Towards Ontology-Guided Design of Learning Information Systems[1]

Aldo de Moor

STARLab, Vrije Universiteit Brussel
Belgium
ademoor@vub.ac.be

Abstract. Courseware increasingly consists of generic information and communication tools. These offer a plethora of functionalities, but their usefulness to a particular learning community is not easy to assess. The aim should be to develop comprehensive learning information systems tailored to the specific needs of a community. Design patterns are important instruments for capturing best practice design knowledge. Ontologies, in turn, can help to precisely capture and reason about these patterns. In this paper, we make the case for ontology-guided learning IS design, and sketch the ontological core of a potential design method. Such methods should enable communities to specify and access relevant best design practice patterns. Design knowledge can then be reused across communities, improving the quality of communication support provided, while preventing wheels from being reinvented.

1 Introduction

Courseware has become an essential means of course-based, Web-supported e-learning. A plethora of tools and environments exists. On the one hand, there are more or less comprehensive courseware platforms: commercial platforms like WebCT and Blackboard, and open platforms which can be completely or partially open source. However, e-learning is increasingly also being supported by collections of generic collaborative tools, such as blogs and wikis, e.g. [1].

Courseware platforms and tools offer many functionalities which can be used to support communities of users in their individual and especially collaborative needs. Much research and development efforts concentrate on producing increasingly advanced components for knowledge sharing and learning [2, 3]. However, many design and use problems arise as such functionality is becoming more widely available, including un(der)used functionalities, gaps between required and available

[1] This research was supported by the EU Leonardo da Vinci CODRIVE project B/04/B/F/PP-144.339. The author wishes to thank Jurrit de Vries and Jaap Wagenvoort for their help with the experiment described in the example.

This paper will be published in the Proc.of the International Workshop on Ontologies, Semantics & E-Learning (WOSE'05) of the OTM 2005 Federated Conferences, Agia Napia, Cyprus, Oct.31 - Nov.4, 2005. Lecture Notes in Computer Science, Springer:Berlin.

functionalities, and conflicting requirements of different stakeholders. Such mismatches between collaborative requirements and available functionalities can be characterized as socio-technical gaps [4].

Practical evaluation methods are necessary for effectively and efficiently designing comprehensive and useful learning information systems. These we define as sets of interrelated courseware and tool functionalities that satisfactorily serve the information and communication needs of a particular learning community. Many evaluation methods score and weigh the quality of functionalities. However, mere numerical evaluations have no clear semantics, and thus are hard to compare and reuse across learning communities. In order to provide learning communities also with semantic building blocks for designing their information systems, we examine in this paper the role that ontologies can play. To this purpose, we make the case for ontology-guided learning information system design patterns. Design patterns capture the essential lessons learnt by a community about which technologies work for it for what purposes. By formalizing these patterns using ontologies, the effectiveness and efficiency of its design processes, and thus of the resulting information systems, can be improved. In Sect. 2, we present a Social Context Model for communication processes, which provides the core patterns for the ontology. Sect. 3 examines the idea of ontology-guided learning IS design. We sketch the contours of a possible design method in Sect.4. We end the paper with conclusions.

2 A Social Context Model for Communication Processes

E-learning communities are communities of practice, which involve a domain of knowledge defining a set of issues, a community of people who care about this domain, and the shared practice that they develop to be effective in their domain [5]. Such communities require sophisticated system designs to support their communicative needs [6]. The problem with current analysis approaches is that they often classify communication technologies only in terms of the basic communication moves they afford, such as the creation of issues or arguments pro/contra. Needed, however, is a way to analyze collaborative technologies and the communication processes they enable in their full *context of use*. While a mailing list may work perfectly fine as a coordination tool in a small research community, it generally works badly as a tool to coordinate the multiple assignments given to many groups in a particular course. Another example is a divergent discussion process, which may be very useful in a friendly social community, but quite insufficient when used to support official document authoring processes with large interests at stake.

In [7, 8], we developed a Social Context Model (SCM) for the analysis of the functionalities provided by communication tools. The model helps to position communication technologies and processes in terms of their usefulness for particular communication purposes. The model examines communication processes from a process *context* dimension consisting of communication process layers, and from a process *structure* dimension describing the configuration of the elements of the communication processes.

The SCM consists of four layers of communication processes, each higher level process providing a context that embeds the lower-level processes. From high to low-level processes these are: collaboration, authoring, support, and discussion processes:

- **Collaboration processes** *give purpose* to the collaboration activities, discussions, and documents.
- **Authoring processes** *produce* the structured outputs of the collaboration processes, such as documents.
- **Support processes** focus on the *organization* of the discussions between the participants, ensuring that they contribute to the creation and interpretation of the output.
- **Discussion processes** are the actual conversational exchanges in which the argumentation between participants takes place.

Each communication process, whether it is a discussion process or one of its embedding context processes, has a structure comprised of certain process entities. First, there are the *process elements* (the elements a process itself is made of, such as goals, roles, and objects). Second, there are the processes constructed out of these elements. These we subdivide into *actions* (which constitute the actual communication workflows) and *change processes* (meta-processes in which the communication process can be adapted).

In both actions and change processes, there are *norms* that describe the acceptable behavior in the community, by defining the authorizations of the participants in the process roles that they play. All communities have such norms, either explicitly laid down in charters and by-laws, or only implicitly defined, but no less strong in impact [9, 10]. These context-grounded norms are the core elements needed to see which collaborative patterns may, must, or may not be present in a particular socio-technical system, and are thus essential in systems design.

We have applied the SCM in various ways. One use is to chart collaborative requirements in some community of practice. Another application is comparing the usage contexts of collaborative functionalities afforded by various tools. Once these data are known, matches between requirements and functionalities can be performed to identify socio-technical gaps, rank tools in terms of usefulness, etc.

	Process Elements *(Goals, Roles, Objects)*	Actions *(Workflows, Norms)*
Collaboration Proc. *(Why?)*		
Authoring Proc. *(What?)*	• Author • Editor	• Author – May – Add_Position_to_Section • Author – May –Take_Position_in_Section • Editor – Must – Edit_Section_Introduction • Editor – Must – Edit_Section_Conclusion
Support Proc. *(How organized?)*		
Discussion Proc. *(How done?)*	• Discussant • Position • Argument	• Discussant – May – Create_Position • Discussant – May – Add_Argument

Figure 1 An SCM Affordance Analysis of GRASS

An example of an examination of the affordances provided by a collaborative tool, is an analysis of the GRASS (Group Report Authoring Support System) tool, which allows adversarial stakeholder communities to assess the amount of consensus they have reached on conflict issues [11]. Fig.1 shows a simplified analysis of this example (amongst other things not taking into account the change processes). It shows that GRASS is especially strong in typical issue-network functionalities (creating positions and arguments pro/contra positions or other arguments). In addition, it allows these issue-networks to be part of group reports, by permitting authors to add positions (plus their associated argumentation trees) to particular sections, and to take positions as desired. These norms are privileges (permitted actions). Examples of responsibilities (required actions) are that editors *must* create section introductions and conclusions. The analysis also shows that two types of communication processes are *not* supported by GRASS at the moment: support processes (how to organize discussions, for example facilitation or summarization of discussions) and collaboration processes (the purpose of the reports produced, e.g. societal conflict resolution or class assignments).

Although the SCM has proven its value in practice, a limitation is still that so far it only has been used for informal, qualitative analysis. Its usefulness would increase, however, if part of its representation and associated analysis processes are formalized. One way to do so is by using the SCM as the basis for ontology-guided design of socio-technical systems.

3 Ontology-Guided Learning IS Design

Ontologies are useful instruments for improving the reusability of collaborative knowledge. An ontology is a shared conceptualization of an application area that defines the salient concepts and relationships among concepts, and as such can assist in modeling and implementing solutions to application tasks [12]. Three terms are especially important in this definition: *shared*, *conceptualization,* and *application*. In

order to be *shared*, there needs to be a process in which conceptualizations get agreed upon. The *conceptualization* itself requires some formalism, in the sense of knowledge representation plus reasoning procedures. Finally, this shared conceptualization should contribute to solving an *application* problem.

Reaching shared meanings is not trivial, since community members may have different interests and different visions of the domain under consideration [13]. The difficult social process of sharing and reaching consensus on joint meanings is an important area of ontological research [14, 15]. However, here we focus on the conceptualization itself, with the application area being design of learning information systems.

In the design of complex (socio-technical) systems, design patterns are very important. Humans are very good at using patterns to order the world and make sense of things in complex situations [16]. A pattern is a careful description of a perennial solution to a recurring problem within a […] context (Alexander, in [17]). Thus, patterns are context-dependent and relatively stable. They need to be concrete enough to be useful, while also being sufficiently abstract to be reusable. Such patterns can be captured and used in conceptual models like the SCM and ontology-based knowledge systems.

In our work, we have adopted conceptual graphs to model and use design patterns [18]. Conceptual graphs are a flexible and extensible method for knowledge representation [19, 20]. Their strength is not in the range of semantic constraints they can represent, other formalisms such as ORM provide much richer semantic constraint primitives [21]. What makes conceptual graphs particularly useful for pattern analysis are the generalization hierarchies, not only of concept and relation types, but also of complete graphs. The formalism contains a set of powerful operations such as projections, generalizations, and joins that make use of the properties of the graph hierarchies and their components. Furthermore, they not only support the description of complex domain knowledge, but also the validation of that knowledge against meta-knowledge about the domain (such as quality criteria that domain graphs have to match with). Thus, conceptual graphs are well suited to represent and reason about pattern and context knowledge.

Next, we outline some requirements and elements of a possible approach, using examples from a learning IS design case.

4 Towards a Design Method

There is no such thing as THE design choice in a community of practice. Each community has its own requirements, over time developing its own unique perspectives, common knowledge, practices, supporting technologies, approaches, and established ways of interacting [5]. Thus, to properly support a community in its systems development, design patterns need to evolve over time as well, generally becoming more specialized and tailored. Yet if communities are so unique, how then can their design best-practices be shared?

We illustrate the design problem and the use of conceptual graphs as the core of a potential design method by examining material from a real learning IS design case.

The main point we want to make in this paper is that ontology-based design is needed to create better learning information systems. Our aim is not not present a full method for ontology-guided learning information system design. Both the learning problem and conceptual graphs-based ontological analysis presented are much simplified examples of what is needed in practice. However, in an educational technology field predominantly focusing on standards and technology platforms, the case should illustrate that a systematic, formal semantics-based approach to learning information *systems*-design is an important part of the puzzle as well.

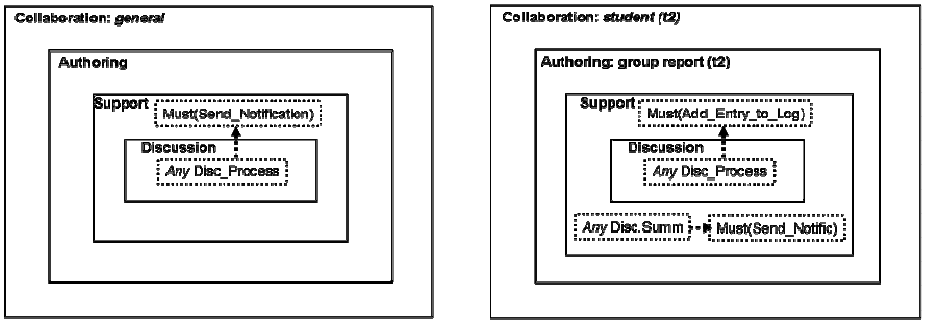
4.1 An Example: Refining Notification Patterns

Providing computer support for the synchronization of asynchronous collaborative learning activities is vital for learning communities [22]. For example, an important synchronization functionality is notification of events through e-mail. As soon as a document is added to a particular folder, the members of a workspace often receive an e-mail, including a clickable link so that they can view the change immediately.

Although conceptually simple, developing appropriate notification support by a learning information system requires answering many questions: who is to be notified? All users, or just a relevant subset? How to determine to which users a particular change is relevant? How often to notify users: with every event, daily, weekly? Which technology to use: a 'push' technology like e-mail, or a 'pull' technology like some web tool?

This issue came to the fore during an experiment with building and using a learning information system consisting of a set of generic information tools. In the Fall 2004 semester, 15 Information Management students taking a course given by the author on 'Quality of Information Systems' were given the assignment to jointly write a report on a societal theme. To this purpose, they were to use a set of blogs to collect and interpret source materials for the report, and use the GRASS tool mentioned earlier to write the report. Since discussions were scattered over many fora, we initially adopted the design pattern 'send a notification whenever some discussion process takes place' (Fig.2a). To our great surprise, of 13 students having participated in the evaluation held after the experiment, 10 'completely disagreed' and 3 'disagreed' with the proposition that the notification of changes in the report was valuable. This suggested that the general notification design pattern of Fig.2a should be discarded in a student population having the goal of writing a group report. Instead, the correct pattern seemed to be that no notification should be sent at all in case of an addition to the discussion. Still, this design pattern also seemed too extreme: to a follow-up question 7 students answered they did not want to receive any notification at all, but 2 still preferred a daily, and 4 a weekly digest. This, plus the fact that there had been some technical problems resulting in a flood of notification mails at one point in time, suggest that it was not so much the notification functionality itself that was problematic, but its high frequency and indiscriminate nature. An alternative design, therefore, could be to add an entry about any discussion event to a Web-accessible change log, while still sending each student an

e-mail about rare, 'quality'-events, such as the posting of a discussion summary by an editor (Fig.2b).



(a) The default notification pattern (b) The revised student notification pattern

Figure 2 Notification Pattern Refinement

4.2 Pattern Analysis with Conceptual Graphs

This example shows how design patterns can evolve in practice. It is important to note that both requirements (e.g. practices of students working on group reports) and the (mappings to) supporting functionalities (e.g. notification process functionality) become more specialized over time. Ontologies in general, and conceptual graphs in particular, allow for these patterns to be efficiently analyzed and reused when embedded in appropriate design methodologies. To show proof of principle, we now explain how the above example could be formalized using conceptual graphs.

Conceptual graphs are constructed out of concepts and relations. A concept has two fields: a type and a referent field. The type-field contains a type label that is part of a concept type hierarchy. The referent-field designates a particular entity with the mentioned type. This field is optional: if not specified, the concept is considered to be generic, and is by default existentially quantified. A conceptual relation links two or more concepts. Each conceptual relation has a relation type. It also has one or more arcs, represented by arrows, each of which must be linked to some concept. A conceptual graph is a combination of concept nodes and conceptual relation nodes. It can also consist of a single concept node.

Generalization/specialization semantics are a core feature of conceptual graphs. Besides concept type hierarchies, also a relation type hierarchy and a generalization hierarchy of graphs are distinguished. This enables powerful abstraction and comparison operations to be carried out, which are core building blocks of pattern analysis.

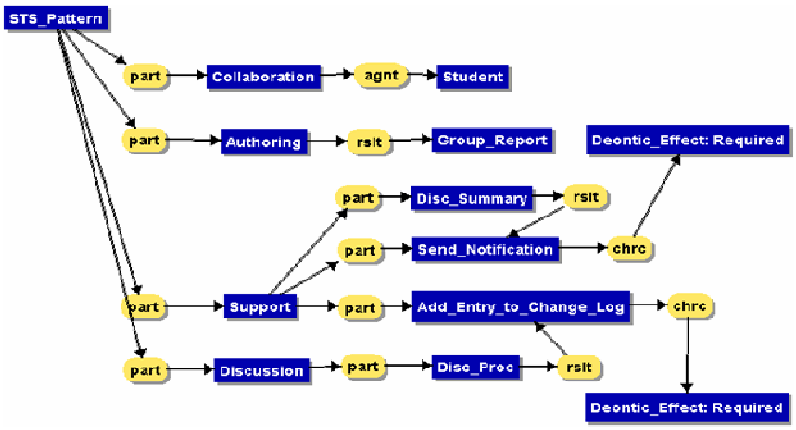
Returning to the notification example: the design patterns identified there can be easily translated into conceptual graphs. For example, Fig.3 contains the (partial) conceptual graph representation and associated concept type hierarchy of the informal notification design pattern depicted in Fig.2c. The key concept of the type hierarchy is

the STS_Pattern (socio-technical system-pattern). The hierarchy contains the four main types of SCM communication processes: Collaboration, Authoring, Support, and Discussion processes. A Deontic_Effect can be attached to concepts to indicate their normative status (i.e. required, permitted, forbidden). Only one type of Actor is currently distinguished, namely Student, while two types of functionalities are described: Send_Notification and Add_Entry_to_Change_Log. Three objects are Group_Report, Discussion_Summary, and Discussion_Process. The latter is labelled as an object, since in the system it is the representation of the discussion process (for example an addition to a discussion thread) that, for instance, triggers the addition of an entry to the change log. Of course, the discussion process as a *process* also needs to be of interest and included in the type hierarchy, but is not shown here to reduce the complexity of the example.

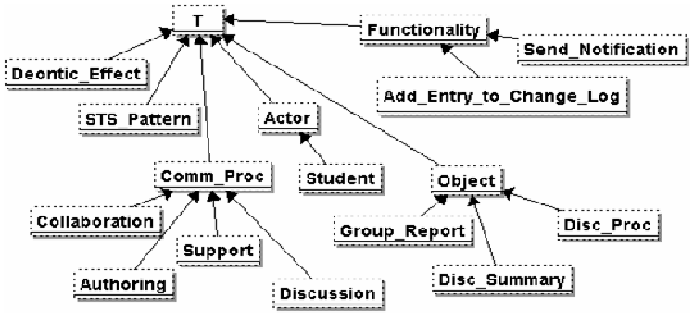
The notification pattern graph that is built on top of the semantics of this type hierarchy shows, for example, that a Support-process exists in which a Discussion_Summary-object being added triggers the Send_Notification-functionality, and that the latter has an associated Deontic_Effect (i.e. normative status) of being required. Note that the (agent, part, result, and characteristic) relations are often used in conceptual graph theory, but other, community-defined relations can be used just as well.

Having represented design patterns is only the first step. The second step is to use these representations in the design process. One way is to develop *query graphs*: using the standard conceptual graph-*projection* operation, all graphs from a set of socio-technical systems design patterns that are specializations of the query graph will be returned. This can help a community in reusing existing ideas for its own design patterns.

In [18], we present a more elaborate method based on socio-technical systems patterns that allows for collaboratory improvement. A similar method could be developed for learning information systems design.



(a) A conceptual graph of the revised notification pattern



(b) The associated concept type hierarchy

Figure 3 A Conceptual Graph of the Notification Design Pattern

5 Conclusions

Learning information systems, meaningfully constructed out of constellations of courseware platforms and generic communication tool components, will play an increasingly important role in supporting learning communities. Design patterns are useful instruments in facilitating the community-driven systems development process. Although useful, informal design patterns such as based on the Social Context Model, are not sufficient when reusability of patterns is a goal. Ontologies, in the form of shared, formalized conceptualizations of a domain and application area, can help promote reusability and usefulness of such patterns.

The main point of this article was to show the need for ontology-guided design of learning information systems. We only gave a brief sketch of a possible method for supporting this process using conceptual graphs, and did not give more details about the particular ontological approach to use, since that was not the point. In future work, at the very least, an extension of the SCM concept (and relation) type hierarchy is needed, like more refined goals, roles, objects, workflows, and norms, although the

particular extensions will depend on the (category of) learning community. By building on a common ontological core, each community can define its own semantics, while still allowing for comparisons with other patterns to be made. Ultimately, such conceptualizations should become full (learning IS) pattern languages that allow communities to express rich collaboration requirements, i.e. along the lines of the Pattern Language for Living Communication described in [17]. Such a language, in turn, is the starting point for the design of a proper community design *methodology,* which was outside the scope of this paper but equally important.

To capture and reason about the formalized patterns, conceptual graphs are not the only formalism to be used. Related ontology formalisms and methods are extensively being researched in the Semantic Web and business process modeling research communities, e.g. [23]. Multiple knowledge representation formalisms can and should complement and enrich each other [24]. One of our research objectives therefore is to extend the powerful pattern comparison feature of conceptual graphs with the rich semantic constraints provided by Object-Role Modeling [21]. This research will take place in the context of the DOGMA approach to ontology engineering, developed at STARLab [25]. This approach aims to satisfy real world needs by developing a useful and scalable community-grounded ontology engineering approach, allowing for much more subtle representation and analysis of design patterns. Such a combination of a pragmatic ontological engineering methodology informed by high-quality, learning community defined design patterns, could open the road to much more customized, high-quality learning information systems.

References

1. Efimova, L. and S. Fiedler. Learning Webs: Learning in Weblog Networks. in *Proc. of the IADIS International Conference on Web Based Communities 2004, Lisbon, Portugal*. 2004: IADIS Press.
2. Roschelle, J., et al., Developing Educational Software Components. *IEEE Computer*, 1999. 32(9): p. 2-10.
3. Bieber, M.e.a. Towards Knowledge-Sharing and Learning in Virtual Professional Communities. in *Proc. of the 35th Hawaii International Conference on System Sciences, Hawaii, January 5-7, 2002*. 2002.
4. Ackerman, M.S., The Intellectual Challenge of CSCW: the Gap Between Social Requirements and Technical Feasibility. *Human-Computer Interaction*, 2000. 15(2): p. 179-203.
5. Wenger, E., R. McDermott, and W. Snyder, *Cultivating Communities of Practice*. 2002, Cambridge, MA: Harvard Business School Press.
6. Etzioni, A. and O. Etzioni, Face-to-Face and Computer-Mediated Communities: A Comparative Analysis. *The Information Society*, 1999. 15: p. 241-248.
7. de Moor, A. and R. Kleef. Authoring Tools for Effective Societal Discourse. in *Proc. of the 15th International Informatics for Environmental Protection Symposium: Sustainability in the Information Society*. 2001. Zurich, Switzerland.
8. Kleef, R. and A. de Moor, Communication Process Analysis in Virtual Communities on Sustainable Development, in *Environmental Online Communication*, A. Scharl, Editor. 2004, Springer: Berlin.

9. Preece, J., *Online Communities : Designing Usability, Supporting Sociability*. 2000, Chichester ; New York: John Wiley.
10. Wershler-Henry, D. and M. Surman, *Commonspace: Beyond Virtual Community*. FT.Com Books. 2001, Toronto: Pearson.
11. Heng, M. and A. de Moor, From Habermas's Communicative Theory to Practice on the Internet. *Information Systems Journal*, 2003. 13(4): p. 331-352.
12. Musen, M.A., Ontology-Oriented Design and Programming, in *Knowledge Engineering and Agent Technology*, J. Cuena, et al., Editors. 2000, IOS Press: Amsterdam.
13. Gruninger, M. and J. Lee, Ontology: Applications and Design. *Communications of the ACM*, 2002. 45(2): p. 39-41.
14. Euzenat, J., Building Consensual Knowledge Bases: Context and Architecture, in *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, N. Mars, Editor. 1995, IOS Press: Amsterdam. p. 143-155.
15. Edgington, T., et al., Adopting Ontology to Facilitate Knowledge Sharing. *Communications of the ACM*, 2004. 47(11): p. 85-90.
16. Kurtz, C.F. and D.J. Snowden, The New Dynamics of Strategy: Sense-Making in a Complex and Complicated World. *IBM Systems Journal*, 2003. 42(3): p. 462-483.
17. Schuler, D. A Pattern Language for Living Communication. in *Participatory Design Conference (PDC'02), Malmo, Sweden, June*. 2002.
18. de Moor, A., Improving the Testbed Development Process in Collaboratories, in *Proc. of the 12th International Conference on Conceptual Structures (ICCS 2004), Huntsville, AL, USA, July 2004*. 2004. p. 261-274.
19. Sowa, J.F., *Conceptual Structures : Information Processing in Mind and Machine*. 1984, Reading, Mass.: Addison-Wesley.
20. Mineau, G.W., R. Missaoui, and R. Godinx, Conceptual Modeling for Data and Knowledge Management. *Data & Knowledge Engineering*, 2000. 33: p. 137-168.
21. Halpin, T., Object-Role Modeling (ORM/NIAM), in *Handbook on Architectures of Information Systems*, P. Bernus, K. Mertins, and G. Schmidt, Editors. 1998, Springer-Verlag: Berlin.
22. Lundin, J., Synchronizing Asynchronous Collaborative Learners, in *Communities and Technologies*, M. Huysman, E. Wenger, and V. Wulf, Editors. 2003, Kluwer Academic: Dordrecht. p. 427-433.
23. Sheth, A. and R.A. Meersman, eds. *Proc. of the NSF-EU Workshop on Database and Information Systems Research for Semantic Web and Enterprises, April 3-5, Amicalola Falls and State Park, GA, USA*. 2002.
24. Markman, A.B., *Knowledge Representation*. 1999, London: Lawrence Erlbaum.
25. Spyns, P., R.A. Meersman, and M. Jarrar, Data Modelling versus Ontology Engineering. *ACM SIGMOD Record*, 2002. 31(4): p. 12-17.