

Transitioning to ontologies in Mondeca
Methodology & practical issues
Bernard Vatant bernard.vatant@mondeca.com

Paper issued of ICT TAO project www.tao-project.eu

Common pitfalls, aka “what does not work”

Since 2001, Mondeca has built ontologies for dozens of customers and projects, and to do so has developed a pragmatic approach and identified a certain number of bottlenecks. Since it's easier to identify what does not work than finding the killer methodology, let's start by listing a **certain number of things that *do not work***.

1. **Build ontologies “from scratch”** out of “tacit knowledge” of domain experts, using ontology editors. Domain experts have generally no or poor expertise in knowledge representation tools and languages, and the learning curve is too steep to expect them to get such expertise quickly in a project life-time. Supposing domain experts have the needed expertise, working this way would be costly and most of the time wasting their precious time, and in most cases likely to miss the target, if the task is not supported by a previous audit of the legacy of data, documents and schemas, and a specification of requirements for the target system.
2. **Re-use complex domain ontologies** built for the domain in similar projects. The more complex an ontology, the more tied it is to its original context of development and use, and the less likely it is to fit another context. It's often as difficult and costly to trim such ontologies in order to keep only the relevant parts than to re-build those parts completely.
3. **Build domain ontologies “top-down”** as extensions of “foundational ontologies”. Foundational ontologies are often highly abstract and constraining, and almost never adapted to business requirements. They bear strong constraints that are rarely part of the requirements for the system to build.
4. **Build ontologies without knowing the technical and functional requirements** of the system that will implement them: meta-model, components, architecture, interfaces, queries, automatic feeding and publication etc.
5. **Build ontologies without considering the data** legacy which will be used to populate the knowledge base(s) of the target system. At some point the explicit semantics of the target system will have to match the implicit semantics embedded in the data structure of the source. Otherwise migration of such data will be impossible.

Generally speaking, *what does not work is to build ontologies without a complete specification of the target system they will be part of*. Having in mind just a static knowledge representation is not enough. Ontologies are developed to be parts of a target information system, and have to fit smoothly with all components of this system, even more so as they are backbone components, upon which all other components will rely.

General guidelines, aka “best practices for efficient transition”

The following points are what Mondeca considers as “best practices” to avoid the pitfalls mentioned above.

1. Design ontologies as engineering systems

Ontologies are knowledge representation artefacts built to be integrated in, and generally control, an information system, and have to be considered as the result of an engineering activity. As such, they must follow the general guidelines of any engineering activity, namely specification of need and requirements, iterative evaluation of the relevancy of a solution against requirements, integration with other components and general system architecture, etc. In short, quoting Tom Gruber, *It Is What It Does*¹

Typical requirements on what ontologies can “do” include

- ✓ Migrate data from legacy sources
- ✓ Control integrity of data against a set of constraints
- ✓ Support sophisticated user queries
- ✓ Federate data as description of business objects
- ✓ Federate applications through common definitions of business objects

2. Build ontologies as part of a *transition process*

The engineering tasks leading to the building and integration of ontologies can most of the time be considered as part of a transition process from a *legacy system* (document corpus, data, data schemes, vocabulary ...) to a *target system* which will be ontology driven. The ontology is the backbone of the target system, and part of it has to be extracted, inferred, or otherwise migrated from the legacy system(s). In general, no explicit ontology is defined in the legacy system, but some implicit or latent ontology is present in terms of *data structures*, and will be identified by a careful audit of the legacy to migrate: existing data bases (schema if available, content, available export formats), document corpus to index, classify, or mine, terminology, controlled vocabulary, entity lists.

Another part of the ontology will be defined from the target system requirements, but there is of course no way to extract this part from the legacy.

3. Know the target system technical constraints

The ontology will be implemented in a software environment, of which technical characteristics have to be known before developing the ontology. The technical architecture and meta-model used in the target system (e.g., Mondeca ITM meta-model) will put specific technical constraints on the ontology ‘species’ and allowed constructs. If reasoning facilities are expected in the target system, the ontology constructs will have to be limited to those constructs supported by the inference tools.

4. Specify the target system functional requirements

The added value of the target system against legacy system is generally defined as a set of extra functional requirements. Those requirements have to be specified clearly and the ability of the ontology to meet them evaluated from a qualitative and quantitative viewpoint (performance). In particular, type and number of queries likely to be performed against knowledge bases, user interfaces expected in read and publication mode, are to be assessed whenever a modelling choice is open. In any case, the “good” modelling decision should not be the one which “represents the best” the “domain reality”, but the one which meets the functional requirements with the best performance.

¹ <http://tomgruber.org/writing/cidoc-ontology-2003.pdf>

5. Put the business objects at the core of the ontology

The ontology backbone taxonomy is built around *core business object types*. Those types are generally the ones known by all system users, from business experts to end users. They can be identified by several methods, such as the objects most frequently queried, those which appear as primary keys in data bases, as main taxonomy categories, terms in controlled vocabulary etc. Those objects will define the “core classes”, along with their main attributes. The core classes are not necessarily the “upper classes” of the ontology, but the most likely to be instantiated and the most often queried in the target system. Those core classes and attributes are generally not many, since they represent the business core. Whatever the way to extract them, their very definition is almost always the opportunity for domain experts and system users to go through a “conceptual audit” of their core business objects, going through clarification of semantics and business logic, and disambiguation of terminology. In this task the role of the knowledge engineer is critical. She must push towards and facilitate the conceptual audit, but keep agnostic on its results. The knowledge engineer is not the domain expert. Stabilization of this core ontology, and its assessment against data migration and functional requirements should be the objective of a prototype system.

As necessary, core classes will be further extended by more generic (abstract) classes, generally in order to federate attributes. The generic classes defined this way are technical, and do not necessarily appear in the end user experience.

6. Manage both business terminology and logic, but don't confuse them

The business logic which is formalized in the ontology is considered as distinct from, but not independent from, the terminology used to represent the concepts. The distinction is not always easy to grasp by knowledge experts, for whom the logic is strongly embedded in terminology. One task of the knowledge engineer, and particularly at the beginning of the process, is help making this distinction clear, show how the same business logic can be represented by different users using different terms, or the other way round, discover ambiguity of terms hiding several distinct business logic.

7. Consider data throughout the transition process

In an iterative way, and as often as possible, the capacity of the data to be represented using the ontology has to be assessed on samples of legacy data. This task has to be conducted on real data, using the workflow that will be used in production. It has to take into account the data sources and original format, their extraction and transformation in the best ad hoc format (tabulated text, CSV, XML, RDF ...), import into the target system with integrity checking, test suite of queries against the resulting knowledge base.

The data migration is often a bottleneck in the process. Legacy data actually are rarely what their administrators think they are, or would like them to be. Assessing integrity of data samples against the ontology and finding inconsistencies can lead either to refine or correct bugs in the ontology, or to help legacy administrators to clean or improve their data in order to make them fit for migration. In most projects, both adjustments are needed.