Knowledge maps: An essential technique for conceptualisation

A. Gómez[a], A. Moreno[a], J. Pazos[a], A. Sierra-Alonso [b,*]

[a] *Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, s/n, 28660 Boadilla del Monte, Madrid, Spain*
[b] *Universidad Carlos III. Escuela Politécnica Superior, Avda. de la Universidad No. 30, C/Butarque 15, 28911 Leganés, Madrid, Spain*

Received 23 November 1999; accepted 30 November 1999

Abstract

The process of conceptualisation is a fundamental problem-solving activity and, hence, is an essential activity for solving the problem of software systems construction. This paper first analyses the process of conceptualisation generally, that is, not as applied specifically to software systems, and establishes a general-purpose conceptualisation process, composed of three activities: analysis, synthesis and holistic testing. A proposed instantiation of this framework for the process of conceptualisation in knowledge-based systems (KBS) construction is then presented. The paper focuses on an activity that is frequently overlooked in conceptualisation, that is, holistic testing, and on a technique that is proposed to address this phase, known as the knowledge map (KM). This technique integrates the static and dynamic perspectives of the reasoning employed by the expert to solve the problem. This paper discusses the foundations and an application of this technique.

1. Introduction

The four main components of software development can be called "the four Ps": Products, Process, Persons, Problem, and not necessarily in this order. As discussed at length elsewhere [1], emphasis was initially placed on the product, was later switched to the process, and subsequently turned to people; however, little, and by no means enough, emphasis was placed on what we see as a necessary condition for good software development: the problem, its understanding and conceptualisation [8]. This failure to address the problem has probably strongly influenced the difficulties encountered in software development.

According to an age-old aphorism, a well-defined (or alternatively, correctly modelled) problem is half solved. This is evident. What is not so evident, however, is how to correctly define or model a problem. Indeed, finding a means to model a problem in the best possible manner is in itself a complex technical issue. Actually, it is very difficult to formulate complex problems fully and

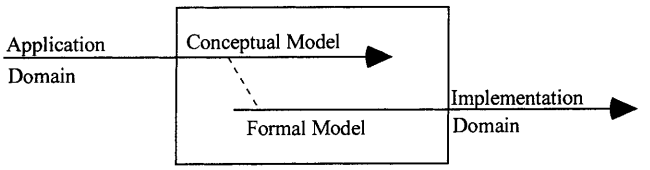

Fig. 1. The essential software Process.

accurately. So, there is a growing tendency to think that a problem is seldom what it appears to be at first sight. This is because the initial modelling is not, though it should be, valid, bearing in mind that the problem is subject to continual changes and gradual refinement during the problem-solving process. Often we do not know whether or not the problem has or has not been correctly formulated until the end. When we go about setting out a problem, it is because we already have an idea about how it can be solved. This is the difference between a poorly defined problem and critical sophism. The solutions dictate the problems, and there is no a priori rule for finding solutions. This once led Gauss to say in reference to one of his brilliant mathematical discoveries that now he had found the solution, all he had to do was to ascertain the logical process that led to it.

Blum [2] identifies a transformation from the identification of a need in-the-world into a set of computer programs that operate in-the-computer within software system-based problem-solving process. Fig. 1 shows this transformation based on two models – conceptual and formal models (as they are termed by Blum) – which represent the problem raised and the software system built, respectively. This paper concerns conceptual models, that is, the models used for representing the user problem, and, in particular, a technique, the knowledge map (KM), as an aid for conceptualisation in knowledge-based systems (KBS) development.

Section 2 defines what conceptualisation and conceptual model mean, as well as a general-purpose process for performing this task, composed of three phases: analysis, synthesis and holistic testing. Section 3 presents a proposed instantiation of the above phases for the process of KBS conceptualisation. Traditional KBS construction methodologies have notably failed to pay attention to the holistic testing phase, and, hence, Section 4 centres on a technique for addressing this phase, called the KM. Section 5 presents an example of KM construction. Section 6 summarises how KMs can benefit the conceptualisation process, from the viewpoint of both their construction by the knowledge engineer and of their validation by the expert. Finally, Section 7 summarises the conclusions drawn.

2. Conceptual modeling

Before focusing on conceptualisation as a phase of software systems construction, this section analyses conceptualisation as a means of solving any problem. Thus, the following sections discuss some considerations concerning problem conceptualisation and its role in problem solving. A brief description is also given of the components to be taken into account during problem conceptualisation. Finally, a general-purpose process is presented, which describes the phases involved in any conceptualisation.

The importance of conceptualisation in any problem-solving activity is never stressed enough. This was perhaps best expressed by Chesterton [3], who claimed that the worst thing was not being unable to find the solution, it was not being able to see the problem. Indeed, human understanding of reality is determined by two interactive components. The first is the sensory component and its amplifications, equipped with systems to gather, either directly or indirectly, information from the environment. The second is the conceptual component that extracts from all the above information the relevant concepts for solving the problem at hand, their internal relations and the reasoning used to arrive at the right conclusions. So, *conceptualisation is modelling by the problem solver*. This modelling is represented by means of a conceptual model. This means that there are many ways of conceptualising, that is, many conceptual models for a problem. For example, if we were to calculate the height of a building using a barometer, the problem could be conceptualised in many different ways, ranging from calculating the pressure on the ground floor and on the roof, through measuring it in barometer units, to dropping the barometer from the roof and measuring how long it takes the noise it makes when it hits the ground to reach the roof, etc., and they are all valid. Alternatively, a particular conceptualisation of a problem may make it impossible to express the characteristics of and, hence, to solve the problem. This is precisely the case when light is conceptualise d as waves to express reflection and as particles to describe refraction. Again, a conceptualisation may not prevent given types of knowledge from being expressed, but it may make it more difficult, as is the case of the geocentric conceptualisation of solar movement. This ontological promiscuity, making several conceptualisations apt for one and the same problem, means that only the alternative most useful for the purpose at hand should be retained.

All the conceptual models that can be built for a problem, that is, any of the possible conceptualisations, must adhere as strictly as possible to the following four methodological rules of the Cartesian method [5]:

1. *Rule of evidence*: Never accept anything as true which is not clearly and distinctly seen to be so. That is, take care not to act hastily or with prejudice, and admit only those judgements that appear to the mind so clearly and distinctly as not to afford the least doubt.
2. *Rule of analysis*: Divide each of the difficulties under examination into as many parts as possible.
3. *Rule of synthesis*: Order knowledge, beginning with the objects that are the simplest and easiest to know and so proceed, gradually, to knowledge of the more complex, also somehow ordering knowledge that is naturally devoid thereof.
4. *Rule of proof*: Always make lists and inspections that are exhaustive enough to assure that there are no omissions.

2.2. Elements of conceptualisation

We can consider a conceptualisation as composed of a threesome: concepts, relations and functions. The following sections give a detailed description of these components.

2.2.1. Concepts

The notion of *concept* is used in a fairly broad sense. It refers to both concrete (objects, persons, etc.) and abstract (number 2, set of all integers) things, elemental concepts (electron)

or compounds (atoms), fictitious (unicorn) or real (detective) entities. Concepts are the mental building blocks used by human beings to think. Indeed, they are a sort of mental representation of things or experiences. Stripped of the ability to form concepts, each individual object (each shoe, each door, etc.) would have to have a name that would differentiate from any other in the world, and it would be impossible to think. Fortunately, to conceptualise is to overlook irrelevant differences, and the concept car, for example, is valid for naming each of the millions of cars there are on the road. So, a concept is anything about which something is to be said and could, therefore, also be the description of a task, function, action, strategy, reasoning process, etc. The set of all the concepts involved in a problem is referred to as the universe of discourse.

A problem solver usually works with a small number of concepts. So, if too many concepts emerge during conceptualisation, their number should be constrained. Two, fairly well-known, powerful techniques – generalisation and abstraction – are used to reduce the number of concepts. Formally, generalisation is the move from considering a concept to considering a set containing that concept or the move from considering a small to a more comprehensive set of concepts containing the smaller one. Abstraction basically consists of retaining only the concepts of what is relevant to problem solving, running the risk, of course, of omitting details that could be important. Modelling is an example of abstraction.

People do not only invent objects, they also create concepts that make the job of problem solving easier. The creation of units of measure, such as litres of petrol consumed per 100 km, passenger/kilometre per litre or kilometres per full tank are just some examples of the above. These are used to correctly set out and later solve a problem. Indeed, suppose you want to acquire vehicles to transport members of an institution from their homes to their place of work and vice versa. The problem would be formulated in such a manner as to answer the following question: Which is the vehicle with the lowest consumption that can be used to transport the employees of the institution? Another conceptualisation, correct in this case, would be to first consider the number of persons for transportation and then determine the most efficient means of doing so. For example, if a lot of employees lived near to each other, the different vehicle types should be evaluated bearing in mind passengers/kilometres per litre. That is, to purchase vehicles with a higher seating capacity, even though they consume more than others with fewer seats that consume less.

Of course, it is not always necessary to create ad hoc concepts to solve a problem. It is sometimes possible to use concepts that exist in other domains, which are transplanted or transferred by analogy. Concepts are transplanted from one domain to another when the above concepts are directly transferred. Such is the case, for instance, of the use of the physical concept of transitory state, which is instantly transferred to the study of certain business management problems. On the other hand, concepts are transferred by analogy when a phenomenon is examined from one viewpoint or perspective and the underlying concerns are viewed from another, as is the case, for example, of the study of living cell mortality and equipment wear.

Indeed, by far the most powerful way of solving problems is to acquaint oneself with the notions and concepts invented by others, which provide tools to conceptualise and solve various problems. For example, by learning concepts of algebra and geometry, concepts that have been refined over the centuries, it is possible to solve a host of problems, which, otherwise, would be very difficult or impossible to solve.

2.2.2. Relations

A concept is such only by virtue of the way in which it is linked and connected to other things, which are also concepts. In other words, the property of being a concept is a property of *connectivity*, a quality that stems from their integration into a sort of network, and this alone [6]. So, concepts are like structural or even topological properties of vast, maze-like, entangled and sticky mental *spaghetti*. That is, the source of the intellectual problem-solving power of concepts is their mutual interconnection.

These interconnections are expressed by means of relations. This means that a relation is an interconnection between concepts in a universe of discourse. In a conceptualisation of part of the world, some relations are stressed and others are ignored. The set of relations in a conceptualisation are called the basic relational set.

Table 1 shows some of the most common relations that emerge when a problem is modelled.

2.2.3. Functions

Functions are special cases of relations, which, owing to their importance and specificity, should be considered separately. Examples of functions are:

Mother-of: Person $\rightarrow$ Woman
Squared: Number $\rightarrow$ Positive-Number
Car_Price: Model $\times$ Year $\times$ No. kilometres $\rightarrow$ Price

A function is, therefore, a type of interrelation between concepts in a universe of discourse, where the value of the last concept is unique for the preceding concepts (for example, the value of the concept Woman is unique for the value of the concept Person, when it is expressed as the relation Mother-of). Although many functions can be defined for a given set of concepts, some functions are usually stressed and others ignored in order to conceptualise part of the world. The set of stressed functions in a conceptualisation is called the basic functional set.

By defining the concepts, relations and functions, the conceptual model also shows the sequence of steps according to which problem solvers will solve the problem in question, the inferences made and how the information, that is, data, news and knowledge, will be processed. This model represents how problem solvers behave when solving the problem in question, describing what information is involved and where, how, when, why and for what purpose it comes into play.

2.3. The process of conceptualisation

As follows from the rules of the Cartesian method discussed in Section 2.1, the phases involved in building the conceptual model of a problem have been traditionally categorised as *analysis* and *synthesis*. During the analysis phase, the modeller has to understand the problem. This understanding is usually gained by decomposing the problem into simpler parts. This knowledge is employed during the synthesis phase to put together the parts obtained during analysis until a picture/modelling of the problem is gained, encompassing all its components and relations and, consequently, its solution. So, analysis can be said to identify the components of the domain, and synthesis to organise these components to form the conceptual model.

Analysis and synthesis are, therefore, two complementary phases. Analysis is a more passive task, where the modeller's activity is confined to observing and decomposing, whereas synthesis calls for a more active attitude, where the modeller has to make a mental effort to structure all the

Table 1
Some of the most common relations in a problem

Relation type	Description	Example
Equivalence	Establishes the equality or equivalence between two apparently different expressions	Profit = Income – Expenses
Taxonomic	Classifies a specific concept as part of a general concept	A is B A can be classified as B, C or D
Structural	Describes how a concept or a system of concepts can be decomposed into parts or subsystems	A is part of B A is composed of B and C A forms part of B A is disjunct of B A is not disjunct of B A and B are mutually disjunct A and B are exhaustive partitions of C A and B are partitions of C
Dependence	Special type of relation ranging from an association, through responsibilities, parenthood, property, etc., to participation	A is part of B A and B are associated to build B A is responsible for B
Topological	Describes the spatial distribution of physical concepts and interconnections between these concepts	A is to the right of B A is above B A is below B A is inside B A contains B A intersects B A is connected to B A is in contact with B
Causal	Describes how given states or actions induce other states or actions	A is the cause of B A needs B A fires B
Functional	Describes the conditions for actions and reactions to take place and possible consequences of the actions	A enables B A needs B A fires B
Chronological	Describes the time sequence in which events occur	A occurs before B A occurs after B A and B occur simultaneously A occurs during B A starts before B ends
Similarity	Establishes which concepts are equal or analogous and to what extent	Valve A is open = admits petrol
Conditional	Defines the conditions in which given things take place	For highly accelerated small masses moving at low speed, use quantum hypotheses
Purpose	Establishes the whys and wherefores of concepts	Codes were created by human beings to transmit ideas

pieces and model the problem. Therefore, analysis and synthesis address the same objects, although the order is reversed such that the end of analysis is the start of synthesis. Analysis is referred to as regressive or backward reasoning, that is, it goes from the whole problem to its

individual components, whereas synthesis is referred to as progressive or forward reasoning, that is, it goes from the individual components to the solution.

The analysis phase is governed by Descartes' rule of evidence, used to determine whether the available knowledge is either known or considered to be true, and the rule of analysis as a guide for performing this phase. Synthesis, on the other hand, is based on the rule of synthesis, as a means of piecing together a solution to the problem on the basis of the knowledge extracted during analysis.

However, as discussed by Crick [4], these two activities alone are not sufficient to understand and model all the points of interest in a problem and its solution: "three main approaches are needed to unscramble a complicated system. One can take it apart and characterize all the isolated bits – what they are made of and how they work. Then one can find exactly where each part is located in the system in erlation to all the other parts and how they interact with each other. These two approaches are unlikely, by themsevles, to reveal exactly how the system works. To do this one must also study the behaviour of the system and its components while interfering very delicately with its various parts, to see what effect such alterations have on behaviour at all levels". This third activity can be referred to as holistic testing. The aim of this activity is to observe whether the behaviour of the model is similar to real-world behaviour. That is, to validate the knowledge reflected in the conceptual model generated during synthesis in order to check that the above knowledge adequately represents the desired/observed behaviour. This, therefore, involves examining how the solution to the problem works before it is implemented, thus analysing how it affects all the parts of the problem. This task is guided by Descartes' rule of proof that ensures that the process is complete.

3. Role of conceptualisation in the development of knowledge-based systems

This paper centres on the process of conceptualisation in knowledge engineering, where the conceptual models represent the problem solver's view of the problem and, particularly, how a particular expert solves the problem at hand. In this case, the conceptual models also determine how the KBS under development is to operate, as the objective of the system is to simulate the behaviour of the above expert. Therefore, the conceptual models act as a specification of what the software system is to do.

Conceptualisation comes between knowledge acquisition and formalisation in the KBS construction process. Knowledge acquisition is the task of gathering information required to build the KBS from any source, where information is taken in its broadest sense, that is, data, news and, especially, knowledge. This information is used during the conceptualisation process to gain a clear understanding of the structure and relationships within a particular system or problem. Finally, the aim of the formalisation stage is to express the conceptualised knowledge of the problem and problem solving as structures that can be used by the computer. These structures are what Blum terms formal models in Fig. 1.

Fig. 2 presents our proposal for addressing the conceptualisation process in KBS. This proposal adopts the concepts discussed in the preceding section: explicit differentiation between the activities of analysis and synthesis during the conceptualisation phase, and the inclusion of holistic testing.

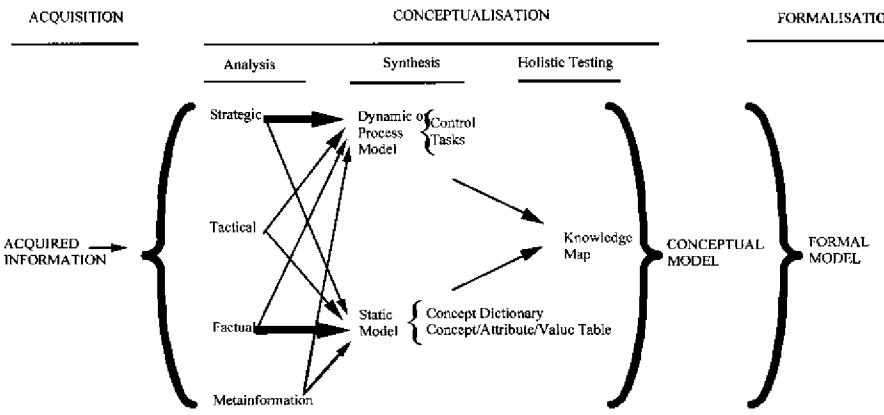

Fig. 2. Analysis and synthesis for conceptualisation.

The objective of analysis is for the knowledge engineer to understand all the parts of the problem solved by the expert at several levels (strategic, tactical and factual). Table 2 describes what actions are to be taken during this activity, and the possible conceptual modelling techniques that can be employed. The three stages described in Table 2 basically involve the following:

1. *Identify strategic knowledge.* Strategic knowledge specifies the steps to be taken to solve the problem. What we are looking for in this stage are the steps into which the task performed by the expert can be divided and the order in which these steps are performed.
2. *Identify tactical knowledge.* Tactical knowledge specifies the inferences to be made in each step of the task. What we are looking for in this stage are the inferences, calculations and uncertainties used by the expert.
3. *Identify factual knowledge.* Factual knowledge defines the structure of the domain and is managed by the knowledge of other levels to get correct inferences. What we are looking for in this stage are the properties, concepts, relations and functions in the domain, used by the expert for tactical or strategic reasoning.

Note that these stages are referred to as "identification" to stress the idea mentioned in the previous section that analysis is an activity of searching, understanding and decomposing, as separate from the composition and interrelation of the parts. When dealing with complex problems, knowledge engineers find it difficult to retain all the information gathered during analysis, and they, therefore, note down this information in rough copies using the format they consider best suited for setting out the information in question. These representations are not yet what we would call conceptual models and have, therefore, been termed intermediate representations in Table 2.

During the synthesis phase, the knowledge engineer has to organise and interrelate the knowledge identified during analysis in a static and a dynamic model, which represent, respectively, the structure and functionality of the problem and how it is solved by the expert. Table 3 describes in detail the actions and possible techniques to be used during the synthesis phase. Section 5 includes an example showing part of the static and dynamic models of a conceptualisation.

Table 2
Analysis phase of conceptualisation

Phase I: analysis	Description	Purposes	Actions	Intermediate representations
Stage 1: Identify strategic information	Identify the steps into which the task can be divided and the order thereof	Define modular steps Establish control flow	1. Identify high-level steps: – Determine the problem-solving steps and their detailed sequence Number those steps – Describe the process 2. Identify the task substeps: – Describe each substep within the step and what to do – Number the substeps – Represent the sequence of substeps – Check whether it is possible to decompose the substeps 3. Define the substeps: – Establish their purpose – Determine the information required to be able to execute each substep – Describe input sources – Describe the target of each output	Logical nets Functional decomposition trees Charts Reasoning step definition
Stage 2: Identify Tactical Information	Identify what happens in each step identified in stage 1	Establish how the inferences are normally used Fully understand how the conclusions are extracted	1. Understand how the conclusions are extracted: – Identify the attributes required – Assign attributes to inferences – Represent this information 2. How and when to react to new information, hat is: – Identify all the end attributes that can be inferred from the firing attribute – Detect basic attributes that act in conjunction with the firing attribute – Establish a standard structure for inferences – Select the intermediate representation that illustrates how the effects are inferred 3. Describe the inference form: – Determine intermediate levels of inference and link them by means of a net	Decision Tables, when there are not many attributes Pseudorules, when there are a lot of different types of attributes Decision trees, when considering certain characteristics in a specified order Inference graphs, if there are clear causal relationships Organisation charts

Table 2 *(Continued)*

Phase I: analysis	Description	Purposes	Actions	Intermediate representations
Stage 3: Identify Factual Information	Identify concepts, properties, relations and functions in the domain of the task performed by the expert	Organise the information gathered about each attribute	1. Create standardised problem attribute definitions: – Identify the attribute name – Determine the cases in which the attribute is important – Establish how the attribute is used	Attribute definition Concept definition Tree diagrams
		Classify which attribute will influence each relationship	2. Classify attributes according to their importance:	
		Who supplies and uses the attributes	– Identify the concept describing each attribute – Group together attributes of similar concepts	
			3. Organise (identify and group) facts that are independent of the case or concept with which they are associated	
			4. Organise the relationships between the above facts	

Table 3
Synthesis phase of conceptualisation

Phase II: synthesis	Description	Purposes	Actions	Intermediate representations
Stage 1: Build the static or structural model	Establish the domain structure and its components	Output a concept/attribute/value table that is as complete and relevant as possible	1. Build a glossary of terms and an explanation as exhaustive and clear as possible	Double-entry tables for – Concepts/Attributes/Values – Concepts dictionary – Set of basic relationships – List of terms and description
			2. Design the concept/attribute/value tables 3. Define the set of basic relationships. That is, define all the relationships between concepts, describing their type and constraints 4. Build the concept dictionary: – Identify the concepts – Output their synonyms and antonyms – Define their function – Establish their source – Expressly state the cross-references – Route to the concept/attribute/value tables	
Stage 2: Build the dynamic or functional model	Define the functions to be performed to solve the problem	Identify the tasks and subtasks establishing a hierarchy and priorities	1. Establish the goals and subgoals and their dependencies and priorities 2. Describe the procedures 3. Define the task and subtask hierarchy	– Dependency trees – Standard description of procedures – Petri nets – Organisation charts

The remainder of the paper concentrates on phase 3, which we have termed holistic testing. Table 4 briefly outlines this process. Our aim is to draw attention to the importance of this activity, not explicitly recognised in current KBS development methodologies, like KADS [9], for example. As stated in Section 2.3, this is a fundamental task, because it means that the conceptualised knowledge can be examined globally and, consequently, its validity and completeness can be assured before it is implemented in the software system. We propose to address this task by constructing a model, called the KM, which integrates the knowledge represented in the static and dynamic models. The objective is to be able to jointly validate the knowledge expressed in both models, which, ultimately, represents the knowledge to be simulated by the KBS. Without this KM, the process of validation to be performed during conceptualisation would be confined to checking the knowledge reflected in the static and dynamic models separately, and there would be

Table 4
Holistic testing phase of conceptualisation

Phase III: holistic testing	Description	Purposes	Actions
Build the KM	Mental map, which somehow encompasses all the above and improves thinking, by means of colours, drawings, etc.	Give an overview of problem solving	1. Start with the concepts that define the problem 2. Add the properties for inferring the goal 3. Add other properties to the map to get the properties of 2 4. Record multiple uses of the elements of the map, especially attributes 5. Check that the map is complete, that is, whether all the attributes involved in the reasoning process are represented 6. Verify the map with problem solvers

no certainty as to whether or not the operation of the entire conceptual model was valid. At present, this certitude cannot be gained until a later stage of development, after the system has been implemented and validated by means of the Turing Test, checking the responses of the system against those of the expert. Any operational invalidity is detected very late on in the development process. If the knowledge engineer were able to detect this type of problems during conceptualisation, this would lead to benefits in terms of development time and costs and KBS quality. This is where holistic testing comes in and, hence, the KM, as a tool for representing the overall operation of the KBS, which can, as a result, be validated early on during the conceptualisation phase.

The following sections discuss in detail how to address KM construction and how this model can benefit conceptual modelling.

4. KM construction process

KM is the name given [7] to a type of mental diagram by means of which complex ideas can be easily and quickly set out in a logical order. KMs are a graphic representation of the connections made by the brain in the process of understanding facts about something. As we will see below, they are built starting with the attribute that defines the problem to then develop a graphical diagram that sets out on paper the manner in which the mind comes up with ideas in the process of understanding.

As mentioned in the preceding section, KMs can be considered in the domain of KBS as a representation that will encompass, both explicitly and implicitly, the static and dynamic models generated during the synthesis phase. The KM ultimately seeks to represent the connections between the properties identified during conceptualisation (static side) and the process of inferring values on those properties (dynamic side).

Rules, and, of course, any other formalisms used to represent dynamic knowledge, can contain uncertainty, either because we are not absolutely sure about how the rule has been represented or because the knowledge represented in the formalisms is uncertain *per se*. In either case, the link or

relation between the properties is represented in the KM in the same manner as the links without uncertainty.

The details of KM construction are conditioned, though not determined, by the problem domain. A general-purpose procedure for outputting a KM during the KBS conceptualisation process is given below.

1. *Identify the main goal of the system.* The purpose of the KBS is to make a decision on a concept and, more particularly, on a property or attribute of that concept, which we have termed the goal property. The above goal should have already been decided, as it is essential for drawing up the KM. The attribute or goal property in a medical diagnostics system, for example, would be the disease suffered by the patient.
2. *Design the goal decision block.* Draw a rectangle around the property, specifying to which concept it belongs, using the property/concept form, and the possible values of that property. In the example of the above-mentioned medical diagnostics KBS, the possible values would be the names of the diseases that are to be diagnosed by the system. Fig. 3 shows an example of how to represent the properties in the KM.
3. *Add the properties for inferring or calculating the goal decision.* Place these properties inside boxes around the goal decision. The relation with the goal property is expressed by means of an arrow that will start from the property used to infer or calculate the goal decision. The number of values of the source property of the arrow is specified on this arrow. If the number of values is 1, no specification is required. Each attribute around the goal decision must be involved in inferring or calculating the value of this decision, but this does not mean that all these attributes are used at the same time. We usually recommend the use of from three to seven different attributes used in a number of rules, frames, etc., to infer the values of the goal decision. Most human brains deal with only a small number of attributes simultaneously. If a lot of attributes are used to infer the value of the goal decision, the expert will probably calculate the value of some intermediate attributes in order to infer the goal decision. The number of attributes around the goal decision can be reduced by introducing intermediate attributes.
4. *Extend the KM.* The properties used to infer or calculate the values of each property employed in point 3, that is, the properties used to infer the goal decision, have to be added.
5. *Repeat step 4* until none of the *peripheral* properties are inferred, that is, they are taken from external sources: user input, sensors, files, database, etc.
6. *Check the knowledge reflected in the KM.* These checks come under two categories:
   - Checks related to the validation of the knowledge reflected in the KM with the expert. In this case, the expert, assisted by the knowledge engineer, will have to check that the information stated in the KM correctly reflects how the expert reasons to solve the problem in question. That is, the expert checks that the relations between attributes reflected in the KM are the same as he or she uses and that the underlying reasoning of the KM reflects his or her

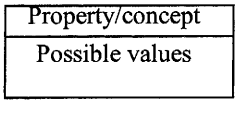

Fig. 3. Property representation in the KM.

problem-solving process. This underlying reasoning is detailed in the dynamic part of the conceptual model.

- Checks related to the verification of the KM against the static and dynamic models generated during the synthesis stage. The most noteworthy checks to be performed include:
  - Check that all the attributes and values are reflected in the Concept/Attribute/Value table of the static model. If not, they should be added.
  - Check that all the attributes of the Concept/Attribute/Value table are somehow reflected in the KM, otherwise either the attribute is over or the KM is incomplete.
  - Check that the values of the respective attributes can be inferred from the information represented in the dynamic model or can be obtained from a default value. Also verify that the value of these attributes, except the goal decision, is used in the left-hand side of a rule or appears in any other place associated with the attribute. If not, examine why it has been recorded as a possible value and has not been used. Similarly, it is important to verify that no value of an attribute is obtained recursively from itself.
  - Check that the attributes in the periphery of the map are obtained from external sources. If not, and it is not clear how these attributes are inferred or calculated, return to the knowledge acquisition phase.

Briefly, the KM represents expert reasoning using both the static (attributes) and dynamic part (relations between attributes represent the inferences identified, although they are not detailed in the KM). Therefore, by combining static and dynamic knowledge, the KM reflects KBS operation and the conceptualisation effected can be validated as a whole. Section 5 shows an example of KM construction and verification.

5. Example of KM construction

This section shows a practical application of the preceding section aiming to give a better understanding of how KMs are built and used. The example is part of a real project and illustrates how the KM is drawn up from the static and dynamic models, which are interrelated in one and the same representation.

The problem addressed in this example is the control of a cement furnace. A series of furnace variables are constantly measured by sensors. Depending on the values of these variables, it may be necessary to take certain actions to control the furnace. As long as these variables have the correct values, no control action will be taken.

Table 5 shows the concept/attribute/value table that is part of the static model of this problem. Tables 6–8 describe the tactical knowledge used to make the inference.

The KM is drafted by first identifying the objective of the system, that is, the goal property. In the example, the goal property is the speed of the ventilator that needs to be adjusted. Fig. 4 shows this property in a thicker edged rectangle. It is clear from this figure that the possible values of this property are real numbers. This knowledge is defined in the concept/attribute/value table (Table 5).

Looking at the tactical knowledge in the dynamic model, we find that the ventilator speed is obtained from the Control_Action property, using the rules in Table 8.

We then look for the attributes required to infer or calculate the attribute Control_Action. Table 7 defines how this attribute is obtained from the Status_BZ and Status_OX. Fig. 5 shows the result of adding these two attributes to the KM under construction.

Table 5
Static model: concept/attribute/value table

Concept	Attribute	Value
Furnace temperature	Reading_BZ	Real
	Ideal_BZ	Real
	Range_BZ	Real
	Status_BZ	{OK, DB, SB}
Percentage oxygen	Status_OX	{OK, High, Low}
Ventilator	Control_Action	{R1 (Reduce speed) R2 (Reduce fuel) I1 (Increase speed) I2 (Increase fuel) Nothing (No action to be taken)}
	Speed	Real

Table 6
Dynamic model[a]

Rule name	Rule formulation
Rule statusBZ1	**If** reading_BZ $<$ ideal_BZ $-$ 2*range_BZ then status_BZ = DB
Rule statusBZ2	**If** reading_BZ $>$ ideal_BZ $-$ 2*range_BZ and reading_BZ $<$ ideal_BZ $-$ range_BZ **then** status_BZ = SB
Rule statusBZ3	**If** reading_BZ $>$ ideal_BZ $-$ 2*range_BZ and reading_BZ $<$ ideal_BZ $+$ range_BZ **then** status_BZ = OK
Rule statusBZ4	**If** reading_BZ = ideal_BZ $-$ 2*range_BZ **then** status_BZ = SB
Rule statusBZ5	**If** reading_BZ = ideal_BZ $-$ range_BZ or reading_BZ = ideal_BZ $+$ range_BZ **then** status_BZ = OK

[a] Tactical knowledge: rules for obtaining the value of the attribute status_BZ.

Table 7
Dynamic model[a]

Status_BZ	Status_OX		
	Low	OK	High
DB	R1, R2	R1	R2
SB	I1	I1, I2	I2
OK	I1, I2	–	I2

[a] Tactical knowledge: Decision table for obtaining the Control_Action value.

Table 8
Dynamic model[a]

Rule name	Rule formulation
Rule Ventilator_Speed1	**If** Control_Action = R2 **then** Speed = Speed (previous status) $*$ 0,5
Rule Ventilator_Speed2	**If** Control_Action = I2 **then** Speed = Speed (previous status) $*$ 0,75

[a] Tactical knowledge: Rules for obtaining the value of the attribute Ventilator_Speed.

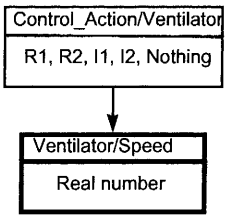

Fig. 4. Starting point of the KM.

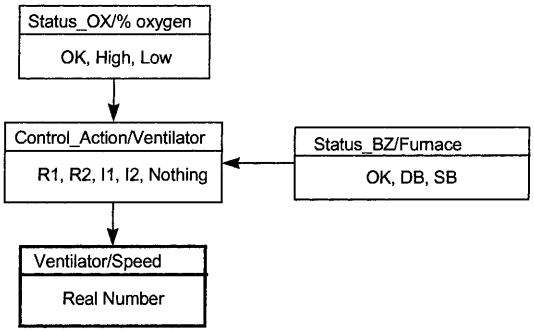

Fig. 5. First KM extension.

The attributes required to infer Status_BZ and Status_OX are then identified. Table 6 defines how this attribute is obtained from the Range_BZ, Ideal_BZ and Reading_BZ. Similarly, Fig. 6 shows the result of adding these three attributes.

As specified in the preceding section, the KM ends when the peripheral attributes are neither inferred nor calculated. In this case, the peripheral attributes are Status_OX, Range_BZ, Ideal_BZ and Reading_BZ. The values of these attributes are obtained from external sources: sensors and user. So, the KM shown in Fig. 6 is the final map.

Now is the time to check the KM:

- *Validation of the knowledge reflected in the KM with the expert.* This validation is performed by arranging a meeting with the expert. At this meeting, the knowledge engineer solves previously selected test cases, using the KM and aided by the dynamic model (Tables 6–8). The expert can check, as the test cases are solved, whether they are reasoned out according to the same problem-solving process as he or she would use. If he or she detects any step in which he or she would reason differently, he or she should say so to the knowledge engineer. This means that the knowledge represented in the conceptual model will be debugged with the aid of the KM. Additionally, we can see how this knowledge debugging is holistic: the static (attributes and their possible values) and dynamic (rules and decision table) knowledge is used at the same time. The stepwise solution of a test case is detailed in Appendix A.
- *Verification of the KM against static and dynamic models.* By comparing the content of the static and dynamic models with the contents of the KM, we are checking that all the information contained in both models has been used to draw up the KM, that the KM contains no more infor-

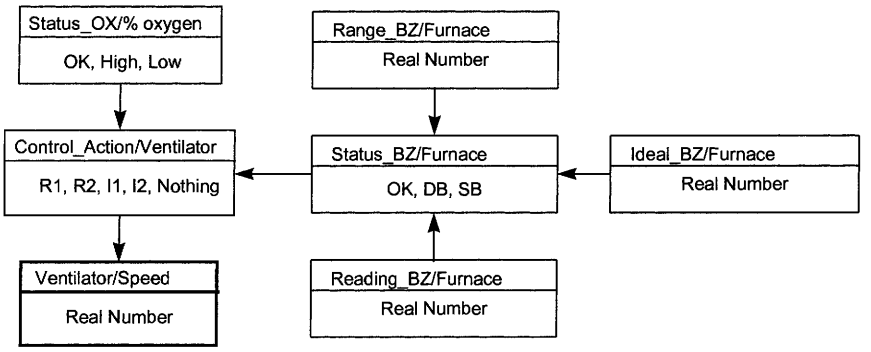

Fig. 6. Second KM extension.

mation than is expressed in the models and, also, that the attributes in the periphery of the KM are obtained from external sources: sensors and user.

This simple example of how to build a KM shows how the information described in the static and dynamic models is integrated into a single model by means of which the operation of the problem to be modelled, in this case, furnace control, can be represented and tested globally.

6. Benefits of KMs

The features of KMs mean that they are suited for carrying out the much needed, albeit often overlooked, activity of holistic testing to conceptualise problems and solutions. Below, we discuss the benefits of the morphology of the above models for performing holistic testing on the conceptual model obtained from an expert. These benefits have been divided into two categories: (1) concerning construction of the KM and (2) concerning validation of the KM and, hence, the operation it reflects.

With regard to the construction of the KM by the knowledge engineer, one of the main benefits is the simplicity of the above model, which stems from the simplicity of the notation used (boxes and arrows). Furthermore, the spatial organisation of the model, structured around "layers" of properties, which gradually represent the relations of dependency among these properties, makes the model easily navigable. This means that the dependencies among the above properties can be easily identified. Furthermore, the semantics of the model is increased by this structure, as the distance (near or far) between the interrelated properties and their interconnections give an idea of how involved each one is in the decision. Thus, in the example, we see that the attribute Range_BZ influences the values of Status_BZ, but not Status_OX.

Another advantage of KMs is that they allow problems to be decomposed into modules and/or subproblems. In Fig. 6, for example, different subproblems could be identified for each of the properties that infer the attribute Control_Action, that is, the influence of the percentage of oxygen, on the one hand, and the temperature in the furnace, on the other. Fig. 7 illustrates these two subproblems. This means that the issues and topics involved in the problem can be gradually processed stepwise and placed in a hierarchy, which enables performance of a critical control function. This approach allows all the information to be applied and implemented in a manner that can be verified, documented and audited. The decomposition shown in Fig. 7 represents a

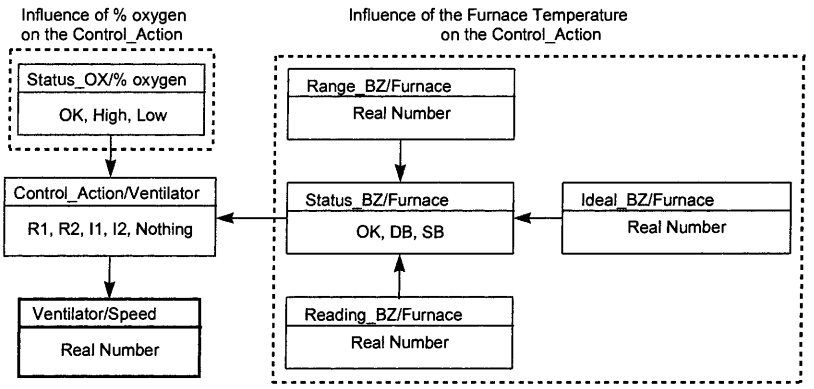

Fig. 7. Representation of subproblems in the KM.

small example, where the modules would not be large. However, if we think of a KM with several tens of properties, this feature would be much more beneficial.

Finally, the fact that KMs are built regressively, that is, from the goal to the data, reduces any possible information contamination to the minimum. This means that this representation can be more or less directly transformed into a computer implementation.

With regard to the benefits of KMs for validation by experts, we can say, on the basis of our experience in their use in real applications, that the very features that make the models easy to build, which we discussed above, also mean that they work well as a basis for discussion between the knowledge engineer and expert. That is, the simplicity of the notation and the spatial distribution are properties that make the model easy to interpret and, therefore, improve the expert's understanding the knowledge it reflects.

Additionally, KMs improve group discussion about a domain, because they are more effective than traditional text for acquiring and visualising important ideas and are as effective as traditional text in acquiring and presenting details.

It is also important to stress another consideration already discussed in this paper, and this is the capability of KMs to represent globally (as opposed to separately, as in the static and dynamic models) the reasoning of the expert and, hence, the operation to be simulated by the KBS under construction. This feature makes it possible for this knowledge to be validated by the expert at an early stage of the development process, before the software system has been built.

7. Conclusions

This paper stressed the importance of understanding and comprehending the problem as a key element in proper software process development. Now, to gain a full understanding of a problem, we must be able to conceptualise the problem effectively and, if possible, efficiently.

Conceptualisation is not an activity exclusively applicable to software development. On the contrary, it is an activity performed by any human being about to solve any problem. Traditionally, and according to the Cartesian method, conceptualisation involves two fundamental activities: analysis and synthesis. Analysis is a descriptive activity, by means of

which the components of the problem under study can be identified, whereas synthesis restructures and reorganises the above models to represent the problem-solving process under examination.

More recently, these two activities have been found not to be sufficient for fully conceptualising a problem. This idea is clearly stated by Crick in reference to work on understanding the human brain. It is not sufficient to decompose the problem into parts and understand the role of each part and the interrelations between these components. It calls for a higher level activity (what we have termed holistic testing), by means of which the workings of the brain can be represented and without which it is impossible for us to tell whether its operation has been properly understood and represented.

Guided by these general notions of conceptualisation, we have tried to apply these ideas to the process of conceptualisation for building KBSs. Thus, we have presented a framework for addressing the process of conceptualisation in KBSs. This process has been divided into three phases: analysis, synthesis and holistic testing. The first two phases are also usually addressed by the well-known KBS construction methodologies, which, however, so far, have lent little importance to the third, despite its being a key activity. Its objective is to reliably test that the information conceptualised really reflects the reasoning used by the expert to solve the problem in question. This global validation of expert reasoning and, ultimately, of the operation of the software system is not traditionally performed until the system is operational. The holistic testing phase allows this process of validation to be performed during conceptualisation, and thus at an early stage of the development process.

A technique, KMs, was presented as an aid for addressing holistic testing. KMs can be used to globally represent the knowledge obtained and represented by means of the static and dynamic models generated during the synthesis phase. They also mean that the conceptual model can be verified and validated globally, unifying the static and dynamic parts.

One of the main reasons why KMs can be widely useful for developing KBSs is the simplicity of the notation used and of its semantics. This means that the model is easy to build and interpret, and thus facilitates communication and interaction with experts for validation purposes.

Appendix A. Example of how to solve a test case using a KM

Test case data:

Status_OX = High
Range_BZ = 52
Ideal_BZ = 85
Reading_BZ = 34.28

Result given by expert:

Speed = 52.5

STEPS

Step 1. The value of the peripheral attributes of the KM are taken from external sources. The result is shown in Fig. 8.

Step 2. The attribute Status_BZ is inferred using the rule RulestatusBZ3 taken from Table 6:

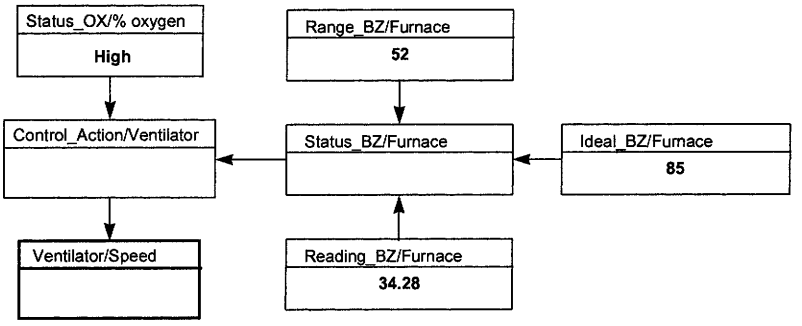

Fig. 8. Step 1: peripheral attributes.

If reading_BZ $>$ ideal_BZ $-$ range_BZ and reading_BZ $<$ ideal_BZ $+$ range_BZ **then** status_BZ $=$ OK

where the condition of the rule is found to be met when the values are entered:

$34.28 > 85 - 52$ and $34.28 < 85 + 52$

and calculated:

$34.28 > 33$ and $34.28 < 137$

and, therefore, the attribute status_BZ = OK. Fig. 9 shows the map with the new inferred values.

Step 3. The attribute Control_Action is inferred using the decision table shown in Table 7. The table entries are for the last row (Status_BZ = OK) and last column (Status_OX = High). Therefore, the value of the attribute Control_Action is I2, as shown in Fig. 10.

Step 4. The attribute Speed is inferred using the rule RuleSpeed_Ventilador2 taken from Table 8. The result is shown in Fig. 11.

If Control Action $= 12$ **then** Speed $=$ Speed(previous status) $* 0.75$

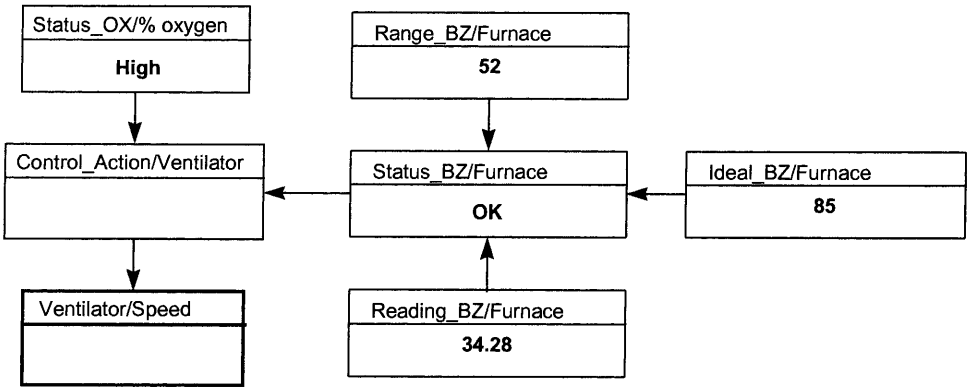

Fig. 9. Step 2.

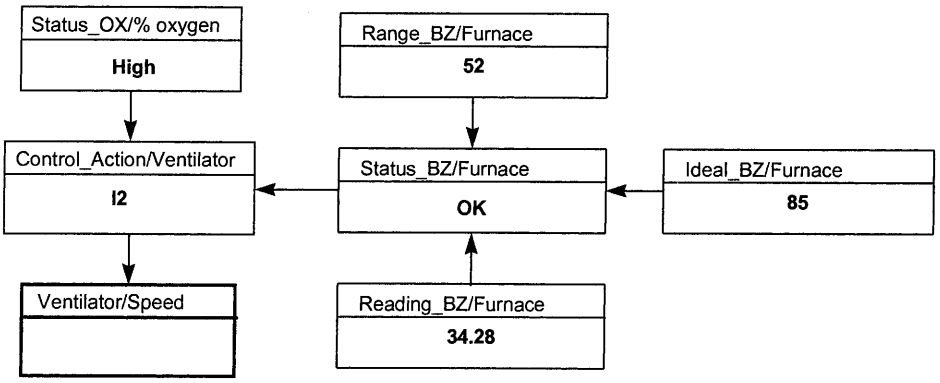

Fig. 10. Inference of the attribute Control_Action.

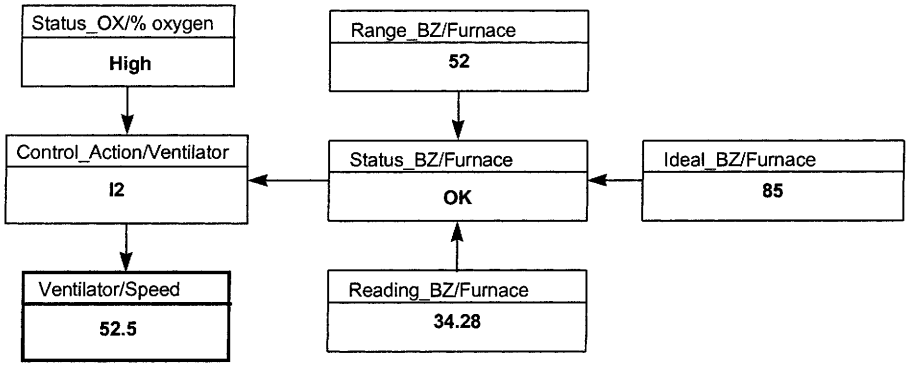

Fig. 11. Obtaining the goal attribute.

The output coincides with the result stated by the expert before using the KM. Therefore, the reasoning used to solve this test case is considered correct. For the KM to be classed as validated, this process would have to be repeated for all the selected test cases.

References

[1] J. Ares, J. Pazos, Conceptual modelling: an essential pillar for quality software development, Knowledge-Based Systems 11 (2) (1998) 87–104.
[2] B.I. Blum, Beyond Programming, Oxford University, New York, 1996.
[3] G.K. Chesterton, The Point of the pin, The Scandal of Father Brown, Cassell, London, 1935.
[4] F. Crick, What Mad Pursuit, Tusquets Editors, S.A., Barcelona, 1989.
[5] R. Descartes, in: C. Adam, P. Tannery (Eds.), Oeuvres de Descartes, first ed., 1638, Paris, 1969.
[6] D.R. Hofstadter, Automating Inspiration? Amer. Scientific, November 1982.
[7] E.C. McCagg, D.F. Dansereau, A convergence paradigm for examining knowledge mapping as a learning strategy. J. Educational Res. 84 (6) (1991).
[8] J. Pazos, Epistemological Approach to SE and KE, Keynote Speech, SEKE '97, Madrid, June 17–20, 1997.
[9] A.Th. Schreiber, B.J. Wielinga, J.A. Breuker (Eds.), KADS: A Principled Approach to Knowledge-Based System Development, Knowledge-Based Systems Book Series, vol. 11, Academic Press, London, 1993.