. Hence, it does not distinguish between 'true' and 'believed' knowledge. This is different from the influential branch of philosophy in which the term knowledge is used exclusively for statements that are true in the world, and where belief is used if truth cannot be ascertained (e.g. [23]). Other philosophical theories [61] have questioned this position, arguing that the logicist, or deductive-nomological philosophical view that lies behind that view is unable to explain major philosophical problems such as analogical reasoning, abduction, and scientific development).

The relationships between data, information and knowledge are illustrated in Figure 2. The grey lines show essential roles of knowledge in data interpretation, information elaboration, and learning of knowledge. The model illustrated will be explained throughout the remaining of this section.

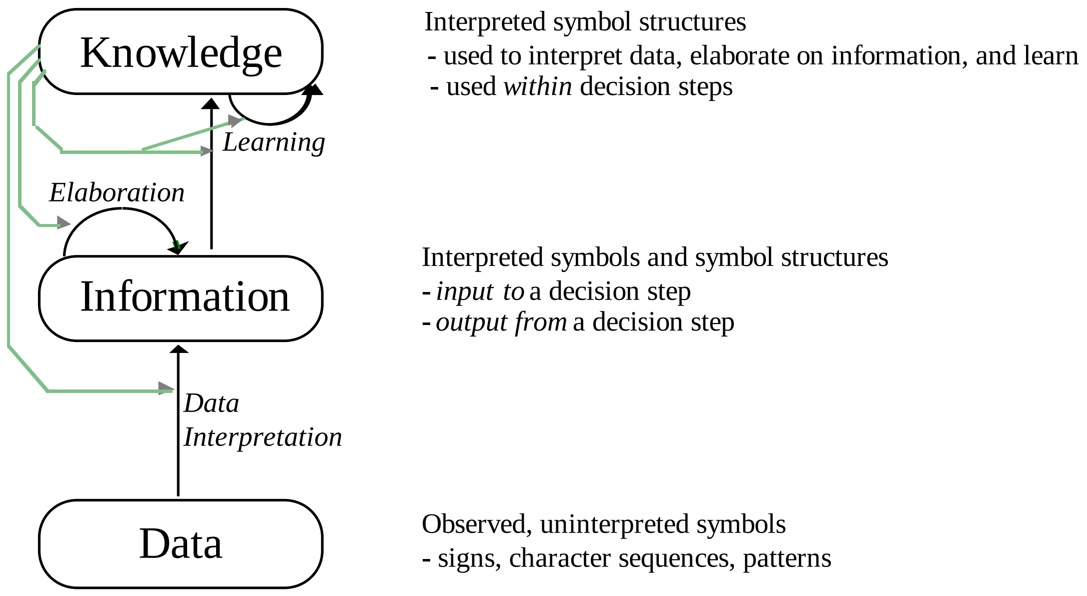
Knowledge
Learning
Elaboration
Information
Data
Interpretation
Data
Interpreted symbol structures
- used to interpret data, elaborate on information, and learn
- used *within* decision steps
Interpreted symbols and symbol structures
- *input to* a decision step
- *output from* a decision step
Observed, uninterpreted symbols
- signs, character sequences, patterns

Figure 2: The Data-Information-Knowledge model.

3.2.1. Data interpretation

A distinction between data and information is that data are uninterpreted characters, signals, patterns, signs, i.e. they have no meaning for the system concerned. Data becomes information after having been interpreted to give meaning. This is illustrated in the figure by the *Data Interpretation* arrow. In order to interpret data into information, a system needs knowledge. For example, "´Q)9§?8$%@*¨&/", or a series of signals from a sensor, is data to most human beings, while the data items "inflation rate", "decreased blood pressure", and "the Cuban crisis" have meaning, and therefore are information. The meaning of these terms may be different for different people, and it is our knowledge about particular domains - and the world in general - that enables us to get meaning out of these data strings. Hence, for data to become information an interpreter is required.

Note that this notion of an interpreter refers to a process that is substantially more complex than simple language interpreters as we know them from programming languages, linguistic grammar analysis, and formal semantics. The type of semantics needed to interpret data into information within a decision process has to be strongly related to pragmatics [9], i.e. interpretation within a real-world context and for a particular purpose, not only to a language-syntactical semantics. It is the interplay between the data interpretation method, and the knowledge that a system brings to bear in that process (the grey line in Figure 2 from the Knowledge box to the Data Interpretation arrow), that determines a system's ability to derive information from data, i.e. to "understand" the data.

In a such data interpretation process, a human decision maker will typically use his cultural background, unconscious intuitions, concrete memories of similar observations in the past, expectations triggered by the specific context, as well as text book knowledge and domain dependent heuristic rules, to determine the contextual meaning of data. Knowledge-based computer systems have by far not reached this degree of sophistication yet - and maybe they never will - but that is an issue of physical realizations of systems, and a different one than the principle issue of computational processes we are concerned with here. (After all, the purpose of a decision-support system is to actively support a human decision make, not to replace him).

It should also be noted that independently of the Data Interpretation process as such, the notion of knowledge, by itself, always assumes an internal interpreter as a necessary part of the knowledge representation (knowledge encoding) system. Knowledge - including its underlying interpreter - is needed within the Data Interpretation process, as just described, as well as within the Elaboration and Learning processes (see Figure 2 and the description below). To avoid confusion between an interpreter in a general sense and the particular Data Interpretation process, we will consistently refer to the latter as "data interpretation", unless this is clear from the context.

3.2.2. Elaboration of information

Once the data has been given an interpretation as information (an initial interpretation, at least) by the process described above, it is elaborated upon in order to be better understood and for deriving (inferring) new information. This is illustrated by the *Elaboration* arrow in Figure 2. Information that may be inferred in this way includes additional problem features, generated hypotheses, consequences of hypotheses, suggested solutions to problems, explanations and justifications of suggestions, critiquing arguments, etc. The elaboration process is the actual problem solving process, i.e. where the core decision-making takes place. Interpretation of data into information may be viewed, simplified, as a kind of pre-processing with respect to the core of the decision making process. However, frequent interaction with the environment during decision making blurs this picture and illustrates the over-simplification of such a view. In a more realistic decision-making process, elaboration and data interpretation are interleaved; initial interpretations get modified during elaboration, elaboration leads to questioning of the environment, which in turn leads to new or revised data being entered, or to revised interpretations of previous data.

Elaboration processes typically apply well-known inference methods such as spreading activation, associative memory retrieval, rule chaining, traversing of inheritance hierarchies, constraint propagation, etc. Elaboration typically involves both control level strategic reasoning and object level inferencing. The elaboration process usually ends in a conclusion. This may be a final decision, or just a temporary halt in order to pose a hypothesis or to question the environment for more information. In both cases the system returns something to the environment which from its own point of view is information, but which from the receiver's point of view is data. It will be turned into appropriate information for the receiver if the receiver is able to interpret the data as intended by the delivering agent. Hence, a common body of knowledge is crucial for meaningful communication between agents.

As illustrated in Figure 2, the interpretation and elaboration processes require knowledge. They may also require other information. But information and knowledge serve different roles in this game: Information is something which serves as input to the elaboration process (after having been interpreted from data), and something which is produced as output from it, while knowledge is something which exists and is brought to bear *within the decision process* itself. Knowledge is an inherent resource of a reasoning agent that enables inferring of new information from existing information. The inferred information may, in turn, lead to the inferring of more information, and so on. For example, if - in a medical decision support system - the information "temperature has increased from 37 to 41.5 degrees during the last hour" is given, a system may use its knowledge to infer "strongly and rapidly increased temperature", from which "possible life-threatening condition" may be concluded, in turn leading to the action "check for life-threatening condition".

Given that knowledge is something that resides and is used inside a decision process, a relevant question is what properties of the internal structures and processes is it that makes it knowledge, instead of, say, data and algorithms in the usual computational sense. There are (at least) two types of answers to this, one from the perspective of an external observer, the other taking an internal method-oriented stance. From an observer's point of view, it is solely a matter of choice whether it is meaningful/useful to describe a particular type of systems behaviour in terms of knowledge or not. This viewpoint is elaborated in the section 5, related to the discussion of the "knowledge level" as an appropriate system description level.

From a method - or mechanism - point of view, on the other hand, to describe computational methods in terms of knowledge and inference, instead of data and algorithms, usually implies some assumptions on the type of underlying processing that is done. The notion of non-determinism is central here. Non-deterministic processing is often the only means available when the input data, and/or optimal ways to process them, are not well understood. The complexity that is involved in this case, and which follows from the type of problems addressed, calls for processing methods at a level of flexibility that is not easily described and realized by strict algorithmic approaches. For this type of computation, the cognition-inspired language of knowledge structures and inference methods has, through years of AI research, shown to be more suitable for characterizing and realizing the necessary processing methods.

3.2.3. Learning of knowledge

A system's knowledge grows and gets modified through interaction with the environment. This process is what we call learning. A widely shared view is that learning is the integration of new information into an existing body of knowledge, in a way that makes it potentially useful for later decision making. New knowledge may also come from inference processes within the knowledge body itself. This is illustrated by the two *Learning* arrows in the figure, respectively. We here focus on learning as the process that produces knowledge, this also says something important about what knowledge is.

First of all, learning is viewed as an integration process, in which new information is integrated into an existing knowledge structure. Knowledge should be viewed as an *integrated totality*, and it is the tightly connected network of interrelated subcomponents that gives knowledge its power of data interpretation, information elaboration, and learning. Second, knowledge as the outcome of a learning process links knowledge to its *potential use*. Learning, as a process, is always related to a purpose, a way to make future use of what is learned [19, 27]. Even if we, as humans, will try to generalize our experiences into general patterns, and abstract our observations into universal principles, there is always a purpose behind this. When we over-generalize during learning, we are quick to specialize again - for example by noticing exceptions - when we realize the mistake [37].

The inherent property of learning as a generalization-making process points to a third characteristic of its product, namely that knowledge is *flexible*. Even when formed within a particular context and for a particular type of use, it may be reused dynamically in future situations that are different. The 'tension' within a body of knowledge to be both specific and related to the use from which it originated, as well as being general and flexibly usable, is a feature that distinguishes it from data and information.

4. Knowledge-based systems and the frame of reference

As computer scientists we frequently use the terms knowledge and information without making clear *whose* knowledge or information we are talking about, i.e. what their *reference* is. For example, does "knowledge" in the term "knowledge-based system" refer to knowledge of the system designer, knowledge of the user, or the computer system's knowledge? This issue is of crucial importance in order to thoroughly understand the individual properties and multi-agent roles of the components that make up an integrated system. In the following, we will for simplification reasons discuss the frame of reference of *knowledge,* and later relate the results to information and data. In order to get the appropriate perspective on the problem, we begin with a brief review of the foundation of the knowledge-based paradigm in AI.

4.1. The knowledge-based paradigm

At the very heart of the notion of knowledge-based systems, and a fundamental assumption of the knowledge-based systems paradigm, is the conjecture that knowledge is something that can be identified and explicitly represented. This is captured by the so-called "knowledge representation hypothesis", e.g. as expressed by Brian Cantwell Smith [51]:

The Knowledge Representation Hypothesis:

Any mechanically embodied intelligent process will be comprised of structural ingredients that
a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and
b) independent of such external semantical contribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.

The hypothesis says that within an intelligent system there exist "structural ingredients" which we as observers take to represent the system's knowledge, and that these structural ingredients not only exists, but also play a causal role in producing the system's intelligent behaviour.

The capturing of knowledge in explicit symbol structures, based upon the knowledge representation hypothesis, is tightly linked to the notion of a "physical symbol system" [39]. A physical symbol system is a type of system - realizable as a physical system - which is able to represent and manipulate symbols in a way that enables intelligent action to be produced. Humans are examples of physical symbol systems, according to Newell, as are (or may be) knowledge-based computer systems as well. To represent knowledge in computer systems therefore requires a language in which all relevant concepts, propositions and complex relationships can be syntactically expressed, and an internal interpreter (a coherent set of inference methods) which ensures that the semantical contents of the representation - as viewed by the computer - is sufficiently close to the real world semantics as viewed by the human designer/user.

Note that no claim is being made here that the knowledge structures within a computer are of the same physical "kind" as the knowledge we ascribe to human beings. The only claim is that they *functionally* serve the same role in the behaviour of artificial intelligence systems as they do in humans. In essence, the knowledge-based systems paradigm assumes that explicit symbol structures is an appropriate way to describe knowledge in general, and a suitable way to represent it within computers.

4.2. The frame of reference of knowledge, information, and data

The "structural ingredients" that represent a system's knowledge contain or are associated with inference methods, i.e. low level interpreters, that capture the semantical contents of the structures within relevant reasoning contexts. Since these structures and interpreters in principle are local to each reasoning agent, the knowledge of an agent will always have to be subjective. Hence, the structures and interpreters have the local system itself as its frame of reference, and may therefore be referred to as the system's knowledge. A knowledge-based system, per definition, has knowledge -

it's "own" knowledge - and ways of processing that knowledge.

According to this view there is no such thing as objective knowledge in the strict sense, since a collection of agents always will have different histories, experiences, environments of operations, etc. However, it may still make sense to talk about 'common' or 'objective' knowledge, but then the assumption has to be made that the agents are similar systems, that they have similar experiential and/or cultural background, etc. Two mathematicians discussing a mathematical problem, for example, share practically the same contextual background, and hence interpret data in a common way. The same may be the case for two knowledge-based systems operating within an intensive care unit, and whose tasks are to monitor patients and suggest actions on alarms. Therefore, when agents share an interpretation context, the respective knowledge may be called objective with respect to those agents. A way to make agents functionally similar with respect to interpretations is through teacher-learner relationships, for example between a human teacher and a computer learner. Given that we have established a frame of reference for knowledge, what does this tell us about the frame of reference for information? The answer is rather trivial: Information is the result of a knowledge-based data interpretation or elaboration process. The knowledge applied within these processes determine the resulting information content. Hence information will have to have the same frame of reference as knowledge.

As seen from the Data-Information-Knowledge model (Figure 2), a system cannot possess information without having knowledge, i.e. without being what we here refer to as a knowledge-based system. The term *information system*, however, is usually used for systems that do not necessarily have knowledge and reasoning capabilities. They are systems intended to store and process structures that are to be interpreted as information *for the user*. An important distinction between a knowledge-based system and an information system is therefore that while the frame of reference of information in an information system is the system user, the frame of reference of knowledge in a knowledge-based system is the system itself.

From a system user's point of view this difference may not be very significant, since the data that a system present to the user will be interpreted into information for the user, irrespective of whether it has been processed by the computer system as information, knowledge or as mere data. There may be differences in the flexibility and intelligibility of the dialogue and way of interaction, but not necessarily so. The difference is significant, however, when it comes to methods for realizing the two type of systems. Unfortunately, in the history of AI it has often been the case that knowledge-based systems have been designed as systems that capture information and knowledge for which the user is the only frame of reference, as pointed out by Dough Lenat, the designer of the CYC system [30], among others.

Data, being a purely syntactic notion, can in general be regarded as global and neutral, and not in the need for a particular frame of reference. (This neutrality only applies to a data item itself, however, and not to the way data is produced and selected, which is a decision problems in its own right, subject to pragmatic, social, and other contextual constraints.) Data is of little value in itself, it is a source of information and knowledge which gets "elevated" through the successive steps of data interpretation, information elaboration, and learning (Figure 2). When databases contain data that are readily interpreted as information by human users, the difference between a database and an information system - from a frame of reference perspective - vanishes. An interesting type of system, from an integration point of view, is a deductive database. Here, an inference method - deduction - is applied within a database system to derive data from other data. Knowledge, often represented as a set of logical axioms, is used in this process. According to our model, a deductive database can be described as a knowledge-based system, which uses its knowledge to interpret data into information, and applies knowledge-based inferencing to derive new information, which in turn is stored - as data - in the database.

It may be interesting to note that an alternative view to the frame of reference problem has been suggested by some AI researchers who question the physical symbol systems hypothesis, and - based on a framework developed by Winograd and Flores [72] - advocates a 'situatedness' view to the understanding of decision making and other cognitive processes (e.g. [16]). According to that view, knowledge is not represented in explicit structures, but dynamically constructed in an interactive process between an agent and the rest of the environment in which the agent is situated. Hence, the frame of reference of knowledge is not a particular reasoning agent, but the total system containing the environment and its interacting agents. We have elsewhere discussed, and argued against, this view [5], as have others as well (e.g. [48, 69]).

Note that our definitions of data, information, and knowledge should imply that, strictly speaking, it does not make sense to talk about "knowledge in a book", or "information in a library". That is, unless the book or library has reasoning capabilities. To be precise, we should talk about *data* in a book, and about books as *sources* of information and knowledge. Our common sense use of these terms may therefore differ substantially from the definitions we have present here. This does not necessarily mean that the common sense notions are nonsense. Firstly, this issue relates to our previous comment about 'objective' knowledge as knowledge commonly shared among agents due to a common scientific background, historical development or culture. Secondly, an explanation of the different uses of terms should take into account the differences of contexts - the different purposes and perspectives - of a common sense account and a computational one.

To sum up the last two sections, the roles and frame of reference of knowledge ascribe to a particular agent the ownership of a certain body of knowledge. Knowledge is knowledge *for* that particular agent, implying that the agent is able to generate intelligent behaviour based on it. Correspondingly, information is data interpreted by and for a particular agent, and is therefore also 'owned' by that agent. Data is uninterpreted patterns, and has no particular frame of reference. By arguing for the plausibility of the model, showing its consistency with respect to the characterization of different system types, and indicating its robustness even with respect to some common sense notions, we hope to have delivered a well-supported argument for the model.

As stated before, the main purpose of the model is to establish a sound platform for developing integrated systems. To be able to perform the necessary analysis and conceptual level modeling of the various type of components of such a system, a suitable modeling perspective and language is needed. In this next section a conceptual modeling framework is outlined, based on the definitional model described in the last two sections.

5. A knowledge level framework for integrated systems

5.1. Knowledge acquisition background

Recent research in knowledge acquisition has produced several methodologies for analyzing and modeling knowledge and information at a conceptual and implementation-independent level. Well known examples are the KADS methodology [70], the Components of Expertise (CoE) framework [52], and the Generic Tasks [15] approach. Adopting the view of knowledge acquisition as constructive modeling [38] - as opposed to a "knowledge transfer" view - a growing part of knowledge acquisition research is focusing on describing problem solving behaviour at this level. Attempts to unify several existing viewpoints and methodologies are also under way, as exemplified by the multiple perspective approach of the CoE methodology [54], and by CommonKADS [71].

A knowledge acquisition methodology establishes a certain perspective, and provides an associated set of analysis and synthesis techniques to describe the essential classes and structures of domain knowledge, problem solving methods, and application tasks, given a particular type of application. This level of system description is often referred to as the *knowledge level*, after Newell's influential paper [40]. In that paper the knowledge level was proposed as a distinct level of description of computer systems, defined to lie above the level of data structures and programming languages - which is referred to as the *symbol level*. There are also other description levels in Newell's model, for example the register-transfer level, the logic circuit level, and the electronic component level. Each level is characterized by a particular *medium* and a *behavioural law*. The medium is what is being processed, and the behavioural law is the type of mechanism used to realize a system behaviour from what is expressed through the medium. The logical circuit level, for example, has zeroes and ones (logical on or off) as its medium and Boolean logic as it behavioural law. At the symbol level, the medium is symbols and symbol structures (data structures and programs), and the behavioural law is the sequential interpretation of procedures.

5.2. The knowledge level

The knowledge level has knowledge as its medium and the principle of rationality as its basic behavioural law. A system is described at the knowledge level as an agent with its own goals and with knowledge related to the achievements of these goals. The principle of rationality states that an agent will always use its knowledge in a way that ensures the achievement of its goals - provided the agent has the knowledge needed. Hence, the notion of rationality is an ideal one, that disregards any pragmatic constraints of computational resources, time constraints, etc. This idealization has made the original knowledge level notion difficult to use for practical systems description and modeling. However, the notion of knowledge level has undergone some modification over the years, from

Newell's highly intentional and purpose-oriented way of describing a system, to a more structured and focused type of description (e.g. [3]). This transition has lead to various 'operationalizations' of the knowledge level notion, associated with terms such as the knowledge use level [52], a knowledge level architecture [55], and the notion of bounded or tractable rationality [65]. The original idea of the knowledge level has been extended by introducing high-level structural and methodological constraints. This makes the knowledge level more practically applicable for conceptual modeling purposes, while retaining its competence-oriented and implementation-independent aspects. We will here refer to knowledge level modeling in this extended sense.

Although there are a variety of knowledge modeling methodologies, most of them start out from the following three types of component structures:

• Tasks	- what are the goals of the system, what should it do?
• Problem solving methods	- by what methods will the system accomplish its tasks?
• Domain knowledge	- what knowledge is needed by the methods in order to accomplish these tasks?

Tasks as well as methods and domain knowledge are structured in class hierarchies and inter-related in various ways, depending on the modeling methodology being used. The structuring of the model space into the three component types listed above, is analogous to the high level structuring of information types made in many information systems methodologies.

As an example of a modern knowledge level modeling method, that also incorporates information modeling, the Components of Expertise methodology (CoE), is summarized in the following, within the context of the previous discussion. It should be noted that this methodology is still under development. A workbench, called KREST, has been developed to support the methodology, particularly by providing a library of reusable knowledge level modeling components, and a linking mechanism between knowledge level models and symbol level programs [33].

A knowledge level description of a system is made by splitting the analysis and design work in three, basically corresponding to three perspectives of a knowledge level description. These perspectives are within the CoE framework called tasks, methods, and models, respectively. In other methodologies thay may have different names and slightly different borderlines.

Tasks are the tasks (and subtasks) identifying what to do in order to solve an application problem. *Methods* are problem solving methods for decomposing a tasks into subtasks, or for solving a task without further decomposition. There are two types of *models*: "Domain models" are general knowledge models of a domain, for example associations between a set of possible findings and a set of possible faults. A domain model may also be a set of decision rules, or models of functional relationships between devices. Domain models correspond to models of general, object level domain knowledge. The other model type is "case models". A case model is a description of an actual problem that is being solved, for example a set of actual measurements and observations. It may also be an instantiation of part of a domain model for a particular problem, e.g. an association between the findings observed for a particular car that will not start, and the fault identified for that particular car. Case models, according to our previous definitions, correspond to *information*. Although our definition of data, information, and knowledge is not an explicit part of the Components of Expertise framework, the notion of a case model makes it a suitable knowledge level methodology from our integrated systems point of view. (Note that the term "case" as used in case models does not quite correspond to its use in case-based reasoning, where a case also denotes a permanent storage of a past problem with its solution and possible annotations.) Domain models, correspondingly, may describe information models as well as knowledge models.

Tasks, models, and methods are structured into task decompositions, model dependency diagrams, and control structures, respectively. This is illustrated in Figure 3. In CoE, a task decomposition relates tasks to their subtasks in a part-subpart hierarchy. For example, a task may be to diagnose a car, with the subtasks to observe symptoms, to decide further tests, to perform tests, and to identify likely faults. The leaf nodes in the hierarchy are tasks that are solved without further decomposition.

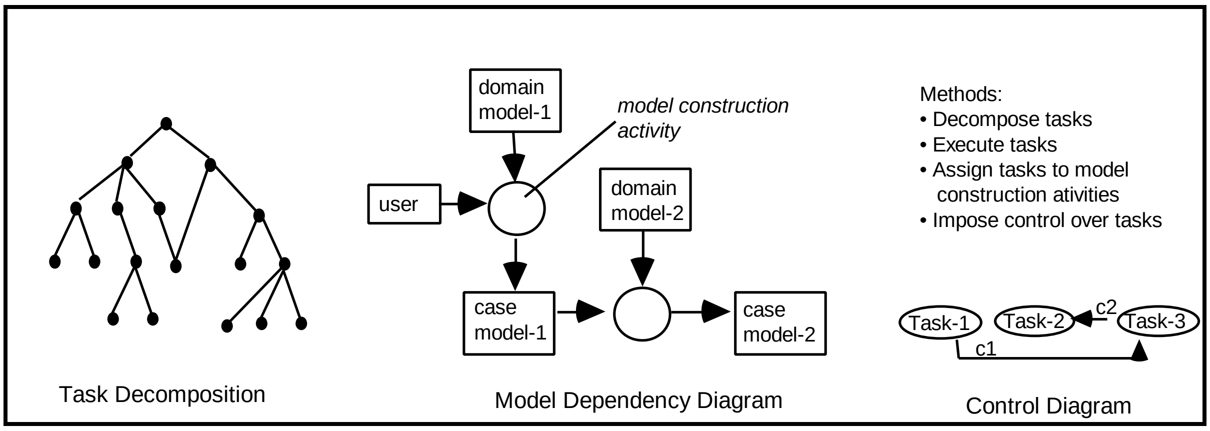

Figure 3: Components of Expertise diagrams

Model dependency diagrams are used to inter-relate the various domain knowledge types that are needed to construct a new case-model on the basis of existing case models. As shown in Figure 3 (middle part), an initial case model is constructed from information provided by the user together with knowledge found in a domain model (e.g. a partial model that infers additional problem descriptors). The resulting case model - called case-model-1 in the figure - becomes input to another construction activity, which takes another domain model (e.g. a causal model, or a combined structural and associational model), and constructs a second case model. Model dependency diagrams are organized into abstraction hierarchies. By expanding a model construction activity, model dependencies at a more detailed level are described. This is therefore a suitable means to relate knowledge and information types at different levels of details. Model construction activities will typically be identical to tasks in the task decomposition.

Problem solving methods are applied to tasks in order to accomplish them. There are two main types of problem solving methods: Task decomposition methods - which return a decomposition of the task it is applied to, and task execution methods - which execute a task directly without further decomposition.

The power of using the three perspectives for knowledge level modeling lies in how the perspectives interact. When making a description according to one perspective, the other two perspectives may be used to make the description more detailed and application focused. As an example of interacting perspectives, let us take a task perspective, and see how a task may be decomposed. A task may be decomposed in two principle ways: By a method-oriented decomposition, or by a model-oriented decomposition. In the former, subtasks of a task are determined by the type of task decomposition method chosen for the task. For example, a method called Cover-and-differentiate will decompose a task into two sets of subtasks: One which will try to find solutions that cover for the observations made, and another that tries to differentiate between possible solutions in order to find the best one. In a model-oriented decomposition, the subtasks of a task are chosen according to what type of domain-models they relate to and the type of case-models they produce, regardless of the problem solving method used to achieve the decomposition. An example would be to decompose a task into one group that handles input of component information, another that deals with process information, a third that deals with the acquisition and use of domain knowledge in terms of functional relationships between subsystems, etc.

From an information system point of view, an analysis in order to specify functional requirements for information systems in a conceptual, implementation-independent language, clearly corresponds to a knowledge level analysis for a knowledge-based system. The perspective and focus are different in the two types of methodologies, of course, since a conceptual information system model tries to capture information so that it can be shared among human users, while an aim of a knowledge level model for knowledge-based systems is the capturing and utilization of knowledge for reasoning by a computer system. If these two perspectives are combined, however, a knowledge level description can be viewed as an extension of a conceptual information model. In the Structured Analysis method [22], for example, the top level components are processes, data flow, external entity, and data store. Processes describe what to do, and map readily to task type components. Data flow specifies the type of input and output to a process. This is included in the task description of a knowledge level model. External entity and data source describe the content and form of what goes into and comes out of a process, which can be viewed as analogous to a high-level domain knowledge model. Specific modeling methods (e.g. OOP [17], PPP [59]) describe the content types and structures of information system components at more detailed levels. Knowledge level modeling adds the perspective of computer systems as goal-driven agents, and knowledge modeling methods enable the description of task structures, methods, knowledge types, and information types needed for agents to accomplish their goals.

5.3. *From the knowledge level to the symbol level*

Given a knowledge level model in some state of development, the important next question is how such a model can be used in the design and implementation of a computational system. That is, we need methods which enable us to bridge the gap between a descriptive and a prescriptive knowledge level model, and an operational computer system. In general, a knowledge level model should be viewed as a *basis for* subsequent design and implementation of the artefact system, without necessarily assuming that a complete transformation is possible - or even wanted.

Although we still have a substantial distance to go before we are able to realize all the types of intelligent behaviour we would like to see in our systems, AI research is steadily producing new and better methods along the way. AI methods are rapidly migrating into systems development techniques and tools in general (for example database design systems [56]). The ability to implement more comprehensive, more competent, more robust, and more adaptive knowledge-based systems - i.e. systems capable of addressing increasingly complex real world applications - has been significantly improved through recent advances within several areas of AI research. Increased understanding of the problems we face has come out of research on intelligent architectures [66], frame-based representation languages [67, 30, 37], case-based and mixed-paradigm reasoning systems [45, 2, 8, 34], integrated learning methods [60, 43], and knowledge-intensive methods for integrated problem solving and experiential learning [64, 10, 6]. These and other related results enable us to develop knowledge-based systems that can deal with real world application domains and problems that are substantially more complex than what we could with the previous generation of basically rule-based techniques. This fact has also motivated the research on knowledge level modeling, since the increased complexity and depth of symbol level models have made it more important to make a thorough analysis at the conceptual level .

A number of tools have been - and are being - developed to support knowledge level modeling as well as system design and implementation. They range from relatively general tool boxes, such as the KEW system developed in the Acknowledge project [25], to strongly methodology-driven workbenches, usually including libraries of reusable modeling components. Some of these approaches are aimed at knowledge level modeling only (e.g. [71]), while others attempt to provide direct links to symbol level components, in order to support reusability of symbol level models (e.g. [27, 54]). This is in line with the general trend in software engineering towards reusable programming constructs and automatically generated programs from specifications.

5.4. *A language for bridging the knowledge level to symbol level gap*

In order to enable a computer to capture integrated information and knowledge models, a coherent description language is needed at a more detailed level than what CoE offers. Knowledge level modeling methodologies emphasize the analysis and description of *types* of knowledge and information components, but are in general weak in representing the actual specific domain models and their inference methods. For this type of modeling, we will also need a conceptual, knowledge level language, in order to express knowledge and information informally, to analyse their types, constraints, and specific relationships (e.g. causality), and to perform some tests and comparisons. A knowledge representation language in the normal sense is needed for symbol level implementation, however. Given their different requirements, the languages may be realized as two separate languages, or they may be combined into a single, unified language. A single representation language does not imply that a knowledge level model can be directly used as the symbol level operational system. As mentioned in section 4.3, transformation and rewriting will usually have to be done. Anyhow, to work within the same representational environment both for knowledge level conceptual modeling, and for system level design and implementation, has major advantages [68]. An object-oriented frame system language, aimed to capture both knowledge level and symbol level models, has been developed an used for this purpose [1]. Variations of this language has also been used as a knowledge representation language in the KEW workbench of the Acknowledge project [25], and in a knowledge-based front-end to an information retrieval system [58]. It is an example of a language suitable for our purpose, since it emphasizes capturing knowledge *content,* and uses a procedural semantics to achieve this. Its open nature makes it suitable for information modeling as well as knowledge modeling and representation. Its most developed version is the *CreekL* language [4] for integrating general domain knowledge with case-specific knowledge, as will be elaborated in the next section. Its underlying assumptions and main characteristics are as follows:

concept (frame)	*relation (slot)*	*value-type (facet)*	*value*
patient			
	subclass-of	value	person
	has-identification	value-class	patient-id
	has-history	if-needed	(patient-journal-lookup patient-id)
	has-age	value-range	(0-120 years)
obstructive-respiratory-disease			
	subclass-of	value	pulmonary-disease
	has-subclass	value	bronchitis, asthma, emphysema
	causes	value	dyspnea
causes			
	instance-of	value	causal-relation, uncertain-relation, transitive-relation
	has-inverse	value	caused-by
	used-to-describe	value	physiological-state, disease
	has-value-type	value	finding, physiological-state

Figure 4: Example frames

A model of concepts defined by their inter-relationships constitutes a semantic network. In order to be able to express each concept as a separate object with particular properties, value types, and values, a frame-based description formalism is used. Frames are objects, and a frame representation system is an object-oriented approach to knowledge representation, originally intended to capture stereotypical concepts and situations [36]. A frame is an object which consists of a list of *slots* that define a concept by relating it to other concepts. A slot consists of a slot name and a slot value.

Referring to the semantic network view, a frame corresponds to a network node, a slot name corresponds to the name of the link going out from the node, and a symbolic slot value corresponds to the node at the other end of this bi-relational link.

Slots are typed according to what role the slot value serves. Slot value types are called *facets*. Typical facets are regular values, default values, value constraints, procedural definitions. Three example frame are shown in Figure 4. Note that explicit procedures, heuristic rules, and semantic relations are regarded as concepts as well, and represented as frames.

A set of basic inference methods enables the interlinked frames to be interpreted as knowledge. Abductive inference - often referred to as "inference to the best explanation" [24] - is the basic type of inference in CreekL. Property inheritance, constraint propagation, frame matching, and plausible concept chaining constitute the main inference methods. Inheritance is not absolute, however, and unless otherwise specified an inherited property may get overridden by a specific, local property (default inheritance). Property inheritance may be regarded as an inference method for retrieving a value given a particular concept. Frame matching, on the other hand, infers a matching concept given one or more property values. Constraints are specified as facets and used to check possible values, to ensure storage of legal values, and to infer values if a local value is unknown. The "value-class" facet, for example (see Figure 4), may be used to infer the superclass of a slot value if the local value is missing.

In this section we have briefly presented and exemplified the *knowledge level analysis* approach to systems modeling. We have also pointed out the problem of going from a knowledge level model to a symbol level implementation. One way to do this is to relax the highly top-down oriented development methodology and incorporate a more iterative knowledge modeling process. A useful tool in this process is an expressive, object-oriented modeling language such as CreekL. To emphasize on methods that enable systems to learn from experience is a way to achieve a continuos, iterative development, where a system adapts and evolves as it is being regularly used. In the section to follow we will focus on a method that seems particularly promising in this respect: Case-based reasoning. A case-based approach is suitable for the type of integrated system we have previously argued for, since it provides a means to develop systems that may gradually evolve from information systems (where the user do the interpretation and reasoning from cases) to knowledge-based systems (the computer do - part of - the case interpretation and reasoning).

6. Integrating information and knowledge - a case-based approach

6.1. Case-based decision support

The type of integration we aim at should combine the functionality of a clever assistant with that of a competent discussion partner. A decision support system should be able to

- provide the user with the right information when it is needed,
- provide suggestions and criticism to the user in his decision making,

- learn from its own experience and continually adapt to its environment.

The system's job is to increase the overall quality and effectiveness of the user's work. Therefore, it is the *total problem solving ability* of the human user and the computer - in cooperation - which is the factor to optimize, not that of the computer system itself. For many applications the information system part would be the one to emphasize, while the knowledge-based part will be restricted to handle just a few tasks. The task for the knowledge-based component in an information-oriented system is to handle information in a meaningful and intelligent way, for example by deriving expectations and consequences of the received information. Another important task is information focusing and filtering, i.e. to find the type of information that is *relevant* in a particular context. Given the huge amount of information available for most professional tasks, and the accelerating increase of information, this is currently a problem that concerns many researchers in the information system and AI fields [18].

Since they do not have to solve problems entirely on their own, knowledge-based decision support systems that cooperate and interact heavily with their users are sometimes viewed as simpler systems from an AI point of view, and easier to realize, than AI systems in general. This is not quite correct, however, since a particular requirement is placed on these systems: They have to model and represent their knowledge and information (i.e. knowledge and information for which the frame of reference is the system) in a way that makes it intelligible to a human user. Facilities for explaining its reasoning chain and its choices, for example, requires that a system's semantic and pragmatic interpretation of represented terms is close to how a human user interprets them. This is difficult to realize unless we apply methods that explicitly represent the terms that a system is to reason about in a way that captures the appropriate context of these terms. A plain neural network approach, for example, does not satisfy this requirement. Here the input and output terms are the only concepts that are communicated, and there is therefore no way that an explanation facility, for example, can be directly realized [14]. (What we refer to here is neural networks as a modeling paradigm. A neural network may also be used to *implement* an explicit concept-representation system, but that is a different issue.) This is a requirement which is also hard to fulfil in a first-order predicate logic approach, where the basic semantics of terms is defined through strict deductive inference (modus ponens) only. This leads to difficulties in capturing the semantics and pragmatics of real world concepts, such as polymorphic and prototypical concepts and relationships [11]. A more flexible, procedural semantics is a way out of this problem, although the lack of an explicitly defined semantics may lead to problems.

The most common way to realize knowledge-based decision support systems has been, and still is, as rule-based systems. Many such systems have been fielded and are doing their job well. Some well known problems, however, include their brittleness - i.e. their inability to deal with problems not pre-anticipated and captured in the rule set, their insufficient explanation capability, and their maintenance difficulties [32]. Second generation expert systems [20] try to cope with the brittleness problem by providing deeper and more principled knowledge in addition to rules. This does not help with the knowledge maintenance problem, however, which is how to update and modify the knowledge base as the system is being used and experience is gained. This is serious because a system then easily gets a static character, i.e. changing knowledge content is something that should be avoided. What is needed in integrated systems is a much more dynamic view: Information is added and gets modified continually, and as information changes, knowledge will change. Human beings usually learn something by integrating new information into existing. If we want tightly integrated, cooperative decision assistants, our computer systems should have this ability as well. Case-based reasoning (CBR) [47, 29, 7], is an approach in this direction. It enables systems to reason by retrieving and reusing previous problem cases, and to learn by updating their case memories after each new experience.

A case-based system solves a problem by trying to 'remember' a previous similar case, and then reusing the solution from that case - possibly after modification. By retaining the problem just solved as a new case, or by other modifications to the case base, the system learns from its experience. Figure 5 illustrates a simple model of the CBR cycle. Learning is facilitated in a case-based system, since learning by retaining new cases is much easier than the updating of general knowledge. The primary knowledge of a case-based system can be viewed as a large memory of past cases. While a rule-based system derives conclusions by applying a set of *general rules*, a case-based system derives its conclusions by retrieving and adapting *specific cases*. Previous experience is available in its most direct form, instead of as abstracted and generalized associations. A case will typically contain a description of the previous problem and its solution, but it may also contain other items, such as a justification of the solution, an explanation of it, the results of having applied it, etc. Negative experiences as well as positive ones may be kept, in order not to repeat earlier mistakes and to reuse previous successes.

Cases are particularly interesting from an integrated systems point of view: A case description as such can be viewed as data, ready to be interpreted by whatever agent has the ability to do so. For a human decision maker a set of previous cases in a computerized case base is information, and provides a means to gather and organize specific information that belong together. Cases which are part of the human decision maker's own memory can be used as knowledge in a decision process - through a human case-based or analogy-based reasoning method. Cases become information and knowledge by interpreter mechanisms (of humans or computers) that are able to sufficiently understand the items of a case description, to use this understanding in order to identify similar cases, to adapt a previous case to the present context, etc. Case retrieval, reuse and learning may also be regarded as knowledge-based processes in their own right, where a body of general and often rather deep knowledge is utilized as part of the case-based reasoning method [6].

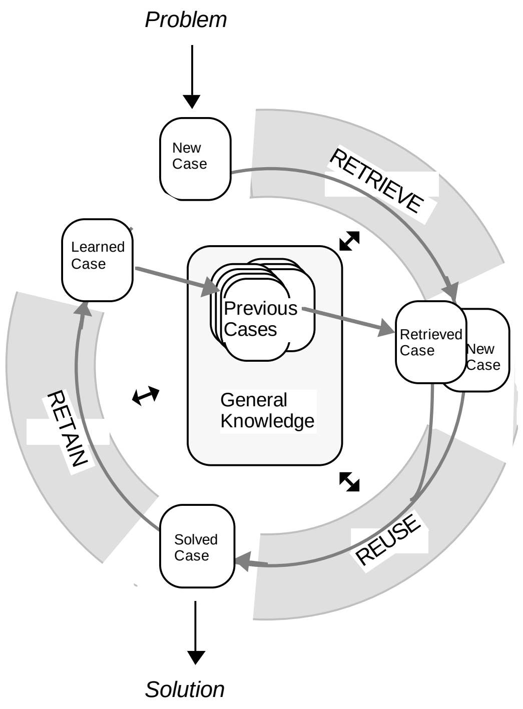

Figure 5: Main steps of the case-based reasoning cycle

The next section exemplifies the case-based approach, by briefly reviewing the system architecture called CREEK (Case-based Reasoning through Extensive Explicit general Knowledge). This architecture also incorporates models of general knowledge as support for, and alternatives to, the case-based methods. In the final section of the section, case-based architectures for integrated systems are discussed with respect to the different types of functionalities for active decision support systems.

6.2. CREEK - a case-based system architecture

CREEK [4, 7] contains, at the top level, four modules integrated within a common conceptual basis as depicted in Figure 6. Each module represents a particular sub-model of knowledge or information. The four modules are an object-level domain knowledge model, a strategy level model (for example a model of diagnostic problem solving), and two internal meta-level models - one for combining case-based and other types of reasoning, and one for sustained learning. CREEK integrates problem solving and learning into one functional architecture. The user is able to interact with the system in all its phases of problem solving and learning.

The CREEK architecture is targeted at building knowledge-based system, hence its submodules are usually interpreted as knowledge and information modules - where the frame of reference is the computer. This is the perspective taken in this overview, since the extension from a knowledge-based systems architecture to including the information system perspective is simpler than the other way around. For the purpose of integrated systems, what will be referred to below as knowledge modules may be viewed as modules of information types where the user is the knowledgeable interpreter and the frame of reference (see section 3.3.2).

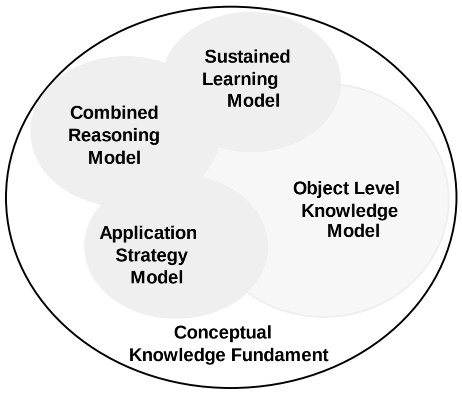
Sustained Learning Model
Combined Reasoning Model
Object Level Knowledge Model
Application Strategy Model
Conceptual Knowledge Fundament

Figure 6: The knowledge modules in CREEK

Previously experienced cases and general knowledge of domain relationships are held in the object level knowledge model. The other models contain general knowledge in the form of concept models and/or rules. A use of cases also for control level reasoning is interesting (see [31] and [44]), but has not been explored within the CREEK architecture. It is important to note that all the concepts in this way get 'glued together' in a unified model. Diagnosis task concepts in medicine, for example, such as "symptom" and "diagnostic-hypothesis" (part of the application strategy model), and learning task concepts, such as "case-indexing" and "failure-generalisation" (part of the sustained learning model), are defined within the same unified representation structure as general domain concepts like "appendicitis" and "fever", and case-related domain terms as "Patient#123456" and "current-radiation-dosage" (which all are part of the object level knowledge model).

All types of knowledge and information are encoded in the frame-based representation language CreekL, briefly summarized at the end of the previous main section. Cases are separate object structures integrated into the common knowledge network. This is illustrated in figure 7, where cases are 'pulled out' of the semantic network structure for clarity, and where a few of the links from cases into the general model are indicated. All symbolic terms that are part of a case description - and object terms as well as relations - have a concept definition in the knowledge network.

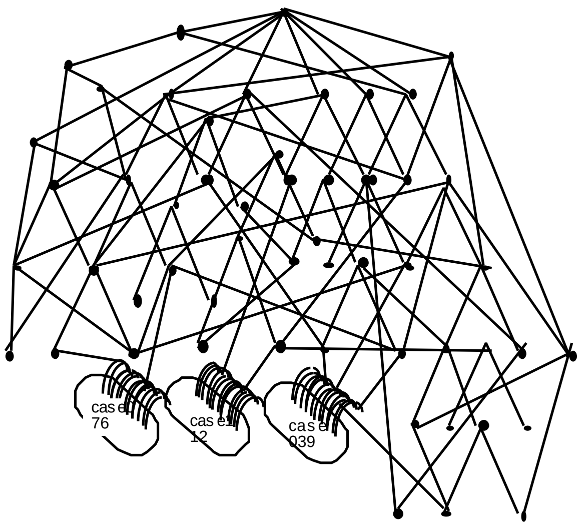

Figure 7: Integrating cases and general knowledge

Below is an example case structure as described in a CreekL frame (the a.b.c notation means frame a's slot b's value c). A case can be a rather complex structure, since it is often useful not only to retain the association between a set of input findings and a solution, but also how the solution was derived, for example in the form of an explanation path in the knowledge model justifying a conclusion (see the has-diagnosis-explanation slot, where a partial explanation is illustrated as a nested list structure).

The part shown explains how myocardial ischemia may lead to increased heart rate reserve, via their relation with 'poor effort', which is an intermediate state in the diagnostic model and a concept node in the network. The differential cases are closely related cases, but with different solutions, and tried out if there is only a weak match between a new problem and case-334. The diagnosis of the case shown was obtained by use of case-152, and no modification of that solution was needed.

```
case-334
   instance-of               value      diagnostic-case
   has-status                value      diagnosis-accepted diagnosis-not-confirmed
   has-relevant-finding      value      mild-obstructive-pattern
                                        PaO2-at-rest.has-value.low
                                        heart-rate-reserve.has-state-change.increased
                                        diffusion-capacity.has-value.decreased
                                        ---
   has-diagnosis             value      myocardial-ischemia
   has-explanation           value      (myocardial-ischemia.associated-with.poor-effort
                                             poor-effort.indicates
                                          .(heart-rate-reserve.has-state-change.increased))
                                        ---
   has-differential-case     value      case-129   case-52  case-20  case-258
   diagnosis-based-on        value      case-152
   diagnosis-modified-by     value      none
```

The interpreter in CREEK contains a three-step process of 1) activating relevant parts of the semantic network 2) generating and assessing (explaining) derived information within the activated knowledge structure, and 3) focusing towards and selecting a conclusion that conforms with the goal. This *activate-explain-focus* cycle is a general mechanism that will normally have to be specialized and instantiated for a particular goal and task type.

The process of remembering (retrieving) and reusing previous cases may be used extensively in all three steps, but a more typical pattern is to use general knowledge in the activation step, cases for generating hypotheses and in parts of the assessment process, and a limited number of cases together with general knowledge for the focusing step. If no case can be found which matches the problem description closely enough, rule-based reasoning will be tried. Finally, as a last resort, an attempt is made to solve the problem entirely by use of the deep knowledge model. See Figure 8.

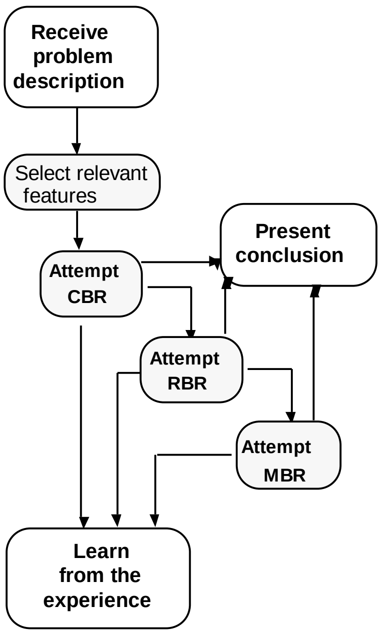

Figure 8: Combined reasoning in CREEK
CBR= case-based reasoning, RBR=rule-based reasoning, MBR=(deep) model-based reasoning.

The general algorithm of the case-based reasoning process in CREEK, i.e. an expansion of the "Attempt CBR" box in Figure 8, is illustrated in Figure 9.

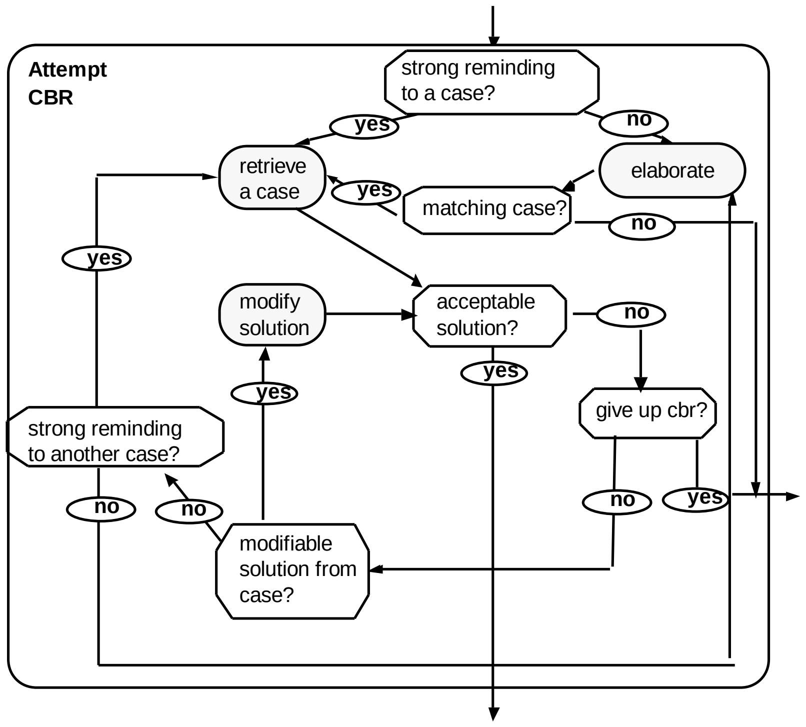

Figure 9: Case-based reasoning in CREEK

CREEK learns from every problem solving experience. If a successful solution was copied from, or just slightly modified on the basis of, a previous case, the reminding to that case from each relevant feature is strengthened. No new case is stored. If a solution was derived by significantly modifying a previous solution, a new case is stored and difference links between the two cases are established. A new case is also created after a problem has been solved from rules or from the deeper knowledge model alone.

The main target for the learning process in CREEK is thus the case base. But a system may also update its general knowledge through interaction with the user during problem solving. Since heuristic rules are represented within the conceptual model, they are available for the same tasks as the conceptual model in general. A rule is viewed as a shallow type of relationship, and may also be used to support case-based problem solving as well as learning. Even if the explanatory strength of a shallow relationship in general is low, it may add to other explanations for the same hypothesis and, thus, contribute to a justification of a hypothesis.

Compared to other case-based reasoning methods, the problem solving and learning methods of CREEK relies more heavily on general knowledge as support for the case-based processes. These processes are *explanation-driven* [6], i.e. subprocesses such as the extraction of relevant features of a problem, assessment of similarity between cases, decisions on whether and how a case solution may be modified for a new problem, and the identification of what to learn from a case just solved, are all depending on sufficiently strong explanations produced as support by the general domain knowledge. In addition, the user is asked for support or confirmation when needed.

6.3. Modes of integration

As previously mentioned, the processes of reusing and retaining case knowledge can be viewed as human reasoning processes as well as computer-based ones. For a human user a case base is an extended memory of previous episodes and situations. What use can an architecture such as CREEK have for designing and implementing integrated decision support systems? A case-based architecture for integrated systems enables several modes of integration, depending on the role of cases within the computer system:

1. **Cases are data for the computer system.**

   This is the simplest mode, where the system does not do case-based reasoning as such, but applies case-based and other knowledge-based methods for case indexing and retrieval only. Since the system views cases as data only, it does not have knowledge of the items that describe case contents, only the items that index the cases.

   This integration mode is close to a knowledge-based information retrieval system, where the data base contains a collection of previous cases. The strength of computers as data managers and information handlers (where the frame of reference for information is the user) is combined with the strength of human beings for intelligent decision making [28]. Help desk systems, a fast growing application area for case-based reasoning exemplify this type of integration.

   These systems will not be able to learn, in the sense of updating its knowledge by retaining new cases, since it is not able to interpret *case contents* as information or knowledge. However, a weak type of learning can be said to occur, since the systems will have the necessary knowledge about the *case indexes* in order to appropriately integrate new cases into the index structure of the case base.

2. **Cases are information for the computer system.**

   This mode of integration requires knowledge in the computer that is able to interpret and utilize case contents as information. Any knowledge-based system approach will in principle do, for example a rule-based case interpreter, or a deeper model-based one.

   This mode has strong similarities with current approaches to integrated information and knowledge systems, where all information items are related to knowledge models. The difference is that a substantial part of the system's information is organized as cases. Information that belongs together, in the sense of sharing the same local context, is stored together.

   The system learns cases in the sense that it can retrieve and interpret their contents as information input to a future reasoning process.

3. **Cases are knowledge for the computer system.**
   This is the case-based reasoning approach, i.e. the case base is not merely a source of information, but a knowledge base which is actively used in the system's reasoning processes. The full flexibility of viewing a case as data, information, and/or knowledge is therefore achieved. This is advantageous, since new cases containing data that the system is not currently capable of meaningfully interpreting, can at least be kept as data to be interpreted by the user.
   Cases may be the only type of knowledge in the system - as in most current CBR systems - or they may be combined with other knowledge types - as in CREEK.
   These systems exhibit learning in the full sense, since they incorporate new cases in a way that makes them immediately ready to be used in the solving of new problems.

From these different modes, we see that a case-based system architecture facilitates a gradual transformation from a pure information system ('mode 0'), through a mode 1 and/or mode 2 system, to a full-fledged mode 3 system. In this way a system will always have its data available in a non-generalized form, and their active use can be incrementally put into effect by adding interpretation and reasoning capabilities to the system as more cases are acquired, and as the use of the system identifies what active decision support users really want. A major strength of the approach is the combination of the automatic learning mechanism inherent in case-based reasoning, with manual procedures for iterative system development.

7. Conclusion

This paper started out with the aim of achieving a better understanding of the fundamental issues underlying the integration of data, information, and knowledge in computer-based decision-support systems. A model defining the core terms within such a perspective was presented, and used as the basis for discussing appropriate analysis and modeling methods for systems development. A basis for achieving the synergy we want is found within recent developments of AI. At the conceptual level, recent progress in knowledge level modeling provides us with a modeling perspective and a set of techniques for implementation-independent description of systems. At the design and implementation level, an object-oriented knowledge representation approach enables us to capture the richness of a dense and multi-relational type semantic network. Case-based reasoning provides a way to solve problems by reusing previous specific experience, and to learn from each such problem solving session. Real world learning usually takes place at the 'fringe' of what is already known, and our approach assures a strong guidance to the learning process both from the system's existing knowledge and from its interaction with its environment. This enables a system to gradually refine and extend its domain knowledge, and to become an increasingly active partner for a human decision maker.

As a feedback to AI research, the view of intelligent systems as user-interactive agents that integrate database, information systems and knowledge-based systems functionalities, provides a focus which guides research in a sound way, namely in the direction of open, situated systems, embedded within their environments of operation.

So far, the principal advantages of a case-based approach to incremental development and evolution of integrated information and knowledge systems have not been met by other known approaches. It is a challenge for future research to further investigate in what way the advantages can be realized and brought into modelling and development methodologies for integrated systems. What we have presented here is a framework within which this discussion hopefully can be conducted in a fruitful and productive way.

Acknowledgements

Klaus-Dieter Althoff, Dietmar Janetzko, and Inge Nordbø provided helpful comments on earlier drafts of this paper.

References

[1] G. Aakvik, A. Aamodt, and I. Nordbø, A knowledge Representation Framework Supporting Knowledge Modelling. *Proceedings EKAW-91*, Fifth European Knowledge Acquisition for Knowledge-based Systems Workshop, Crieff, Scotland, May 1991.

[2] A. Aamodt, Knowledge-intensive case-based reasoning and learning. *Proceedings of ECAI-90*, Ninth European Conference on Artificial Intelligence, Stockholm (August 1990) 1-6.

[3] A. Aamodt, A computational model of knowledge-intensive problem solving and learning, In: B. Wielinga, J. Boose, B. Gaines, G. Schreiber, M. van Someren, *Current trends in knowledge acquisition* (IOS Press, Amsterdam, 1990) 1-20.

[4] A. Aamodt, *A knowledge-intensive approach to problem solving and sustained learning*, Ph.D. dissertation, University of Trondheim, Norwegian Institute of Technology, May 1991. (University Microfilms PUB 92-08460)

[5] A. Aamodt, A case-based answer to some problems of knowledge-based systems. *Proceedings of SCAI-93, Fourth Scandinavian Conference on Artificial Intelligence*, Stockholm, May 1993 (IOS Press, 1993) 168-182.

[6] A. Aamodt, Explanation-driven case-based reasoning, in: S. Wess, K. Althoff, M. Richter (eds.):*Topics in Case-based reasoning*. Springer Verlag, 1994, pp 274-288.

[7] A. Aamodt and E. Plaza, Case-Based Reasoning: Foundational issues, methodological variations, and system approaches, *AI Communications*, Vol.7, No.1, March (1994) 39-59.

[8] K-D. Althoff, Machine learning and knowledge acquisition in a computational architecture for fault diagnosis in engineering systems. *Proceedings of the ML-92 Workshop on Computational Architectures for Machine Learning and Knowledge Acquisition*. Aberdeen, Scotland, July 1992.

[9] A.J. Ayer, *The origins of pragmatism*. Ch. 3, Pierce's philosophy of science, A. The three kinds of reasoning. Freeman, Cooper & Co., 1968. 63-80.

[10] E.R. Bareiss, *Exemplar-based knowledge acquisition: A unified approach to concept representation, classification, and learning*, (Academic Press, 1989).

[11] L. Birnbaum, Rigor Mortis: a response to Nilsson's "Logic and artificial intelligence", *Artificial Intelligence*, 47 (1-3) (1991) 57-77.

[12] G. Boy, *Intelligent assistant systems*, (Academic Press, 1991).

[13] M.L. Brodie and J. Mylopoulos, *On knowledge base management systems*, (Springer, 1986)

[14] B. Chandrasekaran, A. Goel and D. Allemang, Connectionism and Information Processing Abstractions, *AI Magazine*, (Winter 1988), 24-34.

[15] B. Chandrasekaran, Task-structure analysis for knowledge modeling. *Communications of the ACM*, 35 (9) (Sept. 1992) 124-137.

[16] W.J. Clancey, The frame of reference problem in the design of intelligent machines, In K. VanLehn (ed.), *Architectures for Intelligence* (Lawrence Erlbaum, 1991) 357-423.

[17] P. Coad and E. Yourdon, *Object-oriented analysis* (Yourdon, New York, 1990).

[18] *Communications of the ACM*, Special issue on Information Filtering 35(12) (December 1992).

[19] P. Compton and R. Jansen, A philosophical basis for knowledge acquisition, In J. Boose, B. Gaines, J.G Ganascia (eds.), *EKAW 89, Third European Workshop on Knowledge Acquisition for Knowledge-Based systems* (ECCAI/AFCET/ARC, Paris, 1989) 75-89.

[20] J-M. David, J-P. Krivine and R. Simmons (eds.), *Second generation expert systems* (Spinger, 1993).

[21] H. Gallaire, J. Minker and J.-M. Nicolas, Logic and databases: A deductive approach, *ACM Computing Surveys* 16 (2) (June 1984) 153-185.

[22] C. Gane and T. Sarson, *Structured systems analysis: Tools and techniques*, (Prentice-Hall, Englewood Cliffs, 1979).

[23] C.G. Hempel, *Aspects of scientific explanation.*, (Free Press, New York, 1965).

[24] J. Josephson and S. Josephson, *Abductive inference, computation, philosophy, technology*, (Cambridge University Press, 1994).

[25] C. Jullien, N. Shadbolt and B. Wielinga (eds.): *Acknowledge Project Final Report*. ACK-CSI-WM-DL-007, Cap Gemini Innovation, 1992.

[26] L. Kerschberg (ed.), *Expert database systems*, *Proceedings from the Second International Conference* (Benjamin/Cummings, Redwood City CA, 1989).

[27] G. Klinker, C. Bohla, G. Dallemagne, D. Marques and J. McDermott, Usable and reusable programming constructs, *Knowledge Acquisition*, 3 (1991) 117-136.

[28] J. Kolodner, Improving human decision making through case-based decision aiding. *AI Magazine* (Summer 1991) 52-68.

[29] J. Kolodner, *Case-based reasoning* (Morgan Kaufmann, 1993).

[30] D. Lenat and R. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project* (Addison-Wesley, 1989).

[31] B. Lopez and E. Plaza, Case-based planning for medical diagnosis, In: *Methodologies for intelligent systems: 7th International Symposium, ISMIS '93*, Trondheim, June 1993 (Springer 1993) 96-105.

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

[33] A. McIntyre, *KREST Manual 2.5*, Knowledge Technologies n.v. (Brussels, 1993).

[34] M. Manago, K.-D. Althoff, E. Auriol, R. Traphoener, S. Wess, N. Conruyt, and F. Maurer, Induction and reasoning from cases. in: M. M. Richter, S. Wess, K.-D. Althoff & F. Maurer (eds.), *Proc. 1st European Workshop on Case-Based Reasoning (EWCBR-93)*, SEKI-Report SR-93-12 (Universität Kaiserslautern, 1993) 313-318.

[35] J. Minker, Perspectives in deductive databases, *Journal of Logic Programming*, 5 (1988) 33-60.

[36] M. Minsky, A framework for representing knowledge. In P. Winston (ed.), *The psychology of computer vision.* (McGraw-Hill, 1975) 211-277.

[37] T. Mitchell, J. Allen, P. Chalasani, J. Cheng, O. Etzoni, M. Ringuette and J.C. Schlimmer, Theo: A framework for self-improving systems, In K. VanLehn (ed.), *Architectures for Intelligence* (Lawrence Erlbaum, 1991) 323-356.

[38] K. Morik, Sloppy Modelling, In: K. Morik (ed), *Knowledge representation and organization in machine learning*, Lecture notes in artificial intelligence, 347, (Springer, 1989) 107-134.
[39] A. Newell, Physical symbol systems. *Cognitive Science* 4 (1980) 135-183.
[40] A. Newell, The knowledge level, *Artificial Intelligence*, 18 (1982) 87-127.
[41] M. Nygård and A. Aamodt, Road transport informatics; Conceptual framework and technological components, in: *Proceedings IEEE VNIS '93, 4th IEEE International Conference on Vehicle Navigation and Information Systems.* Ottawa, Canada, October 12-15 1993. pp 281-286.
[42] M. Nygård, Reflections on state-of-the-art within RTI / IVHS, in: *Proceedings IEEE VNIS '94, 5th Intern. Conf. on Vehicle Navigation & Information Systems*, Tokyo, Japan. (1994).
[43] E. Plaza, A. Aamodt, A. Ram, W. Van de Velde and M. Van Someren, Integrated learning architectures, in: Proceedings of the ECML-93, European Conference on Machine Learning (Vienna, 1993) 429-441
[44] E. Plaza and J-L. Arcos, *Reflection, memory, and learning.* CSIC/CEAB, Report de Recerca IIIA 93/2. (Blanes, 1993)
[45] B. Porter, R. Bareiss and R. Holte, Concept learning and heuristic classification in weak theory domains. *Artificial Intelligence*, 45 (1-2) (September 1990) 229-263.
[46] W.J. Rapaport, Cognitive Science, In A. Ralston and E.D. Reilly (eds.), *Encyclopaedia of Computer Science and Engineering*, 3rd edition (Van Nostrand Reinhold, New York, 1993).
[47] C. Riesbeck and R. Schank, *Inside Case-based reasoning.* (Lawrence Erlbaum, 1989).
[48] J. Sandberg and B. Wielinga, How situated is cognition?, In *Proceedings of IJCAI-91*, (Morgan Kaufmann, 1991) 341-346.
[49] G.A. Silver and M.L Silver, *Systems analysis and design* (Addison-Wesley, 1989).
[50] E. Smith and D. Medin, *Categories and concepts* (Harvard University Press, 1981).
[51] B. C. Smith, Reflections and semantics in a procedural language. *MIT Technical Report*, LCR-TR-272 (1972).
[52] L. Steels, Components of expertise, *AI Magazine*, 11 (2) (Summer 1990) 29-49.
[53] L. Steels and B. Le Pape, *Enhancing the knowledge engineering process: Contributions from ESPRIT*, (North-Holland, Amsterdam, 1992).
[54] L. Steels, The componential framework and its role in reusability, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems* (Spinger, 1993) 273-298.
[55] J. Sticklen, Problem solving architecture at the knowledge level, *Journal of Experimental and Theoretical Artificial Intelligence* 1 (4) (1989) 233-271.
[56] V.C Storey, A selective survey of the use of artificial intelligence for database design systems, *Data & Knowledge Engineering* 11 (1993) 61-102.
[57] G. Strube, The role of cognitive science in knowledge engineering, In: F. Schmalhofer, G. Strube (eds.), *Contemporary knowledge engineering and cognition: First joint workshop, proceedings*, (Springer 1991) 161-174.
[58] I. Sølvberg, I. Nordbø and A. Aamodt, Knowledge-based information retrieval. *Future Generation Computer Systems*, 7 (1991/92) 379-390.
[59] A. Sølvberg and D.C. Kung, Information systems engineering (Springer, 1993)
[60] G. Tecuci, A multistrategy learning approach to domain modeling and knowledge acquisition. *Machine Learning - EWSL-91, European Working Session on Learning, Proceedings.* Porto, March 1991, Springer Verlag 1991, 14-32.
[61] P. Thagard, *Computational Philosophy of Science*, (MIT Press/Bradford Books, 1988).
[62] P. Thagard, Explanatory Coherence, *Behavioural and Brain Sciences* 12 (1989) 435-467.
[63] J.D. Ullman, *Principles of database and knowledge-base systems, Vol.II: The new technologies,* (Computer Science Press, Rockville MD, 1989).
[64] W. Van de Velde, *Learning from experience*, Ph.D Dissertation, Vrije Universiteit Brussel, Artificial Intelligence Laboratory. Technical Report 88-1, (Brussel, 1988).
[65] W. Van de Velde, Issues in knowledge level modelling, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems* (Spinger, 1993) 211-231.

[66] K. VanLehn (ed.), *Architectures for Intelligence* (Lawrence Erlbaum, 1991).
[67] K. Van Marcke, *KRS: An object oriented language,* Ph.D. Dissertation Artificial Intelligence Laboratory, Vrije Universiteit Brussel, AI-MEMO no 88-9, (Brussels, 1988).
[68] J. Vanwelkenhuysen and P. Rademakers, Mapping a knowledge level analysis onto a computational framework. *Proceedings of ECAI-90*, Ninth European Conference on Artificial Intelligence, Stockholm, August 1990. (ECCAI, 1990) 661-66.
[69] A.H. Vera and H. Simon, Situated action: A symbolic interpretation, *Cognitive Science* 17 (1) (1993) 7-48.
[70] B.J. Wielinga, A.Th. Schreiber and J.A. Breuker, KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 4(1), 1992.
[71] B. Wielinga, W. Van de Velde, G. Screiber and H. Akkermans, Towards a unification of knowledge modelling approaches, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems* (Spinger, 1993) 299-335.
[72] T. Winograd and F. Flores, *Understanding computers and cognition*. Ablex, 1986
[73] L. Wittgenstein, *Philosophical investigations* (Blackwell, 1953) 31-34.