



# **Strategic White Paper**

## **Simplifying the Software Development Value Chain**

Through Ontology-driven Software Artifact Generation

**By Dieter E. Jenz, President, Jenz & Partner GmbH**

**First Edition June, 2003**

Jenz & Partner GmbH  
Hainstr. 40a  
63526 Erlensee  
Germany  
Phone : ++49-(0)6183-9100-0  
Email: [info@jenzundpartner.de](mailto:info@jenzundpartner.de)  
Internet: <http://www.jenzundpartner.de>

Jenz & Partner was founded in 1985. Dieter E. Jenz serves as the company's president. The company provides a range of industry analyst, business and technical consulting, and educational services. It is widely known for its contributions to distributed applications, object-oriented development, and relational database technology. For additional information, Jenz & Partners' Website URL is [www.jenzundpartner.de](http://www.jenzundpartner.de).

Jenz & Partner has developed a business process ontology, which can be made available to clients in the context of consulting engagements. Jenz & Partner provides training in ontology definition and in software development process optimization in general.

This White Paper focuses on human-oriented business process (workflow) aspects, which primarily execute within the organization boundaries, and are thus usually referred to as private business processes. It leaves so-called public business processes aside, which support the collaboration of business partners across enterprise boundaries.

The Protégé-2000 ontology and knowledge base editor has been used for the definition of the Jenz & Partner business process ontology. Protégé-2000 was developed by Stanford Medical Informatics at the Stanford University School of Medicine with support from various agencies.

Although this information is believed to be accurate at the time of publication, Jenz & Partner GmbH cannot and does not warrant the accuracy, completeness or suitability of this information or that the information is correct. Jenz & Partner GmbH and/or Jenz & Partner GmbH analysts and consultants cannot be held liable for any damages caused or alleged to be caused directly or indirectly by decisions made using any Jenz & Partner GmbH material. This publication and features described herein are subject to change without notice.

Names appearing in this document that are registered trademarks are not mentioned as being so, nor is the trademark symbol inserted with each mention of these registered trademarks. Product names and company names mentioned in this publication are the property of their respective owners. In no way does Jenz & Partner GmbH have the intention of infringing on any registered trademark.

Copyright © 2003 by Jenz & Partner GmbH. All rights reserved. No part of this publication may be reproduced by any method whatsoever, without the prior written consent of Jenz & Partner GmbH.

## Table of Contents

Executive Summary .....	1
Getting requirements right – still an issue .....	2
Putting the business analyst in the driver’s seat .....	4
The case for an integrated approach .....	5
Ontologies in the realm of business process design .....	7
Process semantics .....	8
Business activity semantics.....	8
Business object semantics .....	11
Business document semantics .....	12
Role semantics .....	12
Business rule semantics.....	12
Creating various artifacts from an ontology .....	13
What are the real benefits? .....	18
Conclusion and outlook .....	18

### **Executive Summary**

As of today, there is still no such thing as a comprehensive and continuous software production process. The software development value chain still has missing links, resulting in a significant number of failed projects. Recently introduced approaches, such as Agile Software Process Management and Extreme Programming are all but partial solutions.

The most significant issue still is getting requirements right and then transforming them - without information loss - into a semantically rich specification, from which various types of software artifacts can be derived. Viewed from an organizational perspective, the issue is how to make non-IT domain experts and IT experts understand each other by speaking the same language.

Although various tools support requirements gathering and formal definition, a semantic gap still exists. All too often, the emphasis is on the definition of functions rather than business processes. However, during the past years, companies have begun to revise their software development processes to the effect that business process definition plays a much more important role. This triggers the need for a coherent approach, which ensures high quality business process definitions.

Since quality is the result of fulfilling stakeholder requirements, expressing requirements in a consistent and machine-processable fashion helps eliminate many of today's weak points in the software development process. Once business process semantics are machine-processable, various software design artifacts, as well as code, can be generated from business process definitions.

Ontologies, explicit formal specifications of terms and relationships among them, play a pivotal role as enabling method for the description of semantically rich, machine-processable business process definitions. Representing business process definitions and related artifacts (e.g. business documents, business objects, etc.) as knowledge, based on ontologies, is a novel approach. This paper discusses a practical approach, which requires almost no up-front investment.

## Getting requirements right – still an issue

Business process modelling is playing an increasingly important role in many software projects. Typically, a software project would start out with business process analysis by interviewing domain experts, followed by business process design, performed by persons with a solid IT background. Tool suites, such as ARIS, Proforma and MEGA Suite, which enable business process modellers to graphically define, simulate and optimize business processes, are seen more often now.

The practical value of such tool suites cannot be disputed, since they effectively contribute to risk minimization, allowing the early detection of inconsistencies and bottlenecks through simulation. However, tools vary widely in their capabilities to allow for the creation of semantically rich business process models.

Business process definitions implicitly reflect functional requirements. However, it is not adequate to just design the context of process control flow and business activities connected through the control flow. To represent the full set of requirements, a process definition needs to specify entities that participate in the process explicit. For example, a business activity is performed by a role and interacts with business documents and business objects. Business activities (e.g. “Create Order”) and business objects (e.g. “Order”) implement business functionality.

The business process designer faces a dilemma. There is no single tool on the market, which is built on a comprehensive meta model to express process-related functional requirements, let alone non-functional requirements, such as “performance” and “reliability”. On the other hand, the project risk is highest during the business modelling phase. If the requirements are not properly understood, chances are that missing functionality remains unnoticed until very late in the development process. Hence, every approach that allows to detect errors as early as possible helps to minimize risks.

Since semantically rich business process definitions represent functional requirements, a formal requirements specification can restrict itself to a description of general items, such as project drivers, project constraints, and non-functional requirements (e.g. based on ISO/IEC 9126). The ontology can reference external documents, of course.

Process modelling tool suites generally suffer from weak integration with traditional CASE tools. In a worst case scenario, software designers need to manually re-enter business object definitions in CASE tools. As a consequence, round-trip engineering encompassing business process modelling and object design is rendered an elusive dream.

Many of today’s purpose-built requirements engineering tools provide for traceability of requirements through integration with CASE tools. Traceability provides a way to relate requirements expressed in a requirements specification with software artifacts, helping with the verification and validation of a system. In addition, traceability provides an effective means for assessing the impact of change. However, given the heterogeneous nature of the software tools used in development, providing traceability is far from being an easy task.

When requirements and software design artifacts have to be manually kept in synchronization, there is an almost 100% probability that synchronization will invariably be lost at some point in time. Software tools provide very little help in this respect. For example, in practice, business process models and object-oriented design artifacts, such as UML class diagrams, usually get out of synchronization very quickly.

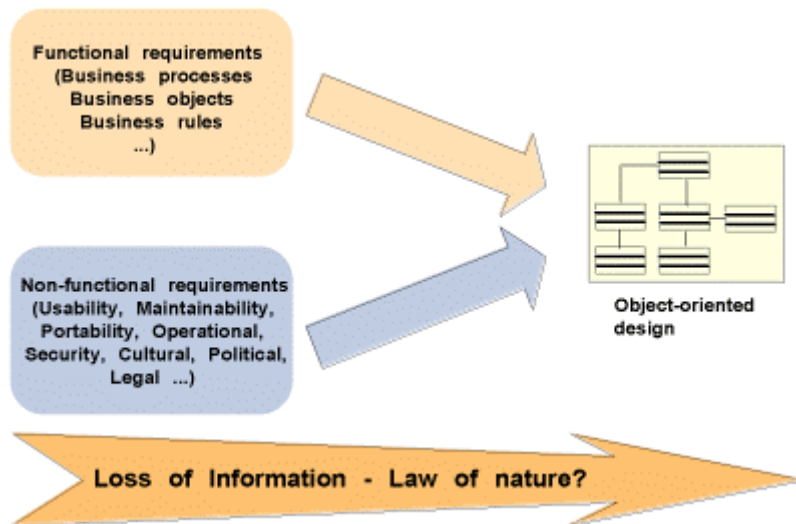


Figure 1: Loss of information increases project risk

The main general reason for the loss of synchronization is the complete misalignment between the requirements meta model and the object-oriented meta model, which forms the foundation for the vast majority of today's software development projects. While functional requirements typically depict user interfaces, algorithms, data objects, etc., the object-oriented meta model centers around the concept of classes and their relationships, where classes represent properties and behavior.

Practitioners have long observed a gaping open space: 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. However, this comes at a cost: there is an information rift and neither business people nor software engineers have a coherent and consistent view on project requirements. It is no wonder that so many projects either fail or require significant efforts to remedy early design errors.

As a consequence, a two-pronged strategy needs to be implemented:

- Assign the task of requirements specification to business analysts,
- Employ an integrated approach for making information machine-processable.

### Putting the business analyst in the driver's seat

Significant productivity gains have been reached in code generation during past years. Development managers report increased developer productivity, reduced programming skills requirements, simplification and streamlining of the development process, and reduced testing requirements. Many standard software development activities, such as deployment script development, can be fully automated. In general, between 40% and 100% of the code can be generated thanks to the effective use of modern CASE tools.

In stark contrast, obtaining requirements and deriving the software design, is still a laborious and error-prone task in most organizations and generally results in the loss of valuable information. The general lack of precision and meaning of available information kept in multiple places makes it all but easy to interpret the data. As a logical consequence, information should be kept in one central place. In order to significantly speed up the software development process, it must be possible to express domain knowledge in machine-processable terms.

The task of translating a project's goal and vision into a set of actionable requirements is best assigned to a business analyst. A business analyst has both a profound understanding of the business side as well as the IT side. Since business people and IT people generally have different views of the problem domain, both sides are in constant danger of grossly misunderstanding each other. To remedy the situation, business and IT must be able to speak the same language and share a common understanding of the business vocabulary and grammar. Business analysts will take on the challenging task of defining the semantics.

As a consequence, the business analyst assumes the most critical role in a software project and thus in the software development value chain. It is the business analyst's responsibility to define requirements in a way that software design artifacts can be derived using a straightforward and repeatable approach. As such, actual software design results in pattern-based refinement.



*Figure 2: The business analyst as the conduit between stakeholders and software development team*

In the traditional role, the business analyst focuses too much on establishing a requirements baseline and then shifts attention towards the management of the requirements specification and verification of the fulfilment of the requirements. The business analyst of the future will play a much more integrative role and will be key in the venture to simplify the software development value chain.



An ontology establishes a bridge between the world of natural language and the world of machines. Viewed from a different perspective, an ontology creates a much needed bridge between domain experts and IT experts, making it possible to keep related information in a single place.

A knowledge base is the result of instantiating an ontology. There are three major reasons for creating a knowledge base:

- Definition of a glossary of terms, which serves as a central repository for domain experts and software engineers alike.
- Definition of requirements-based test cases. A business analyst can develop requirements-based test cases, which help validate the ontology early on, hence mitigating project risk. Of course, requirements-based test cases can also be used later on for acceptance testing.

Clearly, requirements-based tests are only as good as the requirements. A requirement can be completely incorrect and still be testable. Regular review meetings with domain experts help to identify any incorrect requirements. In addition, review meetings help detect requirements that have been overlooked.

- Definition of specifications for subsequent generation of various artifacts. Purpose-specific generators interpret the machine-processable specification and produce specific output, such as UML class models and business process definitions.

In comparison with current software design techniques, the ontology-based approach stands out as an integrated approach. There is no need for using a bunch of purpose-specific tools, such as requirements management tool, CASE tool, etc. There is one central repository that all persons playing a role in the software development process can access, which can be based on a meta model that suits the needs of the enterprise, and from which various artifacts can be generated.

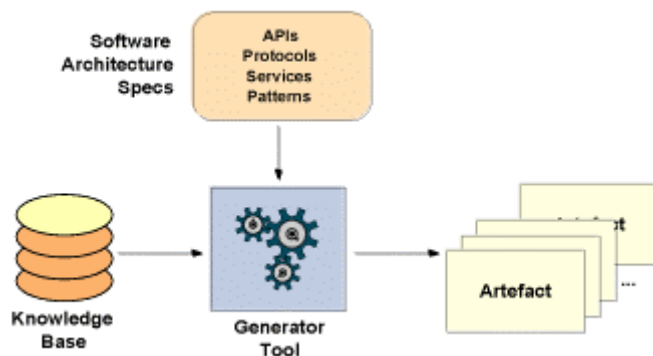


Figure 4: Generating artifacts from a knowledge base

## Ontologies in the realm of business process design

Business process design is at the interface between domain experts and IT experts. As such, an ontology would be an adequate and suitable means for information sharing in the same “language”.

A business process ontology serves two distinct purposes. Firstly, it makes knowledge explicit and allows for knowledge sharing among domain experts and IT people engaged in software design and development. Secondly, since it contains machine-readable definitions of concepts, it serves as a requirements specification from which a number of software artifacts can be generated.

Two prerequisites must be met before a business process ontology can be created. The first step provides for the definition of a meta-model, which defines concepts and relationships among them. In practical terms, this task is very similar to creating an object-oriented class diagram. Concepts are represented by classes, whereas each class may have several properties to describe the various features and attributes of a concept. The second step focuses on the creation of instances from concepts (i.e. classes), which results in a knowledge base.

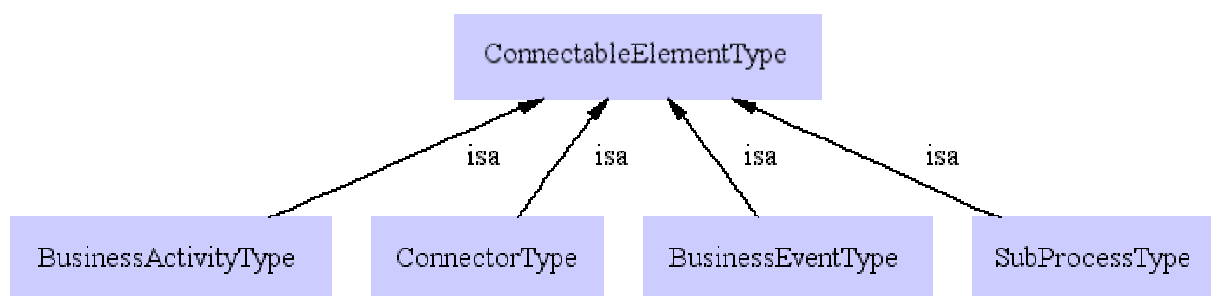


Figure 5: High-level excerpt from the business process ontology

The meta-model of the business process ontology can be defined in a way that suits the domain expert, the requirements engineer and the business process modeller, who may not be an IT expert. An ontology is under full control of the people who have the right to access it, meaning that it can be changed and extended as required. In contrast, most CASE tools are very restrictive regarding modifications of the meta model.

A business process ontology would describe all concepts related with a business process. In particular, it would define entity types such as business activity, business document, business object, business event, business rule, role, resource and control flow. In addition, relationships among entities are defined. Synonyms and homonyms can be defined in order to help eliminate the “terminology gap” between domain experts and IT experts.

The real value of a business process ontology lies in the ability to extend the software development value chain. By adding instances to the knowledge base, a business process designer can easily perform prototyping and verify and validate the meta-model, effectively contributing to risk minimization. The knowledge base can also be subjected to reviews by domain experts, long before any other artifacts are created.

Today, there are already numerous tools that support the definition of ontologies. 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.

### Process semantics

By popular definition, a business process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc., and contributes to the achievement or realization of a defined business objective.

A static representation of a business process can be constructed from activities as core building blocks. Activities need to be identified and then arranged in the sequence in which they are performed. The following figure shows a high-level abstraction of a business process expressed with some artificial and simplified syntax. The business process consists of activities (A) that are connected with each other. The arrows indicate that activities are performed in a sequence, representing the static flow. A decision point indicates that there is an alternate flow. Which route is taken, is determined at run time. The process also has a defined start point and an end point.

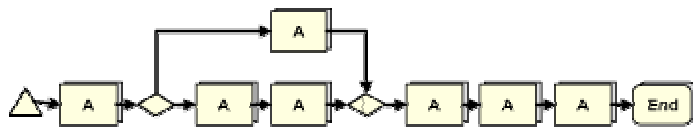


Figure 6: Abstract representation of a business process

The business process ontology defines which elements are connectable through arcs and also defines valid relationships. For example, there are four basic connectable element types regarding control flow: “Business Activity”, “Business Event”, “Connector”, and “Sub Process”.

Experience has shown that a semantically rich description of business activities is key to improving productivity in the software development value chain. The business process ontology can be custom-designed to meet this requirement.

### Business activity semantics

There is no single universally agreed upon definition as to what constitutes a business activity. Some business analysts tend to define coarse-grained business activities, while others focus on defining fine-grained activities. Generally, the concrete Business Process Management System (or Workflow Management System) influences the granularity of business activities to a large extent.

However, experience has shown that it is advantageous to view a business activity as an atomic unit of work, which is performed by a single role (e.g. sales clerk), whereas the role can represent a human or a system. Hence, a business activity (task) defines a definite piece of work that is performed by one role at one place at one time. It forms one logical step within the business process. A business activity is implemented by a service (i.e. an application).

Every business activity executes in a defined context, which includes the following items:

- Role: A logical abstraction of one or more physical actors, usually in terms of common responsibility or position. An actor may be a member of one or more roles.
- Business document: The set of information components that are interchanged as part of a business activity.
- Business object: A representation of a thing active in the business domain. As such, it reflects the reification of some abstraction that is important in the business

domain, and that implements meaningful, computational behaviors. A business object is, by definition, independent of any single application.

- Resource: A real object that can be identified.

A well-defined business process ontology enables the business analyst to describe business activities with their full context. A graphical tool can produce a visual representation of a context as shown in the figure below.

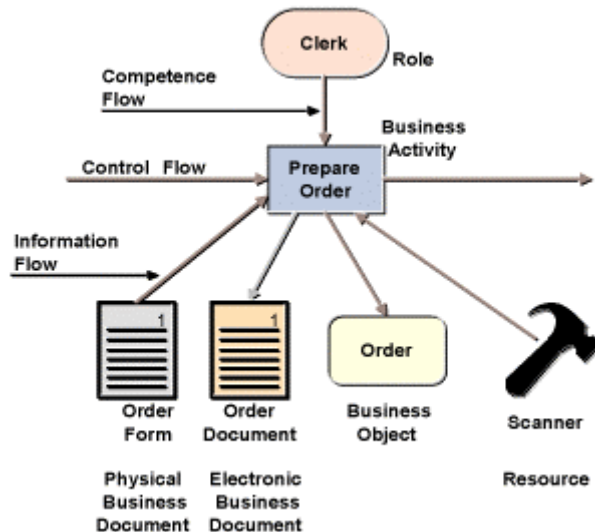


Figure 7: Semantically rich description of a business activity (Document flow not shown)

To describe the execution context of a business activity in a semantically rich fashion, it has proven helpful to use different arc types to associate business activities with roles, business documents, business objects, and resources. The meta-model distinguishes two different arc types:

- Competence flow: A business activity is performed by a role, which in turn has one or more competencies.
- Information flow: Describes the relationship between a business activity and a business document or between a business activity and a business object. A business activity may read, update, or create a business document or business object.

The document flow, which complements the other three arc types (control flow, competence flow, information flow) connects business documents with each other, thus making the document flow among business activities visible.

There are basically three types of business activities:

- manual business activity: activities of this type do not rely on the support of an IT system (e.g. placing goods in storage);
- semi-automatic business activity: this kind of activity is performed by a human with the support of an IT system. It is typically implemented by an application that exposes a graphical user interface;
- automatic business activity: humans are not involved in the execution of automatic business activities.

Every semi-automatic or automatic business activity is associated with an application (a service), which implements the activity. Since many business activities are fairly fine-grained, it may often make sense to design a service so that it implements multiple business activities.

In an ontology, the business activity concept is represented by a class, which, in concept, is very similar to an object-oriented class. A class has one or more attributes, called slots. Using the knowledge base editor, a business analyst can instantiate classes and create instances, which helps with the early validation and verification of the ontology. In fact, the business analyst can do early prototyping, which helps to find incoherencies and inconsistencies long before a software designer gets involved.

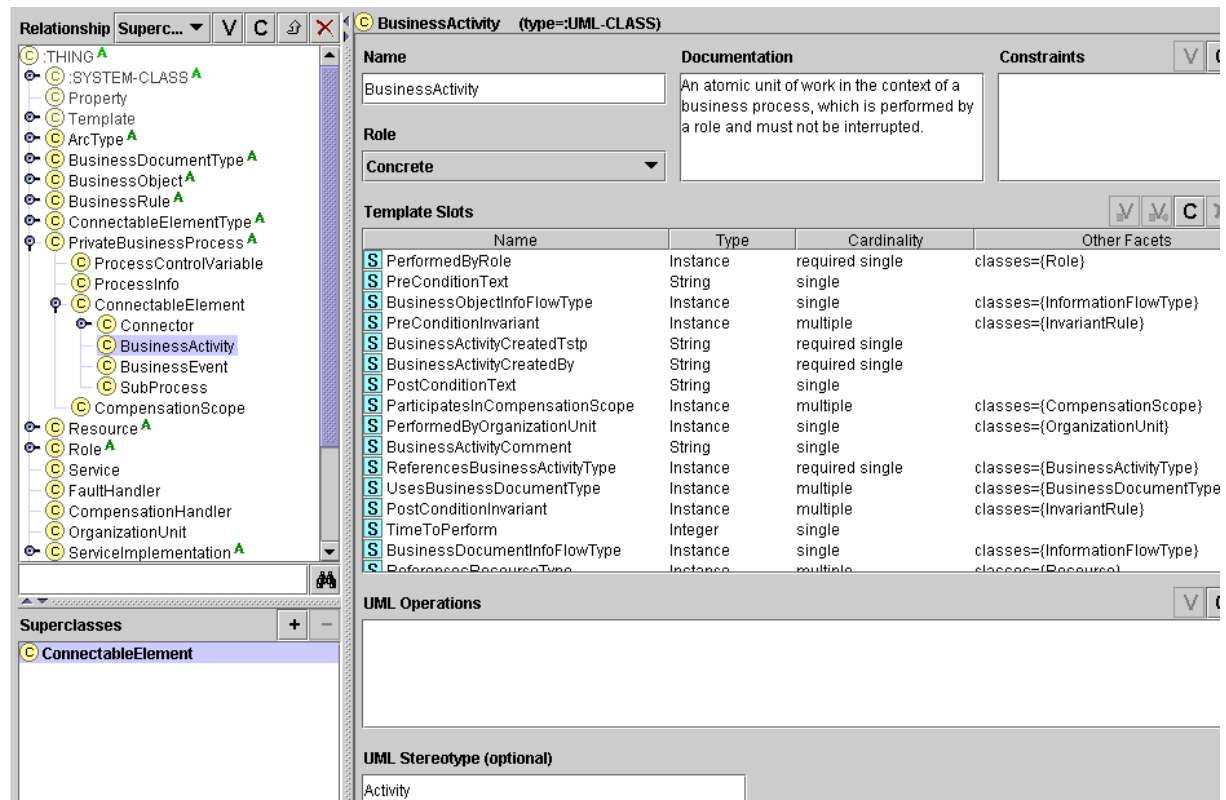


Figure 8: Business activity definition in Protégé

Ontology design is at the discretion of the business analyst. It is advantageous, however, to describe business activities much like UML use cases. Hence, the definition encompasses pre- and post-conditions, constraints, the normal flow of events, exception conditions, etc. In addition, it contains a mandatory reference to a role and optional references to business objects, business documents and resources (see figure 6). It makes competence and information flows (optional) explicit.

The ontology can be instantiated, thus creating a knowledge base. Using the knowledge base editor, the business analyst can describe business activities.

<b>BusinessActivityCreatedBy</b> V C + - <input type="text" value="Adams"/>	<b>ReferencesBusinessActivityType</b> V C + - <input type="text" value="Evaluate Mortgage Requester's Financial Status"/>
<b>PreConditionText</b> <input type="text" value="Electronic Mortgage Request form was checked and is valid"/>	<b>PreConditionInvariant</b> V C + - <input type="text" value=""/>
<b>PostConditionText</b> <input type="text" value="Evaluation result is either 'green' (application approved), 'yellow' (to be decided upon by supervisor) or 'red' (application rejected)"/>	<b>PostConditionInvariant</b> V C + - <input type="text" value="FinancialStatusRating='green' OR FinancialStatusRating='yellow' OR Fin"/>
<b>NormalFlowOfEvents</b> <input type="text" value="The user checks the mortgage requester's financial status by resorting to internal records, such as the applicant's loan history, and by resorting to external data from business partners. Internal and external data are made available by the system and are displayed on the user's screen together with the amount of the requested loan and the system's rating of the mortgage requester's financial status. The user either accepts the system's rating proposal or overwrites it if there are other sources of knowledge."/>	
<b>ExceptionConditions</b> <input type="text" value=""/>	
<b>PerformedByRole</b> V C + - <input type="text" value="Financial Services Clerk"/>	<b>PerformedByOrganizationUnit</b> V C + - <input type="text" value="Private Mortgages"/>

Figure 9: Description of a business activity using the knowledge base editor (clipping)

## Business object semantics

The business analyst usually discovers a whole lot of business objects in the course of business process modelling. Business objects represent entities of the real world, such as “order” and “invoice”.

A business object has a relationship with one or more business activities through information flows. A business activity may either consume a business object (read-only), create a business object (write), or combine both operations (read-write).

Business process modelling expresses the requirements of domain experts and does not concern itself with implementation issues in any way. Business analysts tend to find various business objects, which are to be treated as business object candidates in the first instance.

It helps enormously to identify business objects as early as possible, since it is far easier to detect synonyms and homonyms. Moreover, domain experts can help in the definition of business objects in terms of describing properties and behavior from a business viewpoint. Properties and behavior may be added later during object-oriented design, since each business object needs to be integrated with the software architecture.

Business objects are usually composed of a number of basic and aggregated entities such as “postal address”, called business information entities (BIEs). Since standards bodies have already defined a number of generic BIEs, reusing them to the maximum extent is certainly a reasonable strategy.

Business object definitions in the knowledge base are the source for the generation of class definitions in UML syntax, which can be imported into a CASE tool later on.

### Business document semantics

Business documents and business objects have many characteristics in common, but are not exactly the same. Multiple business documents may be associated with a single business object. In addition, the set of supported operations is limited, since every operation must be related to the manipulation of a business document. For example, the operation “withdraw funds” would be valid for the business object “bank account”, but not for a business document.

In many respects, however, the description of a business document resembles that of a business object. Likewise, class definitions can be generated from business document descriptions.

### Role semantics

Roles specify the actor types (e.g. purchasing officer, sales clerk, system) that participate in business processes and perform business activities. Roles correspond to UML actors.

For maximum flexibility and re-use, business activities are assigned to roles rather than to named users. Using roles instead of assigning a real user’s name makes changes easy to manage. For example, if an employee leaves the organization, business processes do not require modification. In addition, it enables a user to easily delegate and reassign business activities (provided the user is authorized to do so).

Every role has a set of “competencies”, which describe in business terms what the people associated with that role are allowed to do. For example, the role “purchasing officer” may sign orders up to a limit of \$10,000.

Business Process Management Systems use proprietary role definition formats. A generator can create BPMS-specific role definitions that can be imported into the BPMS.

### Business rule semantics

Business rules represent an organization’s codified policies and decision-making practices. 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 department head must approve a loan application if the amount of the loan is greater than or equal \$1,000,000”. Provided that the “loan” and “department head” 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.

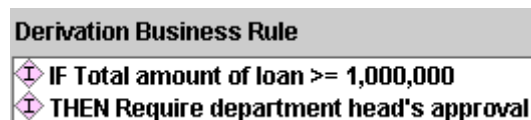


Figure 10: Derivation Business Rule

## Creating various artifacts from an ontology

As can be witnessed very often, software designers start out with designing use cases and class diagrams. UML Activity diagrams are not so popular among designers, mainly due to relatively poor expressiveness. While UML Use case diagrams and descriptions are useful for discussion with domain experts, UML Class diagrams are not. In activity diagrams, there is not necessarily a discernable relationship between use cases and activities.

A well-designed business process ontology will provide a coherent view of a business process. Hence, it forms a basis for the generation of various software artifacts. In essence, the following artifacts could be generated from an ontology:

- BPMS-neutral process definition: Domain experts find it helpful to visualize business processes in a flow-chart format. A business process description can be produced for use in review meetings, which provides comprehensive documentation and makes the semantics explicit.
- BPMS-specific process definition: There are numerous Business Process Management Systems (BPMS) on the market, many of which use proprietary syntax. Provided that the BPMS vendor provides a suitable import feature, product-specific process definitions (i.e. the execution language) can be generated from an ontology.
- UML class diagram: Using the popular XMI interface (XML Metadata Interface), the ontology tool can export various design artifacts, most notably business objects and business documents. The ontology tool evaluates classes and relationships and creates an export file that can be imported into a CASE tool that also implements the XMI interface.

A business process definition with its control flow can be visualized in a knowledge base tool, such as Protégé. A graphical representation facilitates reviews by domain experts and helps them determine whether their requirements are met.

Business managers have been requesting better alignment of IT with business for years. Ontology-based business process design will result in the ability to streamline the software development process. A business analyst would instantiate an ontology, thus creating a knowledge base, which contains comprehensive definitions of business processes and related artifacts.

Software designers augment the knowledge base, providing implementation-oriented information, such as the relationships to service definitions. A service implements a business activity and may already exist. A service definition provides platform-neutral and product-neutral information about the characteristics of a service, i.e. interfaces, properties and behavior. At this stage in the development process, service characteristics are normally not complete and need to be defined in more detail during object-oriented design.

Various artifacts can be generated from the knowledge base. A product-specific transformation script is required to create product-specific artifacts, such as the process execution language. Generic transformation scripts suffice for the generation of design artifacts, such as UML class diagrams.

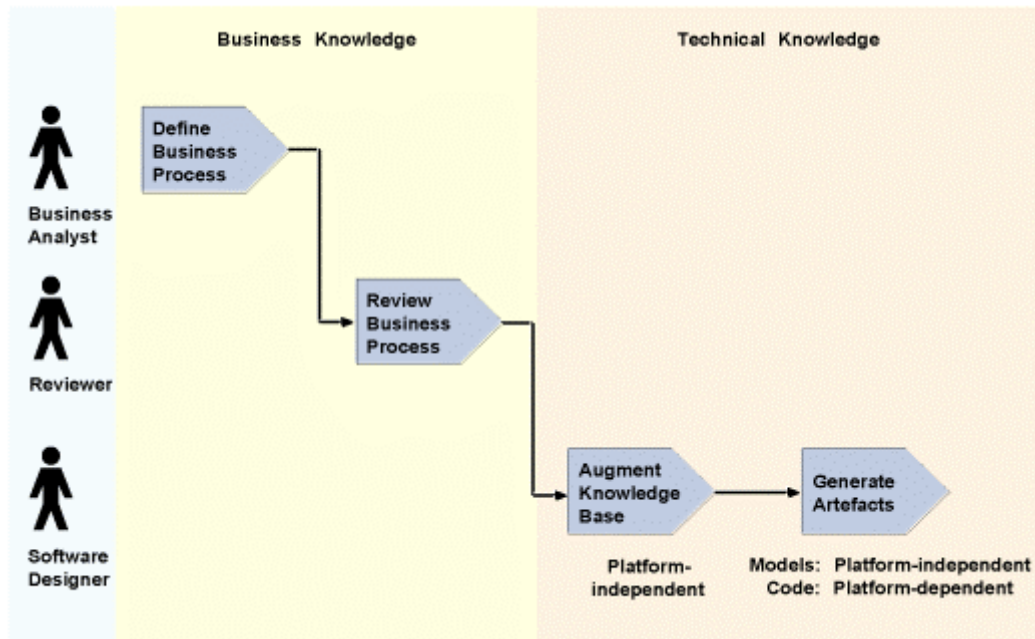


Figure 11: Major tasks in the generation of artifacts from a knowledge base (simplified view)

The business analyst is primarily concerned with defining entities that are related to business processes: business activities, business objects, business documents, business rules and roles. Over time, as the number of entities continues to grow, the business analyst will be able to re-use existing entity definitions.

Once a business process definition has reached a state which allows it to be subjected to a review, a BPMS-neutral representation of the business process definition can be generated for discussion by the review team. The business process is best presented in a graphical view, which makes it easier for domain experts to detect inconsistencies and omissions. An extended version of the vendor-neutral BPMN (Business Process Modelling Notation) would be the preferred notation, since it can be easily understood by non-technical persons.

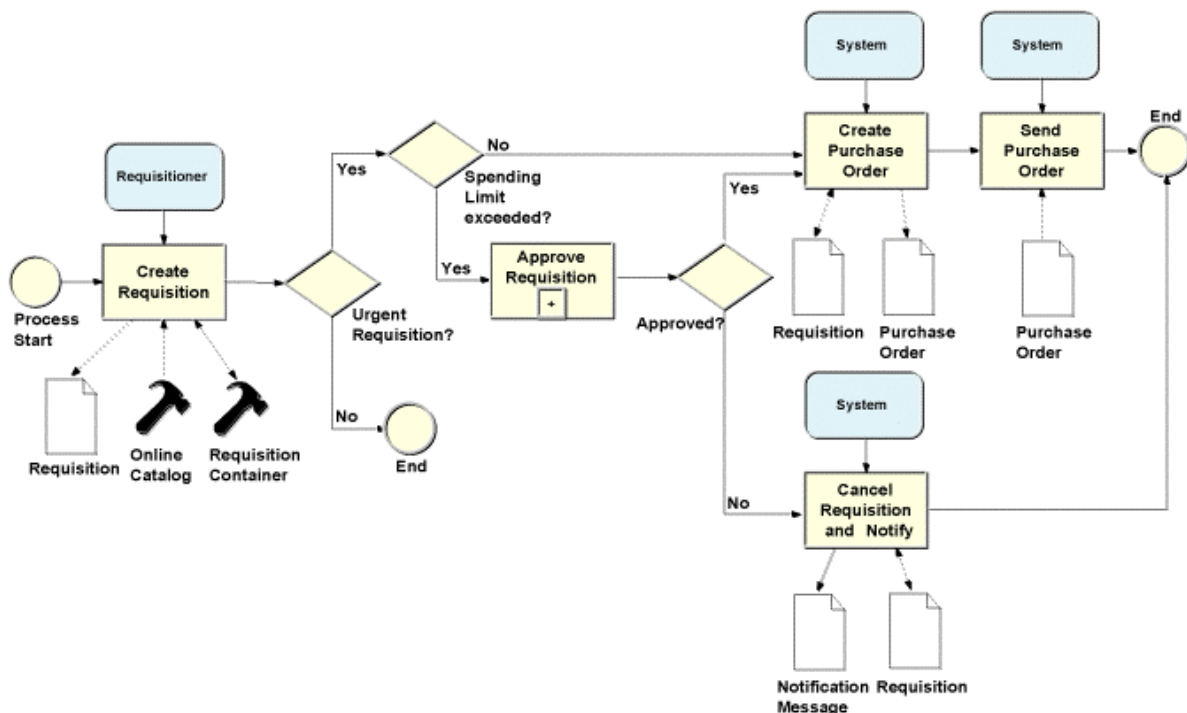


Figure 12: Process definition in extended BPMN notation

In addition, a report exposing activity definitions, entity definitions and constraints will prove helpful in review meetings. For example, a business activity description will expose pre- and post-conditions, business rules, the normal flow of events, and so on. Business activities are described much like use cases.

If the outcome of the review meeting demands changes to existing definitions, the business analyst applies the necessary changes to the knowledge base. If the review team is happy with the business process definition, it can be turned over to the software designers for adding technical information necessary for the generation of executable business process definitions and software design artifacts.

While a business process chart only provides a static view of a process, process execution simulation exhibits dynamic aspects as to how a process actually performs in a simulated environment with defined resources. Process simulation is a very reasonable alternative to experimenting with the actual process in the real world, without having to do time-consuming and costly experiments. When introduced at the beginning, simulation will uncover snags at an early stage, thus saving precious time and resources.

Once again, the business process knowledge base is the source for the generation of process definitions, which are imported either into a purpose-specific simulation tool or into the Business Process Management System (or Workflow Management System).

One of the major tasks of the software designer is to provide a linkage between business aspects and technical aspects. For example, the software designer needs to connect business activities to services, which implement them. Both the business analyst and the software designer work on the same ontology, but the interface is clearly defined. While the business analyst is not concerned with the technical implementation, the software designer is not concerned with business aspects.

Once the software designer has linked business activities to their service implementations and has supplied all the necessary technical details, the stage is set for the generation step. Based on semantics-rich definitions in machine-processable form, artifacts for theoretically any number of target products can be generated from the knowledge base.

The generator tool can basically create two types of artifacts:

- BPMS-specific process definition: Although standards bodies, such as the Workflow Management Coalition (WfMC), have defined standards to support the exchange of business process definitions, software vendors still stick to proprietary or semi-proprietary formats. To make use of generated process definitions, the BPMS must support an interface that allows for the import of process definitions. If this requirement is met, business process definitions can be deployed into the BPMS for subsequent execution.
- UML class definitions: Since properties and behavior of business activities, business objects, business documents, etc., can be defined in the knowledge base, the generator can create class definitions in UML syntax and use the product-neutral XMI Format (XML Metadata Interchange) as output format. Provided that the CASE-tool supports XMI import, the software designer can import class definitions and edit them in the CASE tool.

The figure below illustrates how various artifacts can be generated from a single source. The business process ontology represents the central source of knowledge, while generated artifacts only represent subsets. If changes need to be applied to a UML class diagram after generation, they possibly need to be reflected in the ontology as well. Provided that XMI can be used for data interchange, this task can be at least partially automated.

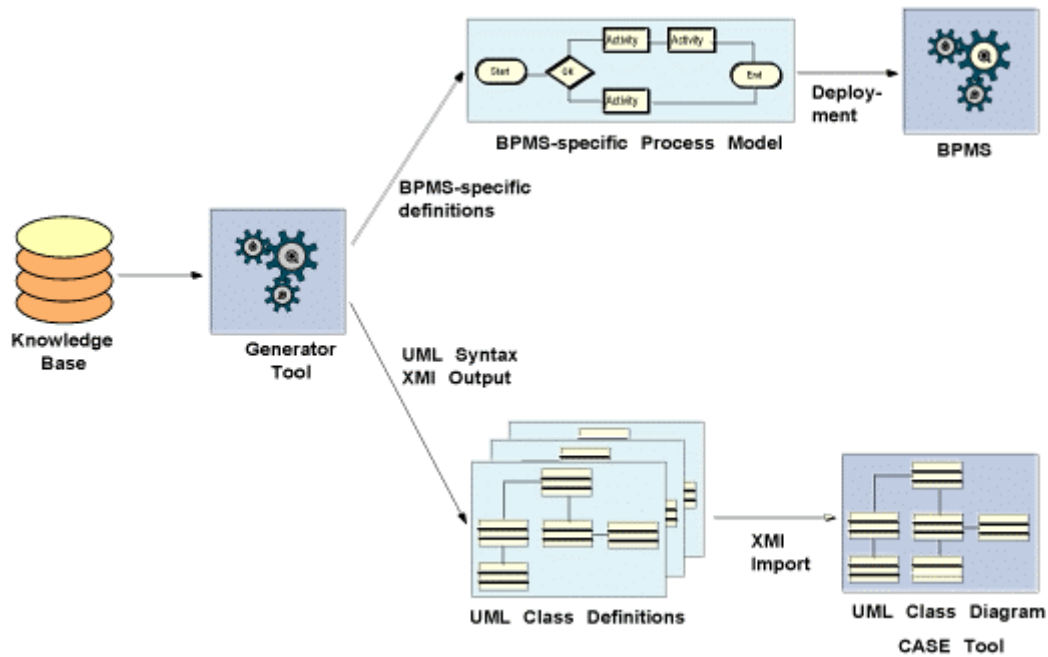


Figure 13: Generating artifacts from the knowledge base

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 is UML-biased. It says little about 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 will be higher as well.

In MDA terminology, the output of the generator represents a platform-independent model. Since it conforms to the XMI specification, it is also CASE tool agnostic, meaning that it can be imported into any CASE tool that supports XMI data import. CASE tool vendor support for XMI is growing slowly but steadily.

After import into a CASE tool, the software designer can work on the UML class diagram and focus on technical aspects. Since stereotypes and tagged values can already be defined in the ontology and knowledge base editor, the software generation value chain effectively includes the knowledge base. Hence it is possible to implement a two-stage generation process: the first generation stage results in the generation of executable process definitions and UML class diagrams, while the results of the second generation stage are implementation-oriented artifacts, such as executable code in a target programming language and deployment descriptors.

It is worth noting that the two-stage generation process fits in well with current software generation practice. Usually, software engineers start out with UML diagramming, which then function as input for code generation. Now, UML models are generated from the knowledge base.

The figure below shows a UML class diagram after import into Poseidon, a popular CASE tool. Please note, that class definitions, associations and stereotypes have been generated from information in the knowledge base. In fact, artifact generation requires no extra effort and reflects “out-of-the-box functionality” of Protégé. UML class definitions are created

implicitly when the UML backend stores a knowledge base in XMI format. Stereotypes and tagged values can be assigned to classes and slots in the knowledge base editor.

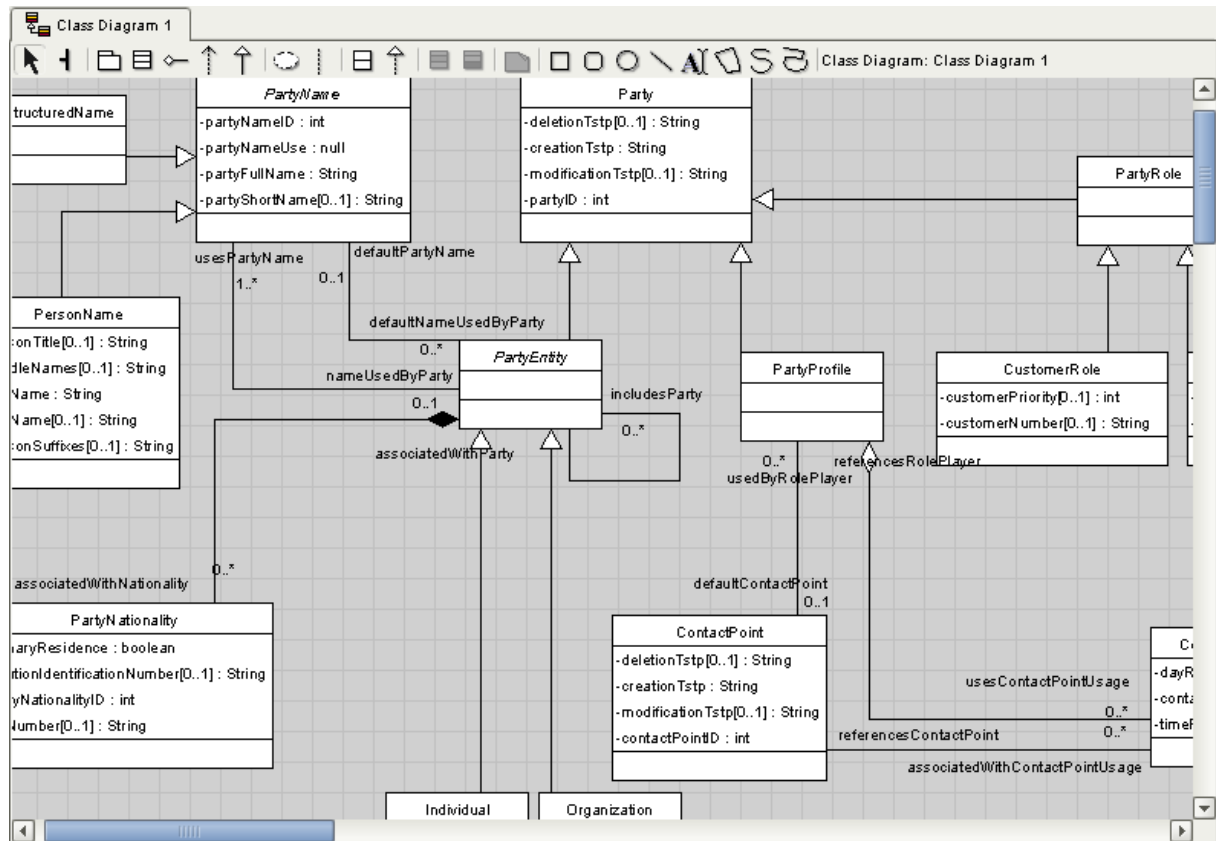


Figure 14: UML Class diagram after import into a CASE tool (clipping)

### What are the real benefits?

An ontology provides an effective and efficient means for the definition of semantically rich business processes and related entity types. One of the striking benefits lies in the fact that an ontology can be fully tailored to the needs of the organization. An ontology can be designed in a way that it allows business analysts to describe functional as well as non-functional requirements in a single place. In addition, an ontology can be easily adapted to meet changing requirements, should the need arise.

Ontologies provide for the much needed “understanding bridge” between business and IT. Ontologies and knowledge bases can be created and maintained by non-IT people, provided that they have received adequate training.

However, there are other key benefits:

- Rapid prototyping helps save cost and mitigate risk through early ontology validation,
- The generator approach is supported through machine-processable semantics,
- Reduced dependency on vendor and tool strategies,

Since knowledge bases can be populated very early in the life of a project, business analysts find themselves in the position to be able to do rapid prototyping. On such basis, the business analyst can easily get a sense of practical usefulness of the ontology long before software designers are involved. As such, early prototyping effectively helps with risk mitigation.

Provided that business processes are well described, various kinds of artifacts can be generated from a single source. The software development value chain is enhanced to the effect that the semantic gap between business process design and object-oriented design no longer exists. Although an organization would need to develop various generators, the benefits still outweigh the cost.

Compared to the OMG’s Model-driven Architecture (MDA) approach, there is no need for an organization to make a commitment to a UML-based and UML-biased methodology. The organization remains in control as to which methodologies it supports and, in addition, minimizes dependencies on software vendors.

Many organizations have learned lessons the hard way in recent years. The software tool landscape is in constant flux, which has resulted in tools falling behind technologically. Over time, many organizations have experienced tools to become more of a burden, since obtaining desired results requires “outwitting” the tool. As a consequence, ontology-driven artifact generation empowers organizations to “feed” specific-purpose tools from a single source, effectively reducing dependencies on tool vendors.

The ontology-driven approach dramatically loosens dependencies on specific software products. While it is a legitimate strategy for software vendors to tie-in users into their software by implementing product-specific features, user organizations are exposed to the threat that the software product may prove inadequate as time progresses.

## Conclusion and outlook

We are currently experiencing the beginning of a silent transition towards a higher level of requirements description. The software development process will reach a higher integration level in that it will be based on machine-processable semantics. Non-IT people, such as business analysts will be able to define ontologies or re-use “off-the-shelf” ontologies, which serve as the basis for the generation of various types of software artifacts. An important step towards the elimination of the “semantic gap” between business and IT will be taken.

Ontologies bring together business process design, object-oriented design and business rules. In consequence of this, the biggest issue at the early stage of the software development value chain can be solved. Moreover, industry-specific ontologies as well as problem-oriented ontologies, such as the Jenz & Partner business process ontology, will be developed based on a standardized language, the Web Ontology Language (OWL). Organizations can use such ontologies as a solid foundation for the population of their own knowledge bases.

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.