USING ONTOLOGIES TO SUPPORT AND CRITIQUE DECISIONS

Yannis Kalfoglou
Advanced Knowledge Techologies (AKT)
School of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
y.kalfoglou@ecs.soton.ac.uk

***Abstract*—Supporting decision making in the working environment has long being pursued by practitioners across a variety of fields, ranging from sociology and operational research to cognitive and computer scientists. A number of computer-supported systems and various technologies have been used over the years, but as we move into more global and flexible organisational structures, new technologies and challenges arise. In this paper, I argue for an ontology-based solution and present some of the early prototypes we have been developing, assess their impact on the decision making process and elaborate on the costs involved.**

A. Keywords

design paradigms, decision support, decision critiuing, organisational memories, cost-benefits analysis

I. INTRODUCTION

Since the early days of decision support research, the notion of group-targeted technologies has been at the centre of attention, based on the assumption that groups are the core functional teams within an organisation. At the same point in time, AI technology was becoming more applicable and appealing to wider audiences and industry, which led to investigations of potential synergy between AI products, like *Expert Systems*, and *Group Decision Support Systems* (GDSSs) [4]. As we move on to the 21st century though, and the Internet technology is becoming indispensable asset for the survival of organisations, the notion of groups has changed significantly. Other less-formal, but more focused and motivated structures are emerging, like *Communities of Practice* [26].

This shift has an effect on the research agenda for decision support systems as Shim and his colleagues point out in [31]. They predict that technologies like *data warehousing*, *online analytical processing- OLAP*, *data mining* and *web-based DSS* will be the focus for the next decade of research on decision support systems. Alongside with these technologies the focus of engineers is to build and support the necessary infrastructure upon which decision support can be made available to human users. For example, in the AI realm, knowledge engineers are working for over a decade now on the provision of reusable, computational forms of domain knowledge, ontologies, and knowledge management systems, like organisational or corporate memories. Although, their aim and target is more generic than providing support for decision making, they are well suited technologies for any form of knowledge management support.

I emphasize this argument in this paper by presenting some example uses of ontologies which could support decision making, especially in the early stages of design. By design here, I mean mostly software systems design, and the decision making affects software designers as opposed to other social groups, like government policy decision makers. I present an overview of design paradigms in section II before moving on to illustrate an example use of ontologies in decision support via an organisational memory in section III. Decision critiquing is a relatively new but fruitful area of decision support which could benefit systems design if certain assumptions hold as I discuss in section IV. The benefits of using an ontology-based solution need to be carefully assessed against the costs involved as I discuss by concluding the paper in section V.

II. DESIGN PARADIGMS

In ([24]-page 3) the authors argue that four broad descriptions of design paradigms exist in the literature. We elaborate on the impact of these paradigms in systems' design in [21] but I briefly recapitulate them here along with the types of knowledge engineering technology that can be classified under each of these. It is intended to show the involvement of knowledge engineering in design and decision support rather than acting as a directive.

Design as decomposition and synthesis: this paradigm dates back at least to 1974 with Alexander's work on architecture [6]. Design is taken to be the reshuffling of components developed previously, then abstracted into reusable components. Modern expressions of this approach include object-oriented design patterns, ontologies, and Problem Solving Methods (PSMs). These are the dominant paradigms in the contemporary knowledge and software engineering.

Design as search: early work in this paradigm dates back to 1969 with Simon's work on AI [32]. Design is taken to be the traversal of a space of possibilities, looking for pathways to goals. Modern expressions of this approach include most of the AI search literature, *knowledge-level* search, the SOAR papers [29], the SVF project [19], certain KA approaches [23], etc. In a *knowledge-level* search, intelligence is modelled as a search for appropriate operators that convert some current state to a goal state. Domain-specific knowledge is used to select the operators according to *the principle of rationality*: an intelligent agent will select an operator which its knowledge tells it will lead the achievement of some of its goals. In this paradigm we see preliminary work on modelling the domain-specific knowledge in computational forms, like domain-specific ontologies.

Design as negotiation and deliberation: this paradigm dates back to at least 1973 with Ritell's work on *wicked problems* [28]. Wicked problems have many features, the most important being that no objective measure of success exists. Designing solutions for wicked problems cannot aim to produce some perfectly correct answer since no such definition of *correct* exists. Hence, this approach to design tries to support effective debates by a community over a range of possible answers. Modern expressions of this approach include the requirements engineering community. Requirements engineering is usually complicated by the incompleteness of the specification being developed: while aspecification should be consistent, requirements are often inconsistent. Requirements engineering researchers such as Easterbrook [16], and Finkelstein *et.al.* [17], argue that we should routinely expect specifications to reflect different and inconsistent viewpoints. Traditionally, this paradigm has relied much on manual methods. In recent years more automatic techniques have been used. For example, the work of [25] on conceptual modelling and in particular the use of declarative meta-models in requirements engineering and the work on negotiation in multiagent systems [14].

Situated design: Schon's work on the *reflective practitioner* [30] is among the first in this paradigm. In that approach, design mostly happens when some concrete artefact *talks back* to the designer - typically by failing in some important situation. That is, reflective design is less concerned with the creation of some initial artefact than the on-going re-interpretation and adjustment of that artefact. Modern expressions of this approach include the situated cognition community [12], certain approaches to design rationale [10], and knowledge engineering techniques that focus on maintenance rather than initial design [23].

The design paradigms surveyed by Moran and Carroll do not explicitly mention possible combinations. For example, when designing new components we infer that their design is a combination of search, negotiation and deliberation, and situated design.

III. DECISION SUPPORT

We wintess a shift in the decision support literature from data-oriented processing systems to more integrated with the human intellect and organisational processes systems [9]. These systems have been studied in the knowledge management literature and among the most visible ones, are organisational memories (OMs). They have been studied as means for providing easy access and retrieval of relevant information to users. There are several technologies which support the implementation and deployment of OMs (some of them identified in [1]). Having the ideal OM in place could assist in decision making as, crudely speaking, any information regarding the organisation could be made easily accessible.

However, there is relatively little support for the initial set-up of an OM. When implementing and deploying an OM, it is difficult to identify the right information to include. This task is, normally, a knowledge engineer's job, to identify relevant information and populate the OM accordingly. This process though is time-consuming, manual and error-prone, given the diversity and quantity of resources to be analyzed for relevance. Semi-automatic methods and techniques exist, but these are bound to individual technologies. On the other hand, it is always the user who has to initiate search in the OM. This however, requires the user to formulate a query, sometimes with the help of semi-automatic support, and then the OM system has to parse the query successfully, retrieve information deemed to be relevant according to some pre-defined notion of relevance, and present it to the user.

This problem has been identified in field surveys [15] as well as in implemented systems [2]. It is a multi-faceted problem because it is not only concerned with the elicitation of resources that will be presented to the user or used for retrieving relevant information, but also with the nature of these resources which could be: (a) used by other systems within the organisation, which incidentally also serve users in their quest for valuable information, (b) 'unspecified', in that they are vaguely expressed, need to be composed by a number of related resources or are external to the organisation, (c) and once these resources are identified and put into use, they act as a qualitative measure for the OM. That is, if an OM's users are not satisfied with the quality of information presented to them, it is unlikely that they will return, especially when there

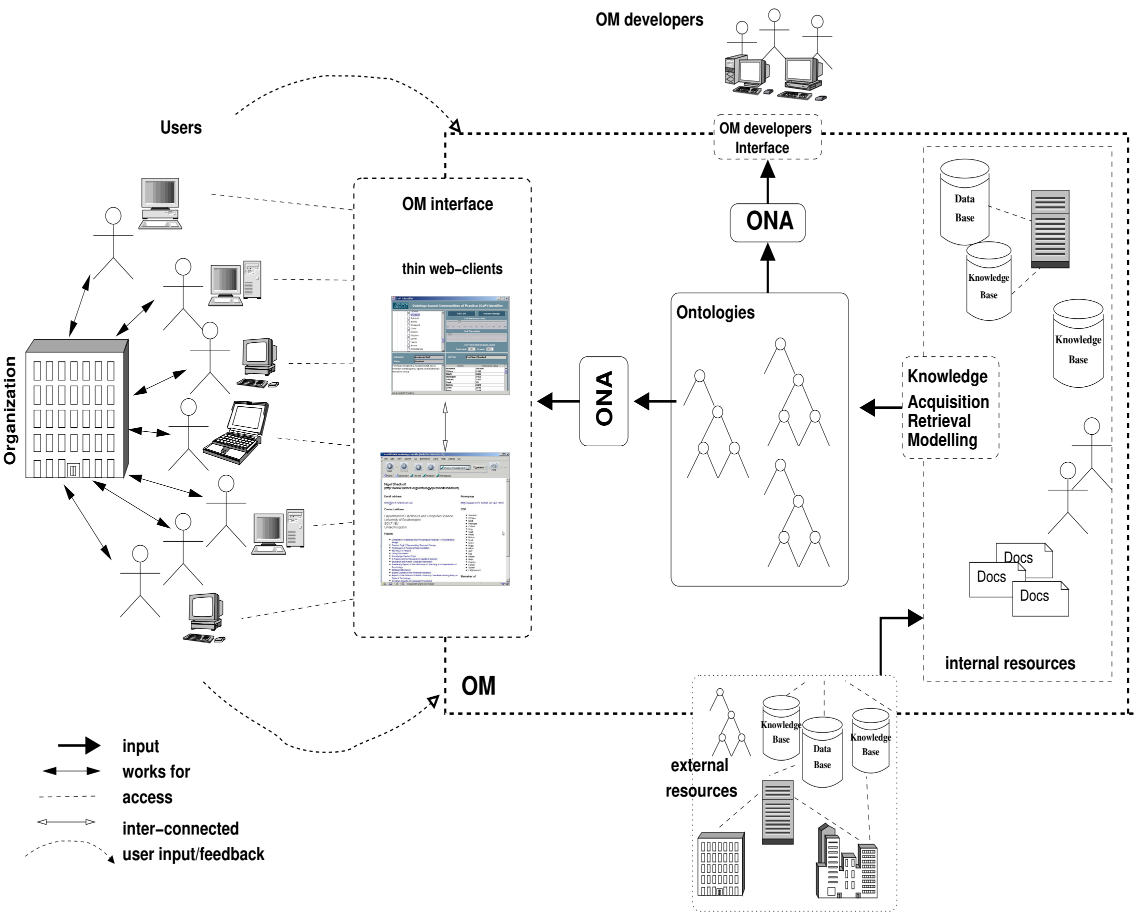

Fig. 1: Applying ONA at different phases of OMs: to push knowledge to users as well as help developers tune their OMs.

are other conventional information-seeking systems in the organisation that users used to use before confronted with an OM.

A way of tackling this resource-selection problem is by identifying the purpose of the OM: what are the users' needs and what will the OM be used for. This has been reported as one of the first phases in building an OM [15]. The techniques and methods for achieving this rather ambitious goal are mostly taken from requirements analysis and elicitation research. They stem from Computer Supported Collaborative Work (CSCW) research, from systems design research, and from the cognitive science literature.

However, we should be cautious when we are calling upon requirements engineering to elicit the needs when building an OM. As Zave and Jackson report in their survey [34], vague and imprecise requirements are always difficult to formalize and subsequently convert to specifications, in the early phases of software development[1].

[1]In our case, the early phase of developing an OM.

This refinement is necessary, the authors continue, "to bridge the gap between requirements and specifications", thus emerging with a specification that could satisfy users' needs and meet the requirements.

The vagueness and incompleteness of requirements from prospective OM users led some designers to decide to build their OM around an existing workflow process engine, as for example in the *KnowMore* OM [2]. We discuss the adaptability of this approach and its advantages of achieving a 'near perfect' integration with existing IT organisational infrastructure and satisfying users' (pre-defined) needs in [5], but here I would like to focus on the importance of having a comprehensive OM from its initial set-up. By comprehensive I mean an OM that includes a lot of resources that have been automatically extracted rather than waiting the user to initiate the extraction process. The side-effect of having this sort of OM in place is that we can tackle the 'cold start' syndrome identified in [18] in which the authors reported that they had relatively few knowledge assets in their OM during the first operational month which led to low access rates from its users as they couldn't see the value-added of the OM. The problem was eventually solved, but at a cost: more systems and methods had to be used to chase users for contributions in order to enrich the content of the OM, thus leading to an increase in the OM's knowledge assets and consequently in increased access figures.

Our approach to this problem is to use ontologies. The hypothesis is that since we already use ontologies in OMs for the purposes of semantic interoperability and reuse, we could also use them in other ways. We could analyse their structure by taking into account relationships between their constructs, based on a tunable spread activation algorithm, yielding the nodes that are most "popular". These are assumed, in the absence of contradicting evidence, to be the most important ones. The spreading activation algorithm also identifies nodes similar to a specific node. This is the premise underlying our hypothesis.

It could be argued that our analysis is not a qualitative one, but merely a quantitative one. However, as Cooper argues in [13], quality can be measured in two ways, in terms of popularity or importance. Our analysis yields concepts that are the most popular in the network, and since the network is about an ontology which by default represents important concepts, then these concepts are also important.

To operationalize our hypothesis, we assume that (a) ontologies will be available in the organisation in which we want to deploy an OM, and (b) these will be populated. It is clear that these assumptions are strong and indeed are ongoing research issues in the knowledge engineering community, especially the latter. However, we should accept and anticipate that ontologies are popular in organisational settings nowadays, in the form of database systems, other knowledge sharing formalisms more common to the AI research community (KIF) or indeed in emerging semantic web standard formats (RDF).

Using ontologies as the foundation for an OM is not a unique idea, but the use of ONA to provide initial information for populating the OM is novel. We should also mention that using an ontology at the start of an OM's lifecycle allows us to provide support to users in formulating their queries from an early stage. Normally, users have to formulate initial queries unaided since there is no prior information available, as no retrievals have been made yet. In applying ONA, we support users in formulating queries by providing them with ontological information regarding the starting node for initiating an ONA-based search. This information is readily available in existing slots in the underlying ontology (such as the documentation slot).

In figure 1 I depict a high-level diagram of an OM. This is not meant to be a reference architecture for OMs. This figure emphasizes the dual role of ONA and the supportive role ontologies play in our scenario. On the left-hand side of the figure we have users of an organisation performing their regular tasks. In the centre we have an OM which is composed, at this abstract level, by two interfaces to users and OM developers, a port to external resources, and internal resources existing in the organisation's repositories. The latter could have several forms, ranging from tacit knowledge possessed by experts to explicit knowledge expressed formally in KBs or digital discussion spaces. In the centre of our abstract OM, lie the ontologies which underpin the entire OM. These are either existing resources or are constructed (semi-) automatically with the aid of knowledge acquisition, retrieval and modelling techniques. I do not refer to these in this paper as the focus is on the use of ONA: the two rectangular boxes denoting "ONA" are placed between the ontologies and OM interfaces to users and developers. The genericity of ONA makes it possible to use it for pushing knowledge to users but also as an aid for the OM's developers. They could apply ONA to the organisation's ontologies in order to identify which concepts should be presented to certain types of users. For instance, assuming that there is a workflow engine in the organisation, and developers are looking for ways of linking the OM to it, they could either engage in modelling techniques such as those used in linking the *KnowMore* OM with workflow processes [2], or they could use ONA to help them identify concepts from the underlying ontologies that could be used to map onto the ones of the workflow's processes. This activity requires inspection and familiarization with only one end of the prospective link: that of the workflow processes. The developer then, uses the concepts found in the workflow processes as a starting node for his/her ONA. This could reveal whether further linking is feasible (or otherwise), thus saving development time and allowing developers to deal with ontologies that they are not familiar with. The approach taken by the *KnowMore* OM, requires a careful analysis and possibly, modelling of both workflow processes and ontologies before a link between them could be implemented. ONA can ease the analysis on the ontology end of this prospective link.

I also include two curly dotted arcs in figure 1 linking users with the OM. These denote users' feedback and input. This is an important, probably the most important, element of any OM architecture as an OM can be improved over time by user feedback and input. In our abstract architecture, we implemented light-weight feedback mechanisms, like thin Web-clients, accessible through Web browsers, as a means for eliciting feedback on an OM's resources. Finally, the OM interface to its users is light-weight and accessible from distributed clients on the Web. We have developed two kinds of interfaces for accessing our dedicated tools in the AKT project: a dedicated OM interface, where the user can state preferences in selecting the appropriate node to search for related information, or there could be a customized rendering of information into a user's Web browser. The latter is extracted automatically after applying ONA to the underlying ontology, whereas the former requires user input to tune the search criteria.

The method described above is not infallible nor adaptable to any existing OM setting. We identified potential caveats on using ONA to bootstrap OMs and categorize them in three broadly defined areas: (a) *Information overload:* a progressive and query-based interaction with the OM from initial set-up acts as a safeguard against unwanted information overload. However, progressive interaction means that the initial set-up suffers from cold-start syndrome – not enough information will be available; query-based interaction requires expertise and domain familiarization from the users to get the most out of an OM; (b) *Context-awareness:* this has been recognized as the Achilles' heel for OMs. One proposed remedy, advocated by proponents of marrying workflow processes and OMs [3], seems to work well only in settings where workflow processes are either existing, or are relatively easy to identify and model; (c) *Domain-independence:* this is a desired feature for OMs. But, the proposed ONA approach is not specific to any kind of ontology, or indeed to any ontology at all! This makes it possible to apply ONA to more than one ontology as are likely to exist in large organisations.

IV. DECISION CRITIQUING

When decisions are made, designers need to verify, validate and eventually trace them in case a review is needed. Among a variety of tools that have been available – some of which are outlined in the report of [31] – we found it practical to connect the underlying techology used for supporting reasoning, ontologies, with *Experience Factories* (EFs). These stem from the software engineering community and were first operationalised in early nineties [8] as a means to support exchange of all kinds of experiences in the life-cycle of a software project. The main focus of an EF is to support 'learning from experience' on a technology-independent organisational level. An EF stores the collected experiences in an *Experience Base* (EB). In [7], it was argued that Case-Based Reasoning (CBR) plays an important role in the EFs paradigm. As CBR provides both the technology and a methodology for 'learning from experience', it was natural to use it for implementing continuous learning in an EF style. The value of a repository of cases is also acknowledged in the knowledge management literature as O'Leary writes with respect to the conversion of individual to group available knowledge: "Although individuals might have generic knowledge to contribute, case histories are particularly robust" [27].

We applied EFs in an ontology development and deployment life-cycle as a means to manage experiences collected from various agents participating in that effort [20]. To accomplish this task, we built a generic architecture which I depict diagrammatically in figure 2.

The left-hand side of figure 2, depicts the task of verification. In particular, we are interested in verification of ontologies at the application level [22]. That is, we verify that ontological constructs are not misused by applications that adhere to an ontology. After applying our verification mechanism we accumulate, temporarily, the results in an EB. These are code-testing results and we regard them as experiences. The EB is then imported by an experiences editing tool which allows for further additions and modification of the description of existing experiences. It allows us to customise the experiences to the particular project as it provides a way of expressing information usually not obtainable through code-testing. These are decisions about the artefacts at question and our aim is to provide support for entering as much information as possible about the decision making process.

For example, in the context of decisions about the ontology usage, we used a proposed typology for accumulating ontology projects information [33] that the designer need to fill-in and can later use to facilitate future tracing and reuse: *purpose* of the ontology, *representation languages and paradigms* used, *meaning and formality* of the ontological constructs, what is the *subject matter*, information regarding the *scale* and *development status* of the ontology, what is the *conceptual architecture*, *mechanisms and techniques* used, and the *implementation platform.*

We then select the experiences we want to validate and send them to a designated tool for verifying their correctness with respect to test results. This tool embodies the verification mechanism we deploy in the first step but here we apply it to verify the correctness of the results themselves. After the selected experiences have been validated we store them in the final EB to be part of the EF.

This cycle can be repeated as many times as we wish in the same or other ontologies to collect and manage the knowledge accumulated during verification and testing. Ultimately, this will result in an EF of ontology verification and testing that can be deployed in similar projects in order to facilitate ontology use and decision making for the designers. It is important to mention that the similarity of projects to benefit from this sort of EF dictates the ability to transfer experiences and decisions across projects.

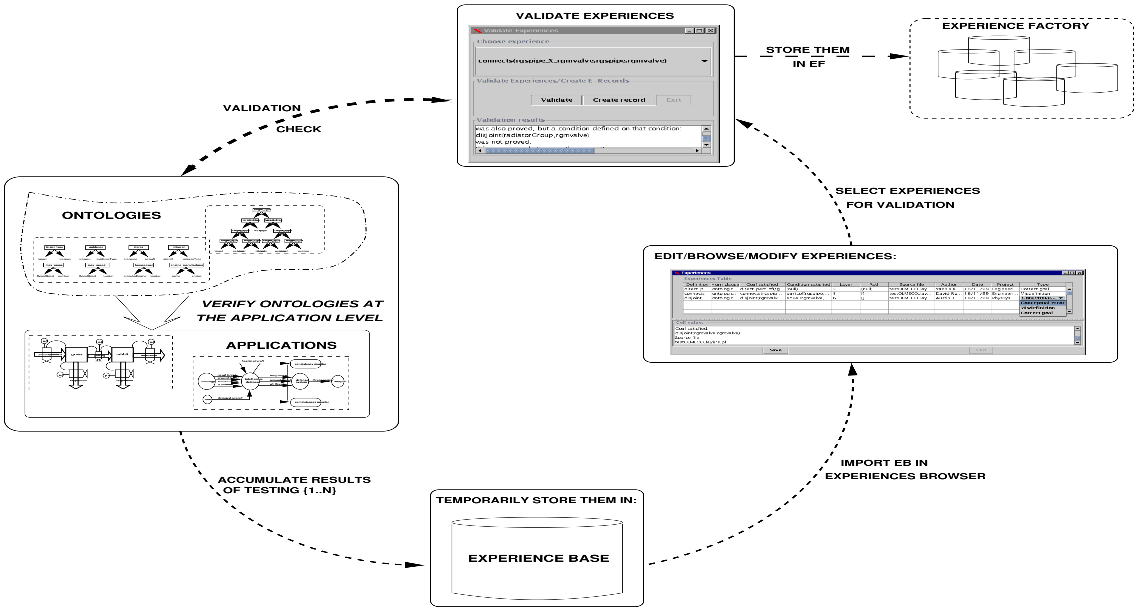

Fig. 2: Experience-based architecture to support ontology verification.

V. COST-BENEFIT ANALYSIS

There are benefits from applying ontology technology to support and critique decision making for designers, but there are also costs involved with using this technocology in the first place. In a field survey we conducted few years ago [21], we identified the following, broadly speaking, categories:

Construction cost: an ontology aims to represent knowledge, and knowledge comes at a cost. Whether we build the ontology from from scratch or adopt it from others there is a 'cost of construction' following that investment. In the former case, that cost refers to pure construction issues such as, for example, choice and adoption or creation of a design methodology. In the latter case, there is an adoption of a pre-existent ontology (for example, residing in a public library), and the cost is related to familiarisation and installation issues. Empirical evidence from the knowledge engineering field reveals that this cost is often a drawback and those who had to afford it were skeptical for the effectiveness of the approach. This is apparent in generic ontologies. For example, the CYC project where the developers had to devote more than 10 years of effort to build the CYC ontology. However, when we move to domain or task specific ontologies the situation changes as these are easier to build and often developed incrementally as new knowledge regarding the domain of interest is acquired and pushed into the system.

Reuse cost: an alternative to building ontologies from scratch is to reuse pre-existenting ones. This is actually the mainstream in knowledge engineering and most of the applications reported in the literature follow this approach. Empirical evidence from recent expirements in ontology reuse show that particular types of ontologies are more useful than others. In the context of the HPKB [11] project, it was found that very generic ontologies provide less support and are less useful than domain-specific ones. The latter scored a constant 60% rate of reuse in the HPKB study in contrast with the poor 22% rate of reuse scored by generic ontologies. However these results should not undermine their role in structuring knowledge: "Although the rate of reuse of terms from very general ontologies may be significantly lower, the real advantage of these ontologies probably comes from helping knowledge engineers organise their knowledge bases along sound ontological lines."[11].

Prior to reusing an ontology an engineer has to locate the right ontology first and then familiarise herself with it. We have seen some efforts to facilitate the selection task, as for example the reusable libraries of ontologies and in the same context ongoing work to produce frameworks which characterise and classify ontologies [33]. However, the familiarisation task remains a problem.

One practical approach here is the EF where its EBs contain characterized and classified experienceware described in different levels of abstraction. The approach supports a goal-oriented, similarity-based retrieval that helps to find the right piece of knowledge in the right time and within a justifiable effort. Based on a continuous evaluation of the EB the user herself decides whether or not more abstract knowledge is provided.

However, statistical results from reusing components in the software engineering domain tell us that just because we can access reusable compoments (like ontologies), this does not necessarily mean that we can use them as a productivity tool. Ontologies must be learnt prior to use and this learning curve may have a non-trivial impact on the overall cost.

Maintenance cost: in the long run this cost might hinder further deployment. It is not easily predictable and quantifiable since there are various angles of viewing this problem. For instance, if we accept that ontologies should rarely stabilise then we should expect to include in our budget along with the cost of constructing, costs for maintaining the ontologies we use as well as the system which uses them. How common is ontology instability? We don't know since we have very little experience with the long-term use of large libraries of ontologies. However, this is a debatable point and we find projects where ontologies were deployed on the rationale that they were stable, and projects where this is not taken for granted as ontologies are expected to change over time.

Purpose: apart from these cost-related factors, other issues emerge when we decide to use ontologies. These are the level of formality, often related to the purpose of use, level of support, technical obstacles to overcome, etc.

In particular the level of formality is a traditional point of contention in many fields. Despite the strong claims made by opponents of formality evidence shows that formal ontologies can be operationalised [21], which makes them suitable for enforcing automated consistency checking. On the other hand, this sort of use is not the mainstream and, arguably, not a cost-effective approach. This may justify the wide usage of semi-formal or even informal ontologies we see in the literature, which is mostly directed to deliver reuse and knowledge sharing rather than improving reliability.

The level of support for using ontologies and technical obstacles which should be overcomed, raised in industrial experiments where the ontologies used were outsourced. For example, as we report in [21], the translation activity involved was intensive and lack of automatic support was an important disadvantage. However, the situation changes when we look at other similar efforts when ontologies are build in-house rather than outsourced were no such complaints were made.

Acknowledgements: *This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. The views and conclusions containted herein are those of the author and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of the EPSRC or any other member of the AKT IRC.*

VI. References

[1] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn, and M. Sintek. Toward a Technology for Organizational Memories. *IEEE Intelligent Systems*, 13(3):40–48, June 1998.

[2] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn, and M. Sintek. Context-Aware, Proactive Delivery of Task-Specific Knowledge: The KnowMore Project. *International Journal on Information Systems Frontiers (ISF)*, 2(3/4):139–162, 2000.

[3] A. Abecker, A. Bernardi, S. Ntioudis, G. Mentzas, R. Herterich, C. Houy, S. Muller, and M. Legal. The DECOR toolbox for workflow-embedded organizational memory access. In *Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS'01), Setubal, Portugal*, July 2001.

[4] M. Aiken, O. Liu Sheng, and D. Vogel. Integrating expert systems with group decision support systems. *ACM Transactions on Information Systems (TOIS)*, 9(1):75–95, 1991.

[5] H. Alani, Y. Kalfoglou, K. O'Hara, and N. Shadbolt. Initiating organizational memories using ontology-based network analysis as a bootstrapping tool. *BCS-SGAI Expert Update*, 5(3):43–46, October 2002.

[6] C. Alexander. *Notes on the Synthesis of Form*. Harvard University Press, July 1974. ISBN: 0674627512.

[7] K-D. Althoff, F. Bomarius, and C. Tautz. Using Case-Based Reasoning Technology to Build Learning Software Organizations. In *Proceedings of the ECAI'98 Workshop on Building, Maintaining, and Using Organisational Memories (OM'98), Brighton, UK*, August 1998.

[8] V. Basili, G. Caldiera, and D. Rombach. Experience Factory. In *Encyclopedia of Software Engineering*, volume 1, pages 469–476. John Wiley & Sons, 1994.

[9] C. Carlsson and E. Turban. DSS: directions for the next decade. *Decision Support Systems*, 33:105–110, 2002.

[10] G. Casady. Rationale in Practice: templates for Capturing and Applying Design Expertise. In *Design Rationale: Concepts, Techniques, and Use*, pages 351–372. Lawerence Erlbaum Associates, 1996.

[11] P. Cohen, V. Chaudhri, A. Pease, and R. Schrag. Does prior knowledge facilitate the development of

knowledge-based systems? In *Proceedings of the Sixteenth National Conference on Artificial Intelligence, AAAI'99, Orlando, FL, USA*, pages 221–226, July 1999.

[12] P.J. Compton and R. Jansen. A Philosophical Basis for Knowledge Acquisition. *Knowledge Acquisition*, 2:241–257, 1990.

[13] W.S. Cooper. On selecting a measure of retrieval effectiveness. In *Readings in Information Retrieval*, pages 191–204, 1997.

[14] R. Davis and R. Smith. Negotiation as a metaphor for distributed problem solving. In *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1998.

[15] R. Dieng, O. Corby, A. Giboin, and M. Ribiere. Methods and tools for corporate knowledge management. *International Journal of Human-Computer Studies (IJHCS)*, 51:567–598, 1999.

[16] S. Easterbrook. Handling conflicts between domain descriptions with computer-supported negotiation. *Knowledge Acquisition*, 3:255–289, 1991.

[17] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multi-perspective specifications. *IEEE Transactions on Software Engineering*, 20(8):569–578, 1994.

[18] C. Gresse-vonWangenheim, D. Lichtnow, A. von-Wangenheim, and E. Comunello. Supporting Knowledge Management in University Software R&D Groups. In *Proceedings of the 3rd International Workshop on Learning Software Organizations (LSO'01), Kaiserslautern, Germany*, pages 51–66. Springer-Verlag, September 2001.

[19] J. Josephson, B. Chandrasekaran, M. Carroll, N. Iyer, B. Wasacz, and G. Rizzoni. Exploration of Large Design Spaces: an Architecture and Preliminary Results. In *Proceedings of the 15th National Conference on Artificial Intelligence, AAAI'98, Madison, Wisconsin, USA*, July 1998.

[20] Y. Kalfoglou. Maintaining ontologies with organisational memories. In *Knowledge Management and Organizational Memories*, Kluwer Academic Publishers, ISBN: 0-7923-7659-5, 2002.

[21] Y. Kalfoglou, T. Menzies, K-D. Althoff, and E. Motta. Meta-knowledge in systems design: panacea...or undelivered promise? *The Knowledge Engineering Review*, 15(4):381–404, December 2000.

[22] Y. Kalfoglou, D. Robertson, and A. Tate. Using Meta-Knowledge at the Application Level. Research Paper 956, Department of Artificial Intelligence, University of Edinburgh, 2000.

[23] T. Menzies. Towards situated knowledge acquisition. *International Journal of Human-Computer Studies*, 49:867–893, 1998.

[24] T.P. Moran and J.M. Carroll. *Design Rationale: Concepts, Techniques, and Use*. Lawerence Erlbaum Associates, 1996. ISBN: 0-8058-1567-8.

[25] H. Nissen, A. Jeusfeld, M. Jarke, G. Zemanek, and H. Huber. Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modelling Technology. *IEEE Software*, 13(2), March 1996.

[26] K. O'Hara, H. Alani, and N. Shadbolt. Identifying Communities of Practice: Analysing Ontologies as Networks to Support Community Recognition. In *Proceedings of the 2002 IFIP World Computer Congress, Montreal, Canada*, August 2002.

[27] D. O'Leary. Knowledge Management Systems: Converting and Connecting. *IEEE Intelligent Systems*, 13(3):30–33, June 1998.

[28] H. Rittel and M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4:155–169, 1973.

[29] P. Rosenbloom, J. Laird, and A. Newell. *The SOAR Papers*. The MIT Press, 1993.

[30] D.A. Schon. *The Reflective Practitioner*. Harper Collins, 1983.

[31] J.P. Shim, M. Warkentin, J.F. Courtney, D.J. Power, R. Sharda, and C. Carlsson. Past, present, and future of decision support technology. *Decision Support Systems*, 33:111–126, 2002.

[32] H. Simon. *The Science of the Artificial*. MIT Press, 1969.

[33] M. Uschold and R. Jasper. A Framework for Understanding and Classifying Ontology Applications. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods(KRR5), Stockholm, Sweden*, August 1999.

[34] P. Zave and M. Jackson. Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology*, 6(1):1–30, January 1997.