Better Knowledge Management Through Knowledge Engineering: A Case Study in Drilling Optimisation

Alun Preece, Alan Flett*, Derek Sleeman
Department of Computing Science
University of Aberdeen
Aberdeen, UK

David Curry, Nigel Meany, Phil Perry
Baker Hughes OASIS
Aberdeen, UK

Abstract: In recent years the term knowledge management has been used to describe the efforts of organisations to *capture*, *store*, and *deploy* knowledge. Most current knowledge management activities rely on database and web technology; currently, few organisations have a systematic process for capturing *knowledge*, as distinct from *data*. The paper presents a case study in which knowledge engineering practices are being used to support knowledge management by a drilling optimisation group within a large service company. The three facets of the knowledge management task are illustrated: (1) Knowledge is *captured* by a knowledge acquisition process in which a conceptual model of aspects of the company's business domain is used to guide the capture of cases. (2) Knowledge is *stored* using a knowledge representation language to codify the structured knowledge in a number of knowledge bases, which together comprise a knowledge repository. (3) Knowledge is *deployed* by running the knowledge bases within a knowledge server, accessible by on the company intranet.

1 Introduction

In recent years the term knowledge management has been used to describe the efforts of organisations to capture, store, and deploy knowledge [1, 2, 3]. Organisations are interested in acquiring knowledge from valued individuals, and analysing business activities to learn the lessons from successes and failures; such captured knowledge then needs to be made available throughout the organisation in a timely fashion. Most knowledge management activities are a combination of business processes and information technology [4]. As currently practised, knowledge management" is a broad-based activity:

- ***Document management systems*** allow workers to find existing documents relevant to the task-at-hand. Essentially these are multi-source search/information retrieval systems, tied into an organisations intranet

* Author's current affiliation: Interprice Technologies, Berlin, Germany.

(possibly extending to the public internet). A number of commercial products such as those of Autonomy and Verity are available.

- ***Discussion forum systems*** promote dissemination of knowledge within communities of practice. Workers subscribe to forums relevant to their interests, exchanging questions and answers, lessons learned, announcements, and industry gossip. Such systems are easily implementable with freely-available web software, in addition to commercial products.
- ***Capability management systems*** allow an organisation to "know who knows what" [5]. Essentially, they can be thought of as databases of suitably-structured CVs, and as such are implementable with off-the-shelf database software. The goal is to put people together by matching someone's need for expertise with someone else's listed skills.
- ***Lessons-learned knowledge base systems*** are designed to allow workers to tap into past experience, by storing that experience in the form of structured *cases*. These systems support sophisticated queries, typically supporting "fuzzy" retrieval of "similar" cases. While simple systems can be built using conventional database software, special-purpose case-based reasoning or knowledge-based system software is needed for full functionality.

In terms of technology, most current knowledge management activities rely on database and internet systems: if knowledge is stored explicitly at all, it is typically stored in databases either as simple tables (for example, in relational databases) or semi-structured text (for example, in Lotus Notes). There is little use of sophisticated knowledge representation systems such as Classic, Loom, or G2. Also, few organisations have a systematic process for capturing *knowledge*, as distinct from their conventional *information*-capture procedures.

This paper argues that technology and processes from the knowledge engineering field are significantly under-utilised in current knowledge management practice, despite recent efforts to promote their use [8]. Specifically, this paper focuses on:

1. use of knowledge acquisition processes to capture structured knowledge in a systematic way;
2. use of knowledge representation technology to store the knowledge, preserving important relationships that are far richer than is possible in conventional databases.

In support of this viewpoint, the paper presents a case study in which knowledge engineering practice is being used to support knowledge management by a drilling optimisation group within a large service company in the oil and gas industry. The three facets of the knowledge management task (capture, storage, and deployment) are illustrated: (1) *Knowledge capture* is performed by a systematic knowledge acquisition process in which a conceptual model of aspects of the company's business domain is used to guide the capture of cases and rules. (2) *Knowledge storage* is performed by using a knowledge representation language to codify the structured knowledge in a number of knowledge bases, which together comprise a knowledge repository. (3) *Knowledge deployment* is performed by running the knowledge bases within a knowledge server, accessible by standard web browsers on the company intranet, and capable of answering far more complex queries than is possible using conventional database systems.

The paper is organised as follows: Section 2 examines the relevant technology and processes from knowledge engineering; Section 3 introduces the application domain of the case study; Section 4 looks at the implemented system in detail; Section 5 examines the implemented system; and Section 6 concludes.

2 Applying Knowledge Engineering to Knowledge Management

In the 1990s, knowledge engineering emerged as a mature field, distinct from but closely related to software engineering [3, 6]. Chief among the distinct aspects are:

- A range of techniques for knowledge elicitation and modelling.
- A collection of formalisms for representing knowledge.
- A toolkit of mechanisms for implementing automated reasoning.

The knowledge engineering process in outline is as follows [3, 7]:

1. ***Requirements analysis***: identify the scope of the knowledge-based system, typically in terms of the competency it will be expected to have (for example, kinds of queries it will be able to answer).
2. ***Conceptual modelling***: based on the scope defined in (1), create a glossary of terminology (concepts) for the application domain and define interrelationships between the terms, and constraints on their usage. An explicit conceptual model of this kind is commonly called an *ontology*.
3. ***Knowledge base construction***: using the conceptual model/ontology from (2) as a collection of *knowledge containers* (or schemata), populate the knowledge base with *instances* of domain knowledge (often in the form of rules, facts, cases, or constraints).
4. ***Operationalisation and validation***: operationalise the knowledge base from (3) using automated reasoning mechanisms, and validate its competence against the requirements from (1). If satisfactory, release the system; otherwise, repeat (1-4) until satisfactory.
5. ***Refinement and maintenance***: after delivery, the system will continue to evolve, as knowledge changes; this will involve repeating of (1-4) throughout the life of the system.

Any knowledge management system that involves the explicit representation of knowledge is amenable to development using at least part of the above process. In fact, it can be argued that it is *always* worth applying at least part of this process when undertaking any knowledge management activity that involves the explicit representation of knowledge. For example:

- ***Document management systems***: as a minimum, apply (1) at the outset to ensure competency criteria are defined. This will ensure at least that the right tool is selected; it may reveal a need for a more structured approach.
- ***Discussion forums***: as a minimum, apply (1-2) to ensure that the scope of the system is well-understood, and that the forums are organised so as to effectively support existing (or desired) communities of practice.
- ***Capability management systems***: as above, apply (1-2) to define the metaknowledge that will serve as knowledge containers/schemata to capture workers' capabilities; then, the "CV database" is populated in step (3).

- ***Lessons-learned knowledge base systems***: these *are* knowledge-based systems, and should follow the entire 5-stage process.

It is particularly important to employ knowledge engineering techniques when an organisation seeks to employ a range of knowledge management approaches. This is becoming common in larger organisations: such organisations are already comfortable with a multiplicity of information systems, typically tied into an intranet, and see a multifaceted knowledge management system as normal. For example, such a knowledge management system may include a capability management system, discussion forums, a document management system, and several lessons-learned knowledge bases. In such cases, the key challenge becomes that of *knowledge integration*: linking together the various sources at the level of knowledge content.

In this context, the knowledge engineering process is used to define an organisational knowledge model, or *knowledge map* [9], which becomes the set of relationships that are used to bind together the multifaceted knowledge management system at the level of knowledge content. (The actual software-level bindings can be implemented by hyperlinking, remote procedure calling, or any one of a host of distributed computing techniques.) Therefore, even when an organisation embarks on its first, single-facet knowledge management project, it is very likely to be a worthwhile investment to follow steps (1-2) of the knowledge engineering process to define an initial knowledge map.

3 Case Study: Drilling Optimisation Domain

Baker Hughes OASIS is an engineering services subsidiary company of Baker Hughes Incorporated, providing drilling process expertise to a worldwide client base in the oil and gas industry. In particular, Baker Hughes OASIS specialise in *drilling performance optimisation*, which is a knowledge-rich service involving identifying, understanding, and overcoming barriers to improved drilling performance. Drilling performance optimisation engineers need a specialised set of skills, drawn from a variety of disciplines including mechanical engineering, geology, and physics. As a relatively new service, there is a limited community of skilled optimisation engineers, and those within Baker Hughes OASIS are dispersed worldwide.

For these reasons, drilling performance optimisation represents an ideal application domain for knowledge management. Having recognised this in the early 1990s, Baker Hughes OASIS have developed a multifaceted knowledge management approach, which currently includes the following systems components:

- ***Drilling Performance Guidelines***: a semi-structured document base implemented using Lotus Notes/Domino [16].
- ***OASIS University***: on-line training system for optimisation engineers, also implemented in Lotus Notes/Domino.
- ***Drill Bit Advisor***: a rule-based expert system implemented in LISP/CLOS using a custom graphical rule representation [10].

- ***Drilling Knowledge Store***: a technical lessons-learned knowledge base, described further below.

All of these components are inter-linked. For example, a conclusion (recommendation) made by the Drill Bit Advisor is commonly linked via a URL to a Drilling Performance Guideline in the Lotus Notes/Domino system.

The Drilling Knowledge Store is one of the newest components of Baker Hughes OASIS' knowledge management strategy, and has been designed as an open repository of case-based drilling knowledge. Accessed through a Lotus Domino server, its features include:

- A structured search tool that allows users to query the knowledge store for lessons learned in environments similar to a specified environment of interest.
- Forms for entry of new knowledge to promote easy entry of new cases, which are submitted to reviewers for audit and approval before being made available to other users.
- Links to the Drilling Performance Guidelines system to avoid knowledge duplication and ease updating/maintenance.

The Drilling Knowledge Store builds on a knowledge map developed using the standard knowledge engineering process described in the previous section, and incorporates a *Drilling Knowledge Repository*, which is a case-base of documented experience of optimisation engineers. The work was carried out in collaboration with the University of Aberdeen, managed as a Teaching Company Scheme. The development stages are detailed below.

3.1 Requirements Analysis

A series of initial interviews was conducted with a number of optimisation engineers to explore the scope of the Drilling Knowledge Repository. The key finding was that the system would need to be highly open: because drilling optimisation is a relatively new specialism, knowledge in the domain is evolving, so the system would have to be designed to cope with the likely kinds of change. In particular:

- any knowledge containers/schemata would have to be highly extensible (new concepts and relationships may be discovered in the future);
- instances would frequently be added (new cases will grow in proportion to the growth in the drilling optimisation business);
- instances may be reclassified, especially as outdated knowledge is "decommissioned".

3.2 Conceptual Modelling

An initial glossary of terms was drawn-up following the first round of interviewing. The transcripts of the interviews were analysed using the PC-PACK [8] knowledge acquisition software toolkit in an attempt to derive a set of concepts. However, the tool was not found to be sufficiently flexible in dealing with concepts where the "defining" words are not adjacent in a piece of text, and are interspersed with words from other concepts. PC-PACK and similar textual mark-up systems allow the user to indicate only that single words correspond to concepts, attributes, and values. In practice, it is often the case that such entities are defined by a number of words and that these are not necessarily adjacent. For example, the text

"a bus system that links all the suburbs to the centre and to each other" contains the concept *comprehensive-city-bus-network*, but also contains parts of the concept *city* (*suburb* and *centre*).

In view of these limitations of the tool, a manual concept-mapping approach was used instead [11]. The conceptual modelling activity focussed on two areas:

- Defining the concepts associated with the drilling environment, including extensive definitions of geological concepts (leading to the creation of an ontology for representing the rock formations that constitute a drilling task) and concepts associated with the drilling activity (chiefly drill bits, fluids, and related apparatus).
- Defining the "knowledge management" concepts that would allow the capture of useful instances of optimisation engineers' experience (most obviously, the concept of a "case").

Relatively early on, it became desirable to formalise these concepts in order to manage them within a software environment. The Loom knowledge representation system [12] and its associated Ontosaurus browser/editor was chosen for the following reasons:

- Loom is one of the most flexible, least constraining knowledge representation systems currently available.
- Loom's operational mechanisms (chiefly the classification engine) allowed the knowledge engineering team to test the integrity of the conceptual model during its development.
- Ontosaurus provides a web front-end to Loom knowledge bases, allowing multiple users to inspect, query, and modify the knowledge base on a network, using a standard web browser.

3.3 Knowledge Base Construction

By the time a reasonably-complete conceptual model had been defined, a number of sample cases were already available, having been elicited from optimisation engineers as a natural part of exploring the scope of the domain. When formalising the conceptual model in Loom, the opportunity was taken to represent these cases using the knowledge containers therein. However, systematic acquisition of cases was carried out using a distinct approach.

Firstly, a small number of high-performing, expert optimisation engineers were identified. Then, one-on-one, intensive *knowledge acquisition campaigns* were conducted with these individuals. These campaigns were carefully designed to ensure that the experts would contribute actively and positively (mindful of lessons learned from negative experiences of knowledge acquisition during the hey-day of expert systems in the 1980s). The knowledge acquired from each campaign was formalised in Loom, but also written-up in a natural-language *knowledge book* electronic document [13] that could be checked for accuracy by the expert, and also disseminated on CD-ROM throughout the company as an easily-accessible, early result of the OASIS group's activity.

3.4 Operationalisation and Validation

The knowledge base was operationalised naturally by the choice of Loom as representation language. Validation was performed at two levels:

- Indirect validation of the represented knowledge using the knowledge books.
- Direct validation of the represented knowledge using Loom's inference mechanisms.

Further indirect validation came through the development of drill bit selection rules using some of the case-based knowledge acquired in these campaigns. These rules were then validated using existing software in the Drill Bit Advisor expert system.

The next section examines the structure of the Loom Drilling Knowledge Repository in more detail.

4 The Drilling Knowledge Repository in Loom

Philosophically, the system is considered to be a "knowledge storage and retrieval system" rather than a "knowledge-based system". The reason for this is that knowledge-based systems are strongly associated with tasks, whether those tasks are decision support, automated decision-making, training, or whatever. It is well-known that the task for which knowledge is to be used places a strong bias on the form of the knowledge [14]. In this case, it was important that the implemented system be considered task-neutral: it should serve purely as a repository for captured knowledge, without any risk of biasing the form and content of the knowledge toward a particular future usage.

Nevertheless, the system does have at its core a set of automatic deductive facilities (provided by Loom), which operate on the definitions given to the system by the knowledge modeller. The important point, however, is that these inferences operate at the conceptual model level, and not at any action level. That is, actions are left to humans, and the system does not *per se* advise on any action the user should take. So, for example, the system will recognise instances and classify them appropriately with reference to the conceptual model, but this is purely for the purpose of retrieving those instances and bringing them to the user's attention.

There are two main parts to the Loom knowledge store; a *conceptual model* part, and a *database* part. (This is analogous to a database with its schema and data parts.) The conceptual part of the knowledge base is defined using concepts. There are binary concepts (otherwise known as *roles*) and unary concepts (known simply as *concepts*). The database part is populated with *instances* of the concepts in the conceptual part.

This section gives examples of the Loom constructs to illustrate the approach taken in concrete terms. The intention here is to motivate the constructs so that a full understanding of the representation language is not necessary (if the reader is unfamiliar with Loom or similar languages, good introductions are given in [12, 15]).

4.1 Modelling Constructs for Drilling Engineers' Experience

As the knowledge store is chiefly intended to capture experiential cases from drilling engineers, the most important concept is the *case*.

```
(defconcept CASE :is-primitive
 (:and (:exactly 1 formation-sequence)
    (:all decision DECISION)
    (:all observation OBSERVATION)))
```

A case usually describes a *drill bit run* - a continuous period of drilling with a single drill bit. So, if an optimisation engineer experiences some bit run worthy of being recorded in the knowledge store, then the rock formation sequence drilled should be represented, as well as the decisions taken on how to drill that formation sequence, and any associated observations. A decision can refer to a choice of drill bit, mud (drilling fluid), flow rate, and so on. Alternatively, the case need not refer to an actual drill bit run if the person entering it simply has an experience they wish to share.

A *decision* is presupposed to have several different dimensions. These include: *issues*, *actions*, *goals*, an author, a *spin*, and *reasoning*. These dimensions are intended to provide a balance between structured knowledge and free text. The structured knowledge is to enable the formal representation and therefore support powerful searches, while the free text supports semi-structured knowledge.

```
(defconcept DECISION :is-primitive
 (:and (:exactly 1 action)
    (:at-most 10 issue)
    (:at-most 10 goal)
    (:at-most 1 authors-reasoning)
    (:at-most 1 companys-reasoning)
    (:at-most 1 author)
    (:at-most 1 spin)))
```

An *issue* is some informational context that the engineer was aware of when making the decision. The issues in the current KB reflect quite strongly the Best Practice Drilling database (held in Lotus Notes), and the link roles shown below are intended to reflect this. These can be filled with links to other media, including the Notes database itself, using URLs.

```
(defconcept ISSUE :is-primitive
 (:and KNOWLEDGE_MANAGEMENT_CONCEPT
    (:at-most 1 symptoms-and-diagnosis-link)
    (:at-most 1 description-link)
    (:at-most 1 parameters-link)
    (:at-most 1 diagnostic-information-link)
    (:at-most 1 planning-actions-link)
    (:at-most 1 operating-practices-link)
    (:at-most 1 examples-link)))
```

An *action* is the real-world consequent the engineer performed as part of the decision, and includes both structured (`categorical-outcome`) and free text (`textual-outcome`) outcomes.

```
(defconcept ACTION :is-primitive
 (:and KNOWLEDGE_MANAGEMENT_CONCEPT
    (:at-most 1 categorical-outcome)
    (:at-most 1 textual-outcome)))
```

Two kinds of *reasoning* are captured for a decision: *Author's reasoning* is a field of free-text for explanations for example of why a certain drill bit was chosen. This is to allow the possibility of incomplete, inaccurate, and even incoherent explanations for actions being stored; after all, the main reasoning or determinism for the action is the other structured information describing the circumstances in which the action was taken, such as the formation sequence. *Company's reasoning* is a field which expresses the company's commonly agreed upon beliefs for the decision in question.

4.2 Modelling Constructs for the Drilling Environment

The drilling environment is described chiefly in terms of conceptual rock sequences. Representing these was achieved by defining an ontology of geological concepts, including constraints. For instance, if the user wishes to specify the depth and/or length of a particular section of *lithology* (a basic rock type, for example, sand, shale, etc) then that section has to be represented as a *formation*. The super-structure larger than that is the *formation sequence*, which can have one or more *formations*. Each formation can have one or more *lithologies*. A formation is the conceptual modelling granularity at which the users should be representing any part of the wells they feel should have represented interval lengths and depths.

```
(defconcept FORMATION_SEQUENCE :is-primitive
 (:and ROCK_CONCEPT
    (:at-least 1 formation)))

(defconcept FORMATION :is-primitive
 (:and ROCK_CONCEPT
    (:at-least 1 lithology)))
```

To allow users to represent and query formation sequences in a flexible way, a number of relations are defined in the ontology. For example, the relation `comes-in-somewhere-after` relates two formations, the first of which comes in somewhere after the other.

```
(defrelation comes-in-somewhere-after
 :domain FORMATION_SEQUENCE
 :range FORMATION
 :characteristics (:multiple-valued :closed-world)
 :is (:satisfies (?formation-x ?formation-y)
          (:and
           (FORMATION ?formation-x)
           (FORMATION ?formation-y)
           (:or (comes-in-immediately-after
                 ?formation-x ?formation-y)
             (:exists (?formation-z)
                   (:and
                    (FORMATION ?formation-z)
                    (comes-in-somewhere-after
                       ?formation-x ?formation-z)
                    (comes-in-somewhere-after ?formation-z
                           ?formation-y)))))))
```

One important feature of *lithologies* in their *hardness*. While a lithology has, by definition, one rock type (for example, shale), it can have more than one hardness (for example, that shale may have 100m of *very soft rock* with 300m *soft rock*).

```
(defrelation hardness
 :domain LITHOLOGY
 :range HARDNESS
 :characteristics (:closed-world :multiple-valued))
```

The ontology has a collection of functions that relate formation sequences, constituent lithologies, and accumulated hardness, to support the modelling of drill bit runs.

In addition to the generic geological concepts, the knowledge store has representations of the concepts involved in *drilling*. For example, *drill bit*.

```
(defconcept DRILL_BIT :is-primitive
 (:and DOWN-HOLE_EQUIPMENT_CONCEPT
    (:exactly 1 bit-gauge)))
```

4.3 Querying the Knowledge Store

The function `retrieve` provides an interface to Loom's deductive query facility, used for retrieving instances from the knowledge base. *Formation sequence queries* are among the most sophisticated forms of query that can be issued to the knowledge store. The two concepts likely to be of interest are individual formations, and formation sequence. Two common forms of query here are:

- Queries on an overall cumulative amount of a certain hardness of a certain lithology over a formation sequence.
- Queries for formations that have amounts of certain lithologies of certain hardness.

The example query below looks for cases that have a formation sequence which has as its constituents of its formation(s) greater than or equal to 1900 feet of very soft to soft (including all subtypes of soft and very soft) shale (including all sub-types of shale).

```
(retrieve ?case
   (:and
    (CASE ?case)
    (>= (sum (:collect ?lithology-amount-ft
                (:and
                 (:exists (?formation-sequence ?formation
                        ?lithology ?hardness)
                    (:and
                     (formation-sequence ?case ?formation-sequence)
                     (formation ?formation-sequence ?formation)
                     (lithology ?formation ?lithology)
                     (lithology-hardness-amount-ft ?lithology
                        ?hardness ?lithology-amount-ft)
                     (:or
                      (VERY_SOFT ?hardness)
                      (SOFT ?hardness))
                     (SHALE ?lithology)
                     ))))) 1900)))
```

Users are also likely to want to look for cases where specific goals (outcomes) were achieved. The following example query retrieves cases that have a drill bit decision in which one of its goals was *good ROP* (rate of penetration) with *good bit cleaning*.

```
(retrieve ?case
      (:and
       (CASE ?case)
       (:exists (?decision)
             (decision ?case ?decision)
             (DRILL_BIT_PLANNING_DECISION ?decision)
             (goal ?decision GOOD_ROP_WITH_GOOD_BIT_CLEANING))))
```

4.4 Adding to the Knowledge Store

Recall that the knowledge store is comprised of a conceptual part and and a database part. It is expected that the conceptual part is now stable, and it is rare that knowledge will need to be added or modified. However, it is expected that additions to the database part will be regular. The Loom operations used to update the database part of the knowledge base are `tell` and `about`: `tell` is used to assert propositions and facts about the world or domain; `about` references the instance to which those propositions refer. The following example shows how a case instance may be entered. This example case has one formation sequence name and zero or more decisions/observations.

```
(tell (:about Case-Name
        CASE
        (formation-sequence Formation-Sequence-Name)
        (decision Decision-Name)
        (observation Observation-Name)))
```

5 Current Status and Future Plans

The Loom Drilling Knowledge Repository currently contains 1200 concepts and 240 relations, with further expansion planned. The knowledge store is accessible on the company's intranet using a standard web browser via the Ontosaurus system - see Figure 1. While it is relatively straightforward to browse the case base and ontology using Ontosaurus, there are a number of significant problems with the current system:

- it is hard to run complex queries on the Loom knowledge store because Ontosaurus supports only simple queries (retrieve cases with matching simple role values);
- it is hard to add new cases because these require the user to have good knowledge of Loom syntax, which is an unrealistic expectation for optimisation engineers;
- limited capability for multi-user access (crude database locking and limited concurrent access).

In addition to the above issues, it was desirable that the Drilling Knowledge Repository have a familiar interface, preferably that of the existing systems implemented using Lotus Notes/Domino. At the same time it was thought valuable to link the knowledge represented in the Loom repository with information and knowledge relating to the performance optimisation projects during the course of which the stored knowledge was created. For these reasons, it was decided as an interim solution to incorporate (partially) the Loom knowledge map and cases into a Lotus Notes/Domino database of project-related knowledge, to provide structure for technical lessons learnt on each project. A screenshot of the ported system appears in Figure 2. The immediate benefits of this were:

- easy access to the most valuable knowledge in the knowledge store to all optimisation engineers;
- linkage between optimisation-related knowledge and other knowledge from the projects in which it was created;

- familiar interface, seamlessly integrated with other Notes/Domino knowledge sources;
- Domino's scalable architecture, with built-in support for concurrent access.

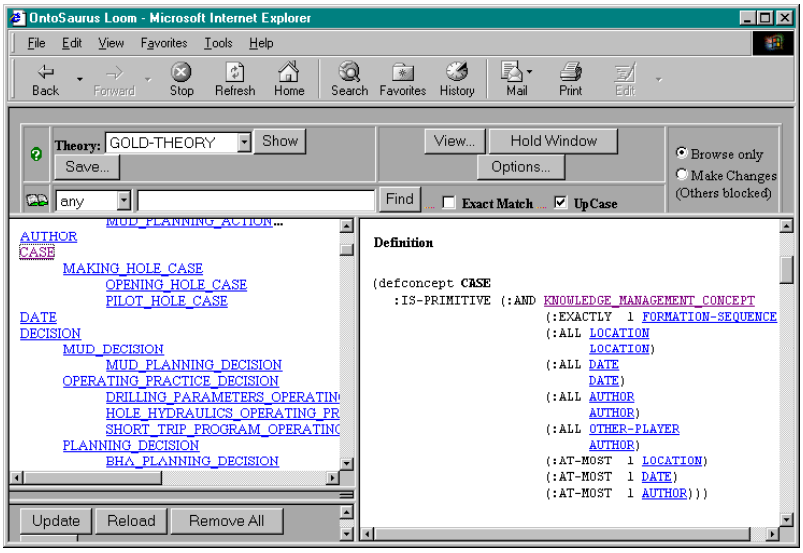

Figure 1: Loom Drilling Knowledge Repository Screenshot

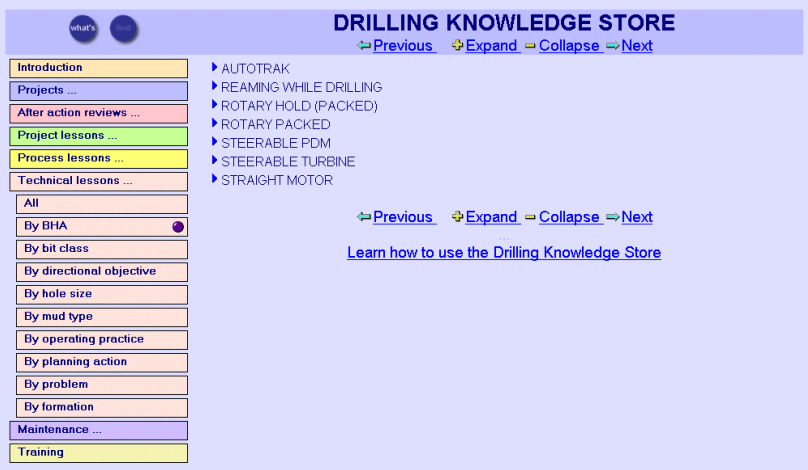

Figure 2: Lotus Notes/Domino Drilling Knowledge Store Screenshot

The port of the Loom knowledge map and cases to Lotus Notes/Domino was carried out manually. Moreover, the Notes knowledge schemata are nowhere near as rich as those in the Loom system. As this is obviously not ideal, future work will focus on providing an automatic conduit for knowledge exchange between the Notes/Domino and Loom systems, allowing knowledge management engineers to

maintain the knowledge map through the Loom system, and optimisation engineers to retrieve and enter cases via the Notes/Domino system.

6 Conclusions

This paper has argued that knowledge engineering techniques bring significant benefits to knowledge management projects. The case study in drilling optimisation reveals three major areas of specific benefit to the knowledge management group at Baker Hughes OASIS:

- The *use of a principled methodology for knowledge acquisition* lead to effective capture of valuable knowledge, and instilled in all staff concerned a good degree of confidence in the process.
- The *development of a common ontology* as a result of the knowledge modelling process had benefit not only for the specific task of acquiring experiential cases from company experts, but also provided a knowledge map with application to multiple company knowledge sources.[1]
- While semi-structured knowledge books were used to provide feedback to participating personnel, the *formal representation of acquired knowledge in an operationalisable form* was very valuable for knowledge verification and integrity checking.

In doing this work, we have detected two weaknesses in current knowledge engineering techniques and technology. Firstly, as noted in Section 3.2, PC-PACK and other textual mark-up systems do not cope adequately with concepts that are defined by a number of non-adjacent words. Thus, we have identified the need for a more flexible tool.

Secondly, it is very difficult to integrate expressive reasoning tools such as Loom with intranet knowledge management environments like Lotus Notes/Domino. It seems reasonable to conclude, therefore, that while knowledge engineering processes are ready to bring significant benefits to knowledge management projects, the knowledge engineering toolbox needs some improvement.

Acknowledgements

The case study described in this paper was supported by Teaching Company Scheme funding from the DTI and EPSRC. The Lotus Notes version of the Drilling Knowledge Store was constructed largely by John Lawton and David Bowden of Transition Associates, who also proposed and developed the OASIS University. The authors would like to thank Tom Russ of ISI/University Southern California, for advice and technical assistance regarding Loom and Ontosaurus.

This paper is published with the kind consent of the Hughes Christensen Company.

[1] We have subsequently developed a knowledge acquisition tool that is driven by the nature of the task and the ontology. We believe that this tool, COCKATOO [17] would be effective in acquiring cases directly from domain experts.

References

1. *Harvard Business Review on Knowledge Management*, Harvard Business School Press, 1998.
2. Liebowitz, J. and Wilcox, L. (1997) *Knowledge Management and Its Integrative Elements*, CRC Press.
3. Schreiber, G., de Hoog, R., Akkermans, H., Anjewierden, A., Shadbolt, N. and van de Velde, W. (2000) *Knowledge Engineering and Management*, MIT Press.
4. Bukowitz, W. and Williams, R. (1999) *Knowledge Management Fieldbook*, Prentice-Hall.
5. Stader, J. and Macintosh, A. (1999) Capability modelling and knowledge management. In *Applications and Innovations in Intelligent Systems VII*, Springer-Verlag, pp 33-50.
6. Motta, E. (1999) *Reusable Components for Knowledge Modelling*, IOS Press.
7. Russell, S. and Norvig, P. (1995) *Artificial Intelligence*, Prentice-Hall.
8. Milton, N., Shadbolt, N., Cottam, H. and Hammersley, M. (1999) Towards a knowledge technology for knowledge management. *International Journal of Human-Computer Studies*, 51:3 615-641.
9. Domingue, J. and Motta, E. (2000) Planet-Onto: from news publishing to integrated knowledge management support. *IEEE Intelligent Systems*, May/June, pp. 26-32.
10. Evans, J. M., Fear, M. J., and Meany, N. C., (1995) A new graphical representation for rule definition and explanation in an expert system. In *Applications and Innovations in Intelligent Systems III*, Springer-Verlag.
11. Kremer, R. (1997) Concept mapping tool to handle multiple formalisms, In *AAAI Spring Symposium on Artificial Intelligence in Knowledge Management*, AAAI Press.
12. MacGregor, R. (1991) The evolving technology of classification-based knowledge representation systems. In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann, pp 385-400.
13. Ermine, J-L. (1998) Knowledge management in the Commissariat a l'Energie Atomique. In *PAKeM98 - Practical Application of Knowledge Management*, London.
14. Clancey, W. J. (1992) Model construction operators. *Artificial Intelligence*, 53:1-115.
15. Brachman, R. J., McGuinness, D. L. Patel-Schneider, P. F., Resnick, L. and Borgida, A. (1991) Living with Classic: when and how to use a KL-ONE-like language. In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann, pp 401-456.
16. Curry, D. A., Singelstad, A. V., and Bowden, D. (1999) Drilling Performance Guidelines - A tool for sharing drilling-related knowledge and experience", SPE/IADC paper no 52804, *SPE/IADC Drilling Conference*, Amsterdam.
17. S White (2000) *Enhancing Knowledge Acquisition with Constraint Technology*, PhD Thesis, The Department of Computing Science, University of Aberdeen.