The ES could probably know more: But man would not make better business decisions

Zoltán Baracskai
Doctus Co.
Hungary
zoltan@doctus.info

Jolán Velencei
Budapest University of Technology and Economics
Hungary
velencei@imvt.bme.hu

Viktor Dörfler
Strathclyde University
United Kingdom
viktor.dorfler@strath.ac.uk

Abstract

We know for a long time that expert system (ES) must not know in a different way than how man knows. We would not expect men to adapt their thinking processes to the machine, would we? Evolution would not let peoples' minds adopt criteria weighting or fuzzy logic. This is why we developed our Doctus[1] *knowledge-based expert system based on symbolic logic – that is the easiest to understand artificial knowledge representation to the decision taker. However, this is all history now... Now we perhaps understand decision takers better – maybe it was worth observing them for nearly two decades. Decision takers, and we mean only the most experienced ones, want to be able to play around a bit with a particular decision and, eventually, they want some help in delegating the decisions they are bored with.*

Keywords: *decision support system, knowledge-based system, expert system, Doctus*

Introduction

We have developed the first version of Doctus based on Simon's conception of bounded rationality, his distinction of programmed and non-programmed decisions, and on our view of knowledge. These we shall briefly describe in the following sections, as they are still valid. It also wore the mark of the time: the available technical/programming facilities and the business environment; these we shall not describe here, as they are outdated.

Once the background is thus prepared, we are going to introduce the 'old Doctus', i.e. we will describe the deductive, inductive and reductive reasoning in Doctus as it is today. We will do this following the timeline in terms of the order of added features; however, we are going to disregard the various ways of implementing these – we will always describe the latest solutions. That way, following a thread of history, we arrive at today.

When Doctus has stepped into the adulthood, we sat down to draw the line and review our experiences. In the 18 years with Doctus, we had around 140 consultancy projects, the vast majority of them for top executives. We had summarized our successes and failures on more than few occasions over these year, we wanted to do something different this time: we wanted to overview which are the features of our ES that decision takers like and make use of and to decide where to go next with the development. In the final section we tell the story of the lessons learned. What we have learned underlies the present development of the 'new Doctus' – suitably codenamed Doctus Light.

[1] http://www.doctus.info/

1. Rationalities, Decisions, Knowledges

According to the idea of homo economicus (Sen, 2002: 19 ff) people are totally (infinitely) rational, meaning that they search for the optimal solution, for the best of all. This would mean that all the decision alternatives are considered and they are compared according to each and every feature. Even if we disregard the simple technical issues, e.g. that the features were all supposed not only to be measurable but measurable in the same of convertible unit, there are still numerous reasons why this is impossible: there are too many solutions to find all of them; identification of some of the attributes would require knowledge that is not yet acquired; if all the decision alternatives and attributes have been identified there is still the task to gather data about each and every alternative, considering each and every attribute; and there is also a time-limit. March and Simon (1993: 157 ff) examined the limitations of rationality; and they show that the above-mentioned limitations on the number of attributes and alternatives are direct consequences of cognitive and organizational limits.

Thus Simon (1997) developed the conception of *bounded rationality* in which he offers two choices to the decision taker:

The first is to take her/his decision according to the total rationality described above, though the searching space – the number of decision alternatives and the aspects considered – and the time-span of comparison get limited. It is necessary to add that this is an impermissible oversimplification of the reality. It is not Simon who simplifies the world impermissibly but the decision taker adopting this framework of rationality. (S)he usually believes that the world and (s)he in it really operates according to the total rationality. (S)he does recognize the limitations, believing that (s)he really does search for the best solution. This still does not mean that this conception of rationality is entirely useless as the experience may sensibly limit the searching space; e.g. we may know that there are only three suppliers in the world capable of providing a certain product or service and that they only compete on a few features.

The other possibility for the decision taker is to define her/his expectations and to start the search afterwards. Once the expectations are defined, the solutions come one-by-one; the decision taker compares the features of a solution to her/his expectations before proceeding to another solution, thus taking a series of decisions. The first solution that satisfies the expectations is accepted and it is called a satisfactory solution. Time plays an important role in this kind of search: if no alternative is accepted for a long while (i.e. no alternative fits the expectations), the decision taker modifies her/his expectations and/or the relations among them.

We give further detailed elaboration of the rationalities elsewhere (e.g. Baracskai, 2000; Baracskai & Velencei, 2004; Dörfler, 2005), here the aim was only to present the conception forming the basis of logic in Doctus' deductive reasoning. The next step is to distinguish the various types of decisions relevant to the different ways of reasoning in Doctus.

Another model of Simon (1977: 31 & 45-49) distinguishes between programmed and non-programmed decisions. Programmed or well-structured decisions are those that frequently occur and thus we can have elaborated procedures on how to handle them; these can often literally be programmed. Non-programmed or ill-structured decisions appear in novel situations that we meet for the first time and thus there cannot be any elaborated procedures available. The programmed and the non-programmed decisions are non-existing extremities (black and white) of a continuum (greyscale) in which the real life decisions reside. As a further development here we describe the decisions using three cornerstones (Figure 1):

Reflex decisions are taken without thinking, i.e. these are reactions not responses of the organization; e.g. paying wages or maintaining stock level belong here. These decisions are to be instant and can be handled by assisting staff. Although we do not address the reflex decisions directly with Doctus, there can be a computer support to these decisions; only they would not normally require knowledge bases, databases and hardwired procedures would suffice.

Routine decisions are taken by managers following some set of explicit rules, such as launching a new product, adapting a new technology, or evaluating suppliers. The knowledge used in these decisions comes from the experience we got by taking similar decisions and we are aware of the decision aspects (attributes) and of the rules between the values of these attributes. Routine decision is the closest real-life resemblance of programmed decisions but it they are not the same. Although routine decisions incorporate a vast amount of attributes and rules, the importance of individual remains in focus. Sometimes we may need a new logical rule between the values of the attributes, or consider some new attributes or ignore some of the old ones. With Doctus we aim at speeding up these decisions by inductive knowledge bases (see later).

Original decisions are those that we take for the first time, such as starting (or continuing) an R&D project or entering a new market. These obviously belong to changes and thus to leadership (see e.g. Kotter, 1999; Kotter & Cohen, 2002). These situations come the closest to the non-programmed decisions, although, they are not the same as we always now something even about the most novel situation. Typically, original decisions are the responsibility of a leader, who does not have any experience with it (as no one has); however, it does not mean that (s)he is novice decision taker. This decision type has an additional feature: there are no well defied attributes. In Doctus the attributes are, in such cases, defined using symbols and metaphors; describing the rules for the deductive knowledge base is a learning process.

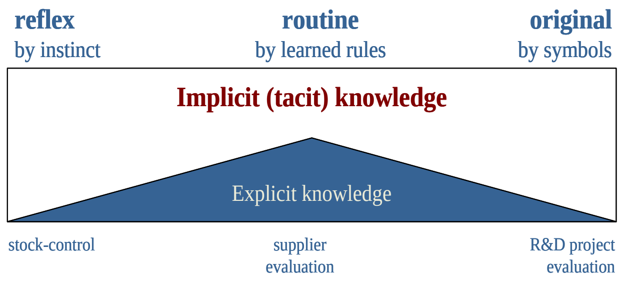


Figure 1: Decision types

In our view managers and leaders of the e.era should not spend any time on reflex decisions. Finding the rules for routine decisions facilitates the delegation of these decisions. Excellence is up to original decisions. If all decisions are considered to be original decisions, the e-business looses on efficiency; if all are considered to be routines, the e-business becomes rigid and dies. We can find some correspondence between the rationalities model and this decision typology: the routine decisions, and thus the inductive knowledge bases, correspond to bounded rationality. The deductive knowledge base in its final form also corresponds to bounded rationality but the essence here is the learning process of structuring the expectations. In both cases with Doctus we design a representation of the knowledge of a manager/leader and the involved domain experts. For better understanding next we give a brief overview of how we classify knowledge.

Our typology is based on Ryle's (1949) distinction of knowing that and knowing how; the same categories were identified by Anderson (1983) as propositional (declarative) and procedural knowledge. Gurteen (1998) identified a further type that he called knowing why. We add further two categories, knowing what and knowing it. To organize these knowledge types into a more comprehensive model we use Polanyi's conception of tacit knowledge (Polanyi, 1966) and, based on this, the distinction of focal and subsidiary knowledge (Polanyi, 1962), also taking on board Minsky's (1994) distinction of positive and negative knowledge. Instead of describing these various models in detail we present our synthesized model (Figure 2) and establish the relations to the mentioned knowledge types.

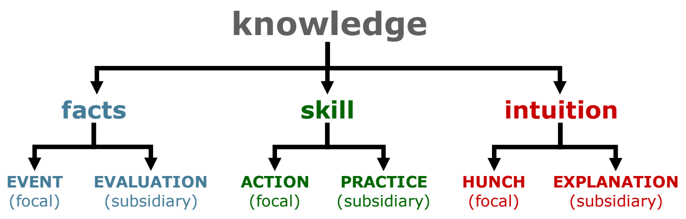


Figure 2: Knowledge types

The focal part of facts is the event. This corresponds to the 'know-it'; this knowledge is tacit. The subsidiary knowledge of facts is the evaluation, i.e. the rules of evaluating; this is a 'know-that', which is explicit. For example, the jump of a pole-vaulter is a fact. Its subsidiary part is the knowledge about the measuring standard and how to use it. The focal part is our experience of the jump; this includes e.g. that the bar did not fall down. As we see, the focal facts are something much richer than what is usually called facts. The second-hand facts (De Bono, 1976: 12-14) belong to the domain of propositions, i.e. they will not be focal but subsidiary facts. The essence of the second-hand facts is not the correspondence to the reality but the controllability; e.g. we can check in lexicons that Lee Harvey Oswald killed Kennedy, so it is a fact even if we cannot be sure if it is true.

The focal part of skills is the action, the 'know-how'. The subsidiary part is the set of rules of practising and second-hand facts about practising, which is 'know-that'. E.g. the act is movement with the bicycle with the subsidiary rules of keeping the balance. Experiencing events and practising skills are both experiences but while the former is passive the later is active.

The focal intuition is the hunch, when one senses the direction, which is 'know-what' or the solution, which is 'know-why'. The subsidiary intuition is the ex post explanation which has to follow the rules of formal logic strictly, regardless whether it was how the hunch happened or not. Similarly to the previous two subsidiary knowledge types, the explanation uses second-hand facts, which belong to 'know-that'. It is a useful image to describe hunch as applying a number of rules tacitly at the same time, as Simon (1987) did. To contrast the focal intuition to the previous two types of focal knowledge we can say that those were both external experiences while the hunch is an inner experience.

If we properly understand the organization of these knowledge types, we see that only 'know-that', i.e. that subsidiary knowledge types can be put into words and thus only this can be used in building knowledge bases. However, we also indicated that we can make countless statements about our experience, and thus by building a knowledge base we convert a small part of the expert's and decision takers knowledge from tacit 'know-it', 'know-how', 'know-why', and 'know-what' into explicit – this is an important added value of Doctus.

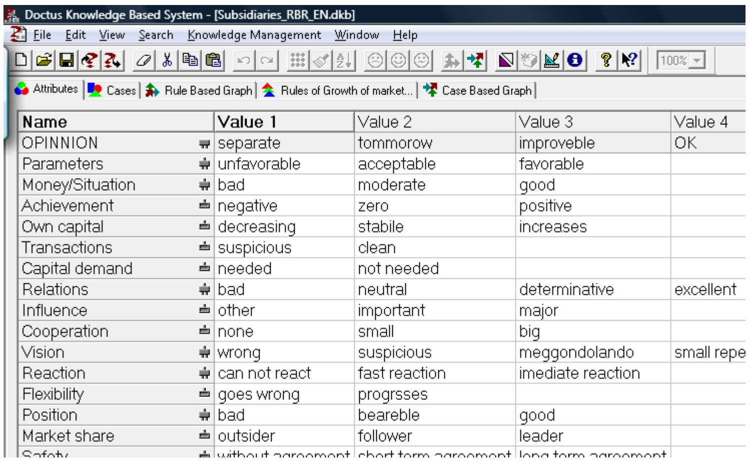
2. The “Old Doctus”

Doctus is a knowledge-based expert system[2] that is aimed at supporting the decision taker by providing her/him with the representation of the expert opinion. As we never believed that people think in numbers, we based Doctus on symbolic artificial intelligence (AI). Nobody thinks that the beautiful is 3.6 times better than the ugly. In symbolic AI the representation consists of concepts, acquired from the experts, which are connected by logical rules in «if... then» format. In this representation the concepts are treated as symbols hence the names ‘symbolic logic’ and ‘symbolic AI’ are used.

The process of creating a representation of the experts’ knowledge in a KBS is called knowledge engineering, and the person doing it is the knowledge engineer. There are two main stages of knowledge engineering: during knowledge acquisition we build the initial model which is altered, throughout the fine-tuning stage, until the experts accept it. Apart from the experts and the knowledge engineer we also invite the decision taker to visit occasionally the knowledge engineering process, this makes it easier to use the knowledge base later.

2.1. Deduction

The first mode of reasoning in Doctus, both historically and in terms of software complexity, is *deductive* or *rule-based reasoning*. In this case the experts are required to articulate the aspects of decision, the «attributes» and the grades of satisfying these aspects, the «values». The values are usually normally defined using ordinal scales but nominal values can also be used. To make the acquisition of attributes and values simpler we provided an interface that we expected the users to handle easily; we designed an Excel-like look-and-feel. (Figure 3)


Doctus Knowledge Based System - [Subsidiaries_RBR_EN.dkb]
File Edit View Search Knowledge Management Window Help
100%
Attributes
Cases
Rule Based Graph
Rules of Growth of market...
Case Based Graph

Name	Value 1	Value 2	Value 3	Value 4
OPINNION	separate	tommorow	improveble	OK
Parameters	unfavorable	acceptable	favorable	
Money/Situation	bad	moderate	good	
Achievement	negative	zero	positive	
Own capital	decreasing	stabile	increases	
Transactions	suspicious	clean		
Capital demand	needed	not needed		
Relations	bad	neutral	determinative	excellent
Influence	other	important	major	
Cooperation	none	small	big	
Vision	wrong	suspicious	meggondolando	small repe
Reaction	can not react	fast reaction	imediate reaction	
Flexibility	goes wrong	progrsses		
Position	bad	beareble	good	
Market share	outsider	follower	leader	
Safety	without agreement	short term agreement	long term agreement	

Figure 3: Attributes and values in Doctus

The attributes are then organized into a multi-level hierarchy called «rule-based graph». (Figure 4) The rule-based graph (RBG) describes the dependencies of the attributes; the leaves of the graph are the input attributes, these are the factors of dependant attributes, these in turn of other dependant attributes and so forth, until we reach the root of the graph, which is the decision the decision attribute. What is important to realize at this point is that the RBG

[2] The term ‘knowledge-based system’ (KBS) indicates that there is a representation of knowledge in such systems; these representations are called knowledge bases. We also speak of ‘expert systems’ (ES) emphasizing that we build a knowledge base of experts’; these two terms are sometimes combined into ‘knowledge-based expert system’ (KBES). Although the three terms differ in emphasis in this paper they are used interchangeably.

only tells us *which* the factors of each attribute are but not *how* the attributes depend on their factors.

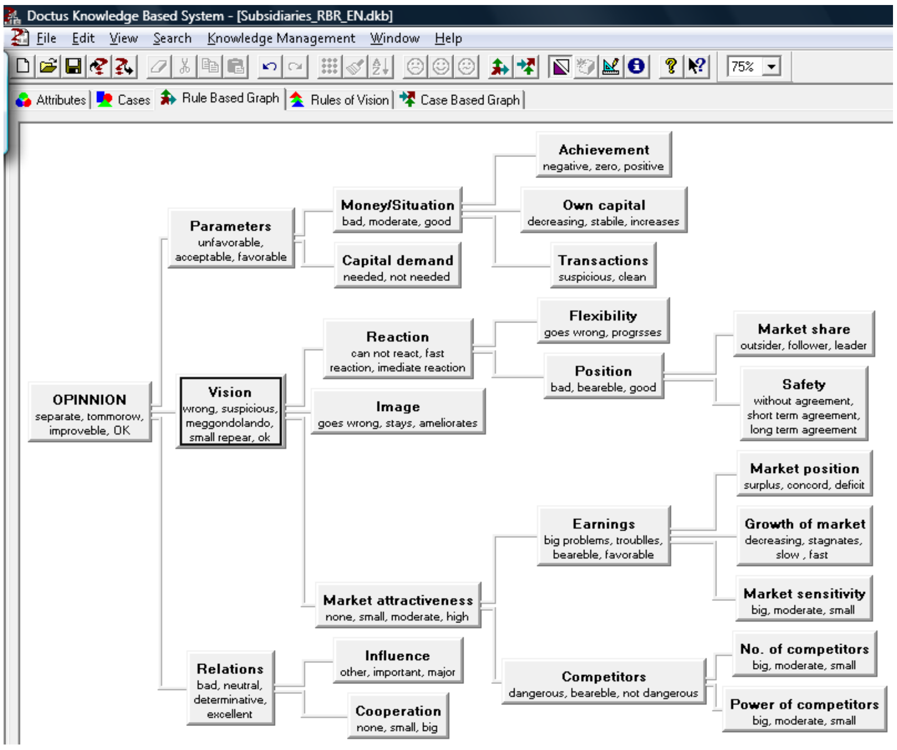
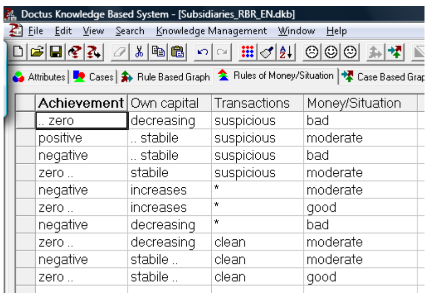
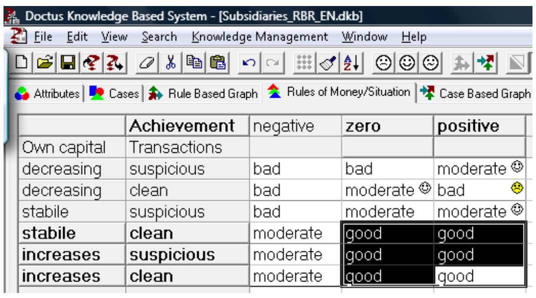

Doctus Knowledge Based System - [Subsidiaries_RBR_EN.dkb]
File Edit View Search Knowledge Management Window Help
75%
Attributes | Cases | Rule Based Graph | Rules of Vision | Case Based Graph
OPINNION separate, tommorow, improveble, OK
Parameters unfavorable, acceptable, favorable
Vision wrong, suspicious, meggondolando, small repear, ok
Relations bad, neutral, determinative, excellent
Money/Situation bad, moderate, good
Capital demand needed, not needed
Reaction can not react, fast reaction, imediate reaction
Image goes wrong, stays, ameliorates
Market attractiveness none, small, moderate, high
Influence other, important, major
Cooperation none, small, big
Achievement negative, zero, positive
Own capital decreasing, stabile, increases
Transactions suspicious, clean
Flexibility goes wrong, progrsses
Position bad, beareble, good
Earnings big problems, troubles, beareble, favorable
Competitors dangerous, beareble, not dangerous
Market share outsider, follower, leader
Safety without agreement, short term agreement, long term agreement
Market position surplus, concord, deficit
Growth of market decreasing, stagnates, slow , fast
Market sensitivity big, moderate, small
No. of competitors big, moderate, small
Power of competitors big, moderate, small

Figure 4: Rule-based graph in Doctus

To answer this ‘how’ the experts also need to articulate a set of «if... then» rules in each node of the graph to connect the values of the attributes; all value combinations need to be covered. Over the years we implemented several features to make the rule input easier. First we introduced the complex rules, i.e. rules covering more than one value of one or more attributes, to reduce the amount of input. (Figure 5, left) Then we developed a two-dimensional (2-D) rule input interface (Figure 5, right) in which one rule corresponds to a cell; a complex rule here would be an area of multiple cells (Figure 5, right, black area). Finally we launched the consistency analysis; this notifies about the potential contradictions based on the ordinal nature of the values. (Figure 5, right, smilies) The same tool can be used to give advices what are the acceptable rule outputs. By these three features we really made the life of the knowledge engineer easier – unfortunately we gave nothing to the decision taker. We also set up a tool for sorting/rearranging/combining the rules; these were quite helpful in teaching about Doctus – so again, nothing for the decision taker.


Doctus Knowledge Based System - [Subsidiaries_RBR_EN.dkb]
File Edit View Search Knowledge Management Window Help
Attributes | Cases | Rule Based Graph | Rules of Money/Situation | Case Based Grap

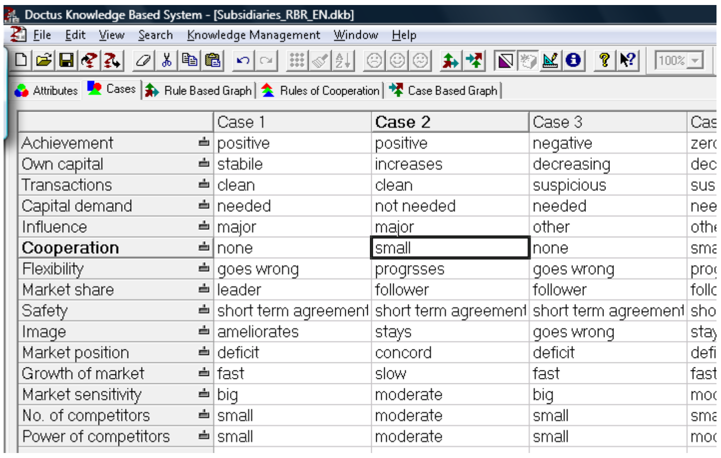
Achievement	Own capital	Transactions	Money/Situation
.. zero	decreasing	suspicious	bad
positive	.. stabile	suspicious	moderate
negative	.. stabile	suspicious	bad
zero ..	stabile	suspicious	moderate
negative	increases	*	moderate
zero ..	increases	*	good
negative	decreasing	*	bad
zero ..	decreasing	clean	moderate
negative	stabile ..	clean	moderate
zero ..	stabile ..	clean	good


Doctus Knowledge Based System - [Subsidiaries_RBR_EN.dkb]
File Edit View Search Knowledge Management Window Help
Attributes | Cases | Rule Based Graph | Rules of Money/Situation | Case Based Graph

	Achievement	negative	zero	positive
Own capital	Transactions			
decreasing	suspicious	bad	bad	moderate ☺
decreasing	clean	bad	moderate ☺	bad ☹
stabile	suspicious	bad	moderate	moderate ☺
stabile	**clean**	moderate	good	good
increases	**suspicious**	moderate	good	good
increases	**clean**	moderate	good	good

Figure 5: Rule input in Doctus

The last input step in deductive reasoning is to describe the decision alternatives, the «cases». This is done by assigning a value of each attribute to each case; these are called the case features. (Figure 6) Here we also added some features, often asked for by the experts, such as value distribution – but we never met an expert who actually wanted to use it, not to mention the decision taker.


Doctus Knowledge Based System - [Subsidiaries_RBR_EN.dkb]
File Edit View Search Knowledge Management Window Help
100%
Attributes
Cases
Rule Based Graph
Rules of Cooperation
Case Based Graph

	Case 1	Case 2	Case 3	Cas
Achievement	positive	positive	negative	zer
Own capital	stabile	increases	decreasing	dec
Transactions	clean	clean	suspicious	sus
Capital demand	needed	not needed	needed	nee
Influence	major	major	other	oth
Cooperation	none	small	none	sma
Flexibility	goes wrong	progrsses	goes wrong	pro
Market share	leader	follower	follower	follc
Safety	short term agreement	short term agreement	short term agreement	sho
Image	ameliorates	stays	goes wrong	stay
Market position	deficit	concord	deficit	defi
Growth of market	fast	slow	fast	fast
Market sensitivity	big	moderate	big	moc
No. of competitors	small	moderate	small	sma
Power of competitors	small	moderate	small	moc

Figure 6: Cases in Doctus

Once the whole deductive knowledge base is thus set up, we can trigger the rules to get the decisions for the cases. The results can appear in the RBG and on the «Cases» tab; we have found the first one more useful. This is only the beginning of the fine-tuning process, the attributes, values, rules, and case features are refined until the experts agree with the knowledge base. To support this we developed «Explain» feature, which takes you to the rule that resulted in a particular output value for a particular case. This was, again, of great use to the knowledge engineer and to a lesser extent to the experts but not much to the decision taker – although this is one of the few features that they showed some interest in.

2.2. Induction

After using the deductive version on Doctus for nearly a decade we realized that the rule-based knowledge based often does not really correspond to the experts' opinion, simply because they were trying to come up with what they thought is expected from them. We also noted that the experts struggle putting their knowledge into rule form – we understood that the experts are treasure houses of experience rather than of rules. Thus, we developed the *inductive reasoning*, in which the experts articulate the attributes and the values in the same way as in deduction but then, instead of describing the rules between the values, they quote cases from their experience, together with the decision outcomes. To infer the rules classifying the cases according to the outcome values we adopted a modified version of the ID3 algorithm developed by Quinlan (1986); later we realized that by applying our modification we came very near to Quinlan's (1993, 1996) later C4.5 model.

As in induction the reasoning starts from the particular cases of experience from which Doctus infers general rules, this kind of reasoning is often *called case-based reasoning* (CBR). However, we try to avoid this term as the CBR was inherited from quantitative decision support and it is still associated with numeric multi-criteria decision making (MCDM).[3] The essence of quantitative CBR is – although often covered by a veil of beautiful graphics – to define some metrics, and distance built on it, which will be the measure of similarity. For a new case the nearest one – the most similar – is searched from the case-base. In symbolic logic we consider the cases described with the same rules to be similar. So the logic is similar but there is a crucial difference: as we noted earlier, we believe that much of the experts' opinion cannot be quantified/measured.

The logic of the inductive reasoning is the following: Consider that the all the cases form a disordered set, where the order is defined having cases with a single output value in each subset. We forms subsets of cases according to the values of each attribute in turn and we choose the one that contributes the most to the order. We call this attribute the most informative one, the one with the highest informativity. (Baracskai, Dörfler & Timár, 2003) Then we repeat the process on all the subsets which contain cases with more than one output value. The result is displayed in the case-based graph (CBG) from which the rules can be read from the root of the graph towards each leaf. (Figure 7)

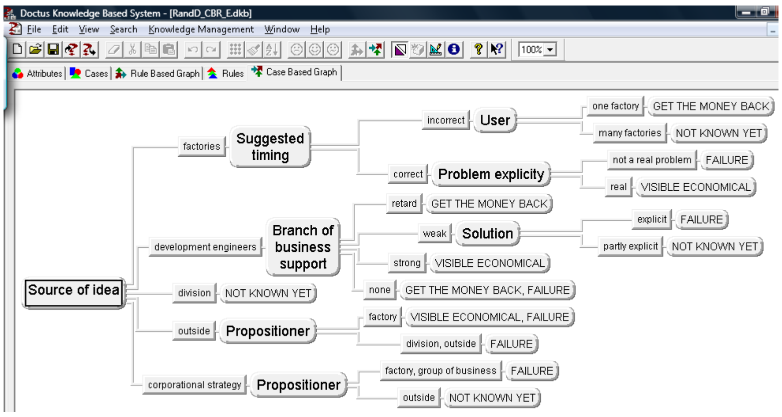


Figure 7: Case-based graph in Doctus

Due to the inductive logic, not necessarily all the attributes will appear in the CBG, as we stop the process as soon as the homogeneity of the output values is achieved. In fact, there are rarely more than 5-7 attributes in the graph. In the vast majority of cases the experts are surprised by the first appearance of the graph and frequently they disagree that it accurately describes their opinion. This is exactly what we expect, and indeed the first version of the graph is usually only the starting point of inductive reasoning; the final version is when the experts rate it accurate. The fine-tuning in this case includes changing/adding/removing attributes and/or values and also removing cases. This was an interesting learning point for us: inductive reasoning does not handle the outliers (non-typical cases) well; thus we implemented a feature to exclude particular cases from the reasoning without removing them from the knowledge base. Occasionally, it is illuminating to understand which cases are the outliers. Another way for fine-tuning is to change manually which attribute will appear in a particular node – as it makes sense only if the informativites are nearly the same, the experts should follow the knowledge engineer's advice when it can be done. When we worked with the same experts on multiple projects, they gradually learned how to provide consistent knowledge representation, but the greatest learning occurs exactly during the fine-tuning stage.

[3] Doctus is indeed a tool for supporting multi-criteria decision taking, as there are multiple attributes with multiple values (a value of an attribute is a criterion) but it is not quantitative and the term CBR as well as MCDM is normally associated with the quantitative approach.

Working with experts on fine-tuning on many-many project we gradually developed an interesting skill ourselves: by looking at the graph we became able to assess the knowledge level of the experts. This is an important thing, as you cannot build an expert system without an expert. On several occasion the only thing we could do was to report to the decision taker that her/his so experts are less then experts in their discipline.

2.3. Reduction

The third kind of reasoning, that we developed most recently, was born from the idea that if it was possible to find out the outcome based on attributes and rules, and it was possible to find the rules based on the attributes and the outcomes, it should also be possible to find the attributes based on the other two. Well, this is not entirely true, as you cannot define the rules and the outcomes without describing the attributes first. But we can also observe, as it was mentioned in the previous section, that the CBG normally does not contain all the attributes, only a few of them. So, although we cannot find the attributes based on the rules and the outcomes, we can find out which attributes are relevant. This is what we can read from the CBG – the informative attributes. So we developed a feature to convert the CBG into a deductive knowledge base by a click of the mouse. As the number of the attributes is thus reduced we call this third type of reasoning reduction or *reductive reasoning*.

It is important that, while we have fewer attributes in reductive knowledge base, it classifies all the cases in the same way as if we used all the attributes. It is possible that there will be no complete rule sets in each node of the new single-level RBG but these will usually indicate impossible case situation, at any rate, a sort of a case that the experts have not seen before. There is an interesting way how we can make a deductive knowledge base denser: first we build a deductive knowledge base as described in Section 2.1; when we have the outcomes, we run the inductive reasoning on the same knowledge base excluding the dependant attributes; finally we extract the rules from the CBG by applying the reductive reasoning. The reductive knowledge base will give the same output value for all the cases only using fewer, often much fewer, attributes. (See a detailed example in Baracskai, Velencei & Dörfler, 2005)

2.4. Export/Import in Doctus

There were two drivers almost at the same time pushing us towards connecting Doctus with external sources/applications. The first was that companies started to use data warehouses and they were eager to try to find some ‘knowledge’ in them. The second was that we wanted an easy-to-use way of using knowledge bases outside Doctus either for reasoning or gathering cases from other people. Therefore we developed the export and import modules of Doctus.

Static exported knowledge is some sort of “snapshot” of the knowledge base with no reasoning capability. Different types of graphs and tables can be generated, presenting graphs or details about the knowledge base. A report in rtf format, containing all tables is also available. Dynamic exported knowledge has capability of reasoning. Built-in templates are available to generate web server applications (php, perl, jsp), client applications (html pages with enabled on-page reasoning, javascript) and desktop applications (c++, xls). The technically most advanced export was that the whole intelligent portal (http://www.doctus.info/solutions/portal) was exported from Doctus – but even we were not able to do this without the help of the programmer. The html page with on-page reasoning became quite popular on the one hand for providing a simple example of reasoning, on the other hand for collecting cases within a company. However, these were ephemeral interests and usually ended up with 'traditional' use of Doctus.

We wanted to make Doctus connectable to all sorts of databases so that we could retrieve the readily available case features as quickly as possible. Therefore we developed a series of application programming interfaces (API) to get data from Excel, comma separated files, Microsoft Query, email, online databases, and finally from anything that can be handled via ODBC. As databases often contain numbers and many experts also indicated that they would like to use numeric data (from databases not what they personally articulate) we added a clustering algorithm to the import module. This way Doctus became capable of data mining. We have never seen a decision taker interested in any of these; and even the experts were exclusively interested in importing from Excel tables. The reason is not that Doctus was not as good at data mining as any other tool; actually we claim it to be one of the best. The decision takers are not interested in data mining.

3. Conclusions and the 'Doctus Light'

Today we understand that the ES must not have the features that the decision taker does not want to use. Decision taking is far too serious issue for men to yield it to a computer. Perhaps we could now develop the m-Doctus, and perhaps it would work as we imagine it, and perhaps we could achieve this within the programmability of the mobile phones. But that would be, again, something that only we thought to be useful. Previously we have spent large amounts of money to develop a data mining enabled Doctus, exported knowledge bases of various kinds – these were our figments that the decision taker is not interested in. Advertising Doctus through its capacity to make the decisions transparent to the decision taker, and if (s)he wants also to others, was another useless idea we implemented – the decision taker does not want to get into the fine details of her/his thinking process.

So, which were the successful parts of the Doctus development over all these years? The two fundamental roles of Doctus to perform deductive and inductive reasoning were very useful; the deductive knowledge bases were mostly used to support original decisions, while the inductive reasoning, as it requires cases with outputs, was typically used to support routine decisions. We also must note that there was far more interest in supporting original decisions of large scope (sometimes called strategic decisions but we might prefer to identify them as expensive ones) than in routine decisions. In fact, on all those occasions when we built an inductive knowledge base for an organization, it was a returning user – so our first decision support was always for an original decision with a deductive knowledge base. The third way of reasoning, the reduction, was also very well received; it way mostly used to simplify the original decision by going through a deductive-inductive-reductive process and only in a few cases directly for delegating decisions or educating the junior decision takers.

The only added feature that the decision takers showed any interest in is the smilies indicating the inconsistencies of the knowledge base, not necessarily to fix them but to think about them. There were two additional features which were useful to the experts rather than to the decision takers: the html export with on-page reasoning and the explanation feature. The Excel-like look and feel was also found useful mostly by the experts only – the decision takers do not use spreadsheet applications.

What are the missing features? What the decision takers and the experts would be interested in that Doctus does not provide at the moment? As with the previous features, the requirements of the decision takers and the experts are very different. Which ones to include in the new development?

The experts liked the analysing tools which facilitate better understanding and would need a few more of the sort; e.g. they would like to have ‘what if...’ and ‘how to...’ type tools on the top of the present explanatory feature. They also liked the external connectivity and the capacity to handle numbers but they would also prefer to have it in a graphical sort of way. Not much of an invention would be needed; the SPSS statistical software has these features they only need to be copied. These are nice-to-have features, not essential requirements. The most claimed feature, the one that is sufficient in itself to prevent the experts from using Doctus is the reporting feature. If we want the experts to want Doctus this is a first priority development. However, we decided otherwise. It was always the decision taker we supported, not the expert. So we shall focus on the needs of the decision taker rather than on the expert. We cut off all of these requirements and will continue to use the present version of Doctus to work with the experts. The new product will support the decision taker.

The decision takers want an easy-to-use way of playing around with the decision. This means that the UI should be intuitively obvious, like driving a car. They would like to have a look at the whole knowledge base from various angles (different types of display) and to be able to amend it quickly and easily. This should also include the possibility to add complete knowledge bases (created and maintained by the experts) on the input side (in deduction), without needing to examine these knowledge bases. To satisfy what we now see to be the expectations of the decision takers we decided to develop a new version of Doctus from scratch; all the features that the decision taker is not interested in will be removed (and that includes most of what we have added in the past decade), and we will try to make it really easy-to-use and playful. To reflect this approach we codenamed the new product Doctus Light. To make it available anywhere anytime, it will be a browser-based application.

References

Anderson, John Robert (1983) *The architecture of cognition*, Harvard University Press, Boston, MA.

Baracskai, Zoltán (2000) A kukában és az Interneten is böngésznek (Browsing the Garbage Can and the Internet), *Vezetéstudomány (Leadersip)*, 31(3), 42-46. [Published in Hungarian]

Baracskai, Zoltán, Dörfler, Viktor & Timár, András (2003) *Doctus Documentation*, retrieved 02/08/2007, from http://www.doctus.info/htms/doc/

Baracskai, Zoltán & Velencei, Jolán (2004)... *i u e-doba odlučuje čovjek* (*Decision Taking in the e-Age*), Sinergija, Zagreb, Croatia. [Published in Croatian]

Baracskai, Zoltán, Velencei, Jolán & Dörfler, Viktor (2005) Reductive Reasoning, *Montenegrin Journal of Economics*, 1(1), 59-66. Electronic version: http://www.mnje.com/Images1/MJ1.pdf

De Bono, Edward (1976) *Practical Thinking*, Penguin Books, London. Originally published as 1973.

Dörfler, Viktor (2005) Rationalities for Decision Taking, *Strategijski menadžment*, 10(1-2).

Gurteen, David (1998) Knowledge, Creativity and Innovation, *Journal of Knowledge Management*, 2(1), 5-13. Electronic version: http://lysander.emeraldinsight.com/vl=5347735/cl=109/nw=1/rpsv/cgi-bin/linker?ini=emerald&reqidx=/cw/mcb/13673270/v2n1/s1/p5

Kotter, John P. (Ed.) (1999) *John P. Kotter on What Leaders Really Do*, Harvard Business School Press, Boston, MA.

Kotter, John P. & Cohen, Dan S. (2002) *The Heart of Change: Real-Life Stories of How People Change Their Organizations*, Harvard Business School Press, Boston, MA.

March, James G. & Simon, Herbert Alexander (1993) *Organizations* (2nd edition), Blackwell, Oxford.

Minsky, Marvin L. (1994) Negative Expertise, *International Journal of Expert Systems*, 7(1), 13-19. Electronic version: http://www.ai.mit.edu/people/minsky/papers/NegExp.mss.txt

Polanyi, Michael (1962/2002) *Personal Knowledge: Towards a Post-Critical Philosophy*, Routledge, London.

Polanyi, Michael (1966/1983) *The Tacit Dimension*, Peter Smith, Gloucester, MA.

Quinlan, J. Ross (1986) The Induction of Decision Trees, *Machine Learning*, 1(1), 81-106.

Quinlan, J. Ross (1993) *C4.5 Programs for Machine Learning*, Morgan Kauffman Publishers, San Mateo, CA.

Quinlan, J. Ross (1996) Improved Use of Continuous Attributes in C4.5, *Journal of Artificial Intelligence Research*, 4, 77-90. Electronic version: http://www.jair.org/papers/paper279.html

Ryle, Gilbert (1949/2000) *The Concept of Mind*, Penguin Books, London.

Sen, Amartya Kumar (2002) *Rationality and Freedom*, Harvard University Press, Cambridge, MA.

Simon, Herbert Alexander (1977) *The New Science of Management Decision* (3rd edition), Prentice-Hall, New Jersey, NJ.

Simon, Herbert Alexander (1987) Making Management Decisions: the Role of Intuition and Emotion, *Academy of Management Executive*, 1(1), 57-64.

Simon, Herbert Alexander (1997) *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization* (4th edition), The Free Press, New York, NY.