Developing Ontologies to Enable Knowledge Management: Integrating Business Process and Data Driven Approaches

Henry M. Kim

Schulich School of Business, York University, 4700 Keele Street Toronto, Ontario Canada M4P1Y4

hmkim@ssb.yorku.ca

Abstract

A knowledge management system must support the integration of information from disparate sources, wherein a decision maker manipulates information that someone else conceptualised and represented. So the system must minimise ambiguity and imprecision in interpreting shared information. This can be achieved by representing the shared information using ontologies. There are typically two approaches to developing ontologies to support decision making. In one approach, ontologies are developed to support new business processes or decisions, but often are not built from existing data repositories. In the other approach, ontologies are developed from existing data repositories, but often may not support new business processes and decisions. In this paper, a methodology for knowledge management that combines both process and data driven approaches to ontology development and builds on them is described. In this methodology, called the BPD/D Ontological Engineering Methodology, competency questions that state the capability of an ontology to support business processes and decisions are specified. Concurrently, architectural requirements that specify aspects of existing systems that constrain ontology design choices are also stated, so that existing data repositories are explicitly considered and built upon when developing ontologies. Information systems tools to support both ontology-based knowledge management system construction and use can then be designed to support the steps of this methodology.

Introduction

Due to the ubiquity of computers, data are represented, presented to users, and interpreted in numerous ways. There are heterogeneous data formats, semantics, and languages even and especially within the same organisation. One way to use information systems to support decisions that require integrating heterogeneous data is to build a new system with uniform data characteristics. Many companies have purchased Enterprise Resource Planning (ERP) software [Cameron et.al. 1996] to ensure that all data that support operational decisions are in the same format, semantics, and language. Alternatively, organisations can modify existing systems so that data with different characteristics can be integrated. The latter approach is more appropriate when non-routine, analytical decisions need to be made by integrating data that reside on established, dispersed repositories; the cost of constructing a new system with unified data cannot be generally justified just to support ad hoc decisions. Therefore, this approach of data integration is appropriate for knowledge management (KM).

Integration for KM can be considered as the means to bring together disparate perspectives based upon what they have in common. One promising way to do this is by manipulating explicit representations of concepts that are common to heterogeneous data. Ontologies from the applied AI field can be used for data integration, since an ontology is an explicit representation of shared understanding; though there are many definitions for the term ‘ontology’ [Fox and Gruninger 1998] [Gruber 1993], this one succinctly characterises why an ontology is useful for data integration. In order to represent a shared understanding, a model of a domain must minimise ambiguity in interpretation by those who are using it to share data. In constructing an ontology, a domain’s key assumptions, vocabulary, and principles are made explicit, represented, and generalised. This reduces the possibility of ambiguity, which often results from different assumptions, vocabulary, and principles implicitly held by those who try to share data.

The state of the art in ontological engineering methodology research can be characterised as follows [Jones, Bench-Capon, and Visser 1998]: There exist few explicit methodologies, as well as records of rules of thumb and principles for constructing ontologies.

Explicit ontological engineering methodologies such as those for the TOVE [Gruninger 1996][Gruninger 1996] and Enterprise Ontology [Uschold and King 1995] projects are used to develop ontologies for specific business software applications (ontology-based systems), where ontology representations are designed to satisfy the applications’ requirements. The ontology-based system is used to support existing or newly planned business processes. When stating ontology requirements, emphasis is neither placed on integrating data from existing repositories such as conceptual Entity-Relationship models and structured web pages, nor designing the ontology-based system in the context of existing information systems architecture and constraints. These methodologies are used to design the content of an ontology for a system to support explicit business processes, and to facilitate re-using ontology representations for system support of additional business processes in the future.

Many efforts explicitly specify ontology development principles, as opposed to a methodology. Some support applications like natural language processing (e.g. µKosmos [Mahesh and Nirenburg 1995]) and common-sense reasoning (e.g. Cyc [Guha et. al. 1990]). They also emphasise designing ontology content to support the application. Some like Sensus [Swartout et. al. 1997] and work of Guarino et. al. [Guarino, Carrara, and Giaretta 1994] state how representations can be defined and constrained to reduce interpretation ambiguity for ontology sharing. The focus is ontology content design for sharing representations, not application support.

Yet others, like ONIONS [Gangemi, Steve, and Giancomelli 1996] and Physsys [Borst et. al. 1996], specify systems architectures and constraints so that ontologies as a whole can be shared, regardless of how a given term in an ontology is defined. The focus is on specifying the appropriate context for ontology sharing, not on ontology content. Ontological engineering efforts such as [Visser and Cui 1998] and KRAFT [Gray et. al. 1997] are of this type, since these investigate systems architectures and constraints for designing ontology clusters for integrating data that reside in disparate repositories.

So different ontological engineering efforts can be characterised according to the following:

- Ontology Development Need: for specific application vs. for sharing and re-use
- Ontology Development Emphasis: content vs. context

Ontologies used for knowledge management should be developed using a methodology suitable for different ontology development needs and emphases. The following principles can be used to design such a methodology:

1. Build upon an explicit state of the art ontological engineering methodology

   For developing ontology content for specific applications, similar to how established methodologies are used.
2. Customised for an organisation's application needs.

   So that an organisation's context can be accounted for in developing an ontology-based application
3. Include steps to systematically capture and incorporate ontology development rules of thumb and principles.

   For developing ontology content and context for sharing and re-use, as is done by several ontology efforts

By incorporating these design principles, an ontological engineering methodology for any organisation, which is useful for constructing business process applications and supporting ontology content and context developments can be designed; it can then be used for business process driven ontology development. The methodology can also be used for constructing applications, as well asused to facilitate ontology sharing and re-use when data repository and information systems architecture and constraints set the context for ontology development; it can also be used for data driven development.

In this report, the resulting methodology called the *BPD/D (Business Process and Data Driven) Ontological Engineering Methodology* is developed and discussed, and examples of its prototypical use are given. Steps in the methodology are detailed in the next section.

BPD/D Ontological Engineering Methodology: Basics

The steps of the BPD/D Ontological Engineering Methodology are classified according to the methodology extension directions stated above.

1) Builds upon an explicit state of the art ontological engineering methodology.
   - The TOVE Ontological Engineering Methodology developed at the Enterprise Integration Laboratory at the University of Toronto serves as the basis upon which the BPD/D methodology is based.
   - The steps in the TOVE methodology incorporated into the BPD/D methodology are:
     - ➢ *Motivating Scenarios*: Detailed narrative about business issues and problems that ultimately an ontology-based system will address.
     - ➢ *Posing Competency Questions*: To pose business questions that are answered using ontology representations, both informally (in English) and formally (in the language in which the ontology is expressed); competency questions characterise the problem solving capability of an ontology.
     - ➢ *Domain Analysis & Statement of Assumptions*: To explore the domain of discourse and explicitly identify key assumptions about the use of the ontology of that domain.

- ➢ *Ontology Development - Data Models*: To explore the domain of discourse and identify key terms (objects) of the domain, as well as models of relations between these terms including classification hierarchies, and attributes of terms.
- ➢ *Ontology Development - Axioms*: To explore the domain of discourse and define the meanings of ontology terms in a formal language that supports automatic deduction.
- ➢ *Answering Competency Questions*: To express the competency question as an axiom, and then to automatically deduce whether the axiom can be entailed from axioms of the ontology.

2) Customised for an organisation's application needs
   - Though TOVE methodology's competency questions can be used to develop ontology content, there does not exist a step in that methodology to explicitly make design choices about the context for the ontology-based application. So within the BPD/D Methodology, *architectural requirements* are stated, and design choices are made to fulfil these requirements.
   - "The organisation has many conceptual data models and accompanying database schema." This is a description of the environment within which an ontology is developed. Through domain analysis, representations for answering representational competency questions are made explicit. Similarly, through application environment characterisation, the rationale for making design choices to fulfil architectural requirements is then made explicit.
   - Also, the *fulfilling architectural requirements* step is performed at the end of the methodology as verification.

3) Includes step to systematically capture and incorporate ontology development rules of thumb and principles.
   - ➢ An interesting body of research in developing ontology context for sharing and re-use of data in existing repositories is ontology clustering. The BPD/D Ontological Engineering Methodology supports ontology clustering by explicitly stating steps for the construction of domain and context dependent application ontologies, as well as general representations organised in shared ontologies, which are more domain and context independent. The steps are called *application ontology development* and *shared ontology augmentation/modification*.

The following table summarises the development of the steps of the BPD/D Ontological Engineering Methodology:

TOVE Ontological Engineering Methodology Steps	**BPD/D Methodology design principles applied to TOVE steps**	**BPD/D Ontological Engineering Methodology Steps**
Motivating Scenario	1) Builds upon an explicit state of the art ontological engineering methodology.	<u>Motivating Scenario</u> 1. Motivating Scenarios
Stating Competency Questions	1) Builds upon an explicit state of the art ontological engineering methodology. 2) Customised for an organisation's application needs	<u>Stating Ontology Requirements</u> 2. Pose Representational Competency Questions 3. State Architectural Requirements
Domain Analysis & Statement of Assumptions	1) Builds upon an explicit state of the art ontological engineering methodology. 2) Customised for an organisation's application needs	<u>Ontology Analysis</u> 4. Domain Analysis & Statement of Assumption 5. Application Environment Characterisation
Ontology Development	1) Builds upon an explicit state of the art ontological engineering methodology. 3) Includes step to systematically capture and incorporate ontology development rules of thumb and principles.	<u>Ontology Development</u> 6. Application Ontology Development – Representational 7. Application Ontology Development – Architectural 8. Shared Ontology Modification/Augmentation – Representational 9. Shared Ontology Modification/Augmentation - Architectural
Answering Competency Questions	3) Builds upon an explicit state of the art ontological engineering methodology. 2) Customised for an organisation's application needs	<u>Fulfilling Ontology Requirements</u> 10. Answering Competency Questions 11. Fulfilling Architectural Requirements

Table 1: BPD/D Ontological Engineering Methodology Steps

In subsequent sections, each BPD/D Ontological Engineering Methodology step is discussed in detail. Discussion of each step includes examples of how the methodology is used for constructing ontologies.

Motivating Scenario

The motivating scenario is a detailed narrative about an enterprise, where special emphasis is placed on the problems that it is facing or tasks it needs to perform. An application built using an ontology would be used to solve this problem. For example, the following motivating scenario [Kim 1999] for a steel manufacturer provides background information about the company, statements of its product quality concerns, the company's quality terminology, explanations of its intended enterprise model use, and how it currently handles defects.

Here is a motivating scenario about the challenges of e-commerce to firms [Forrester 1999]:

- The Commerce Integration Imperative: Internet commerce sites must tie into back-end systems to satisfy on-line customers. To integrate cost-effectively and produce a rich sales and service experience, firms must match their integration investments to specific business needs, market dynamics, and customer expectations.

Stating Ontology Requirements

In order to develop representational competency questions and architectural requirements, the motivating scenario must be parsed. That is, key words and phrases are identified and classified as in the table below. Representational competency questions are questions about the world answerable using an ontology-based model; questions about content can be answered, but questions about context cannot. Architectural requirements are requirements upon the model use that are independent of the content of an ontology. An ontology that addresses the example Motivating Scenario can be characterised this way:

- Ontology of: Sales and Service [content]
- Ontology for: Effective e-commerce [context]

Important terms and phrases from the Motivating Scenario can be parsed as follows:

Key Word or Phrase	Does it describe some content of an ontology? (Is it a useful concept for sales and service?)	Is it useful for setting context of an ontology? (Does it describe something about e-commerce systems?)
"tie into"	N	Y
Back-end systems	N	Y
"satisfy"	Y	N
On-line customers	Y	Y
"integrate cost-effectively"	N	Y
"produce a rich experience"	N	Y
Firms	Y	Y
"match"	N	Y
Integration investments	N	Y
Specific business needs	Y	N
Market dynamics	Y	N
Customer expectations	Y	Y

Table 2: Parsing Motivating Scenario

Representational Competency Questions

Say, there is a firm called ZZZ. Using key words and phrases that document the ontology content, business or management questions (not technical ones) likely to be asked by ZZZ are posed like this:

- Does a certain firm, which is one of ZZZ's on-line customers, satisfy ZZZ's specific business needs?

Such general questions are vague and require clarification. By decomposing a general question to sub-questions like the following, clarification is possible:

- Given that one of *ZZZ's specific business needs* may be to improve its technological leadership, is a certain *firm* a *good customer* from a *technological perspective*?

This question can be hierarchical decomposed:

- Can ZZZ sell new technologies to the customer?
  - What products does the customer currently have?

An ontology used to answer this question must be able to answer the following straightforward questions:

- Is X a ZZZ product?
- Is Y a ZZZ customer?

So, based upon this decomposition of competency questions, the ontology developed must represent the terms, *ZZZ product* and *ZZZ customer*.

Architectural Requirements

Similar to representational competency questions, architectural requirements can be decomposed hierarchically using key terms and phrases from the Motivating Scenario. These requirements are of the type that a technical systems designer (not a business analyst or a manager) would state, like the following:

- The ontology-based system must *build on on-line integration investments* that ZZZ has already made.

This requirement can be hierarchically decomposed:

- The ontology-based system must use as much of the conceptual models with which the operational databases were designed; i.e. it should not require designing a completely different conceptual model, nor should it require building a model that is inconsistent with existing ones.
  - It must be determined if the conceptual language of the ontology is the same as the conceptual language of existing databases, and pros and cons for the design alternatives must be stated.
    - The implementation language of the ontology must be determined. One factor to consider is whether ontology implementation and conceptual modelling languages are closely coupled.
      - The implementation language should support representing user-defined relations.
        - The implementation language should support representing constraints.

This example shows how to decompose higher-level requirements on the ontology-based system to lower-level requirements on concrete features of the ontology language.

Ontology Analysis

Domain Analysis & Statement of Assumption

This step entails analysing how to answer a given representational competency question. It also entails explicitly stating assumptions that bound the scope of the domain. So it provides content for posing and answering competency questions, and hence is driven by them. For instance, in order to answer the question 'Is X a *ZZZ product*,' the following statements and assumptions are made:

- In reality: The term *ZZZ product* is an important term.
- In the ontology: *ZZZ product* should be represented as a *product* with additional characteristics; that is, it should be a specialisation of the term *product*.

- In reality: *Products* are sometimes characterised by their benefits.
- In the ontology: The term *benefit* should be represented. There should be some constraints stating allowable values for *benefit*, otherwise any arbitrary string would be allowed as a benefit.

In this step, key concepts of the modelled world are distilled, and how these concepts will be represented is explored.

Application Characterisation

This step entails analysing how to build an ontology as a module in the overall ontology-based system, and in the current business and systems environment within which the system will work. It sets the context for stating and fulfilling the architectural requirements, and so is driven by them. For instance, the following statements can be made in order to fulfil the requirement, 'The implementation language of the ontology must be determined…"

- ZZZ has some implemented ontologies
- ZZZ has numerous RDBMS tables
- Conceptual models should be the primary source for ontology development
- Some conceptual models are "rich" (e.g. ER models), while some are "semi-rich" (e.g. bubble diagrams of entities and attributes).

In this step, relevant facts about the world that affect the context of the ontology are stated, so that these can be taken into account in designing the ontology.

Ontology Development

Application Ontology and Shared Ontologies Development – Architectural

In this step, the ontology system architecture is developed based upon the Application Characterisation. Features of the system architecture then should fulfil the Architectural Requirements. Shown below is an example partial system architecture.

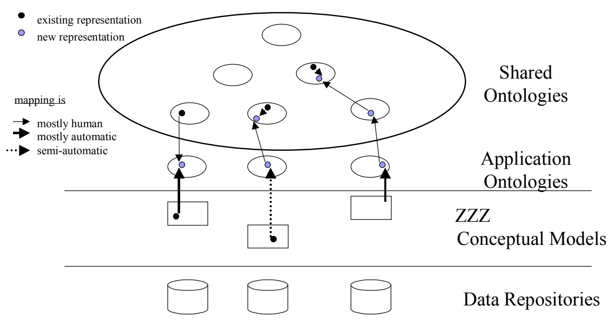

Fig. 1: Example Partial System Architecture for Ontology Development and Sharing

The following are features of this system architecture:

- Application Ontology representations are constructed by mapping from ZZZ conceptual models. The type of mapping can range from automatic translation from one language to the ontology conceptual language if the conceptual models are "rich," or mostly human translation if conceptual models are "semi-rich."

- Application Ontology representations are also constructed by specialising and composing definitions using terms from the Shared Ontologies.
- Application Ontology representations can also be generalised and included in the Shared Ontologies.

Moreover, an ontology implementation language must be able to support the following. This is only a partial list.

- Object orientation
- Classification hierarchy for entities
- Classification hierarchy for relations and attributes
- User defined relations that are mapped from conceptual models

Application Ontology and Shared Ontologies Development – Representational

In this step, ontology representations are developed based upon the Domain Analysis and Statement of Assumptions. These representations then are used to answer the representational competency questions.

Shown below is a partial conceptual model used to construct a products database:

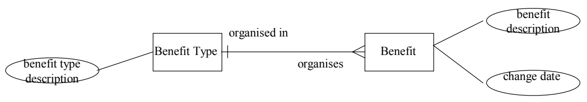

Fig. 2: Partial Conceptual Model for a Products Database

This partial model can be re-expressed in the ontology implementation pseudo-language[1] as in the example below:

```
PRIMITIVE ATTRIBUTE DEFN change_date
SUBCLASS OF: modification_date
DOMAIN VALUES: "yyyy-mm-dd-hh:mm:ss"
```

The other attributes, `benefit_type_description`, and `benefit_description` are similarly defined as primitive attributes of the application ontology:

The relations in the partial conceptual model are expressed as follows:

```
PRIMITIVE RELATION DEFN organises
SUBCLASS OF:            element_of
CARDINALITY CONSTRAINT: 1:M
DOMAIN VALUES:          (benefit_type,
benefit)
INVERSE RELATION:       organised_by
```

With these definitions, the entities, `benefit` and `benefit_type`, can be defined as follows:

```
PRIMITIVE ENTITY DEFN benefit_type
SUBCLASS OF:     feature_type
ATTRIBUTES:      benefit_type_description

ENTITY DEFN benefit
SUBCLASS OF: feature_descriptive
RELATIONS:   organised_by = benefit_type
ATTRIBUTES:  { change_date,
               benefit_description }
```

[1] Since the implementation ontology language has not been chosen yet and it is known that it is object-oriented, it suffices to represent these expressions in a pseudo-language since this language's grammar is that of most object-oriented database languages.

The expressions can be interpreted as follows:

- `benefit_type` is a primitive entity of the application ontology, since it is not characterised beyond the assertion that it is a subclass of `feature_type` and that it has an attribute called `benefit_type_description`.
- `benefit` is a defined entity since it is characterised as a `feature_descriptive` which is `classified_by` a `benefit_type`. In plain English, this can be stated as the following: a benefit is a descriptive feature that is classified by a benefit type. As well, benefit has `change_date` and `benefit_description` as attributes.

With the application ontology expressed, then the shared ontology has to be modified or augmented to ensure that satisfactory definitions exist for the following:

- **`modification_date`** attribute
- **`element_in`** relation
- **`feature_type`** entity
- **`feature_descriptive`** entity

Fulfilling Ontology Requirements

This part of the methodology is not addressed in this document.

Conclusion

This paper documents work that marries two types of ontology research and applies it to an important application. Namely, work in ontological engineering methodology with a knowledge representation focus at the Enterprise Integration Laboratory at the University of Toronto is combined with research in ontological engineering system architectures. The hybrid methodology, called the BPD/D Ontological Engineering Methodology, can be used for both business process and data driven approaches to ontology development. It is comprised of the following steps:

4. Motivating Scenario
5. Representational Competency Questions
6. Architectural Requirements
7. Domain Analysis and Statement of Assumptions
8. Application Environment Characterisation
9. Application Ontology Development – Representational
10. Application Ontology Development – Architectural
11. Share Ontology Modification/Augmentation – Representational

12. Shared Ontology Modification/Augmentation – Architectural
13. Answering Competency Questions
14. Fulfilling Architectural Requirements

The prototyped methodology can be used in the following ways:

- Steps in the methodology can be specialised to give directions for ontology development using software tools. For example, 'Application Ontology Development – Representational' step can be supported by software tools.
- Steps in the methodology can be specialised as more application ontologies are built and shared ontologies are augmented. Not only will there be a richer set of representations that can be classified in libraries, there will also be a richer set of architectural requirements that can also be organised. Ideally then, constructing application ontologies will entail following specific steps that have been distilled from the best practices of an organisation's numerous ontological engineering efforts, and specialising from a rich set of shared ontology libraries as well as putting together architectural requirements from templates of old requirements.

References

[Borst et. al. 1996] Borst, P., Benjamin, J., Wielinga, B., and Akkermans, H. "An Application of Ontology Construction", ECAI-96 Workshop on Ontological Engineering, Budapest, August 13, 1996.

[Cameron et.al. 1996] Cameron, B., Colony, G., Woodring, S., Rhielander, T., Lieu C., "The Prudent Approach to R/3", Packaged Application Strategies, Vol.1, No. 1, April 1996, Forrester Research, Inc., 1033 Mass. Ave., Cambridge, MA 02138.

[Forrester 1999] Forrester Research. The Commerce Integration Imperative [Online]. Available: http://www.forrester.com, 1999.

[Fox and Gruninger 1998] Fox, Mark S. and Grüninger, Michael. "Enterprise Modelling", AI Magazine, AAAI Press, Fall 1998, pp. 109-121.

[Gangemi, Steve, and Giancomelli 1996] Gangemi, A., Steve, G., and Giancomelli, F. "ONIONS: An Ontological Methodology for Taxonomic Knowledge Integration", ECAI-96 Workshop on Ontological Engineering, Budapest, August 13, 1996.

[Gray et. al. 1997] Gray, P., Preece, A., Fiddian, N.J., Gray, W.A., Bench-Capon, T., Shave, M., Azarmi, N., Wiegand, M., Ashwell, M., Beer, M., Cui, Z. et al. "KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases", Proceedings of 8th International Workshop on database and Expert Systems Applications (DEXA'97). 1997.

[Gruninger 1996] Gruninger, Michael. "Designing and Evaluating Generic Ontologies", Proceedings of the Workshop on Ontological Engineering, European Conference on Artificial Intelligence, Budapest, 1996, pp. 53-65.

[Gruber 1993] Gruber, Thomas R. "Towards Principles for the Design of Ontologies Used for Knowledge Sharing", In International Workshop on Formal Ontology, N. Guarino & R. Poli, (Eds.), Padova, Italy, 1993.

[Guarino, Carrara, and Giaretta 1994] Guarino, N., Carrara, M., and Giaretta, P. "An Ontology of Meta-Level Categories", in Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference, J. Doyle, E. Sandwell, and P. Torasso (Eds.), Morgan Kaufmann, San Mateo, CA, 1994.

[Guha et. al. 1990] Guha, R. V., Lenat, D. B., Pittman, K., Pratt, D., and Shepherd, M. "Cyc: A Midterm Report", Communications of the ACM, Vol. 33, no. 8, August 1990.

[Jones, Bench-Capon, and Visser 1998] Jones, Dean, Bench-Capon, Trevor, and Visser, Pepjin, "Methodologies for Ontology Development", Proc. IT&KNOWS Conference of the 15th IFIP World Computer Congress, Budapest, Chapman-Hall, 1998.

[Kim 1999] Kim, Henry M. "Representing and Reasoning about Quality using Enterprise Models", Ph.D. Thesis, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, M53G9, 1999.

[Mahesh and Nirenburg 1995] Mahesh, K. and Nirenburg, S. "A Situated Ontology for Practical NLP", Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, August 19-20, 1995.

[Swartout et. al. 1997] Swartout, W.R., Patil, R., Knight, K., and Russ, T. "Towards Distributed Use of Large-Scale Ontologies", AAAI-97 Spring Symposium on Ontological Engineering, Stanford University, May, 1997.

[Uschold and King 1995] Uschold, M. and King, M. "Towards a Methodology for Building Ontologies" IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995.

[Visser and Cui 1998] Visser, P.R.S. and Cui Z. "Heterogeneous Ontology Structures for Distributed Architectures", Proceedings of ECAI98 W/S on Applications of Ontologies and Problems-solving Methods. August 1998.