ELSEVIER

Research Policy 30 (2001) 1355–1372

research policy

www.elsevier.com/locate/econbase

Expert systems: aspects of and limitations to the codifiability of knowledge☆

Robin Cowan*

MERIT, Faculty of Economics and Business Administration, Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands

Abstract

The creation of expert systems is one way in which knowledge is codified. In creating an expert system, in general there are three aspects to the codification process: creating a model of the knowledge to be codified; creating the language in which to express the model; and writing messages representing the knowledge in that language. These aspects have different relative importance in different contexts. Referring to four case studies, the paper argues further that codification through creating expert systems is not uniformly successful and part of the variation has to do with the nature of the knowledge, or intellectual process or activity being codified, and the difficulty in creating the model. Activities with fixed goals and linear processes lend themselves very well to this form of codification. Processes of categorization and analogy, such as fault detection and repair, are partially successful, tending more often to create systems for experts. Processes involving balancing conflicting goals have, to date, tended to be unsuccessfully codified. These process types involve knowledge or understanding at deeper and deeper levels of abstraction of the overall processes and overarching goals. The more abstract and less concrete the knowledge involved in the task, the more difficult it is to codify it.

Keywords: Expert system; Codifier; Knowledge codification; Tacitness

1. Introduction

Knowledge codification involves turning knowledge, or parts of it, into messages that can be processed as information. Consequently, codified knowledge is constituted by codes or messages. The codes or messages are expressed in symbols, but for these symbolic representations to be useful, both the representational rules (grammar) and the notation (vocabulary) must be stable, and to some extent, standardised. There must be an agreement between or among the codifiers and users of the codified knowledge, who of course could be the same individual or entity, regarding the meaning. Knowledge so codified is easier to distribute, store and recall. These activities are all valuable, and codifying knowledge lowers the costs of all of them.

Whether or not all knowledge is in principle codifiable, in practice some is not. This can be seen simply as a matter of cost — the activity of codification has an output, and this output has value, so the activity has benefits. But it also has costs in terms of labour, capital and material. Thus, whether a piece of knowledge is codified will depend on the relative costs and benefits of doing so at each level of the process. Of course, the evaluation of the rela-

☆ This paper has been prepared under the EC TSER Programme's TIPIK Project (Technology and Infrastructure Policy in the Knowledge-Based Economy — The Impact of the Tendency Towards Codification of Knowledge).

* Tel.: +31-43-3883-878.

E-mail address: r.cowan@merit.unimaas.nl (R. Cowan).

0048-7333/01/$ – see front matter
PII: S0048-7333(01)00156-1

tive costs and benefits can change over time and with circumstances.

Cowan et al. (2000) propose a taxonomy of knowledge as a way into the issue of codification and codifiability. All deliberate activities involve knowledge on the part of the actor. That knowledge can be partitioned in a variety of ways, but for our purposes initially, we can see it as codified or not.[1] To be codified, knowledge must at the least be articulable. So, there are two explanations for the non-codification of a piece of knowledge: either it really is inarticulable and therefore uncodifiable; or relative to the benefits, the cost of codification is too high. In either case, the knowledge will remain tacit. When this is so, any storage, transfer, or use of this knowledge must continue to rest in the hands of the experts. Given that the time and the resources are finite, and making quite natural assumptions about the distribution of features of knowledge it seems safe to state that some knowledge will remain uncodified, and therefore the domain of human experts.

To examine the proposition that some codifications can be too costly (or simply impossible) in some cases, this paper discusses recent attempts to codify knowledge through the development of expert systems in several different contexts. Because writing an expert system is an explicit attempt to transfer knowledge from a human to a machine through some sort of message creation process, this seems a good place to look for insight into the process of codification. In a discussion of knowledge acquisition for expert systems, Tzafestas writes that the knowledge which permits experts to do their jobs, and indeed to *be* experts, is largely tacit. "The knowledge engineer has to explore this tacit knowledge and to present it in a form which can be understood by the experts (for verification) and by the users for beneficial use" (Tzafestas, 1993, p. 30). This paper argues that in the context of expert systems there is some knowledge that can be codified (turned into an expert system essentially in its entirety), some for which this is partly possible, and some for which it is basically impossible given the state of today's technology. These studies we refer to demonstrate attempts to model the kind of knowledge typically employed in industrial settings. We look specifically at the expertise of three different types of workers: the artisan, the repairer and the strategist, and the differences in the nature of their expertise, and ask what it is about these different tasks that makes human expertise easy, hard or impossible to capture in codified form.

But before beginning the analysis, a brief excursus on the nature of the codification process is in order.[2]

2. The codification process: messages, models languages

The most widely recognised aspect of the codification process in which knowledge is transformed into information, is the *creation of messages*, expressing in a symbolic representation existing or created, knowledge. Creating messages often refers to the actual process of distributing information, for example, one person telling another or writing down what he knows. That a message can be successfully created, though, implies the existence of some infrastructure. There must be a language in which the message can be "written" and "read". But a language which is suitable to codify the knowledge in question itself pre-supposes both a model of the phenomenon and a vocabulary with which to name parts of the model and their interactions.

It is important to point out that in creating an expert system, say a system to do medical diagnosis, the codes we refer to are not the messages to the user: "What is the infection? .... From what site was the culture taken? .... Is the organism rod or coccus (etc.)? .... This is a streptococcal infection with a certainly factor of 70%", but rather the representation of the knowledge (of a doctor) that underlies this diagnosis. What is codified, for example, is the tree structure of the inquiry he uses to arrive at his diagnosis, for example: "If the organism stains grampus AND has coccus shape AND grows in chains THEN

[1] We should point out that often activities which appear to use uncodified knowledge are, in fact, using knowledge that has been codified but internalised by the agent. An economics professor, for example, uses the knowledge stored in principles text books all the time, but rarely refers to the books themselves. This is a case of codified knowledge that has been internalised, referred to as an example of a "displaced codebook" by Cowan et al. (2000).

[2] For a more complete discussion of this process, see Cowan and Foray (1997).

there is suggestive evidence (0.7) that the identity of the organism is streptococcus". [3]

In its most general form, then, codification is a process which involves not only the creation of messages, but also the *creation of models*, since modelling knowledge is a prerequisite to transforming this knowledge into information. In this sense, codification is a central method for producing knowledge. Codification cannot in general be considered as a simple transfer or translation operation, then there is always at some stage, an aspect of creation. This aspect of codification can entail fundamental transformations in the way knowledge is organised, so the codified knowledge-base cannot cover exactly the tacit knowledge-base for which, in some sense, it acts as a substitute. This is particularly so in the case of expert systems: "the imitation of expertise, because it is a process of automation of knowledge, is possible only at the expense of the "active" transformation of this knowledge; it is hence in itself a creator of expertise" (Hatchuel and Weil, 1995, p. 25).

Different types of knowledge demand different types of languages — music, mathematics, expert systems, novels, all have different languages associated with their codification. Some languages are "generic" and can express various variety types of knowledge (for instance, it is possible to some degree to write mathematical problems in natural language); others, like some computer languages or the jargon of esoteric academic disciplines, are very specialised. These languages must be developed before any messages can be written. Development may be undertaken as an explicit activity or it may emerge from the activities of those working with models and messages. But in either case there is a cost implied, as people expend time and other resources in activities that tend to the creation of a stable, standardised vocabulary.

A language must exist, but central to any language are concepts and vocabulary. It is the creation of these two things that the modeller is doing. The existence of a vocabulary, which pre-supposes a model (so that if one does not exist it must be created), is necessary for the ability to create messages. [4]

References to the languages mean that a minimal requirement to be a potential user of the information is that one must understand the language in which the knowledge is recorded. [5] This is true whether the language is English or French, a computer language or mathematics. Knowledge is easier to codify and codified knowledge is more effectively diffused within a community made up of agents who can read the codes (We should point out that the ability to read the code can be an important form of tacit knowledge.) Diffusion and the use of codified knowledge are thus dependent upon the initial investment required to build a community of agents, a clique or a network the members of which can read the codes. This ability to produce and receive signals in a language, even a very common one, requires prior and irreversible investment (Arrow, 1974, p. 39).

The relation between the three layers is complex: sometimes pre-existing languages and models are available for the codification process; sometimes codification requires the creation of a new language and some new modelling. In most cases, there is some degree of creation at both the levels of languages and models.

Expert systems designers often refer to the knowledge (or expertise) acquisition issue (see, for example, Tzafestas, 1993, Chapter 2) and the way they describe it can be interpreted in the terms set above. For example, there are two approaches to fault diagnosis expert systems: "model-based approach (using mathematical techniques of analytical redundancy) and knowledge-based approach (which tries to imitate the reasoning of human fault diagnosers and operators)" (Tzafestas, 1993, p. 39). In both cases a model of some sort must be built: in the first case, a model of the machine; in the second, a model of (one part of) the thought patterns of an expert. The

[3] These examples are taken from the MYCIN system, one of the first expert systems, designed for medical diagnosis of certain blood infections (this is rule 037), see Davis et al. (1977).

[4] For a discussion of this and related arguments about the philosophy of science and the philosophy of reference, see the works of Putnam (1987), for example.

[5] The word "recorded" is deliberately ambiguous. The codifiers must understand the language through which the knowledge is systematised and stored. Users must understand the language in which messages are written. So, for example, developers of a medical diagnosis expert system must understand Lisp, but users of it must understand something like a dialect of a natural language.

Tiger fault management system for gas turbines has different layers in which these models can be seen explicitly (Milne et al., 2001). Layer one is simple data gathering through fault detection instruments. In layer two, a model of the turbine is employed to predict the value of a measurement based on a complex combination of several inputs. This prediction is compared with the actual measurement to check whether a fault has occurred.[6] In the top two layers, the output of layer two is combined into higher level conclusions. It is here that the model of the expert's reasoning is embedded.[7]

Of course, a few expert systems are pure cases of one type or the other, and typically building a system involves both kinds of knowledge. Some knowledge is acquired from written documents. Construction of the fault diagnosis system for compressors at the Toyoda Automatic Loom Works used books, manuals and service guides. The service guide illustrated each part of the compressor and gave information about the possible failure causes. The existence of these documents provided the basis for a consistent language, and each contained structured descriptions (which can be seen as a model) of different parts of the machine in question. This greatly simplified building a model of the machine. Other knowledge comes from the experts themselves.[8] Part of their expertise had been codified before the construction of the expert system began. There existed a decision tree diagram, "designed by the human expert ... (and) used to instruct workers ... on the expert's fault finding technique. Some of the significant rules in the knowledge base are comprised of the cause and effect relationships as represented in the tree diagram" (Chen and Ishiko, 1993, p. 197). The diagram is clearly a model of the reasoning of the expert.[9]

3. Four cases of expert system creation

In this section we examine four expert systems, drawing attention to the types of knowledge at issue, problems in their creation and what in some cases appears to be inherent limitations in codification.[10] Each has a different character regarding the extent to which it can be codified, and the problems inherent in its codification.[11]

[6] FAUST3, used for fault analysis in electricity supply networks has a similar structure: "FAUST3 uses a hybrid of 'surface' and 'deep' strategies whereby the surface model is used to produce preliminary hypothesis about the distribution system. These hypothesis are then simulated using a 'deep' causal model and the results are compared against the real observations" (Burrell and Inman, 1998, p. 115).

[7] More recent research on expert systems proposes to adopt an extremely abstract modeling strategy, namely permitting the system itself to build the model through learning algorithms such as neural networks. The suggestion is that this approach will perform better in more dynamic environments, see, for example, Ruiz et al. (2001). Yet another approach would make the deep knowledge of the expert the core of the system. Instead of pursuing an inverted tree structure for the decision process, this suggestion is to model directly the cause and effect belief system of the expert, thereby going straight to the heart of the machine or problem. This would create a highly non-linear model of the technology, as real cause and effect belief systems (which could of course differ from the actual cause and effect mechanisms) tend to contain "cyclical loops and multiply connected nodes" (Leung and Romagnoli, 2000, p. 2490).

[8] The former is sometimes referred to as "deep knowledge", the latter as "shallow knowledge".

[9] Of course, there are now expert systems aimed to assist in creating these crucial models of the reasoning of the expert. For example, Mapcar is "mainly concerned with the problem of the elaboration of idiosyncratic problem solving models, i.e. models that correspond to how human experts solve problems. Its objective is to provide a support to the construction of an abstract vocabulary (modeling primitives) and a problem solving control that will permit capturing domain experts' expertise in an adapted way" (Tchounikine et al., 1998, p. 305).

[10] The cases we discuss in detail are those presented in Hatchuel and Weil (1995). We have used their description of the cases but have re-interpreted them in terms of the theoretical views on codification discussed above and in Cowan and Foray (1997) and Cowan et al. (2000). There are many case studies of expert systems in the literature (see, for example, Kaewert and Frost, 1990; Badiru, 1992; or Tzafestas, 1993 as three collections), and we refer to some of the similarities between these and those of Hatchuel and Weil (1995).

[11] Hatchuel and Weil (1995) refer to three different types of knowledge: doing know-how; understanding know-how; and combining know-how (for a discussion of different types of know-how, see Lundvall and Johnson, 1994). Following Cowan et al. (2000), to the extent possible we avoid discussion of types of knowledge. Instead we focus on types of knowledge activities, attempting to avoid debates about whether we are observing a case of this or that type of knowledge. Further, it is in the end, codification of the knowledge involved in particular activities that is important, rather than codifying certain types of knowledge, thus by focusing on activities rather than knowledge we can, perhaps, avoid having to map from an economic activity to its type of knowledge and thence to its codifiability, and attempt to move directly from a knowledge activity to discussions of the ability or usefulness of codification in that setting.

3.1. Fixed goals and linear processes: the artisan's expertise

The job of the "artisan" involves achieving some well-specified final goal (transforming an input into an output) through a sequence of known steps. Planning production is a typical task: "Planning is a process that searches for a sequence of actions (or operations) that will achieve a goal statement" (Chang and Wee, 1993, p. 291). This expertise can often be represented explicitly as a list of instructions (think of a recipe book or an instruction manual) and is sometimes referred to as technical- or doing- know-how. This expertise allows the artisan to determine the steps between an initial state and a desired final state, passing through intermediate states by well-recognised and determinate steps or actions. It is typically easily and effectively archived, and expresses the "simplest" form of knowledge in the breakdown of industrial job expertise, that is, knowledge that can be transformed into a more or less linear process of discrete steps (or actions/transformations) with a predictable outcome. This is the type of situation that we examine in the first case.

3.2. TOTEM: automated production routings

Around 1992, a medium-sized firm that processes gold, silver and platinum experienced a rapid increase in the number and variety of products it manufactured. This implied a significant increase in the number of "production routings" the firm had to create and execute. A production routing is a document specifying the form and quantity of some required raw material, the machines best suited for each operation in its transformation, and a sequence of operations for producing the final product. For this firm more than half of the firms orders were unique, so each order created a new problem, and the firm typically processed several orders at the same time, often using the same machines. Rather than produce a huge production routing library, and in order to keep pace with an accelerated and more complex production schedule, the managers decided to develop an expert system to generate production routings. The system was constructed in a shell called TOTEM (Traitement Optimisé des Temps et des Matieres), chosen for its ability to represent the planners' expertise in the form of simple rules.

TOTEM began as an empty structure and was gradually fed the expertise needed to produce coherent and accurate routings. Expertise was formally expressed as conditional rules of the "if ... then" type: "If thickness decreases by 10% then length will increase by 10%", which were elicited from the employees' descriptions of their decision-making processes.

As the system was being created, employees enunciated facts about production routings, and each new fact created a new inference rule, thus propagating the system's reasoning until no new facts were being discovered. At this point the system could be used to create complete routing documents. Each rule represented an extremely small fragment of expertise, but adding it to the system must be seen as the creation of a new message — codifying one more piece of knowledge. These messages were meant to express all of the relations among the elements of the metal refining process, including the transformation of various characteristics of the product as it passed through its route, the choice of operations and machines, as well as specifications for machine settings and so on.

While formulating the rules it became clear that to avoid excessively time-consuming searches by the system (which evaluates between 500 and 1000 parameters per routing calculation), it was necessary to structure the knowledge in different levels. To do this effectively involved parsing the knowledge into pieces that fit together "at the same level". Implicitly, this involved creating a model both of the decision-making process and of the knowledge involved in it. Herein lay the modelling process. Level one distinguished knowledge according to product families. Level two broke this knowledge down into autonomous areas of expertise, such that, each could deal with a different aspect of the problem (e.g. expertise on a particular machine or a specified process). The areas of expertise were then arranged hierarchically and divided so that the system only worked on a small number of potentially useful rules at one time. To ease computation requirements, the system was not constructed to search for the optimal routing, but rather merely for one that met the requirements satisfactorily.

The process of codification created several thousand "messages", each one representing a rule used by the firm's routing experts. These "fragments of expertise" corresponded fairly well to the way planners described their own process of creating a routing.

At this level, the creation of the system represented a relatively straight-forward transfer of knowledge from humans to machine code. The model of the knowledge was a relatively simple construction of a set of "if ... then" statements and their implications regarding connections between different parts of the system. While modelling, the knowledge created a coherent system and explicitly described the links between different parts of the knowledge and the process of decision-making it created few strikingly new concepts. Thus, it was possible to express the model using natural language, though due to the "if ... then" structure of the rules there was of course heavy emphasis on Boolean expression. In the production planning process, the transformation of thoughts into decisions represented a relatively linear process and thus one that was not overly complex to model. However, as the number of rules increased, the costs of using this simple model (of a list of decision rules) rose, and in order to maintain efficiency and flexibility, the model was adapted to better suit the requirements of the orders and the constraints of the shop floor. The general expertise was organised into a detailed structure of different areas of expertise. This micro organisation of the rule structure created a macro-level process that closely resembled a manually created production routing, but in fact the "reasoning" of the expert system differed from that of a human route designer. This suggests that there was non-trivial knowledge creation in constructing the model on which the system was based.

This system successfully automated a task that had been done manually, and did so on the basis of the knowledge or expertise of the operators. It codified the decision-making process, though at the micro level the model implemented turned out to be different from the model implicitly used by the manual operators. Nonetheless, the system created a macro structure which, when used, closely mimicked the performance of the humans.

Other examples of similar success stories exist. Digital Equipment Corp (now absorbed by Compaq) developed expert systems in many areas of production and management. MATCHER matches the components used in cancelled orders to those required by orders in progress in order to save re-stocking time. Build-Area Selector assigns production of orders to build areas and schedules the orders of production (Kaewert and Frost, 1990). Texas Instruments has created an automated manufacturing centre in which an expert system controls robots, vision systems, machine tools, sensors and so on. Given data on monthly part needs, the system plans, creates material lists, schedules production and monitors the equipment. Human involvement is mostly restricted to entering parts lists (see Herrod, 1988). A system developed by InSol Inc. creates bills of materials-requirements for warehouse storage systems. Similar to the Texas Instruments system, the operator enters data on requirements, in this case the nature of the structures to be built, and the system produces the bill of materials using processing that is completely transparent to the user (See Jindia, 1990). [12] In all of these cases, the decision making of humans has been successfully represented in an expert system.

TOTEM and these examples have several things in common which jointly make it feasible to create an expert system in which human expertise is (close to) entirely captured. In all of the examples the problem solved by the human and then by the machine expert was a case of straightforward means-ends problems or decision structures. Given the goal, there are specific, necessary steps that must be followed to realise it. In this case, "backward chaining" is possible: one can "backtrack from a goal to the paths that lead to that goal" (Badiru, 1992, p. 24). But backward chaining is possible only when certain conditions are satisfied. First, it must be that the goal of the activity is known in advance, and does not change as the activity progresses. When a goal changes as a consequence of trying to achieve it, backward chaining is simply not possible. Intimately connected to this condition is that the reasoning employed in the activity be relatively straight-forward means-ends reasoning. This means-ends reasoning is itself decomposable into a sequence of linear steps or stages each one of which lends itself to its own backward chaining. So, an identifiable sequence of intermediate steps or goals, leading to the final goal, each one subjected to simple means-ends logic creates a natural situation in which backward chaining can work very effectively. For chaining to be effective, though, one must also be able to enumerate the entire set of problem–solution combinations. Without this completeness, and in

[12] For more examples, see Badiru (1992), especially Chapter 11.

addition a relatively low level of uncertainty in these combinations, codification will necessarily be incomplete, and any expert system will demand significant human intervention.

The TOTEM case exhibited all of these properties, so backward chaining, and a complete expert system, was possible. (The success of the other examples listed above suggests that they too satisfied these conditions, see Badiru, 1992.) The problem–solution rules in these cases (and in TOTEM in particular) tended to be of the simple "if ... then" structure so the codes or messages they created were relatively straightforward, as individual messages and self-contained (and already existed largely intact, complete and coherent in the minds of the human planners). It was also possible to identify all of them. The sequential structure of the building process made this feasible. Any time the shop encountered a situation that was not already included in the codes, the operators were able to describe the rule they used to make the required decision. Simple "if ... then" logic, combined with the fact that the decisions could be taken to a great extent sequentially, created a tree structure for the route design process.[13] Further, the processes involved were very linear and could be described in a sequence of discrete steps which must take place in a particular order. The process between initial and final state could be broken up into a sequence of largely self-contained sub-processes which simplifies greatly the structure of the expert system. This is true both of the physical processes that are being planned (a production path, inserting surplus parts into an existing production plan) but also of the planning itself.

In the creation of the system, TOTEM and the others mentioned all exhibited "exploratory neutrality". That is, the only goal in creating these systems was to automate well-specified operational sequences that already existed. Building the system was not intended to create new knowledge or information. New knowledge was created in TOTEM, in that a new model of the decision-making process emerged, but this can be seen as a new description of an existing process rather than to find new information. Further, the systems were not being designed to make any choices or evaluate any trade-offs between over-arching goals such as quality, cost, delivery times and so on. Development of the knowledge environment did not raise any issues beyond itself. Thus, the task of the codifiers was well-defined, and very circumscribed — a very particular process was to be automated.

The relative simplicity of the implicit logical structure and the presence of exploratory neutrality together imply that the modelling process in TOTEM (and presumably in the other examples as well) was in important respects relatively simple. This is not to say that it was easy or fast, rather it is to say that the model itself was not overly complex. Knowledge was divided into identifiable pieces which interacted in relatively straight-forward ways. When the TOTEM system ran into the performance limits of the first version of the model, the necessary modifications were significant in terms of performance, but not excessively costly. Thus, codification created new knowledge in the form of a new model of the process, but because the costs of doing so were relatively low, codification was relatively complete.

It is also worth pointing out that in the TOTEM example the knowledge environment was relatively stable.[14] Materials, machines and operations may have varied according to the demands of specific orders, but the basic component operations involved in turning the inputs into outputs were not changing, though of course which of them were used and in which sequence changed from order to order. This meant that

[13] In the case of TOTEM, this is in fact a simplification, though not a gross one. The system is given certain parameters (number of output units, size, material, tolerances and so on) and on the basis of those explicit parameters generates as much of the routing code as is possible, using the Boolean decision structure described. Then, having completed that stage, it calculates further, implicit parameters (such as yield coefficients), and makes further inferences based on those parameters. Note that even in this more complex procedure the algorithm is very linear.

[14] See Cowan and Foray (1997) for a discussion of codification in stable and unstable knowledge environments. The general idea is that every knowledge activity takes place within a broader context or knowledge environment. Sometimes this environment is changing, as would be the case, for example, if there was an episode of rapid, extensive advance in knowledge or technology, or if a new field was opening up and there was not yet consensus about how to formalise it or how to proceed in investigating it. On the other hand, often knowledge activity takes place in a stable environment: there is consensus on how to model the class of phenomena at issue, disputes over modeling, jargon, vocabulary and so on have been settled, and much of the activity is standardised and perhaps even routinised. The economics of codification are very different in the two cases.

essentially once the materials were obtained and the machines were available, the expert system could do its work without interference.

3.3. Categorisation and analogy: the repairer's expertise

A very common activity in which expert systems are employed is fault diagnosis and repair. These sorts of systems are used in medical diagnosis, machinery repair, fault-finding in industrial processes and so on. It is almost universally the case, though, that these systems are designed to assist experts rather than to replace them:

> The maintenance of a complex item of equipment involves a diagnostic procedure incorporating many rules as well as judgement decision by the maintenance personnel. Experience is an important factor in determining the ease with which a mechanic can locate a failure in a component and implement the appropriate correction. Expert systems are now being utilised *to assist maintenance personnel* in performing complex repairs by presenting menu-driven instruction guides for the diagnostic task (Badiru, 1992, pp. 294–295, emphasis added).

Diagnosing and repairing any fault requires a combination of activities designed to solve a problem that is often well-specified only in a very general sense, and is in addition often a problem that has not been seen before. This task is typically more complex than that of the artisan, involving as it does both investigation and analysis in order to first find the cause of the problem and then to determine a solution. Rather than simple deduction, typically what is involved is an inductive process in which identifying the cause of a problem is done through generalisation. That is, a problem is recognised as belonging to a certain class of situations all members of which have the same cause, and thus demand the same or similar corrective measures. To complicate matters, however, this "cause" may in fact be "multiple" in the sense that several causes produce this class of effects, or worse yet, the effect may belong to the "cause unknown" class of situations. In the first case, the operation of repair is relatively straightforward, *once the identification is complete*. In the second case, the repair proceeds through trial and error, though on a circumscribed set of causes. In the final case, the repairer's expertise is often manifest as "having the right hunch" or remembering what appears to be a similar repair done in the past. Even in the first case, where a single cause is determined, the identification of this problem as belonging to a specific class can create significant difficulties. While deductive algorithms have been developed and are highly advanced, inductive algorithms are much less so. Thus, a major challenge in codifying the expertise of the repairer lies precisely in this inductive step. Fault diagnosis systems often attempt to do this by classification. Classes of faults are generated through historical records of previous faults, and the system attempts to put the fault it is diagnosing into one of the classes. The systems DIVA and EXACT had similar performance features: their diagnoses were based on historical generalisations and classification. They are fast (EXACT is claimed to be as good as a human in terms of speed and accuracy) but they cannot solve problems "which have never been encountered in the past and are thus not included in the knowledge database" (Chen and Ishiko, 1993, p. 206). [15]

Clearly, repair cannot be described as a linear process from start to finish with a rehearsed and predictable outcome; rather "... it intermingles action and investigation in an ever-changing pattern" (Hatchuel and Weil, 1995, p. 36), sometimes unsuccessfully. Therefore, unlike knowledge which is codified, and in a sense "fixed" in a recipe book or a dictionary, "the repairer's expertise cannot be defined out of context, it requires that the practical and social framework of the reparation (sic) be constructed simultaneously" (Hatchuel and Weil, 1995, p. 37). The difficulty, indeed, lies in the fact that the repairer cannot simply follow a set of rules, he must use his judgement.

3.4. Cornélius

The manufacturing sector has undergone rapid and extensive changes in recent decades, and the introduc-

[15] DIVA is used "to help the operating personnel to interpret the evolution of vibrations and to diagnose the developing faults" in turbine generators (David et al., 1993, p. 212); EXACT is used for compressor trouble-shooting (Chen and Ishiko, 1993). The quotation refers explicitly to EXACT. When DIVA encounters a problem it does not recognise, it gives the user the message "I do not recognise your problem! I guess I am not (yet) competent enough ..." (David et al., 1993, p. 218).

tion of "mass customisation" has increased the complexity of both manufacturing processes and machinery. A firm in the parts manufacturing sector studied by Hatchuel and Weil (1995) responded with an automation programme, thereby increasing the complexity of its machinery base, and increasing the need for maintenance of a new, more complex capital. In this context, an expert system devoted specifically to industrial diagnosis and maintenance, Cornélius, was proposed.

The facility chosen to test the feasibility of automation was a flexible manufacturing cell in the machining workshop. It was thought appropriate for three reasons. First, because the factory's entire output passed through this cell, it was at a risk of becoming a production bottleneck. Thus, the ability to diagnose and repair problems quickly and accurately, and to an extent automatically, was highly desirable. Second, its problems were representative of those experienced by other NC machines in the plant, so an expert system that could handle the problems of this cell could easily be adapted to other areas of the plant. Third, there were several experienced operators and maintenance workers who could lend expertise to the project.

The task of the expert system was to find the source of any failure, to diagnose it and to instruct the operator on how to effect repairs. The underlying rationale was that an important source of savings would be to shift at least some of the repair tasks from maintenance personnel to the operators, who were not explicitly trained in maintenance. Cornélius was chosen as the software package because of the relative simplicity of its basic concepts and what appeared to be the close relationship of its reasoning to that of a theoretically trained technician.

Given the diagnostic nature of the project, a system that represented knowledge in a set of simple "if *A* then *B*" statements was not enough since, due to the pervasiveness of multiple possible causes of defects, the statements would actually be of the form "if *A* then *B* or *C* or *D* or . . . ". Further, since a large proportion of the breakdowns had some unique aspect it would often be unclear which rule would apply. The problem faced by the expert system was much more like unravelling a mystery than simply applying diagnostic rules. To solve this puzzle, a system would somehow need access to a model of the internal functioning of the machine, and in particular to the way in which different components related to each other. Eventually, the system was designed in two parts: a generic application that reproduced a certain mode of reasoning, and a knowledge database which provided the raw data to which the reasoning was applied. These two parts combined to create a model of the machine and the way a repairer would normally address this model.

How does this system diagnose a machine failure? In its representation in the expert system, the flexible cell is divided into its main component processes, such as a door opening or closing, conveyor movement, tool changes and so on. Symptoms of failure lead the system to invoke particular subroutines: if it receives a "door not open" signal, it invokes the "suspect door not open" function. Since there are a multitude of reasons why a door may fail, the key is rapid identification of the correct cause. Figuratively, the expert system represents each component system and function by a diagram, similar to an electric circuit diagram, the point of which is to describe the components according to their relations with each other (These "diagrams" are stored in the knowledge database mentioned above.) During operation of the cell, signals are sent by each of the components describing its states and actions. The system relies on maps of normal and deviating machine behaviour (which are logical translations of the circuit diagrams) to monitor and act.

The expert system scans the maps, gathering information (by inference or by asking the user questions) on the state of the components. The user's answers allows the system to infer either a normal or deviating state of a component. The diagnostic process continues, either by logical inference or by further questioning of the user, until the faulty component is detected. In case one is not, the system suspects a different function according to the way these functions depend on each other, until no further inferences can be made.

One early difficult conceptual issue was to decide whose expertise was to be modelled — the technician who had installed it, the maintenance fitters who maintained it, or the operators who were responsible for restarting it when necessary. Each of these actors could hold relevant pieces of the diagnostic puzzle. The answer emerged during the process of codification: none alone had expertise complete enough to create the expert system. A more "general" expertise had to be constructed and then encoded. Interestingly, language was a potential stumbling block. Different jargons were in use by these different types of agents,

and this created a potential difficulty in codifying their knowledge in such a way that others could use it. Language began as non-standardised.

The first step of the diagnostic process was to determine the most frequently failing functions. As a consequence, the first modelling task was to identify these functions. This identification was a central part of writing the databases that underpin the system. From here, though, the task of modelling the workings of the system, and the technical inter-relations among the different functions, seemed monumental. Fortunately, it was greatly facilitated by an existing maintenance manual which described components and procedures of available tests.[16] This manual provided a complete description of the flexible cell and its components, the most frequent sources of failure, diagnostic procedures and emergency repairs. From the point of view of creating the expert system, one major contribution of the manual was that in the course of its creation the ambiguities in the technical language had been removed. This solved the language problem, and created the necessary standards.

The members of the group agreed on clear and distinct definitions for the cell's components, and hence much time was saved in the course of formalising the knowledge and in developing the language in which to do so (the procedure took only 2 months.) The modelling process was trickier. It could not consist of a mere translation of functional plans for the cell. Rather, the important diagnostic elements had to be extracted from the exhaustive list of relations in the functional plans.

What caused a great deal of difficulty in designing the reasoning rules was that the typical serviceman's approach could not be described in diagnostic terms alone. In fact, servicemen relied much more on past experience; their first question seemed to be whether a particular failure was the same as or related to one seen in the past. Thus, it was suggested that this system should not start with a list of questions designed to define a cause, but rather with a list of actions to perform based on the preliminary analysis of the causes of the most frequent failures. While modelling the machine was relatively straightforward, explicitly modelling the casual links among the parts of the machine was more difficult. But a dynamic model of the repairer's expertise, which would be necessary to make his expertise useful to a non-expert, turned out to be much more difficult. This is due in large part to the fact that it involved considerable pattern recognition, and an iterative process of recognition and action.

Interestingly, after the first version had been tested, the system's designers re-wrote the knowledge databases to reduce the number of components described, and revised the diagnostic maps to be more economical. The dynamic process of creating the expert system led to the creation of new knowledge, necessitating the creation of new messages and models with which to structure them. Even though in the beginning it was not clear to the project leaders what the final outcome would be, in the end, this knowledge turned out to be extremely useful in increasing the overall reliability of the cell, as part of the knowledge created indicated precisely where the most important and common failures were occurring and to what they were linked.

This particular expert system, however, has not been used in the way it was originally intended, which was to make it possible for non-experts to take on a more active role in repair. It has turned out to be extremely expensive (and time-consuming) to create and maintain the needed knowledge databases. Further, even though specifically designed to be user-friendly, it has proved difficult to use for those without significant amounts of expertise ex ante. The knowledge codified in this system was useful to those who were in a sense the sources of this knowledge, as it formalised their knowledge and the processes they followed. This permitted them to work faster and more reliably. However, use of the system was dependent on their own pre-existing expertise, and in particular on their practical experience.

We observe here an interesting interplay between environmental stability and codifiability. Contrary to a general tendency wherein stability of environment increases net benefits to codification, in this case the opposite seems to be true. Because the environment was stable, the experience accumulated by the operators retained its value and led to ever-increasing ability to make good and rapid judgements about where to

[16] The manual had been prepared recently by a group consisting of a workshop supervisor, two artisans, a representative of the maintenance service and a representative of the innovation group. Notice that this group included several different types of expertise, each of which contributed to understanding how the cell functions.

begin and how to proceed. Again the issue is pattern recognition and generalisation, a venue in which humans in general still seem to have an advantage over computer technologies. Imagine, though, for the sake of contrast, a repairer of either a new machine or a machine that is constantly changing. Here, experience is of little use, and human ability to generalise from past experience does not help. Of more value will be the user or instruction manual, which by definition contains codified knowledge, and is thus more easily embedded in an expert system to aid in the repair. Stability of the environment is central to the ability to codify the expertise of the artisan, and indeed raises dramatically the value of doing so. But here the opposite seems to be the case. While stable knowledge makes codification possible, the stability of the environment makes the value of the codified knowledge decrease relative to the unstable environment case. [17]

In the case of a repairer, the final goal is fixed, and readily identifiable. Nonetheless, backward chaining is not possible. The problem is that the steps to achieve the final goal are not clear before the fact. There are too many possibilities and initially not enough is known to be able to tell the effects of acting on them.

The Cornélius project was only a partial success. While the initial goal was to make it possible for non-experts (namely, operators) to effect repairs to the machine, in fact this did not happen. Nonetheless, some knowledge was codified, and in the process some created, and this codified knowledge was of value to repair specialists. Cornélius is not unique. Descriptions of the other fault-diagnosis expert systems indicate the same feature. General Electric's CATS-1, now called DELTA, was perhaps the earliest and best-known system for trouble-shooting. This system *helps maintenance personnel* in railroad repair shops to isolate and repair a variety of diesel-electric locomotive problems (Pratt, 1984, cited in Chen and Ishiko, 1993). Ford's DEX.C3 *provides assistance to mechanics* at Ford dealerships in diagnosing faults in three-gear automatic transmissions. The PAR (Pulse Acquisition Radar) system of the US Army "assists a PAR mechanic in troubleshooting .... It is intended to complement technical manual instructions and mechanic expertise" (Badiru, 1992, p. 298). "Bell's Automated Cable Expertise ... was designed to ... *help managers* to investigate the data contained in the complaint system" (Chen and Ishiko, 1993, p. 196; cf. Rauch-Hindin, 1986). AT&T Bell's interactive Repair Assistant (IRA) "... *provides troubleshooting advice* to the telephone company's mobile field technicians" (Chen and Ishiko, 1993, p. 196; cf. Horton et al., 1988; Callahan, 1988). [18]

What we observe in the Cornélius case, and what is clear in the descriptions of so many fault-diagnosis systems is, that the knowledge or expertise involved in fault diagnosis and repair is not simply raw material that needs to be memorised and recorded. Instead it is a theoretical construction interlinking choices and raw data, and making inferences from one to the other. The key to effective "understanding" is very abstract and the expertise includes generalisation and pattern recognition as central activities. With contemporary technology, both computer hardware technology and technologies of representation of (abstract) knowledge, these activities are both still very difficult. Both DIVA and EXACT simply stop when they encounter a situation they "have not seen before". Thus, modelling the knowledge used is extremely expensive, and possibly still impossible. The inference-making aspect of this expertise implies that it can be highly context-dependent since the context will determine which parts of both data and logic should be employed. Without a coherent model of the artefact or process being "repaired", messages could not be organised in a way that was easily accessible to non-experts. Experts, though, have internalised this model so the messages were coherent to them. Thus,

[17] In the jargon of Cowan et al. (2000) by the time the environment has stabilised, the codebook has been displaced — the repairer has gained enough experience that he knows pretty well what is in the manual, and his expertise lies in knowledge (gained from experience) of things not in the manual.

[18] The descriptions in the text here are either quotations or paraphrases of descriptions of these systems. (In every quotation, the emphasis has been added.) The idea that appears over and over in descriptions of fault diagnosis systems is the assistance that they provide to already-skilled technicians. There are, it should be noted, a small number of exceptions: *Cooker* checks large steriliser cookers for Campbell Soup. Used day and night to "diagnose cooker problems without the need of consulting human experts". Termi-point Expert Maintenance System (TEMS) (an expert system for supporting wire-bonding machine maintenance) "can diagnose certain machine problems in a few minutes while the human experts solve the same problems in an hour or more" (Chen and Ishiko, 1993, p. 196; cf. Dallimonti, 1987).

the use of the system cannot be separated from the expertise of the user, and codification will necessarily be incomplete.

3.5. Balancing conflicting goals: the expertise of the strategist

Strategy involves making plans and finding the most effective means to carry them out. This contrasts with the tasks of the artisan or repairer, both of whom take their final goals or desired end state as given and merely seek effective means with which to achieve them. The strategist, often a corporate leader, strategic planner or the co-ordinator of a team of professionals must formulate both plans and means simultaneously. Here, the expertise has three aspects. One concerns the tasks to be performed, resources to be mobilised, goals to be met and so on, all of which can be relatively easily defined and recorded. The second involves combining, arranging and co-ordinating events, activities and individuals into a coherent unit or network to decide upon and achieve the desired ends through the means chosen. The third, and most difficult, involves making trade-offs among conflicting goals or objectives. The forms and methods of decision making involved in these activities can be very difficult to formalise.

The problem is that backward chaining is simply not possible. As the problem is explored, the goal can shift, and when this happens, it becomes extremely difficult to compare (in a mechanical way) different courses of action. Of course, there are final goals that do not change: maximise profits of the organisation; get all trains safely and quickly through a station and so on, but the distance between the immediate means and the final end is quite large, and there is no obvious (linear) path linking the possible actions with the final outcome. There can be many possible actions, and tracing each of them to see their effects on the final, immutable, goal is simply too difficult. In this case, a common strategy is to retreat to intermediate goals. For example, the profits of a conglomerate are intimately related to the profits of each subsidiary, thus if it is too difficult to plan for the conglomerate profits, we should look instead at the profits of all the subsidiaries. But notice that in making this move we have changed from a single final goal to several intermediate goals, and these intermediate goals can, and probably typically do, conflict with each other.[19] This is the point at which strategic expertise enters: how to trade these goals off against each other.

3.6. The "Naval" project

An on-going problem for a large multi-national oil corporation was to allocate oil rigs across a variety of prospects. Because both rigs and prospects are very heterogeneous, co-ordinating its activities and those of its subsidiaries so that the most appropriate rig was used at each prospect at the appropriate time could be a significant source of savings. Similarly, to get this co-ordination wrong could be quite costly. Traditionally, this activity involved a great deal of negotiation on the part of corporate planners. To a great extent it was an ad hoc process which evolved continuously as problems in projected assignments were encountered and addressed. In an attempt to make the co-ordination more effective and less costly, the firm embarked on the Naval project. This was an attempt to develop an expert system that could generate the most coherent compromise between each of the operations of the multi-national, given geographical, physical and time constraints.

The system would in principle use two distinct types of knowledge: one that made it possible to list prospective rigs and drilling prospects, and to rank each rig as a candidate for a particular prospect;[20] and one that made it possible to combine different preferences across a large number of prospects into a single coherent plan for rig deployment.

[19] Chang and Wee (1993, p. 291) put it this way: "Normally an original goal statement is too complex or too abstract to achieve through a single operation. It is often necessary to break down the original goal into smaller, simpler subgoals. In an ideal situation, a subplan for each subgoal can be formulated independently and a compete plan can be derived simply by combining subplans. However, in practice, subplans often interact. That is, achieving some goals can actually prevent the accomplishment of others". They were discussing a "simple" production planning problem in which the conflicts were relatively straightforward to resolve. They were considering none of the complexities involved in the compromises discussed in this section.

[20] This issue here is that rigs differ from each other in terms of performance, cost, availability and so on, and prospects differ from each other in terms of technical characteristics, cost of drilling, geological value and so on.

The expert system worked in two stages. In the first stage, the basic elements of rigs (type, weight of derrick, capacity, etc.) and prospects (geology, drilling cost, time constraints, etc.) were described and simply given to the system as data. It then created a list of advantages and disadvantages of each possible assignment, and used those lists to create a score for each rig-prospect combination. More was needed, however, to generate an optimal plan for rig assignment since these scores were based on technical issues, and took no account of the objectives of the different subsidiaries involved. Crucially, these scores made no attempt to balance any conflicts in these objectives. Underlying the list of advantages and disadvantages were constraints, and these were central in the derivation of the assignment scores. But when looked at from different points of view (e.g. from the points of view of different subsidiaries, or when considering different strategic objectives of the form) different constraints had different importance. To bring this into the analysis, the user of the system had to intervene, giving weights to these constraints. This permitted the system to generate a "weighted if ... then" allocation algorithm.

Using this weighted system of potential assignments, the expert system was able to build an optimal programme. The "compromise algorithm" represented a significant advance in the artificial intelligence, codifying a process formerly done by negotiation within a committee. In this sense it was creating expertise (or perhaps unifying it) out of the disparate activities of the committee members in negotiation. Unfortunately the system was limited in several ways. The initial assignment rules had to be simple enough to remain operational, yet general enough to apply to a broad range of situations. Also, although formalised rules contained much, and diverse, content, they were all handled in the same way. Only the weights attributed to them differentiated their importance. But assigning weights in the first place proved to be a daunting task.

The limits to codification arose because it was impossible to assign objective values a priori since the weights (and to some extent the parameters themselves) only had meaning relative to one another, and relative to the shifting goals of the organisations. Furthermore, there was no system of weights that could adequately (or accurately) account for the tactics previously used by planners in a rich and dynamic environment. Each specific context required a modification of the weights to correspond with the most suitable tactic (an issue that was brought to light when the recommendation of the expert system was compared with the programmes habitually negotiated for real cases in the past). Nonetheless, Naval proved useful in generating acceptable programmes *once the experts had learned enough about the system's reasoning* and the effects of different weights. Still, a full-time operator was necessary to master several types of expertise involved in running the system. He had to have a sound knowledge of rigs and prospects, an understanding of the reasoning processes of the expert system, complete knowledge of general planning and compromise activities in the industry, and finally he had to understand the over-arching goals of the conglomerate and the inter-relations among the goals of its subsidiaries. His expertise was not only necessary for assigning weights and guiding searches, but also for explaining and justifying the results. For this reason Naval turned out to be "a system for experts rather than an expert system" (Hatchuel and Weil, 1995, p. 214).

What we see here is that this assignment activity was too complex to be codified completely. In essence, the system was being asked to make compromises that formerly had been worked out through an iterative negotiation process. While some parts of this were encoded, it was only with the help of significant input from an operator who understood why and how some constraints were more important than others (and that this could change over time) that the system could be usefully employed. Combining and negotiating the goals of different subsidiaries involved and creating a plan that enabled these goals to be met to the extent possible involved too much iteration between means and ends. Because the goals as initially stated were incompatible, creating an insoluble problem, they had to be "re-negotiated" as the assignments were developed.

If the goals were to automate, and duplicate, the rig assignment that had been carried out by the committee previously, the Naval project was a failure. The experts were able to provide the necessary rules to create sensible single rig-prospect combinations (also introducing economic rules and preferences concerning contractors or the origin of rigs). However, in trying to create the compromise programme there was an almost total absence of "expertise" that could be formulated by the identified experts. No one person

had handled the task before, and thus it was unclear what exactly was being modelled. Eventually it was seen that this was the negotiation or compromise process itself rather than the skills of any expert. There seemed to be no systematic way of preparing these programmes since the procedure had been handled previously by several committees working continuously, intervening when necessary, trying to reconcile, through negotiations, the most obvious contradictions. In this case a huge innovation in knowledge modelling would have been necessary, involving not only modelling the way in which compromise is struck, but also, and especially, the way goals are adjusted to make compromise possible. The first part, finding a compromise among different objectives, was achieved (and was no mean feat), but in its entirety this innovation was simply beyond reach. The key difficulty was in codifying the weighting procedure which allowed trade-offs to be made among conflicting goals and constraints. The weights having been assigned, the artificial intelligence experts were able to find algorithms to create reasonable assignments, but the key step, assigning weights, had to be done manually.

In this enterprise, codification was not an imitation or simulation of expert reasoning, but a pure innovation in the context of expertise within the firm. Unlike the TOTEM and Cornélius projects in which expert know-how was transformed, in the Naval case a method completely unknown to the experts was incorporated. But while Naval provided the specialists with added support and legitimacy in their co-ordination tasks, the experts could not be done away with.

3.7. GESPI

The GESPI project (Gestion Prévisionnelle des Itinéraires) was an attempt to automate the routing of trains through an urban railway station in France. For us, it provides another example of the limits to codifying the strategist's expertise.

In any station, planners create a plan at the start of each day. But in this case there were 640 possible routes to enter the station, 30 tracks through it, and 6 tracks serving 30 platforms. This is an enormous combinatorial problem: a train has 1400 possible routes through the station. But advances in artificial intelligence technology (and the facility of computers in dealing with combinatorial problems) suggested that an expert system could be an attractive way of solving this repeated problem.

Any complete expert system would have to cope with three management levels: creating the long-term service plan twice a year, defining basic summer and winter services; a middle-term plan making basic track assignments for each day of the week; and real-time traffic control in which plans are sent to the signal cabin for dispatchers' use in directing traffic. The third tier of management takes place in real time as it includes on-the-spot, last-minute evaluations of incidents and adjustments to the plans. After looking initially at the possibility of automating daily routing, the project team decided to restrict their plans to the middle-term assignment of trains and tracks to avoid the technical difficulties of operating in real time.

In building the GESPI system the engineers discovered that there were far too many possible combinations of available facilities and constraints in the station to index them, so the planners broke the problem down into rules of routine and rules for managing interventions. There were rules to define trains at risk of conflict, and rules defining trains that were (or were not) assigned regularly to a track. These two types of rules let the planners work out a programme in stages. Initially the system checked the state of facilities and work in progress and ensured that all regular traffic was conflict-free. It then attempted to assign platforms and tracks to the new or relocated trains (about 250 per day) and ensured a clear route for each, checking that the assignments conformed to technical, commercial and traffic constraints. If there were trains it could not schedule, it asked the planner to route them.

The system was built iteratively. When a piece of expertise was recognised, it was formalised and codified. In this it was much like TOTEM, and each piece of expertise was translated into a message. This process constituted a stepwise creation of a model of the information processing and decision-making process. As the model was built, it was tested against actual decisions and revised, much like building a model in any scientific endeavour. This process increased the quantity of the experts' knowledge that was codified (more "rules" were identified for example) and it improved the principles on which the system was based. Initial results were encouraging as "the experts expressed their reasoning in a form close to the heuristics and rules used in artificial intelligence" (Hatchuel

and Weil, 1995, p. 181). A small number of simple rules, codifying the constraints that existed, formed the basis of the system. In the first stage of building the system, technical constraints were addressed, as these were the simplest and most standardised. At the second stage of modelling, commercial constraints were included. These were more complex in part because they involved other departments such as sales and marketing, and thus implicitly involved trade-offs between the goals of the various departments. Further, they existed in a less formalised and more diffuse state (diffused both in terms of their natures and in that they originated in different parts of the organisation) which made codification more difficult. The language in which the expertise was expressed was again much like a natural language, but the model of the planning process became more and more complex as codification proceeded.

As a system, GESPI was able partially to automate the planning task. Users remark, however, that checking GESPI's plans is more work than creating them from scratch. One can conjecture that this is the case because in creating a plan, the logic of the plan is developed, and when a planner knows the general logic of it, it is possible to see whether different parts fit with the general logic. The difficulty in altering a plan of this nature is the risk of creating a cascade of secondary changes that must be made to accommodate the first one. If one understands the logic of the overall plan, certain possible responses can be ruled out immediately because they will cause these cascades later in the routing. If one does not understand the logic, one can only proceed by trial and error. When reading a plan created by GESPI the logic is unknown and has to be deciphered before the internal coherence of the plan can be evaluated. This is a part of the expertise that is extremely difficult to codify. It is in a sense the same issue as the evolving means-ends problem of Naval. The evolution of means and ends is essentially an evolution of the logic of the process. It is in understanding and manipulating this logic that the expertise lies. [21]

Again, in this case it was not possible to transfer expert knowledge completely. Even though there appeared to be a linear approach to the problem of co-ordinating trains in and out of the station, involving mostly an identifiable goal, fixed constraints, and a limited (if large) number of possible actions, it was the practical experience of the planners that enabled them to judge in advance a potentially dangerous situation and intervene if necessary. It was not possible for the programme to imitate the expertise involved in this kind of decision making. Paradoxically, a system designed to be more rapid than human planners was in fact less flexible when it came to making last-minute modifications because it had to reformulate an entire day's programme, whereas a planner could make these changes just prior to sending out his plan or at any time throughout the day, as discussed in the previous paragraph. GESPI used the rules and experience provided initially by the planners to combine the main artisanal elements with the skills required to find the best possible compromise, but the programme was unable to imitate their reasoning. Close knowledge of the logic of the station and the consequent logic of the plan were central to being able to make (especially last-minute, ad hoc) adjustments to an existing plan. This logic proved too difficult to model, and is possibly even inarticulable. Thus, again the result of the expert system project was to provide a system that was of limited use and could only be used by existing experts who were able to provide the missing logic. [22]

In both the Naval and GESPI projects, "expertise" was recognised at the corporate level. One or several highly skilled people were using their skills to perform some important task. Expertise was involved in what appeared, at first glance, to be a task with a relatively stable, repeated objective. Automation should have been possible. The cases differ in that in one, GESPI, the expertise could be located in a single individual (though there were several who, individually, had it); in the other, Naval, the expertise was constituted at the group level — it did not reside in a single individual but in the dynamic of the group.

[21] In the Naval case, especially to the extent that the logic of the outcome is discussed in normal circumstances, it would almost certainly be as ex post rationalisation. After the process has been completed, the logic has appeared. Before the process is completed, the logic of the ends is itself unclear and evolving. To mimic this process then, an expert system would have to be able to change its internal logic as the process evolves and goals shift.

[22] One use of the system was to check the effects of possible modifications to the station, presumably because this was not a time-critical task, and the key question was whether a proposed modification would create impossible problems for train routing. Using GESPI for this sort of task would not remove a planner from his normal duties.

In the GESPI case, several individual experts were identifiable, but a certain aspect of their expertise remained difficult to harness. They possessed knowledge that was highly abstract, and possibly inarticulable; knowledge that is only gained through years of experience in routing trains. Here, planners were developing a plan with some internal logic, and this logic itself evolved as they worked. But by the end of the process this logic had been internalised. Was it articulable? This is difficult to answer. Certainly, though, it would have been extremely difficult, and hence costly to articulate it in such a way that the expert system could have used it the way the planners themselves did.

In the Naval case, two issues complicated matters. First, an "acting expert" could not readily be identified, hence articulating the expertise was a problem. Indeed, the expertise lay in the interactions of the negotiators. Second, and much more important, was that the key part of the expertise expressed in these interactions was the ability to modify means and ends iteratively, that is, the plan revision resulting from negotiations over the different goals of the actors. The fact that intermediate goals were revised as the planners worked meant that the means-ends connections were extremely complicated, and reflexively modified. These "shifting sands" caused immense problems for codification.

The stumbling block in both cases lay in codifying some highly abstract vision of the entire operation, and its abstract goals, while the details of the problem solution were involved with very intermediate, conflicting goals. These intermediate goals changed as the problems were being solved, but remained connected to the over-arching goal through the internal logic of the solution. Shifting goals, and the need to develop and understand the abstract internal logic of the solution, made codification very incomplete at best.

In both cases the activity under consideration involved generating plans for actions which demonstrated compromise between multiple objectives. While at some high, over-arching level, goals were constant (make profits, get the trains through the station safely and on time) the intermediate goals were very flexible, and, due to the fact that they conflicted with each other, of necessity fluid. It was this fluidity that caused problems.

Before concluding, we should ask briefly about the issue of joint versus individuated expertise: Naval versus GESPI. The two cases are different in that regard. GESPI does represent "organisational expertise" pretty clearly. The simple issue of multiple expertises is not important: "... a combination of independent expertises is very easy to be accommodated (sic) in an expert system. Furthermore, modern expert systems can communicate between them and exchange their knowledge and conclusions" (Tzafestas and Adrianopoulos, 1993, p. 28). This is what the designers of GESPI tried to do. The problem was not in combining their factual knowledge (presumably different members of the team had different knowledge about rigs and prospects), though, the difficulty lay in the negotiation among different goals. Each member of the team brought with him or her ideas about the goals of his subsidiary and about the over-arching goals of the multi-national. These ideas were combined into a single plan by some inarticulable process. Would designing an expert system have been possible had the process been carried out by one individual who was able to bring to the task all of these conflicting goals? Could he have articulated the process by which his decision was reached? It seems unlikely. The problem lay not in the multiple personnel aspect, it lay in the compromising aspect. [23]

4. Conclusions

Codifying knowledge involves three distinct but related aspects: creating models, creating languages, and creating messages. Each of these aspects has its own costs, and at each level, as the process occurs, very typically new knowledge is created. The act of codifying is not merely translating the "expert's" knowledge from his head and onto "paper", but is typically an act of knowledge creation. [24] As we have seen in these studies, different knowledge activities involve more or less complex codification processes.

Generally speaking, relatively linear, direct processes with a known and fixed goal, like those pursued by the artisan, can readily be codified into a list

[23] This is not to say that expertise that lies in a group of persons is necessarily the same as the expertise of an individual. But drawing out the implications for codification of the differences is not possible using these cases.

[24] This point is stressed by Hatchuel and Weil (1995).

of instructions or decision rules which can be implemented by a machine. Activities that involve significant amounts of pattern recognition, generalisation and use of analogy are more difficult. Here a deep abstract knowledge of the system being "repaired" is part of the expertise, and this becomes difficult to model and thus to codify. Forming strategies, trading different, conflicting goals off against each other is harder yet. Goals can change as constraints are encountered, so they co-evolve with actions in the process of addressing whatever problem is to be solved. This involves a vision of the system at an even higher level of abstraction, and this aspect of expertise is extremely difficult to codify. Costs of codification obviously increase as we move from the artisan to the strategist, and these rising costs can contribute to the less complete expert systems that it seems possible to produce when artisanal or diagnostic expertise is involved.

We should also note that environment matters. When an environment is changing it means that the expertise in question has to be both broader (in dealing with more situations) and deeper (to deal with new situations). A "complex" task in a stable environment is likely to be easier to codify than a "simple" task in a changing environment. Speaking again in terms of costs, in the stable environment, the fixed costs of creating models and languages will in general be lower and, further, there is the possibility to spread them over more applications. This seems to be true for many types of expertise. But when the central part of human expertise is created through experience (as is the case of the repairer) a stable environment increases the value of this uncodified knowledge, and it can be this expertise that is difficult to codify and this decreases the net benefits of codification.

The TOTEM project was implemented and considered a success. We can conclude from this that benefits outweighed costs. Interestingly, though, economic factors did not play a large part in the decision making of the project. We can conjecture that costs were relatively small. Benefits came not in the form of immediate productivity gains, but rather in the creation of expertise in a form that was possible to use and evolve. Further, the project created unexpected knowledge (for example, an improved definition of ingots) which gave the firm better control over its production process.

While Cornélius was a technical success, in that it was able to codify expert knowledge, it was not a success in diffusing it beyond the initial experts whose knowledge was codified. The project as an expert system for repair has been discontinued, but the codified knowledge has been put to other valuable uses. Codification of the causal structures of failures enabled the firm to increase the general reliability and productivity of the cell.

The Naval project was discontinued in part because of a major change in the oil market, and in part because the labour costs of the project were not bearable in the firm's new environment. In the new environment, the firm no longer performed tasks that Naval was meant to automate. This, of course, makes it difficult to evaluate its (potential) costs and benefits.

The diagrams produced in the process of creating the GESPI program were similar to those produced manually. The main result of the system, though, was to confirm the previous opinion that the station was operating at or above capacity. But the similarity between the GESPI routing and manual routings implied that the GESPI system could be used to run experiments on proposed station modifications. One highly peripheral benefit that was noted was that GESPI provided a beachhead to further office automation. As an expert system, though, it seems clear that costs outweighed benefits.

These four case studies illustrate that expert systems can codify knowledge embedded deeply in skilled humans, and in doing so often create new, sometimes unexpected and valuable knowledge. The studies also show that different types of knowledge lend themselves with different degrees of compliance to the codification process.

Acknowledgements

I am grateful for the excellent research assistance of Lorri Baier and the comments and suggestions received from colleagues in the TIPIK Project, particularly Patrick Cohendet and Dominique Foray.

References

Arrow, K.J., 1974. The Limits of Organisation, Norton.

Badiru, A.B., 1992. Expert Systems Applications in Engineering and Manufacturing New York. Prentice-Hall, Englewood Cliffs, NJ.

Burrell, P., Inman, D., 1998. An expert system for the analysis of faults in an electricity supply network: problems and achievements. Computers in Industry 37, 113–123.

Callahan, P.H., 1988. Expert systems for AT&T switched network maintenance. AT&T Technical Journal 67, 93–101.

Chang, K.-H., Wee, W.G., 1993. A Knowledge-Based Mechanical Assembly Planning System. In: Tzafestas, S. (Ed.), Expert System in Engineering Applications. Springer, Berlin, pp. 291–306.

Chen, J.-G., Ishiko, K., 1993. EXACT — An Expert System for Automobile Air-Conditioner Compressor Troubleshooting. In: Tzafestas, S. (Ed.), Expert System in Engineering Applications. Springer, Berlin, pp. 193–207.

Cowan, R., Foray, D., 1997. The economics of codification and the diffusion of knowledge. Industrial and Corporate Change 6 (3), 595–622.

Cowan, R., David, P., Foray, D., 2000. The Explicit Economics of Knowledge Codification and Tacitness. Industrial and Corporate Change. 9 (2), 211–253.

Dallimonti, R., 1987. Smarter maintenance with expert systems. Plant Engineering 18, 51–56.

David, J.-M., Krivine, J.-P., Tiarri, J.-P., Ricard, B., 1993. Using Prototypical Knowledge in Classification-Based Expert Systems. In: Tzafestas, S., pp. 209–221.

Davis, R., Buchanan, B.G., Shorthifle, E.H., 1977. Production rules as a representation for a knowledge-based consultation program. Artificial Intelligence 8, 15–45.

Hatchuel, A., Weil, B., 1995. Experts in Organisations. de Gruyter, New York.

Herrod, R.A., 1988. AI: promises start to pay off. Manufacturing Engineering, 98–103.

Horton, E.M., Hsiao, J., Sielinski, J.E., 1988. Interactive repair assistant: a knowledge-based system for providing advice to field technicians. IEEE Communications Magazine 26, 56–57.

Jindia, A.K., 1990. Expert systems remove repetitive tedious work for customer order entry. Industrial Engineering 22 (11), 51–53.

Kaewert, D.W., Frost, J.M., 1990. Developing Expert Systems for Manufacturing: A Case Study Approach. McGraw-Hill, New York.

Leung, D., Romagnoli, J., 2000. Dynamic probabilistic model-based expert system for fault diagnosis, computers. Computers & Chemical Engineering 24, 2473–2492.

Lundvall, B.A., Johnson, B., 1994. The learning economy. Journal of Industry Studies 1 (2), 23–41.

Milne, R., Nicol, C., Travé-Massuyès, L., 2001. Tiger with model-based diagnosis: initial deployment. Knowledge-Based Systems 14, 213–222.

Pratt, C.A., 1984. An Artificially Intelligent Locomotive Mechanic, Simulation 38–39, pp. 40–41.

Putnam, H., 1987. The Many Faces of Realism I. Open Court Publishing Company, Chicago.

Rauch-Hindin, W.B., 1986. Artificial Intelligence in Business, Science and Industry — Fundamentals. Prentice-Hall, Englewood Cliffs, NJ.

Ruiz, D., Cantón, J., María Nougués, J., Espuna, A., Puigjaner, L., 2001. On-line fault diagnosis system support for reactive scheduling in multipurpose batch chemical plants, computers. Computers & Chemical Engineering 25, 829–837.

Tchounikine, P., Choquet, C., Istenes, Z., 1998. Elaborating the problem solving model of a fault diagnosis expert system by knowledge level prototyping. Expert Systems with Applications 14, 303–312.

Tzafestas, S. (Ed.), 1993. Expert Systems in Engineering Applications. Springer, Berlin.

Tzafestas, S., Adrianopoulos, A., 1993. Knowledge Acquisition for Expert System Design. In: Tzafestas, S., pp. 25–51.