Business Process Ontologies: Frequently Asked Questions

by Dieter E. Jenz

President, Jenz & Partner GmbH

July, 2003

Jenz & Partner GmbH

Hainstr. 40a

63526 Erlensee

Germany

Phone : ++49-(0)6183-9100-0

Email: info@jenzundpartner.de

Internet: http://www.jenzundpartner.de

Frequently Asked Questions

General

The business case

Standards and tools

Using ontologies in practice

Answers

What is an ontology?

An ontology defines the terms and concepts (meaning) used to describe and represent an area of knowledge, as well as relations among them. Thus, an ontology includes

- Concepts (things) in the domains of interest
- Relationships between those things
- Properties (and property values) of those things
- The functions and processes involving those things
- Constraints on and rules about those things.

A well-formed ontology is one that is expressed in a well-defined syntax which has a well-defined machine interpretation. Purpose-specific generators interpret the machine-processable specification and produce specific output, such as UML class models and business process definitions.

The ontology concept is nothing really new. Indeed, for example, an ontology has many similarities with database schemas or UML class diagrams. However, an ontology is not about modeling entity types and their relationships and optimizing them, but to gain an understanding of existing things and their relationships in a language close to natural language.

What is a knowledge base?

A knowledge base is the result of instantiating an ontology, i.e. populating an ontology with data. As such, a knowledge base contains structure and data.

To give you a practical example: A knowledge base has much in common with a populated relational database. As you know, a relational database consists of metadata (the database schema) and database records, stored in tables. For example, when John Bloggs places an order, a record would be created in the database, effectively instantiating the database schema.

What does an ontology look like?

An ontology is based on a taxonomy which represents a class hierarchy in the object-oriented world. People are generally used to designing taxonomies. For example, a product catalog is implicitly based on a taxonomy. In the process of designing a product catalog, a domain expert would identify product groups and would layout a product group hierarchy. While the hierarchy already contains implicit relationships between product groups, the domain expert may add additional relationships among product groups.

Nobody has probably ever used the term “ontology” for a product catalog structure. However, that’s what it basically is.

What would a typical case for designing an ontology be?

The scope is information interoperability. A typical case would be the analysis of business concepts. For example, among many others, the business party concept certainly makes a first-rate candidate for an ontology. Business parties are the people and communities that are of interest to the organization for any reason. An individual business party may either be an individual or a community. A community is a household or an organization. An organization is a legal entity, while a household is not.

A business analyst would design an ontology that represents these terms and concepts. Often different terms are used to designate business party. For example, the term "business partner" may be an additional, established term within an organization. An ontology can associate different names with an information entity. This can be done at the concept level (e.g. business party) and at the attribute level (e.g. shipping date, could be identified as ShippingDate or ShipDate). Hence, the business analyst can express that different terms are equivalent, which is very important regarding interoperability.

The business analyst can define constraints in an ontology. For example, the ontology can answer questions, such as "Can multiple people have the same social security number?". This is an extremely important and powerful capability, which helps keep a knowledge base consistent.

A knowledge base comes into existence by populating the ontology with data. For example, the business analyst can create an instance for General Motors, which is an organization, and create an instance for John Bloggs, who is an individual. Early prototyping helps with uncovering incoherencies and inconsistencies in the ontology. A business analyst would use an Ontology and Knowledge Base Editor for designing the ontology and populating the knowledge base.

What is the business case for defining and using ontologies?

The need to better align IT with business has been obvious for many years. Shrinking business cycles demand faster time to market of business applications. Interfaces and protocols change a lot, while process and service descriptions remain fairly stable over time.

In stark contrast to general software engineering practice, business analysts rather than software engineers take on the task of ontology design and maintenance. The ontology-based approach is about shifting power to the business experts, thus representing a business-centric approach. In IT terms, the ontology-based approach strictly supports shifting the focus from implementation to design.

The business analyst becomes an information architect. Usually, several business analysts would be involved in ontology design and maintenance. Hence, an ontology retains the collective memory of business analysts and a shared understanding of core business concepts. The terms used in an ontology should be close to natural language, thus making review easy for domain experts.

Many organizations have decided to outsource software development. Ontologies can be used as an effective means to describe requirements and business concepts, which helps

mitigate the risk of project failure. All too often, financial resources have been wasted due to imprecise or ambiguous requirements. It usually takes considerable time before problems rise to the surface. As a long established project management rule states, failures in the early stages of a project have the highest impact in terms of cost. However, when contractors can rely on precise and semantically rich descriptions of business concepts, the risk of failure is dramatically reduced.

Ontologies and knowledge bases can be the single source for the generation of business processes in product-specific representation for deployment. In addition, software design artifacts, such as UML class models, and even source code can be generated directly from ontologies, thus effectively speeding up the software development process. Ultimately, ontologies enhance flexibility and agility.

Most importantly, ontologies help business analysts and IT experts speak the same language. They have a common understanding of the meaning of concepts and terms, thus mitigating the risk of misinterpretation, which plagues so many organizations.

Is an ontology technical?

No. A business analyst needs no previous knowledge about software engineering methods and techniques.

For example, the Protégé Ontology and Knowledge Base Editor, developed by Stanford Medical Informatics at the Stanford University School of Medicine was originally developed to help medical doctors and researchers design ontologies in areas such as clinical research.

Why has the obvious been disregarded for such a long time?

There are three major reasons:

- For historical reasons, business and IT have been kept separate since the birth of business computing. Tasks related to software design were and still are considered the domain of software engineers.
- There were no languages and easy-to-use tools to help business analysts design ontologies.
- Lack of awareness. Ontologies are well known in the domain of science, but not in the business domain.

Still, most IT experts are not aware of ontology concepts. Clearly, the Semantic Web Initiative helps spark interest in ontologies.

What are the benefits from a business perspective?

Business information (content), and business processes and applications, which use that content, need to be easily integratable so that new business ecosystems can be created in a short time span. This high-level architecture enables metadata driven development, deployment, execution and analysis of business processes, applications and content.

Ontologies provide a clear separation of the „what“ from the „how”, meaning that business analysts can focus on the business aspects, completely ignoring how declarative knowledge translates into some kind of software system. Business analysts can describe concepts in terms that stakeholders understand.

The semantics of the enterprise is fairly stable, while technology advances at a rapid pace. Hence, it makes sense to describe the business semantics and generate various artifacts, such as organization manuals and business process descriptions, as well as software artifacts from a single source. In that regard, ontologies play a major role in quality management.

Using the knowledge base, an ontology designer can easily perform prototyping and verify and validate the ontology, effectively contributing to risk minimization. Prototyping would usually happen long before IT is involved.

What are the benefits from the IT perspective?

Organizations are generally pursuing the goal to isolate their software systems from rapid technology changes. Many organizations have already designed a vendor-agnostic software architecture and, in addition, are following a technology-agnostic strategy, allowing them to keep software design platform-independent. The OMG is promoting the Model-Driven-Architecture (MDA) approach, which is not really new, but meets the needs of most IT organizations.

Ontologies are fully in line with the goals of the MDA approach. They are inherently technology and vendor-agnostic. UML models can be generated from ontologies and UML models can be imported in ontologies.

An ontology describes structure, not behavior. However, an ontology can be instantiated to form a knowledge base, which allows an ontology designer to indirectly express behavior through the contents of the knowledge base.

The ontology concept fits in very well with the Service-Oriented Architecture (SOA) approach, which proposes loosely-coupled and metadata-centered services.

A key problem in achieving interoperability is to be able to recognize that two pieces of data are talking about the same thing, even though different terminology is being used. Bridging the terminology gap is important. Ontologies provide that kind of “terminology interoperability”.

What’s the cost of creating and maintaining an ontology?

Obviously, designing an ontology requires time and effort. Multiple review cycles may be required until an ontology can be declared the baseline for further activities that rely on it. However, there are significant savings later on in the software development process, since the knowledge needs to be acquired and expressed in some formal manner at any rate. In essence, the result is a shift of costs between organization units. What used to be included in an IT project budget would now be part of the enterprise’s organization department’s budget.

Nobody would dispute the necessity to acquire knowledge about a problem domain before software is produced. Moreover, everybody would agree that it's necessary to get requirements right early on in order to mitigate project risk. As many reports indicate (for example, see Standish Group's CHAOS report), costs for eliminating an error are the higher the later it is detected. Since the elimination of design errors cost much more than the elimination of coding errors, making every effort to detect design errors as early as possible makes perfect sense.

What Return on Investment (ROI) can I expect?

The effective and intelligent use of ontologies and knowledge bases will contribute to accomplishing significant productivity gains particularly in the early stages of the software development process. In fact, the effective use of ontologies and knowledge bases results in a significant productivity advancement through the simplification of the software development value chain. Although no experience is available from successfully finished large-scale projects, productivity gains in the 20-30% range all over the software development process are a well-founded expectation.

Will ontology design result in the duplication of work?

No. However, tasks will need reassignment.

In many organizations, software designers create UML models, which represent structure and behavior. One of the most important tasks is the design of the class hierarchy and the relationships among classes. Since there are many conceptual similarities with the design of an ontology, and an ontology is semantically richer than a UML class model, it makes sense to put the emphasis on ontology design. UML models can be generated from an ontology.

In essence, the workload of the business analyst will increase, while the workload of software designers will decrease. The combined workload will decrease, however, since multiple work results needed in the business domain as well as in the IT domain can be generated from the ontology, which forms an information backbone for business and IT.

Duplication of work is what can be seen in most organizations right now. Ontologies help reduce duplication of work.

How long will it take to develop an ontology?

Developing an ontology may take anything from hours to weeks or even months. While there are ontologies that comprise only a handful of concepts, others may become very large. For example, ontologies with more than 100,000 concepts exist in the medical diagnosis domain.

Since most organizations have been using IT systems for decades, developing an ontology has much of a consolidation effort. In a bottom-up approach, a business analyst would analyze existing application systems in order to distill concepts that can subsequently be consolidated. The top-down approach, which is much easier and saves time, represents the clean slate approach. A business analyst would create a taxonomy

first, which conceptually represents a class hierarchy in the object-oriented world. Relationships among concepts are added in the next step.

An ontology design needs to undergo thorough review, of course, before it can be declared the baseline for future software development. Concepts implemented in existing application systems are then mapped to the baseline ontology. Mappings can be removed as existing application systems are phased out.

How about the risks of an ontology-based approach?

The risk of an ontology-based approach is fairly minimal, due to the following reasons:

- The W3C Web-Ontology Language standard (OWL) has already gained wide industry support, which further reduces the danger of falling into the trap of tool vendor dependency.
- It is always possible to transform ontologies into UML models, although transformation may incur loss of information.
- As a kind of last resort, ontologies can be stored in XML format. Therefore, there is no danger of losing valuable information due to a proprietary binary data format, which some vendor may resist to disclose.

Of course, there is always the risk of bad ontology design. As with object-oriented modelling, ontology designers need to "know what they are doing". However, since a knowledge base can be built almost in parallel with ontology design, using a knowledge base editor, ontology designers have a chance to detect design errors very early. This is clearly a distinct advantage over object-oriented design, where design errors tend to surface much later, since testing requires an executable application.

Due to a non-proprietary data format, information stored in an ontology can be transformed into some other representation. In a worst case scenario, transformation scripts would need to be written, which is a fairly minimal effort.

Are there general-purpose ontologies on the horizon?

Early efforts are under way aiming at the design of universal ontologies. For example, the Semantic World's Semantic Business Model project has set as its goal to provide authoritative high quality models of common aspects of business and specific vertical industries. Once such universal ontologies are available, business analysts can adapt existing ontologies in order to provide mappings to existing concepts in the organization. As a result, the software development process can be simplified and streamlined at the same time.

Can I use UML for ontology design?

Yes, this is possible. A software engineer can design an ontology by organizing object classes in a class hierarchy and creating relationships among classes.

However, UML has some limitations which stem from its origin. For example, UML doesn't support the concept of conceptual equivalence, allowing a designer to designate "business party" and "business partner" as equivalent concepts.

In addition, UML has insufficient means for describing restrictions and constraints. Software designers can use the Object Constraint Language (OCL), which is difficult to understand by business analysts. In all, an ontology language provides richer semantics.

What efforts are under way in the OMG camp?

The Object Management Group (OMG) has issued a Request for Proposal for an Ontology Definition Metamodel in March 2003, whose scope is not web-centric. The request is for a MOF metamodel for ontology development with a mapping to UML and a binding to OWL, which will provide for translation from UML to OWL, and for using UML tools in ontology definition (i.e. for a UML presentation syntax for OWL). Initial submissions are due in August 2003.

How is an UML-oriented approach different?

Ontologies and UML class diagramming addresses different roles. Ontologies would be designed by business analysts, while UML class diagrams are generally too hard to understand for business experts. UML class diagramming is the domain of the software designer.

Software designers can, of course, use object-oriented design techniques to express concepts in terms of classes and relationships among classes. The Unified Modelling Language (UML), which has become more popular over the past years thanks to its graphical notation, helps software designers visualize concepts. However, UML class diagrams are all but easy to understand for business experts without IT knowledge.

UML has its roots in software engineering and was designed to provide software engineers with an expressive modeling language for the specification, construction, visualization and documentation of the artifacts of a software system. UML has evolved bottom-up from object-oriented programming concepts.

In contrast, ontologies are not for software engineers in the first place. The focus is more on the formal expression of business concepts.

Are there any ontology standards?

Yes.

The Web Ontology Language (OWL) specification is part of the stack of W3C recommendations related to the Semantic Web.

- XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.
- XML Schema is a language for restricting the structure of XML documents.

- RDF is a data model for objects ("resources") and relations between them, provides a simple semantics for this data model, and these data models can be represented in a XML syntax.

- RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.

- OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

In all, OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine readable content. One of the goals for OWL is to provide semantic data for agents in the Internet (e.g. shopping agents).

OWL is currently (May 2003) at the W3C Candidate Recommendation stage. It should have reached Recommendation status by summer 2003.

How about tools for ontology design and management?

Today, more than 50 tools are available from non-commercial organizations and commercial software vendors that provide ontology editing capabilities. One of the more popular tools is Protégé, developed by the Stanford University School of Medicine. Protégé is an ontology and knowledge base editor with a history going back to the early 1990s. Other tools are SemTalk, OntoEdit, and Unicorn, to name just a few.

Not all of these tools support OWL at this time. However, most software providers have already expressed their intent to support OWL. For example, Protégé will support OWL by Q3 2003.

When will ontology-driven software generation reach mainstream?

Today, ontologies have not yet made their entry into the realm of software development. Only few seasoned software designers seem to even know about ontologies, even though the concept has become more popular in recent years in the context of the semantic web initiative. Most software designers are deeply rooted in object-oriented analysis and design methods. However, in contrast to the software-centric approach, ontologies stand for the business-centric approach, supporting the process of shifting power to the business experts.

It will probably take time for the ontology-driven approach to capture mindshare. Only very few of the big software vendors seem to support relevant initiatives, such as the W3C Web-Ontology Working Group. Moreover, efforts to promote ontologies to a wider audience are virtually non-existent. Today, apart from academic institutions, consultants seem to be the major promoters. Hence, public awareness will probably grow slowly, at least until a handful of project success stories can be read in popular magazines.

Who is supposed to design and maintain ontologies?

At the minimum, there are two roles involved: the business analyst and the software engineer. The business analyst would design an ontology and express business concepts, while the software engineer would augment an ontology to represent bindings to the technical implementation of software. However, business is still strictly separate from technology.

The idea is about representing knowledge so that machine agents (e.g. software generators, business process engines) can understand them. Business analyst and software engineer work together to make machine-processable semantics a reality.

What skills are required for ontology design?

A business analyst acts as a liaison between business people and technology people. The main responsibility is to gather, detail and document requirements in a format that is useful to business domain experts as well as IT experts.

The skill sets that a business analyst must possess include among others: analytical skills, organizational skills, problem solving skills, and communication skills. Hence, a business analyst requires skills that exceeds the skill set.

When would one start with ontology design?

Apart from project management activities, ontology design would be the very first activity in a new development project. However, a business analyst would first check, if an ontology is already available. If yes, the next action would be to check, whether the existing ontology can be enhanced. Only if that is not the case, the business analyst would start with designing a new ontology. At any rate, starting with ontology design as early as possible pays off tremendously.

What is the difference between the tool-centric and the ontology-centric approach?

From the very beginning, software development has suffered from disparate tool repositories. There is a plethora of tools, each one implementing its own, proprietary information model. The effects have been noticeable in many ways and have seriously hampered software productivity improvement.

While Integrated Development Environment suites (IDE suites), such as Eclipse and NetBeans, are based on comprehensive repository information models, compared to the coherent whole, they are still confined to cover a more or less small segment of the entire software development process. CASE tools are generally based on proprietary repository information models, too. The list could go on and on. As a consequence, the nonexistence of a global repository information model results in gross duplication of work and information loss.

Ontologies provide a solution to the issues briefly characterized above. In fact, ontologies emerge as a unifying force, opening up opportunities for the design of a comprehensive

information model, which encompasses all stages of the software lifecycle, from inception to retirement. The result is large scale integration at the semantic level.

To help semantic unification advance, organizations are well advised to think about pursuing an ontology-centric approach. In essence, the ontology-centric approach helps shift focus from a function-oriented, tool-centric view, towards a semantics-oriented, ontology-centric view. The table below identifies major characteristics of both approaches.

Characteristic	Tool-Centric Approach	Ontology-Centric Approach
Semantics	Implicit semantics	Explicit labeling
Basic approach	Person uses a tool's functions. Underlying information model is transparent.	Information model is visible. The object type determines the tool used for creation and manipulation of data values.
Extensibility	Generally limited, tool-dependent	unlimited
Repository Information Model	Generally not disclosed, disclosure is at the vendor's discretion	Defined by the user organization
Information exchange among tools	Generally difficult and error-prone, due to differing semantics	Information exchange not necessary among ontologies
Audience	Persons who want to create or manipulate objects of a type that the tool supports.	unlimited

How about integration with the software development process?

An ontology would be created very early in the software development process. It would allow for the definition of process flows, business objects, service implementations (i.e. applications), and non-functional requirements, to name just the most important concepts.

By employing generators, various artifacts can be generated, which can then be imported into a tool or an execution environment. For example, process flows generated for a specific process engine can be deployed in the process engine, UML models can be imported into a CASE tool, and generated source code can be imported into an Integrated Development Environment tool (IDE). Using ontologies and knowledge bases will not make existing tools obsolete.

In general, only minor changes are necessary to the existing software development process. These changes generally only address the exchange of information between ontologies and other tools.

How about compliance with the Model-Driven Architecture approach?

Ontology-driven software artifact generation and the OMG's Model-driven Architecture (MDA) are complementary approaches and fit together perfectly well. MDA is rooted in the object-oriented domain and focuses on UML and application development rather than business process aspects. In comparison, the strength of the ontology-driven approach lies in its technology-agnostic impartiality.

Systems built using both the ontology-driven approach and the MDA approach exhibit more flexibility and agility in times of ongoing technological change. In addition, due to a more formal and accurate specification of requirements and design right from the beginning, quality and robustness of the software artifacts produced will be higher as well.

What is the relationship between a business process ontology and a business process diagram?

A business process ontology defines the concepts that constitute a business process and the relationships among them. As such, the business process ontology defines structure. A business analyst would instantiate the ontology by creating process related definitions, such as business activities, business events, and the process flow, effectively populating the knowledge base.

A business process diagram is the visualization of business process definitions in the knowledge base, which conceptually represents a repository. In comparison, a process modeling tool would read business process definitions from the tool repository, where business process definitions are typically stored in proprietary format and graphically present the business process definition to the user.

While the meta-model underlying a process modeling tool repository is under the sole control of the respective software vendor, a business process ontology is under the sole control of the enterprise and is vendor-agnostic.

What is the relationship between a business process ontology and business rules?

Business rules represent an organization's codified policies and decision-making practices. Although business rules play a pivotal role in every enterprise, they have played a stepchild role for many years. Very often, business rules are more or less disregarded during requirements gathering and their implementation is left to software engineers.

Business rules are declarative statements, i.e. a business analyst expresses *what* should be done, not *how* it should be done. Business rules use nouns that must be well defined if they are to be consistent. Since, as a matter of fact, business analysts and domain experts know best about business rules, it makes perfect sense to capture business rules as early as possible.

As a simple example, a business rule may assert that "The driver has a valid driver's license". Provided that the "driver" and "driver's license" concepts are already defined in

the ontology, it is fairly easy to define a business rule that relates to existing concepts. Business analysts and software engineers both have a common understanding of the semantics.

Business rules make a very strong case for the use of ontologies, since context information is in a single place and not spread over multiple tool repositories.

Are there examples of companies already designing ontologies?

The ontology concept is not new but is applied implicitly by so many people every day without them being aware of it. However, only very few of the big software vendors seem to actively promote ontologies to a wider audience. As a consequence of non-existent to low-key marketing, it will probably take time for the ontology-driven approach to capture mindshare.

That is not to say that large enterprises are unaware of the potential of the ontology-based approach. There are a number of large organizations that have already developed ontologies with remarkable success. However, in general, public awareness will probably grow slowly, at least until a handful of project success stories can be read in popular magazines.

Is it an evolution or a revolution?

Ontology-based software development is emerging as a natural evolution of existing technologies. Clearly, this is not a revolution from a technical viewpoint. However, it is a revolution considering the change in roles. No longer is it the software engineers that assume the most critical role in a software project, but the business analysts.

