

It is High Time for Pursuing the Ontology-Centric Approach

As enterprises invest in the design of vendor-agnostic software architectures, they effectively pursue a strategy aimed at keeping control of their IT landscape. Being exposed to conflicting software vendor strategies is not a desirable prospect. Hence, making dispositions in good time to achieve maximum independence from unforeseen vendor strategy changes is just appropriate.

In a similar fashion, enterprises are well advised to pursue a strategy that allows them to share and reuse data in a vendor-neutral and technology-independent manner, both in application development and in all flavors of content management, data warehousing and business intelligence. Traditionally, application development and data management related to Commercial Off-the-Shelf (COTS) packages have been two separate activities within an organization. Due to misalignment of strategies, organizations miss opportunities for synergy. Other consequences are redundant skills throughout the organization and duplication of effort.

Clearly, metadata and models need to be used intelligently together. 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.

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. Two of the most serious issues are:

- Different project teams use different terminology to describe the same things. Results are definition overlaps, homonyms (e.g. “bill”) and synonyms (e.g. “bill”, “invoice”). There is no coherent semantics.
- Design artifacts, such as business process definitions and business object definitions, cannot be exchanged among tool repositories due to non-existing or incompatible data interchange features. Having to reenter definitions in some other tool manually is a recurring experience in many organizations.

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.

In the 1990s, software development has gone through a sea change. Object-orientation emerged as a new paradigm in software development, capitalizing to a great extent on its

inherent potential for reuse. One of the great ideas was to design object types only once and reuse them multiple times.

Although organizations have been generally successful in making the transition to object-oriented development, purpose-specific development tools have often been a limiting factor. During the past decades, the tool space has been controlled by software vendors. Most software development tools are still only partly based on principles of object-orientation, although they can be used to produce object-oriented software.

What user organizations demand, though, is a semantically rich and comprehensive business information model, which encompasses metadata about the organization, business process definitions, business object definitions, business document definitions, and provides support for the entire software lifecycle.

The business information model can be defined using an ontology. Concepts (i.e. classes) are defined and related to each other, much like classes in a UML class diagram. Each slot (i.e. attribute) can be associated with a control (slot widget) that allows a person to enter and manipulate data values. For example, an ontology designer could associate a business process definition editor control with the slot that is designated to hold the static process flow. Hence, based on principles of object-orientation, there would be an association between object type and a tool that can be used to create or modify an object of that type. For example, the Protégé Ontology and Knowledge Base Editor contains the Graph Widget, which is useful for visualizing networks of instances and relationships of instances. As such, the Graph Widget can be used to visualize static process flows, i.e. business process definitions.

In the early 1990s, IBM pioneered the “object-action” paradigm in user interfaces. In first generation user interfaces, the user used to start an application and then select the object to be created, viewed or manipulated. For example, the user invoked a word processor application and then selected a file (“action-object” paradigm). Second generation user interfaces introduced the “object-action” paradigm, meaning that the user would select a file, and, in the next step would select the application used to manipulate the object. Hence, the “object-action” approach represents a 1:n approach, allowing to associate an object type with multiple applications. In most practical situations, there would be a 1:1 relationship, however.

In a similar fashion, the introduction of the “object-action” paradigm needs to gain a foothold in the software development tool space as well. In other terms, the sweeping success of object-orientation needs to repeat itself in that space. However, to realize this paradigm, each object type must be unambiguously defined. So, it is far from easy to reach consensus over the semantics of object types. As a practical example, in business process modeling, there is the notion of “activity”. A brief analysis of how this term is understood and used by industry bodies, standards organizations, tool vendors and business analysts, reveals that there is no common denominator. An “activity” may be defined as a process or as a part of a process, which are completely different things.

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

Today, although people in IT seem to speak the same language, they do not understand each other well. Moreover, we need to learn a language that both business and IT people can speak. In some fields, there has been more progress than in others. For example, in accounting, business experts and IT experts do understand each other. In contrast, in the business process space, much needs to be done to eliminate the "semantic gap". Therefore, it is high time for pursuing the ontology-centric approach.