OPAALS PROJECT

Contract n° IST-034824

WP6: Socio-Economic Constructivism & Language

Del6.6 – Proof of Concept for Ontology Prototype

Project funded by the European Community under the “Information Society Technology” Programme

Contract Number: IST-034824

Project Acronym: OPAALS

Deliverable N°: D6.6

Due date: 30 November 2007

Delivery Date: 15 December 2007

Short Description: The business ontology prototype will represent a preliminary study to understand how business models can be related to process models, enabling a possible mapping between Semantics of Business Vocabulary and Business Rules (SBVR) and formalism for process representation, such as Business Process Modelling Notation (BPMN)

Author: Dr. Victor Bayon, Nagaraj Konda and NAICA

Partners contributed: BCU and NAICA (Sub-contractor to BCU)

Made available to: OPAALS Consortium and European Commission

VERSIONING		
VERSION	DATE	NAME, ORGANIZATION
0.1	**18/10/2007**	**BCU, NAICA**
0.2	**15/11/2007**	**BCU, NAICA**
0.3	**21/11/2007**	**BCU, NAICA**

Quality check

Internal Reviewers: Jukka Huhtamaki, TUT and Javier Noguera, TI

Dependences:

Work Packages	WP6, WP10 and WP2. This deliverable addresses at a prototype level a number of interesting aspects that are part of the work being done under these work packages. This includes the areas on business and domain vocabulary, workflows and business process automation, and visualisation. The first part of the report introduces extensively the theoretical and technological foundations of business process modelling. The second part describes and analyses the actual proof-of-concept implementation of a workflow, inspired by the underlying technologies. The third and final part presents more high level concepts and processes of theory-based, real-life orientated business models.
Partners	UniK, SUAS, TI, TUT, IITK
Domains	This report covers the social and computing domains.
Targets	Computing domain

Table of contents

Figures and Tables

Glossary

BML - Business Modelling Language
BPEL - Business Process Execution Language
CORBA - Common Object Broker architecture
DBE - Digital Business Ecosystems
DE - Digital Ecosystems
D(B)E - DBE + DE
FADA - Federated Advanced Directory Architecture
MDA - Model Driven Architecture
OCL - Object Constrain Language
OMG - Object Management Group
P2P - Peer to Peer
RSS - "Really Simple Syndication","Rich Site Summary" and also formally known as "RDF Site Summary"
REST - Representational State Transfer
RPC - Remote Procedure Call
S&D - Search and Discovery
SBVR - Semantics of Business Vocabulary and Rules
SI - Service Interface
SM - Service Manifest
SDO - Service Design Overhead
SME - Small Medium Enterprise
SMID - Service Manifest Identification
SP - Service Plasticity
SSL - Semantic Service Language
UDDI - Universal Description Discovery and Integration
WSDL - Web Services Definition Languages
WS-* - Oasis Web Services Stack specifications
XPDL - XML Process Definition Language

1 Introduction

The business ontology/taxonomy prototype aims at defining connections between concepts upon which business models are built and more specific concepts, related to executable processes and the technological components that realise them. This will allow us to relate business activities to business motivations, goals and strategies.

The business ontology prototype will represent a preliminary study to understand how business models can be related to process models, enabling a possible mapping between Semantics of Business Vocabulary and Business Rules (SBVR) and formalism for process representation, such as Business Process Modelling Notation (BPMN)[1].

Altman et al (1999) define ontologies as scientific models helping clear communication between users and enabling structured storage of information to support automated processing. A popular definition by Gruber (1993) states that "ontology is a formal specification of a conceptualization".

Business Models and Business Processes, the two key concepts under scrutiny in this report, are becoming primary focus for researchers from two key domains – Management Science and Computer Science. From a management science perspective, business models and business processes are seen as socially driven processes greatly influenced by the societal and task environments. The businesses being more interested in this domain agree that there is increase in complexity in defining business models its associated concepts from a meta-level and designing the business processes for specific business model instances that are sustainable supporting their growth needs. This is more relevant towards the small and medium businesses as they face more challenges in the environment than their larger counterparts. As part of this report staying within the business domain, we plan to unravel the complexities by identifying and expanding on the concepts involved in defining the business models and also building a vocabulary of terms that are associated with a business organisation. This approach is expected to provide the following benefits:

1. Defines the concepts and expands them into sub-concepts or business terms to create an easy understanding of their meaning from a business perspective.
2. Improves communication (human to human or machine to machine) and sense making by ensuring that the concepts, sub-concepts and vocabularies are common and that there is a common usage within a community or industry domain.
3. Creation of relationships with business processes to help businesses in gaining better focus on change management.

[1] The seven fallacies of Business Process execution available at http://www.infoq.com/articles/seven-fallacies-of-bpm presents an interesting view point on BPM.

In the computer science view, these concepts are seen from an executable perspective where business process and model interactions are to be automated. This report is organised into three parts – Part 1, Part 2 and Part 3. The first two parts take the computer science view while the final part presents the management science view building vocabularies from the business model concepts. The first part of the work introduces extensively the theoretical and technological foundations of business process modelling. The second part describes and analyses the actual proof-of-concept implementation of a workflow, inspired by the underlying technologies. The third and final part presents more high level concepts and processes of theory-based, real-life orientated business models.

The key contributors to this report are mentioned below:

- Part 1 has been prepared by NAICA (sub-contractor to Birmingham City University)
- Part 2 has been prepared by Dr. Victor Bayon, Birmingham City University and
- Part 3 which presents the business model processes and vocabularies has been prepared by Nagaraj Konda, Birmingham City University.

2 Part 1

Contributor –NAICA

3 Rational

The adoption of ICT inside firms represents a key factor to improve efficiency, decrease costs and create new revenues opportunities in the emerging global, inter-networked and knowledge-based competitive environment. Shifting business processes to an electronic dimension allows enterprises to achieve increases in efficiency and reductions of coordination costs. Anyway, this digitalisation very often lacks in delivering the expected benefits. The magnitude of the problem is not easy to be measured. Some studies argue that a percentage higher than 70% of IT projects either fails or falls short of their goals (SGI, 1994). Even though other analysis (Glass, 2005; Jorgensen et al., 2006) demonstrate that these data show a failure rate higher than its real value, the occurrence of problems in the digitalisation of business processes is undeniable. The most damaging implications for IT projects, in fact, is not the number of those that were abandoned, rather it is those that were completed but offer fewer features and functions than originally specified (Glass, 2006).

One of the reasons behind this problem is a recurrent misalignment between the IT perspective and the business perspective, shown by the difficulties to realise software functionalities that exactly reflect business processes. This misalignment is often caused by design methodologies that, being strictly related with the point of view of software developers, do not allow the business people to specify requirements with respect to his own perspective by using a familiar language (Kleppe et al., 2003)

This work comes from these considerations and is intended as a preliminary study to define an approach to business process digitalisation able to start from models of processes that are understandable or directly realised by business people. The main idea is to provide a basic set of concepts and definitions that future research could use to develop a comprehensive framework for (semi)automated conversion of process models into deployable software components. In this perspective, a business process developed by a business analyst could be directly applied to an engine instead of going through human interpretations and translations into other languages (White, 2006). For this reason, it is fundamental to explore how knowledge embedded in enterprise processes and practices, formalised through the related business model, can be exploited to obtain this objective. More in details, it is interesting to understand how process representation can be realised by means of business languages based on formal logic, maintaining the rigorousness necessary to specific engines to run such processes.

For the purpose of this work, a business process can be defined as a set of heterogeneous activities that aim at the same goal and involve different actors, that can be human beings as well as technological instruments.

Process modelling is thus the capturing of an ordered sequence of business activities and supporting information about how a business pursues its objectives. There are different levels

of process modelling (White, 2006): Process Maps (simple representation of the activities), Process Descriptions (representation extended with additional information, but not enough to fully define actual performance), and Process Models (representation extended with enough information so that the process can be analyzed, simulated, and/or executed). At this first stage of the study, it is not important to focus on one specific level; anyway, the main objective on the long term is to enable a straightforward definition of process models in order to enable their automated processing.

4 Business language for process representation

As described in the previous section, the main objective of this study is investigating the relationships between business languages based on formal logic and formalisms aimed at process modelling and representations. The standards chosen for this purpose are the Object Management Group's (OMG) *Semantics of Business Vocabulary and Business Rules* (SBVR) and the *Business Process Modeling Notation* (BPMN), defined by the Business Process Management Initiative and later adopted as OMG standard.
From an operational perspective, this section aims at achieving the following objectives:

- to introduce the reader to the basic principles behind the OMG's language model, SBVR;
- to provide an overview about BPMN and its importance in the context of process representation;
- to outline an SBVR extension based on BPMN concepts, thanks to the definition of a simplified BPMN metamodel as a set of SBVR constructs.

This last output can be considered as the definition of a *business modelling sub-language focused on process representation*, that is based on natural language and structured enough to allow automatic interpretation of business process models. This means that, within the framework outlined by the Model Driven Architecture (MDA) approach, this sub-language may enable a wide range of inference capabilities. Furthermore, it may support a more direct translation of business-oriented models of processes into formalisms that can drive the actual orchestration of software components that execute these processes. Such a bridge, being a first necessary step to enable a direct connection between process models and deployable software components, is one of the most interesting perspectives in both business modelling and software development. For this reasons, this section proposes a discussion about possible relations with another emerging standard for process execution that is Business Process Execution Language (BPEL).

Another interesting application of this study might directly come from the analysis realised in the definition of the *Business Ontology Prototype* that describes the connections among strategies and business processes and practices. Starting from this point, it is possible to explore the possibility to define a straightforward methodology that relates executable processes (and the deployable components that realise them) to business strategic aspects. In other words, it is possible to figure out a scenario where:

- a change in the business model could imply changes in the technological infrastructure, that could be easily discovered thanks to the ontology prototype;

- a reconfiguration in the technological components adopted could be analysed in terms of the consequential changes in the strategic objectives pursued by the firm, and thus in its business model.

This bidirectional study might open interesting perspectives in analysing the interconnection among a business organisation and its infrastructure.

4.1 Semantics of Business Vocabulary and Business Rules

The Semantics of Business Vocabulary and Business Rules (SBVR) aims at creating a standard "to allow business people to define the policies and rules by which they run their business in their own language, in terms of the things they deal with in the business, and to capture those rules in a way that is clear, unambiguous and readily translatable into other representations" (OMG, 2004). To reach these objectives, SBVR aims at enabling:

- business vocabularies construction, whose definitions represent shared understanding among a community of business people and whose contents are univocally identified by this community;
- rules formalization, based on vocabularies; these rules should be expressed in a language close to natural (business) language but, at the same time, representable from an information technology point of view, in order to allow their sharing and transferring.

In other words, business semantics should be defined by business people in a natural language, using shared terms, in order to enable vocabularies and rules exchange among organizations.

Another important characteristic of SBVR is its fact-orientation. Rules build on facts; in other words, facts make assertions about business concepts and rules constrain and support these assertions (BRG, 2003). Some of the main strengths of this orientation are the following:

- people communicate facts, that is the fact is the atomic unit of communication;
- fact is itself a kind of thing and can therefore be the subject of other facts;
- multidimensional categorisation is allowed (each classification is a separate fact) while the most of other approaches encounter many problems to reach this objective;
- time-changeability and purpose-changeability are supported (a new object is created to represent a new fact);
- the fact-oriented approach enables extensibility and reuse (expansion of business vocabularies introduces new concepts or types of facts without any regard to their being more specific or more general than what is already defined).

4.1.1 SBVR in an MDA perpective

Semantics of Business Vocabulary and Business Rules is positioned to be entirely within the business model layer of the OMG's Model Driven Architecture (MDA), as described in Fig. 1. Business vocabularies and related rules can be conceptually considered as Computationally Independent Models (CIMs): they allow business environment representation, avoiding technical details; moreover, they are thought by and for business people (i.e. domain practitioners).

This positioning has two implications.

- SBVR is targeted at business rules and business vocabularies, including those relevant for usage in conjunction with those rules. Other aspects of business models also have to be developed, including business process and organization structure, but these are to be addressed by the OMG in other initiatives.
- Business models, including the models that SBVR supports, describe businesses and not the IT systems that support them.

In MDA, IT systems are specified using Platform Independent Models (PIMs) and Platform-Specific Models (PSMs). Guidance will be needed for transformation of business models to PIMs, even though such guidance is outside the scope of SBVR (OMG, 2007).

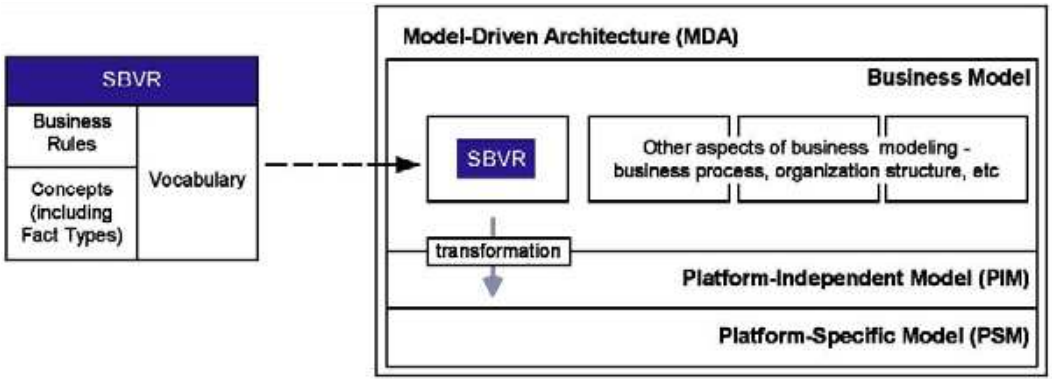


Figure 1. SBVR in an MDA approach

As announced above, the work realised focuses on this positioning to figure out a framework that allows going from CIM models of processes towards process representations that could be easily used to realise technological components. In other words, using SBVR as structured language for process description represents a basic assumption for moving through all the layers conceived in the MDA perspective, in order to derive in a semi-automated way models that are specific for a given platform.

4.1.2 Overview

Fig. 2 illustrates the core concepts upon which the SBVR approach is built (OMG, 2007).

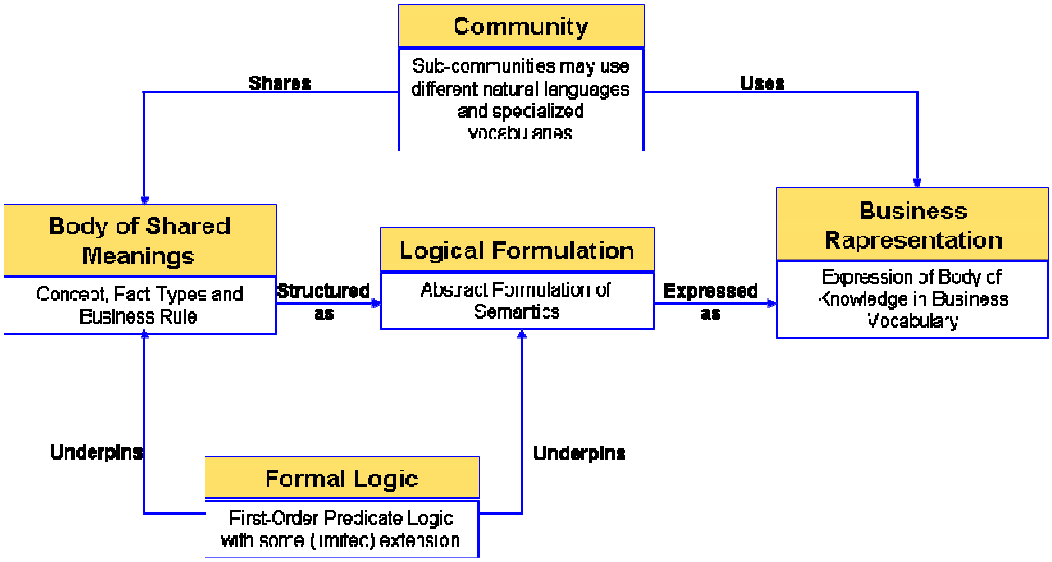


Figure 2. Basic components of SBVR

The basis for business vocabulary is the community. At the business level, communities of primary importance are enterprises for which business rules are being established and expressed. Communities play a central role in capturing business semantics. Each community involved within the business has a different impact in vocabulary construction and rules definition.

A community has a body of shared meanings, comprising concepts (which include fact types) and business rules commonly used and understood by its members. What is shared is the meaning, not the form of expression. This separation of concepts from expressions represents a fundamental characteristic of SBVR.

Logical formulation provides a formal, abstract, language-independent syntax for capturing the semantics of a body of shared meanings. It supports multiple forms of representation, such as: noun and verb fact type forms, reading of associations in both directions. Logical formulation supports two essential features of SBVR, that are the mapping of a body of shared meanings to vocabularies used by communities, and the mapping to XMI that enables interchange of concepts, facts, and business rules between tools that support SBVR.

Business Expression allows to describe shared meanings in a way acceptable and usable by speech communities. SBVR supports mapping of business meaning to concrete language (both natural or artificial) by associating elements of the Body of Shared Meanings with

signifiers. Thus, logical formulations provide the structure and signifiers are placed in logical formulations to provide the expression.
SBVR has a sound theoretical foundation of formal logic, underpinning both logical formulation and the structures of bodies of shared meanings. It uses first-order predicate logic (even if higher-order logic is allowed) and some extensions into modal logic.

4.1.3 MOF/XMI representation

A business vocabulary provides a means of recording and communicating facts. Following OMG's Model Driven Architecture, a business vocabulary developed as an information system independent model of business communication is used to drive the creation of a platform independent Meta Object Facility (MOF) model. The MOF model is, in turn, used to drive generation of Java interfaces (based on Java Metadata Interface i.e. JMI) and an XML schema (based on XML Metadata Interchange i.e. XMI).
The SBVR Metamodel is a MOF-based metamodel that supports a MOF representation of the concepts represented by the SBVR vocabularies. The SBVR Metamodel is available as an XML document. Models of business concepts, business vocabularies and business guidance can be communicated in terms of SBVR using XML documents that conform to an XMI-based XML schema created from the SBVR Metamodel.

4.1.4 Notation

Defining a clear separation between meaning and expression, SBVR does not impose a specific notation for writing vocabularies and rules. Anyway, it proposes a Structured English, that uses some specific styles to assign a precise meaning to the elements embedded in each SBVR expression. The SBVR Structured English is just one of possibly many notations that can be used to express the SBVR Metamodel, and, as a notation, is nonnormative in the SBVR standard (OMG, 2007).
There are four font styles with formal meaning:

term — The 'term' font is used for a designation for a noun concept (other than an individual concept), one that is part of a vocabulary being used or defined. Terms are usually defined using lower case letters unless they include a proper noun. Terms are defined in singular form, but plural forms are implicitly available for use.

Name — The 'name' font is used for a designation of an individual concept. Names tend to be proper nouns (e.g., California). Note that names of numerical values in formal statements are also shown in this style (e.g., 25). capitalization.

verb The 'verb' font is used for designations for fact types, both in the context of showing a fact type form of expression (e.g., 'modal formulation *claims* modality') and in the context of using it in a statement (e.g., "Each modal formulation *claims* exactly one modality.").

keyword The 'keyword' font is used for linguistic symbols used to construct statements – the words that can be combined with other designations to form statements and definitions (e.g., 'each' and 'it is obligatory that'). Different categories of keywords are defined: quantification (e.g., 'each', 'at least'), logical operations (e.g. 'and', 'or', 'not'), modal operations (e.g. 'it is obligatory that', 'it is possible that'), other keywords (e.g. 'the', 'a').

4.2 Business Process Modeling Notation

A process model has to contain different forms of information, such as which are the activities to perform, who is going to do them, when and where will they be done, how and why will they be done, and who is dependent on its being done. Existing approaches to process modelling differ in the extent to which their constructs highlight the information that answers these different questions. The differences result from the various source domains as well as from the application areas targeted (List et al., 2006).

In this work BPMN, a widely adopted standard for business process and workflow representation, is taken into account.

The Business Process Modeling Notation (BPMN) provides a graphical notation to facilitate human communication between business users and technical users, of complex business processes. BPMN is a flow-chart based notation for defining business processes. Such a notation can be used at each of the process modelling levels, to create process maps (i.e. simple flow charts of the activities), process descriptions (i.e. flow charts extended with additional information, but not enough to fully define actual performance), or process models (i.e. flow charts extended with enough information so that the process can be analyzed, simulated, and/or executed) (White, 2006).

The guidelines followed in the BPMN development have been the following:

- acceptability and usability for the business community;
- ability to generate executable processes (e.g., BPEL) through a BPMN Model (a combination of graphical elements and supporting information (attributes));
- applicability to general purpose business problems (although executable processes triggered the development of BPMN);
- guidance for purpose and detail level in modelling given by specific methodologies.

4.2.1 Core set of BPMN elements

The set of core elements described above are represented by means of graphical symbols that constitute the formal notation of BPMN, as shown in Fig. 3. As better described in the following sections, the basic elements defined by the notation are *flow objects* (to describe activities, events and control points), *connectors* (to describe the order of the activities, the exchange of messages, and to link data, information and artifacts to flow objects), *artifacts* (to show information beyond the graphical representation of the process), and *swimlanes* (to partition and organise activities).

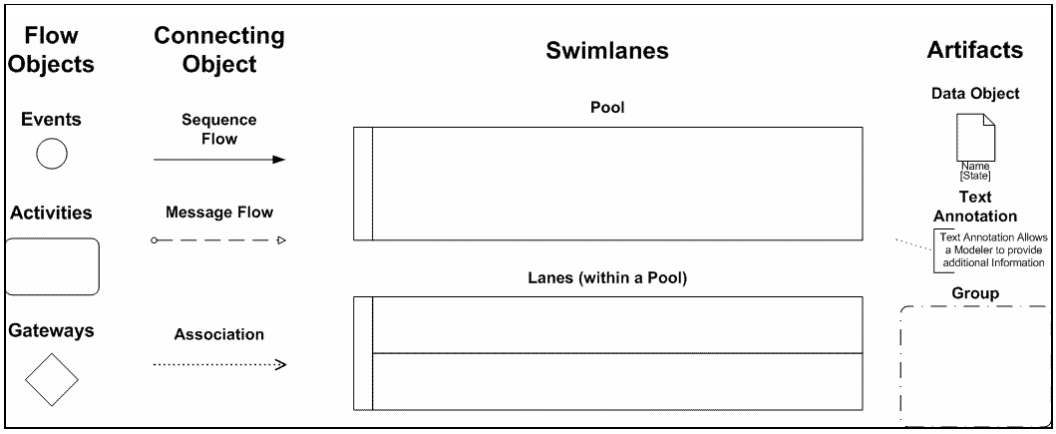


Figure 3. BPMN elements

A sub-set of these elements will be used as the basis for the definition of a simplified process metamodel, that will allow to express simple but complete processes. In the following, such metamodel will be expressed both in a UML Class Diagram like fashion and through an SBVR model (vocabulary + rule set). The proposed sub-set of the BPMN elements includes activities, events, gateways, sequence flows, and lanes.

4.2.2 Flow objects

Flow Objects are the very basic set of shapes belonging to the core of BPMN.

Gateways 	A Gateway is represented by a diamond shape and is used to control the divergence and convergence of the flow of a process. Thus, it will determine traditional decisions, as well as the forking, merging, and joining of paths.

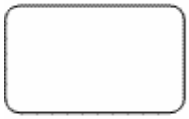
Activities 	An Activity is represented by a rounded-corner rectangle and is a generic term for work that a company performs. In BPMN an Activity can be atomic or non-atomic (compound), but here we consider only the atomic type.
Events 	An Event is represented by a circle and is something that "happens" during the course of a business process. These Events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End (see the figures to the right, respectively). Here we we consider only Start Events and End Events.

Table 1. BPMN Flow Objects

4.2.3 Connecting objects

Flow Objects are connected together in a diagram to create the basic skeletal structure of a business process. Connecting Objects provide this function. The most important element among those belonging to the category of the connecting objects is the Sequence Flow.

Sequence Flow	A Sequence Flow is represented by a solid line with a solid arrowhead and is used to show the order (the sequence) that activities will be performed in a process.

Table 2. BPMN Sequence Flow

4.2.4 Swim lanes

A Swimlane is a mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities.

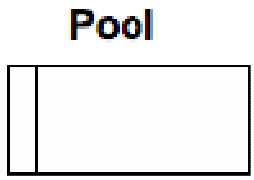
	A Pool represents a participant in a process. It is also acts as a graphical container for partitioning a set of activities from other Pools.

Table 3. BPMN Swim Lanes

4.3 Simplified Metamodel for Process Representation

Starting from the essential sub-set of the BPMN elements that has been introduced and described above, a simplified metamodel for process representation will be defined. Such simplification includes all the elements that are essential in order to represent a simple but complete example process (see section 4.3.2). Advanced BPMN elements are not in the demonstrative purpose of this work and will be analysed in future works. In particular an UML Class Diagram like representation of such metamodel is provided in Fig. 4.

The figure is self-explanatory since it is composed by simple rectangles that represent the main entities of a process and by arrows that represent relationships between them. Each arrow has a label that expresses the name of the relationship. Note that all the relationships are binary (between exactly two entities) except for one that is ternary. Such relationship binds an xor_gateway output flow to a flow object depending on the value of its condition (condition value).

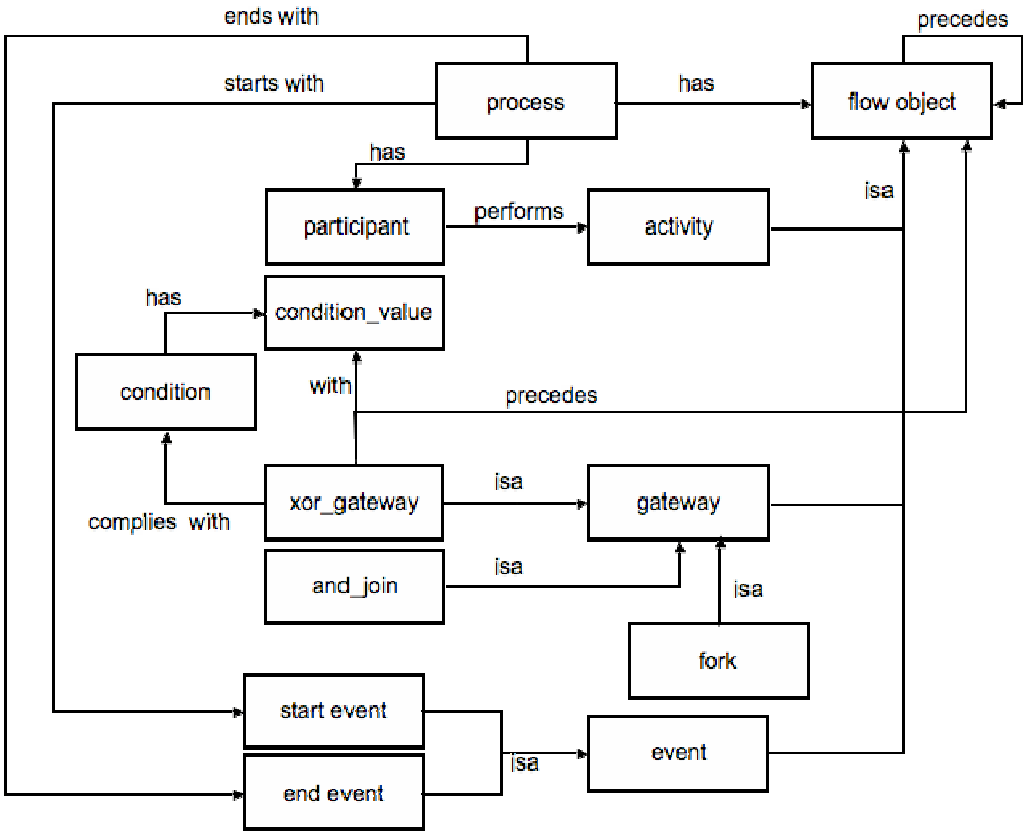


Figure 4. Simplified Process Metamodel

4.3.1 Business Process Vocabulary and Rules

This section provides an SBVR model (vocabulary + rules) that represents the same simplified process metamodel of Fig. 4. Each entity (rectangle) of the metamodel is modelled through a term of the vocabulary, while relationships between entities are modelled through fact type forms. Finally restrictions and constraints are reported as rules.

Activity	Definition: an activity is a generic term that refers to the work that a company or an organization performs via a process.
	General_concept: flow_object
Condition	Definition: a condition is a statement that can have multiple condition_values depending on the status of the runtime environment. The actual condition_value determines the alternative path

	chosen for the flow in an xor_gateway.
condition_value	Definition: a condition_value is the actual value of a condition. The condition_value of the condition of a given xor_gateway determines the alternative path chosen for the flow in the same xor_gateway. Examples of a condition value are: “positive”, “negative”, “true”, “false”, a plain text string, or an integer number.
Event	Definition: an event is something that happens during the course of a process. Events affect the flow of a process. There are two types of event based on when they affect the flow: start_event and end_event.
	General_concept: flow_object
end_event	Definition: the end_event indicates where a process ends its flow. The end_event ends the flow of the process and thus does not precedes any other flow_object.
	General_concept: event
flow_object	Definition: a flow_object is one of the following objects: event, activitie and gateway.
Gateway	Definition: a gateway is a location within a process where the flow merges, forks or takes only one of two or more alternative paths, depending on the value of a condition.
	General_concept: flow_object
xor_gateway	Definition: an xor_gateway is a location within a process where the flow can take only one of two or more alternative paths, depending on the value of a condition.

	General_concept: gateway
fork	Definition: a fork is a location within a process where the flow splits into two or more parallel flows.
	General_concept: gateway
and_join	Definition: an and_join is a location within a process where two or more flows merge into a single one. An and_join is characterised by the fact that the resulting merged flow (output of the and_join) can be started only when all the merging flows (input of the and_join) arrive at the location identified by the and_join.
	General_concept: gateway
participant	Definition: a participant is a person or a role or a system or a program that performs an activity.
Process	Definition: a process is any set of activities performed within a company or organization with a specific aim.
	General_concept: activity
start_event	Definition: the start_event indicates where a process starts its flow. The start_event starts the flow of the process and thus must precede exactly one flow_object. General_concept: event
Fact type forms	process *has* flow_object flow_object *is of* process

	activity *is* flow_object gateway *is* flow_object event *is* flow_object flow_object *precedes* flow_object process *has* participant participant *is of* process participant *performs* activity activity *is performed by* participant process *starts with* start_event process *ends with* end_event start_event *is* event end_event *is* event xor_gateway *is* gateway xor_gateway *complies with* condition xor_gateway *precedes* flow_object *with* condition_value and_join *is* gateway fork *is* gateway
Rules	it is necessary that each process *has* at least one activity. it is necessary that each process *has* exactly

	one start_event. it is necessary that each process *starts from* exactly one start_event. it is necessary that each process *has* at least one end_event. it is necessary that each process *has* at least one participant. it is necessary that each activity *is performed by* exactly one participant. it is necessary that each xor_gateway *precedes* at least 2 flow_object. it is necessary that each fork *precedes* at least 2 flow_object. it is necessary that each join *precedes* exactly one flow_object. it is necessary that a start_event *precedes* exactly one flow_object. it is impossible that a flow_object *precedes* a start_event. it is impossible that an end_event *precedes* a flow_object. it is necessary that each xor_gateway *complies with* exactly one condition. it is necessary that an xor_gateway *precedes* a given flow_object *with* a given condition_value.

4.3.2 Sample Process

This section provides a simple process example both modelled through a BPMN diagram (Fig. 5) and an SBVR model (Fig. 6). The former allows an easy and rapid understanding of the entire flow and of the various elements that are used to compose it, the latter is based on the SBVR Business Process Vocabulary and Rules proposed in 2.3.1 thus demonstrating its use for an effective modelling of simple but complete processes.

The example represents a simplified internet purchase with three actors: the *buyer* that wants to buy a product on internet, a *seller* that has a web-site where he sells some products, and the *bank* that intermediates the financial transaction. The following simplifications are assumed in order to reduce the complexity (but not the completeness) of the example:

- the buyer is already authenticated in the web-site;
- the buyer orders only one product;
- in case of negative response from the bank (the credit card information are wrong or there is some other problem with the financial transaction), the buyer quits immediately the purchase;
- the seller has its own shipment system in order to send the product to the buyer;

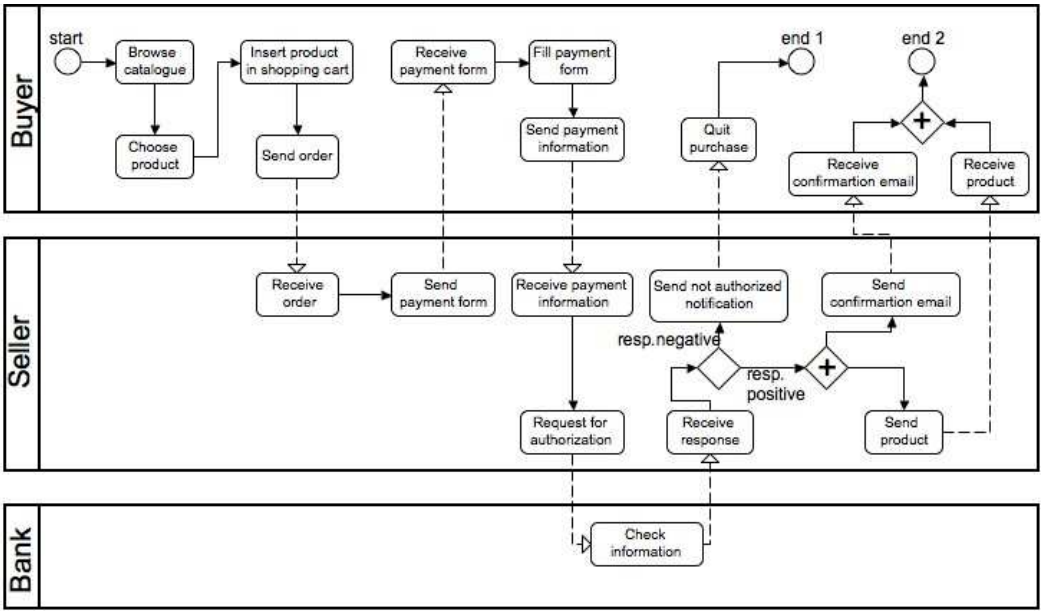


Figure 5. BPMN representation of a simple process example (internet purchase)

In order to model such process example with the SBVR Business Process Vocabulary and Rules, the following steps will be followed:

- the fact type form "process *has* start_event" will be used in order to identify the start event of the process;
- the fact type form "process *has* activity" will be used to identify all the activities of the process;
- the fact type form "process *has* end_event" will be used in order to identify the end event of the process;
- the fact type form "process *has* participant" will be used in order to identify the participants of the process;
- the fact type form "participant *performs* activity" will be used in order to identify the participant that performs each activity of the process;
- the fact type form "activity *precedes* activity" will be used in order to identify the sequence flow of the process;
- the fact type form "activity *precedes* flow_object *with* condition_value" will be used in order to identify the right sequence flow that the process must follow in an xor_gateway when its condition has a specific value;
- the fact type form "activity *precedes* end_event" will be used in order to model the end of a specific flow in the process;

```
process 'internet-purchase' has start_event 'start'

process 'internet-purchase' has activity 'browse catalogue'

process 'internet-purchase' has activity 'choose product'

process 'internet-purchase' has activity 'insert product in
shopping cart'

process 'internet-purchase' has activity 'send order'

process 'internet-purchase' has activity 'receive order'

process 'internet-purchase' has activity 'send payment form'

process 'internet-purchase' has activity 'receive payment form'
```

```
process ‘internet-purchase’ has activity ‘fill payment form’

process ‘internet-purchase’ has activity ‘send payment information’

process ‘internet-purchase’ has activity ‘receive payment information’

process ‘internet-purchase’ has activity ‘request for authorization’

process ‘internet-purchase’ has activity ‘check information’

process ‘internet-purchase’ has activity ‘receive response’

process ‘internet-purchase’ has activity ‘send not authorized notification’

process ‘internet-purchase’ has activity ‘quit purchase’

process ‘internet-purchase’ has activity ‘send product’

process ‘internet-purchase’ has activity ‘receive product’

process ‘internet-purchase’ has activity ‘send confirmation email’

process ‘internet-purchase’ has activity ‘receive confirmation email’

process ‘internet-purchase’ has end_event ‘end 1’

process ‘internet-purchase’ has end_event ‘end 2’

process ‘internet-purchase’ has participant ‘buyer’

process ‘internet-purchase’ has participant ‘seller’

process ‘internet-purchase’ has participant ‘bank’
```

participant ‘buyer’ *performs* activity ‘browse catalogue’

participant ‘buyer’ *performs* activity ‘choose product’

participant ‘buyer’ *performs* activity ‘insert product in shopping cart’

participant ‘buyer’ *performs* activity ‘send order’

participant ‘seller’ *performs* activity ‘receive order’

participant ‘seller’ *performs* activity ‘send payment form’

participant ‘buyer’ *performs* activity ‘receive payment form’

participant ‘buyer’ *performs* activity ‘fill payment form’

participant ‘buyer’ *performs* activity ‘send payment information’

participant ‘seller’ *performs* activity ‘receive payment information’

participant ‘seller’ *performs* activity ‘request for authorization’

participant ‘bank’ *performs* activity ‘check information’

participant ‘seller’ *performs* activity ‘receive response’

participant ‘seller’ *performs* activity ‘send not authorized notification’

participant ‘buyer’ *performs* activity ‘quit purchase’

participant ‘seller’ *performs* activity ‘send product’

participant ‘buyer’ *performs* activity ‘receive product’

participant ‘seller’ *performs* activity ‘send confirmation email’

participant ‘buyer’ *performs* activity ‘receive confirmation email’

start_event ‘start’ *precedes* activity ‘browse catalogue’

activity ‘browse catalogue’ *precedes* activity ‘choose product’

activity ‘choose product’ *precedes* activity ‘insert product in shopping cart’

activity ‘insert product in shopping cart’ *precedes* activity ‘send order’

activity ‘send order’ *precedes* activity ‘receive order’

activity ‘receive order’ *precedes* activity ‘send payment form’

activity ‘send payment form’ *precedes* activity ‘receive payment form’

activity ‘receive payment form’ *precedes* activity ‘fill payment form’

activity ‘fill payment form’ *precedes* activity ‘send payment information’

activity ‘send payment information’ *precedes* activity ‘receive payment information’

activity ‘receive payment information’’ *precedes* activity ‘request for authorization’

activity ‘request for authorization’ *precedes* activity ‘check information’

activity ‘check information’ *precedes* activity ‘receive

```
response'

activity 'receive response' precedes activity 'send not
authorized notification' with condition_value 'negative'

activity 'send not authorized notification' precedes activity
'quit purchase'

activity 'quit purchase' precedes end_event 'end 1'

activity 'receive response' precedes activity 'send product'
with condition_value 'positive'

activity 'send product' precedes activity 'receive product'

activity 'receive product' precedes end_event 'end 2'

activity 'receive response' precedes activity 'send confirmation
email' with condition_value 'positive'

activity 'send confirmation email' precedes activity 'receive
confirmation email'

activity 'receive confirmation email' precedes end_event 'end 2'
```

Figure 6. SBVR representation of a simple process example (internet purchase)

5 Part 2 – Architecture and Implementation of a SBVR inspired Adaptable/Dynamic Service Oriented Workflow

Contributor – Dr. Victor Bayon, (Formally at Birmingham City University)

6 Introduction

In this section we present the design and implementation of a proof-of-concept architecture workflow based on the SBVR previously defined.

The design and implementation of this work has to deal with several design constraints for the purpose of producing a practical implementation of a dynamic workflow implemented within the current Digital Business Ecosystem (DBE) architecture. The DBE architecture was developed through EU-IST programme funded under Framework Programme 6. The six considerations that we are going to take into account are:

1. The SME business model defined in a pseudo natural language (SBVR) and their translation to a real business process model.
2. SBVR as technical framework workflow modelling.
3. Implementation of DBE dynamic workflows using SBVR as a specification language.
4. Scalability of the workflow design.
5. Training/Learning about D(B)E[2] technologies.
6. Practical versus theoretical.

One. We need to consider how to map a pseudo natural language expressed business model into to a sequence of actions using SBVR as a way to define the business process model. In this mapping exercise we need to look into how a small business describes their day to day activities in the sense of "How we do things around here" and see how these natural language expressions could map to SBVR constructions. Part 1 of this deliverable has introduced the ideas of how to capture language into hierarchies of concepts that can be used for defining software components.

Two. We need to consider SBVR as a framework to express not only business rules, as its original design and specification was intended, but also business processes/activities. By business processes/activities we mean the details of the business that could be potentially mapped to business processes, business activities (a business process will be composed of many interweaved business activities) and ultimately to a DBE software workflow based on software services. Part 2 has shown an approach to describe a workflow in SBVR.

One important point to highlight is that that SBVR is not originally designed to define workflows. Its focus on first order logic expressions and lack of temporal logic makes SBVR a framework not suitable for describing expressions such as sequences of events (temporal logic) in an intuitive or obvious manner[3].

[2] D(B)E refers to both the Digital Ecosystem approach and the DBE, being DE the a socio-economical concept and DBE the technical instantiation (the actual implementation such as the ServENT/Execution Environment) of the concept.

[3] http://en.wikipedia.org/wiki/First-order_logic#Difficulty_representing_if-then-else

Three. As our starting point is the current DBE platform, we need to fit the SBVR expressions and its limitations to the current capabilities of the DBE and think about how to convert or automate the definitions to executable software artefacts that can implement a technical business process model. By technical business process model we understand:

- The computable part of a business model;
- The business definition in terms of its internal/external processes/activities that are shared with other SMEs;
- The data definitions (syntax and semantics) that are part of these processes/activities;
- How all these map to DBE software artefacts.

Four. It is necessary to think about the design of the system so that it can scale horizontally, vertically or both in terms of the hardware, infrastructure and services; or at least to provide a reference design that can be configured in a flexible way to suit different needs, such as from simple setups to high performance and high availability configurations.

Five. One aspect of the DBE project was that training was provided to the SMEs that participated on the project. In order to provide leverage to the DBE technology and the training content that already existed we should try to build on the existing learning materials.

Six. A key part of Digital Ecosystems is to provide enhancements on top of the DBE platform that can be used by SMEs and focus on practical results that they can used rather than research-only theoretical aspects. While there should also be research components to the work, this section of the deliverable focuses on achievable and usable results for SMEs so that the real ecosystem of SMEs can potentially emerge from the usage of the platform.

Part 2 of this report is structured in the following way:

First of all in Section 7 we revisit the DBE platform in order to explore its strengths and weaknesses regarding the creation of dynamic/adaptable workflows.

In Section 8 taking into account the lessons learned from the usage of the DBE architecture we introduce a series of design patterns to adapt the DBE architecture for dynamic/adaptable workflows at the service at the service level.

In Section 9 we implement the SBVR/XPDL workflow introduced on the previous chapter using the new service architecture.

In Section 10 we present the experiences gained as a result of this work package and the conclusions.

7 The DBE as a Dynamic De-centralised Runtime Environment: The Cathedral vs. The Bazaar

The starting point of our implementation is the original DBE architecture. Designed as an "onion", the DBE architecture has many layers that build on top of each other to form its overall shape and architecture. The DBE was not designed as a whole single and indivisible unit but as a series of components that could be used on the top of each other to provide more and more advanced and complex functionality. Also, the DBE did not need to run all its components (from P2P to Semantic Registries) in order to provide its basic functionality.

The key dynamic aspect of the DBE was the capabilities of service Search and Discovery (S&D) whereby services and their endpoints were only bound to service clients during runtime via service descriptions and service runtime information. From a service client's perspective, a service bind operation always had to be performed after an endpoint S&D operation, allowing the architecture to run fully de-centralised at different levels.

One of the main differences between traditional Web Services (WSDL, WS-*)[4] and DBE was, for example, that inside an enterprise it is possible to manage the services availability and quality, also including externally exposed web services where the webs services management occurs top-down and where there is a absolute central point for coordination.

We would like to call this model the "Cathedral" model as in the "Cathedral and the Bazaar"[5] . On the other hand, the DBE design principles encouraged services to live in a less organised, more ad-hoc and "chaotic" existence like in a "Bazaar" model without an absolute central point of control.

Figure 7 (below) illustrates the Cathedral model based on a strong top-down hierarchy (bigger nodes) which can happen easily within a single enterprise versus the "Bazaar" model where the nodes are distributed in a less organised manner. For reference, different P2P architectural models were explored during the DBE, such as a hybrid mixture of "Cathedral" and "Bazaar" models based on the Super-Peers/Gradient topology concepts (Dowling et al., 2007),(Biskupski et al., 2007) .

[4] http://xfire.codehaus.org/Stack+Comparison

[5] http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/

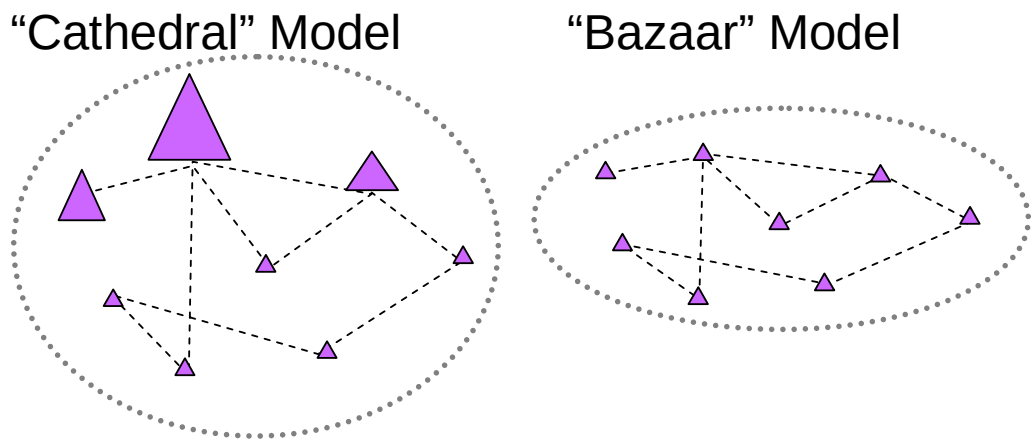


Figure 7. Cathedral & Bazaar Models

The DBE S&D layer-based architecture had several levels which access the S&D facilities, from the lower layers of search from simple keywords (via Federated Autonomous Directory Architecture - FADA entries) to more complex syntactical constructions that can be defined using the Business Modelling Language (BML). From a developers perspective, FADA implemented a simple registry[6]. Also in the DBE there was the notion by design that services or services registries could not be centralised, trading off the benefits of centralisation (i.e. richer description and policies for management, one entry point, speed, caching) for dynamics (i.e. up-to-date environment, less organised), following the Bazaar model. The Super-Peer/Gradient based architecture tried to balance the trades of both by having some nodes that, due to their capabilities and higher availability, could act as temporal/permanent centralised yet independent points of access to the network.

Each DBE endpoint could potentially be bound to 2-3 layers of syntactical, semantic and runtime information that represented the services individually and also as aggregation of services, helping service clients to search, discover and select services at runtime according to their definitions and different levels of expressiveness. The 2-3[7] levels of information available to facilitate the S&D of services endpoints were:

COMPONENT	**ENDPOINT EXPRESSIVENESS**
FADA registry	Keywords (FADA entries)
BML	Attribute/Value pairs, Categories and Taxonomies (from ontologies)

[6] FADA implements a registry (Directory).

[7] 2 or 3 levels depends on how separated BML/SSL are considered. From the DBE architectural viewpoint they were different metamodels but from a users perspective, they were used together.

SSL	Technical interfaces specification

Table 4. FADA endpoints expressiveness

For this deliverable we consider the lessons learned of deploying and using DBE services in real world scenarios from a dynamic systems perspective, revisiting the DBE architecture and finding from its current limitations a new architectural design with the goals to support better and easier dynamic service execution.

7.1 FADA

First of all FADA provided a way to associate keywords (FADA entries) to endpoints. In principle searching for service's endpoints using the FADA registry, we had to provide two distinctive pieces of information:

- The service proxy (style 1)[8] and FADA entries
- The method's name and methods input/output signature (style 2) and FADA entries.

For example Table 5 illustrates a set of 6 FADA entries that can be used to identify the service within the DBE network. FADA entries in principle are very similar to tags/labels of the so called folksonomies[9] or user generated content categories, where service developers could simply label their services individually by adding categories to them.

FADA Entries	**Endpoint**
1-***b77d6d1d-2d40-44e4-b1b5-6e599feab737l*** 2-***FADAWorkflow*** 3-***internet-purchase*** 4- ***seller*** 5- ***seller-send-confirmation-email*** 6- ***send-confirmation-email***	http://dbe.dnsalias.net:2727/seller-send-confirmation-email/CODEBASE

Table 5. FADA entries example

Table 6 illustrates programmatically how to search for a service in FADA using the style 2 approach. Using servENT's[10] *ClientHelper*.*Workspace* class we needed to specify the method

[8] This approach to service execution is not shown on this deliverable as the second approach (Style 2) is preferred.

[9] http://en.wikipedia.org/wiki/Collaborative_tagging

[10] http://swallow.sourceforge.net/

(*methodName*) that we wanted to execute and the methods input/output signature and values (*paramTypes*, *paramValues*).

```
public static ObjectHolder SearchAndExecuteByTags(String methodName,
String[] tags, ObjectHolder input)
      {
            ClientHelper            css            =            new
ClientHelper(ServentInfo.getInstance().getPrivateURL());
            try {
                  Log. write(methodName);
                  Workspace  ws  =    (Workspace)  css.getProxy(null,
tags);
                  Class[]  paramTypes  =  new  Class[]  {String.class,
ObjectHolder.class, ObjectHolder.class};
                  ObjectHolder output = new ObjectHolder();
                  Object[] paramValues = new Object[] {input,output};
                  ws.invoke(methodName, paramTypes, paramValues);
                  return output;
            }
            catch (Exception ex)
            {
                  ex.printStackTrace();
                  return new ObjectHolder(null);
            }
      }
```

Table 6. RCP Searching and Executing service by tags and method name

From a programming point of view, the message passing Remote Procedure Call[11] (RPC) style was less-coupled than Style 1 of the method invocation facilities available in servENT's *ClientHelper* class, which only required **some** syntactical information about the method/service whereas the Style 1 approach required **all** syntactical information about the service.

---

[11] The DBE's execution environment is designed to be able to use a particular style of remote procedures calls. http://en.wikipedia.org/wiki/Remote_procedure_call

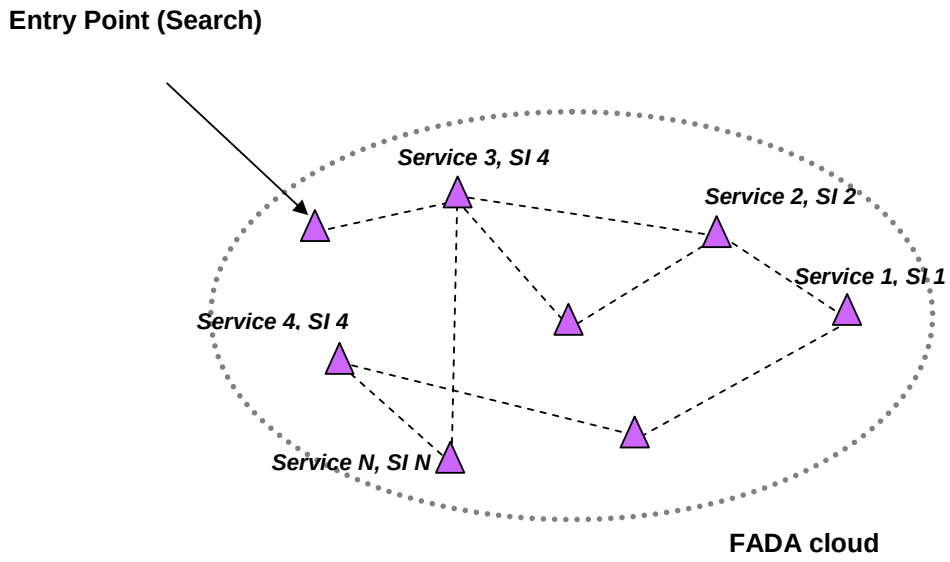


Figure 8. Searching for services and service interfaces

Figure 8 (above) illustrates potentially how a series of nodes (purple triangles) could have one service each[12] implementing different types of SI (SI 1 to N). To search for a service, we will need to specify the required information at runtime (FADA entries, method name and signatures) in order to obtain the service proxy.

FADA not only provided a registry for services and a lookup for service entries, it also provided the mechanism for connecting the nodes to each other in a network scale-free (Barabási et al., 2003) configurations facilitating a broadcast flooding[13] approach to S&D.

To summarise this section, the required steps for finding and executing a service/method endpoint using FADA were:

1. Client specifies FADA Entries & Service info (signature or proxy) and issues a search.
2. The network gets flooded searching for matching services in terms of entries and class signatures.
3. Nodes with matching services respond back to the client.
4. Client chooses which service (proxy) to execute from the dynamic search results list.
5. Client invokes the service/method.
6. Client process the result.

[12] Nodes can have more than one service

[13] http://fada.sourceforge.net/fada3.html

7.2 BML and SSL

7.2.1 BML

In DBE the services description could be specified in the form of BML structures. The information can be as simple as attribute/value pairs or as complex as entries to domain specific taxonomies/ontologies.

BML also allowed the use of technical ontologies such as subsets of OWL DL[14] that were used as expanded data classes in hierarchical models. Once models+data were defined, this information could be associated to the service for S&D.

Figure 9 illustrates BML workflow where a technical ontology model (vCard), is used in a BML model, making the ontology classes available as user data entries.

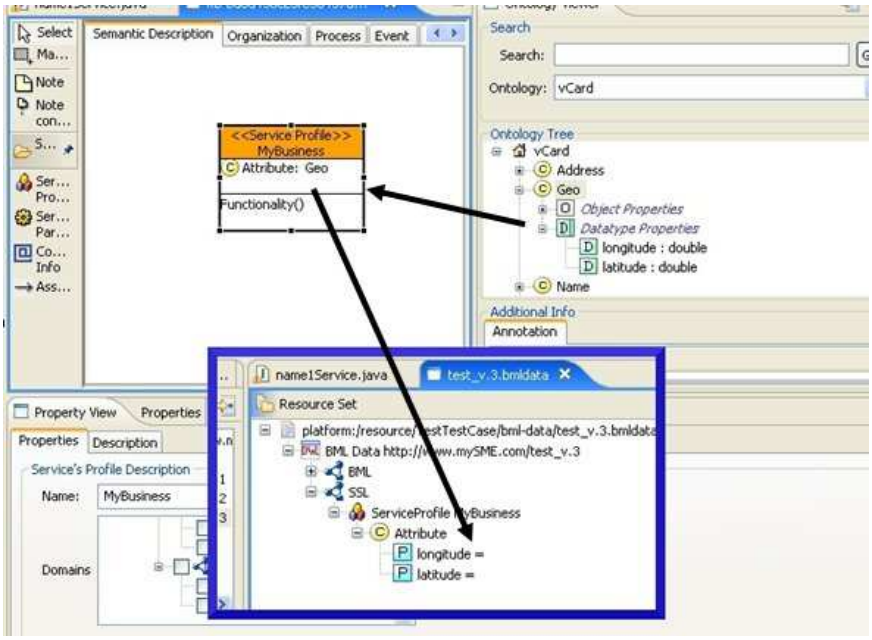


Figure 9. BML workflow

Once a service was deployed special syntactical text constructions (BML queries[15]) could be used to search the service proxy and execute the service.

[14] http://www.w3.org/TR/owl-guide/

[15] http://opensoa.blogspot.com/2006/03/bml-presentation.html

Figure 10 illustrates a search scenario that implemented a specific type of BML model such as domain specific Nanotechnology BML model and a search template that was used to query models implementing the Nanotech BML model.
Companies that used that specific model to express themselves could be ranked according to how "compliant" to the model they were, such as the compliance to generic industry model. If the model for example represented companies capabilities, then the services could be ranked according to such capabilities or lack thereof.

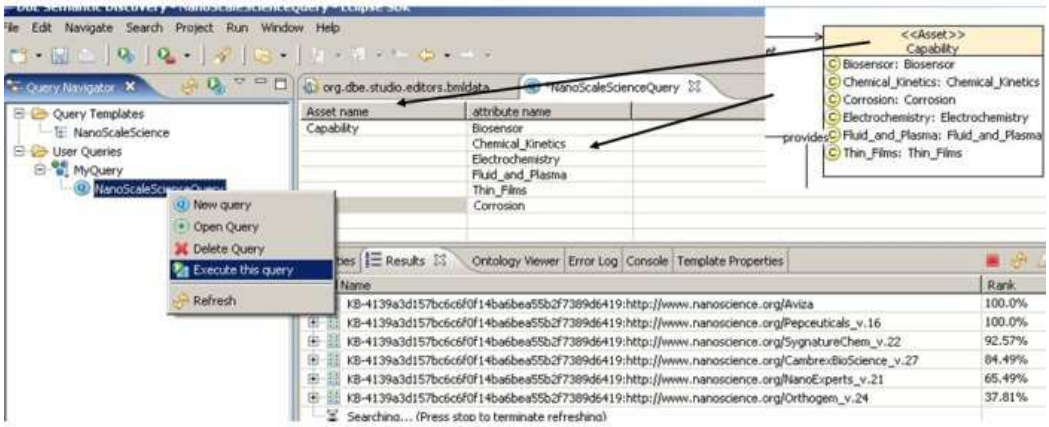


Figure 10. Search template against a BML model

While BML could express services in terms of data attributes, BML still required FADA to find the real services. BML definitions were deployed separately from the service in the form of a Service Manifest (SM)[16]
The SR service ran on specific DBE nodes and contained all the SMs from different service providers. Although the SR provides a centralised repository for storing the SMs (similar to the Super-Peer architecture model), a service provider could choose freely on which SR service to store their SM, and SMs were replicated across SR services according to different runtime parameters.
When a service was mapped with BML, a FADA entry is associated with its SMID (Service Manifest Identifier). The steps for finding a service endpoint using BML were:

1. Client specifies BML query.
2. Query is flooded to all nodes running SR services.
3. SR return ranked results (0%>rank<100%) of matching SMs from all SR asynchronously.

[16] The SM creation was part of the DBEStudio workflow. Slide number 40. http://opensoa.blogspot.com/2006/11/dbestudio-tutorial-ver-022swallow-ver.html

4. Client selects SMs from ranked results
5. The SMID is extracted form the SM and then the FADA process outlined before is repeated in order to obtain the real endpoint of the service associated to the SMID.

Figure 11 illustrates the process of obtaining a SMID, from the original BML Query.

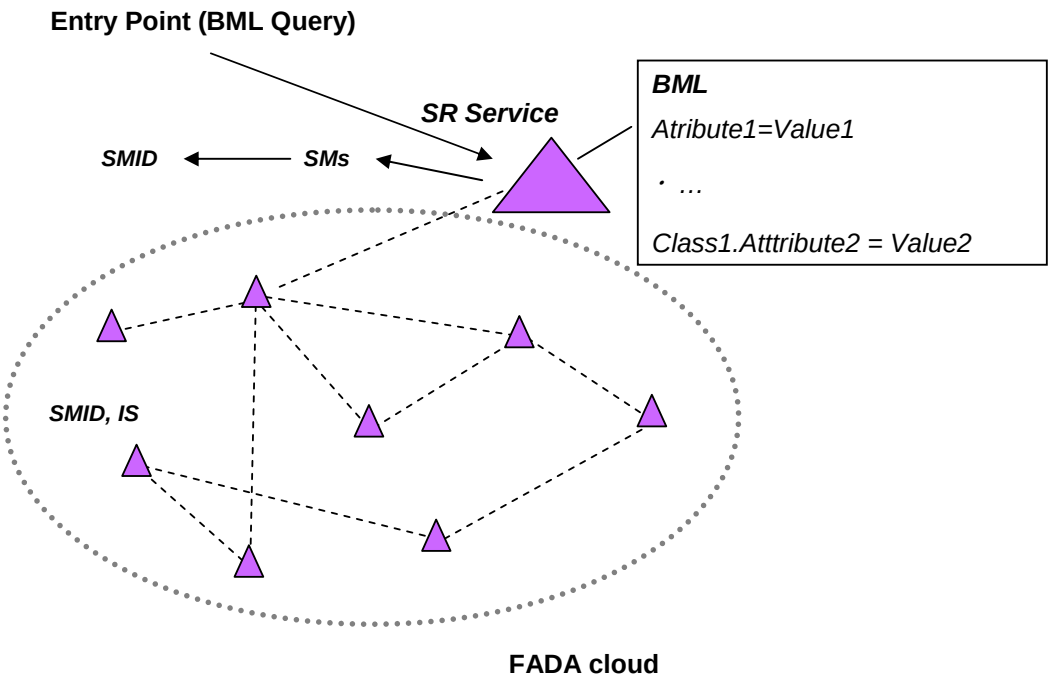


Figure 11. BML query and service search

7.2.2 SSL

SSL, Semantic Service Language[17], was used to defined the Service Interfaces (SIs) and also extra data attributes. It was also used to generate the actual service java interface service stubs. By defining the SSL model, BML search could be extended to also include interface specifications and not just data. For all practical purposes, BML and SSL could be seen as a XML serialised storage of objects and interfaces.

In terms of search FADA entries, BML and SSL acted as a static means for service S&D. SSL was the equivalent of S&D in FADA by specifying implicitly the service proxy. However, with SSL the service interface as a search parameter could be provided as a computer independent model.

[17] SSL was known within DBEStudio as the Semantic Description of the Service Profile. An example could be seen in slides 14 to 19. http://opensoa.blogspot.com/2006/11/dbestudio-tutorial-ver-022swallow-ver.html

All the information provided and included in the SM had to be entered before the service was published and by design, SMs were semi-permanent and detached pieces of information about the service.

BML+SSL trade off search simplicity (FADA entries) for complex definitions that promised to increase the search capabilities and software development efficiencies. However, SMEs had to spend a great deal of time learning the concepts of BML such as modelling, ontology design and how the overall framework fits the DBE architecture. BML could have been useful if there were many almost identical services available from many different providers.

More importantly, as described in D2.1[18], the introduction of complex and more powerful frameworks such as the semantic web stack are limited by important issues such as:

- Useful reusable models that can be used and understood by the targeted developer/user audiences in no time.
- Initial critical mass that is needed to reap the benefits (network effect of many people using the technology).
- Adoption by “non-technical people”.
- Learning and training and marketing efforts required to change attitudes of developers/users.

7.3 Lessons learned: Critical issues for DBE dynamic workflow execution

The previous section has introduced how from basic service descriptions (FADA entries) to more complex syntactical constructions (BML+SSL) DBE services could be searched, discovered and executed.

Like many other approaches, the BML starting point was to define enough service syntactical information following a top-down approach with the goal of generating the boiler plate code from syntactical constructions and to be able to create implementation skeletons ready for service deployment.

While this approach automated some of the processes and provided the basic structures for S&D, it required developers/users to follow complex procedures with the potential gains of “just” to generating some necessary method stubs, and in the mean time make their services very dependent on the interfaces.

On the other hand, FADA entries required minimal configuration and service prototypes could be deployed very fast as developers could just write the methods directly and deploy the services very quickly with a simple cut & paste/file system operation.

[18] http://files.opaals.org/OPAALS/Year_1_Deliverables/WP02/d2_1_Final.pdf

BML separated the service descriptions on a different file descriptor independent from the service itself in the form of the SM and the design/architecture made it not feasible for a service to change its information during runtime without a very complex process. Services were not aware of their own parameters at runtime and this was considered a major trade-off for a dynamic system. For the most part, the DBE services had to be preconfigured at design and the runtime S&D constructs could not be changed ad-hoc or easily[19].

From a user's perspective, the BML and SSL distracted the developer from the core activity which was to develop and deploy services. A DBE with less semantics and definitions (i.e. FADA entries only) and more services could have been more successful than a DBE with a lot more semantics (i.e. BML and services) but less services. In the DBE the answer to "Can a busy web developer do something interesting in less than 10 minutes?" was no.

BML/SSL also introduced certain aspects that made the service execution framework very inflexible. As the services had have specific endpoints which revealed the services' implementation details, it required each service consumer to create a specific type of client to that particular service interface, making the overall system inflexible and not very dynamic at runtime.

7.3.1 How endpoints are defined on Internet Standards

This is a very subtle but important detail that needs to be considered. The internet is based on standards which have been proven to be highly scalable and functional.

One of the design patterns of successful internet services is the separation of the network protocols, the services/applications, the endpoints and application states. For example, it is possible to interact with ftp or http servers while using telnet, and the same with many other standard applications. As far as the services are concerned the endpoints know little or nothing about service specific information. The design pattern is very clear: a Client/Service (Server) interaction is always encoded on the protocol between them and not on the service endpoints.

On the DBE the functionality of the service was exposed directly on the endpoint, including application state (i.e. via calling different service methods). The DBE inherited the idea of interacting on the internet directly with "business objects" at the endpoint level (for example obtaining a "Client" object information such as *Client.getAccountNumber*()) rather than following a simpler approach of message passing/protocol oriented pattern.

---

[19] DBE also implanted a Distributed Intelligent System (DIS) that acted as a service recommender for services. DIS required BML/SSL definitions to operate to make BML/SSL definitions among services and to see which ones were similar and providing service matching according to these relationships and runtime information such as service aggregation/combinations.

This is also a current dilemma among the REST versus SOAP and WS-*. While the SOAP/WS-* communities are building more and more complex specifications that delegate protocol and state the endpoints on top of HTTP, the REST[20] practitioners can quickly integrate web services without dealing with the complexities of SOAP/WS-* stack[21] directly with the less powerful but simpler HTTP semantics.

Another issue that we need to factor in is the availability of data that can be aggregated using simple workflows versus complex workflow environments. For example, there is nothing that imperative-style XML BPEL[22] can do that cannot be achieved with a simple imperative script in a language with libraries that supports WS-*.

While BPEL is helpful and useful inside enterprises, on the internet and for SMEs it is a different story. Today we can see many dynamic composition tools are created using lightweight approaches with REST and lightweight data models such as Atom/RSS/GData such as Yahoo Pipes[23], Google's Mashup editor[24] or many of the hundreds and hundreds of internet Mashups[25] . From a DE point of view, network effects are more important than a complex (and potentially powerful) architecture and Mashups provide an example of what it can be achieved with simple tools and basic standards.

7.4 SBVR: A solution looking for a problem?

Towards the end of the DBE project and the start of OPAALS, SBVR was adopted by OPAALS as a way of using an approach to describe business activities in natural language with the aim of code generation, while supporting the service orientation approach of software design. It was seen that SBVR would provide the capabilities to design DBE services Mashups.

The initial SBVR scoping specification positioned SBVR as a way to allow business users to describe business rules (as in "companies policies and regulations") and not production rules that can be fed to a computational rule engine, leaving a gap in terms of the initial aims of the OPAALS project and the scope of the specification.

---

[20] http://en.wikipedia.org/wiki/Representational_State_Transfer

[21] SOAP versus REST is an ongoing and very long debate. While both approaches have merits, in practice many developers prefer REST as it is more lightweight. According to some estimates, most of the service integration done in platforms such as Amazon or eBay, are done using REST rather than SOAP as it is simple to achieve the same level of functionality.

[22] http://www.ibm.com/developerworks/library/specification/ws-bpel/

[23] http://pipes.yahoo.com/pipes/

[24] http://code.google.com/gme/

[25] http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)

While not the only possible way to describe a workflow in SBVR, the previous section has illustrated how intricate and verbose the use of SBVR can be as it does not include temporal logic in the specification. Surprisingly, even for a simple workflow diagram such as the one shown, it illustrates that an image is worth a thousand SBVR sentences.

We have externally and independently evaluated SBVR and captured the feedback from some of the SMEs that participated in the DBE. Among the West Midlands participating SMEs Domain Solutions[26] is well know for their work on OMG MDA toolsets (CodeGenie[27]) and are familiar with OMG (they are a member), their standards and processes. While SBVR represents a genuine effort towards capturing soft-systems type of business rules, currently the benefits are not clearly visible except perhaps for a business rules analyst/consultant and major changes need to be done in order to capture the interest of SMEs.

David Pilfold, the Managing Director of Domain Solutions, has many years of experience on software modelling and has suggested that until those benefits can be shown clearly, adoption will remain problematic. Among his concerns with SBVR were:

- How to illustrate what the benefits are to SME business owners; show what part of the problem SBVR is solving - the problem needs to be defined.
- How to fully join the process from modelling to code generation (OCL based or action language for service implementation).
- The danger, as with many OMG “standards”, that vendors will implement in different ways and file formats will be incompatible.
- No reference implementation, only a paper specification.
- Current cultural SME practices, from “jump and code” (Baskerville, 2003) to analysis and model driven process.

From a personal perspective SBVR is a controversial technology. SBVR does not build on the lessons learned from the current state of web architectures (from complex architectures such as OMG’s CORBA to the simpler REST approach) and from the experiences working with SMEs and BML as it represents a totally different approach. This is also new to OPAALS researchers and explains the different approaches to SBVR that existed during the phase I of the project.

Now that practical results need to be achieved the shortcomings of SBVR become more evident as the theory is put into practice. There is also some critique as far as the linguistics approach in SBVR is concerned within OPAALS.

Although from a technical perspective SBVR has not got much to offer at this stage, it can still be used as a way to soft-capture business requirements/rules using it as a declarative language to structure business activities. The following sections will introduce a series of

[26] http://domsols.com/

[27] http://www.ooagenerator.com/codegenie.htm

service development patterns/practices on top of DBE architecture in order to bring the lessons learned from the real-world deployment of DBE services, BML usage and some of the ideas behind workflow design using SBVR declarative descriptions that can be used to define service compositions. To illustrate the design and implementation, we will use the online Buyer/Seller BPMN and SBVR models illustrated in the previous Part.

8 Redesigning the DBE Architecture for Dynamic Workflows

As SBVR is loosely defined and with not known open reference implementation, SBVR could mean many things to different people. During Phase I of the OPAALS project there has been several parallel threads of SBVR work dealing with different aspects or approaches trying to come out with results that could in principle be relevant to SMEs (which will be presented in the forthcoming Deliverable 2.2):

- The Indian Institute of Technology Kanpur (IIT) focused on extracting the logical formulation of SBVR expressions (Reference deliverable) to be able to extract activities that could be converted into UML activity diagrams.
- In WP2.1, SUAS worked on obtaining classes (business objects) from SBVR models and to deploy the classes to the Grails architecture[28] in order to be able to create Create-Read-Update-Delete (CRUD) web forms for those business objects.
- Working as a subcontractor for BCU (formerly known as UCE), a parser was created to be able to develop a parser tree from a SBVR specification.
- Also there was a study of SBVR from a linguistics perspective carried out by Kassel University.

While SBVR cannot provide a simple way right now to create a technical specification from the description that can be a machine-executable workflow, it is possible to use SBVR as a soft-systems (i.e. non-technical or computable output, just concepts) enterprise modelling type of approach to provide an outline of how our system should work in terms of processes (the bigger picture) and activities that constitute each project.

Consider Table containing a subset of SBVR statements as illustrated in the previous Part:

process ‘internet-purchase’ *has* start_event ‘start’
process ‘internet-purchase’ *has* activity ‘browse catalogue’
process ‘internet-purchase’ *has* activity ‘choose product’

[28] http://grails.codehaus.org/

process ‘internet-purchase’ *has* activity ‘insert product in shopping cart’

Table 7. SBVR activities definition

From the SBVR model specification, it is possible to extract the 4 definitions important in order to implement the workflow:

- The overall process which is the sum of all the activities (in our case **“internet-purchase”** is the process).
- The activities that form the process (i.e. **“browse-catalogue”**, **“choose product”**, …).
- Roles or participants in the workflow, such as **“Bank”**, **“Buyer”** or **“Seller”.**
- The temporal statements that constitute the workflow (not directly supported by SBVR) and the if-then-else (also not supported on SBVR first order logic approach).

For creating the workflow, we consider activities as services, which could be defined as and endpoint (as read/write URL) in the network that implements that activity. It is important to notice that in the previous chapter, while SBVR is used in a great level of detail in order to define the different facts, fact types and rules that make up for the Buyer-Seller-Bank workflow (the service definition), SBVR does not say what the activities do (the service part).

From a practical perspective, service developers will be interested in the service implementation part rather than in the service design “overheads” if the design overheads do not bring substantial advantages in terms of service development/debug/deployment/maintenance or model useful software artefacts.

From this point of view, we would like to define two service development qualitative concepts that can be used to measure such qualities: “Service Design Overhead” (SDO) and “Service Plasticity” (SP) as:

- **SDO[29]:** Service Design Overhead. is the amount of effort/preparation that the developer needs to put in service implementation that is not actually related to the service implementation but more oriented towards service management aspects (such as endpoint definitions, deployment, etc…). For example, a CORBA service will have a large amount of SDO, a DBE service with BML would be “medium” whereas a REST type service will have very little SDO. Traditionally SDO is related to “Top-

[29] Tool vendors focus for most of the time on increasing SDO at the expense of as a way to create and sell the toolsets that simplify SDO for developers while adding “management” capabilities, such as OMG MDA or WS-* toolsets.

bottom" service design/implementation and large frameworks geared towards management features.

- **SP:** Service Plasticity. It is, in principle, a tangible measure in which we evaluate how easy it is to adapt a service to other platform/requirements. For example, REST type services have a very high level of SP (you could access the service using PHP, Python, etc), whereas DBE services could be high or medium (see next section) and CORBA would have a very low SP. Traditionally we could think of SP as a "bottom-up" approach to service design and implementation.

Although we have presented these two factors as separated, in reality it is often the case that one is related to the other. Any design activity is about embracing the trade-offs in order to provide the most optimum and beautiful solution for the given problem. Although we have not found references to concepts such as SDO/SP, the concepts are implicit among developers ' conversations, such as the current debate among the Web Services stack (WS-*) proponents versus REST[30]

The advantages/disadvantages of higher or lower SDO and SP will depend on the context of the problem we are trying to solve. For example, REST services (Fielding, 200) have very little service definitions and allow very fast deployment whereas SOAP based web services have their endpoint definitions and they allow having UDDI registries or BPEL workflow engines that can take advantage of those definitions.

As the interface objects are defined at the endpoint level, code generators and template engines can read the endpoint definitions and generate stubs, clients and test cases from those definitions.

Another way to look at the SDO/SP relationship is to look at the framework and decide whether it is a data bus (just moving data around such as the internet), a service bus or an object bus; and what facilities the bus has to support the different operations with just data, services and objects.

However the trade-off is that the higher the SDO then the less the flexibility or adaptability of the service at the endpoint level. In the case that the endpoints are high SDO it will also require specialised workflow languages (such as BPEL) to be able to "wire" the services together. Low SDO and high SP can be accessed, for example, with dynamic programming languages such as PHP Python or Ruby or even bash scripts (i.e. reading endpoints with the wget command). A similar argument applies to dynamically typed versus statically typed programming language and their strengths and weaknesses as explained in D2.1.

[30] The current Ruby on Rails version (2.0) has deprecated direct support for WS-*/SOAP while adopting a fully REST full model. http://weblog.rubyonrails.org/2007/12/7/rails-2-0-it-s-done.

This trade off does not mean that an architecture is limited by its own functionality when it can easily be combined with other technologies[31]. As with complex systems such as termite mounds, the complexity and functionality does not come from how complex (or fully featured) the individual parts are, it comes from the interaction among all the different components of the system. In our case, our architecture needs to be adaptable rather than fully featured. As Charles Darwin stated:

"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change."

In hindsight, the DBE made the decision of trying to implement and over engineer in a different way much of the functionality that already existed on the web services space. With a simpler design that could have been easily scaffold into other systems/architectures that provide complementary functionality, the DBE could just have been plugged into to provide its core strength such as the P2P service discovery, while leaving the rest of service side of the design to already existing application containers.
As illustrated in the previous section, the "way the internet works" service endpoints were independent from application syntax and semantics, implementing the full service on the wire protocol rather than exposing the methods or stating on the endpoint itself. While this method has very low SDO, you can write an internet service with 4 lines of code (literally) in many different languages, and it has the highest SP possible. Within the internet services context in an open environment, such has the DBE, ideally required high SP and as low SDO as possible.

8.1 Redesigning the DBE Architecture for Dynamic/Adaptable Workflow Support

This section will introduce a series of 4 deployment/implementation approaches that will reshape the DBE architecture into one that is more dynamic/adaptable based on the experiences with the early version. By dynamic/adaptable we mean a higher level of SP while keeping SDO low. The 4 approaches are:

- To create canonical services definition and Interfaces. Focus on service agnostic endpoints.

[31] See the Unix Design Philosophy. http://en.wikipedia.org/wiki/Unix_philosophy. We can see that these principles are still valid with the design of Yahoo Pipes, following the same principles as the original UNIX Pipes.

- servENT and also REST access points to the service. Execute the services using any architecture/platform easily.
- FADA entries as Folksonomies. Locate the services using tags/labels and/or attribute/value pairs.
- Enabling Caching for Service Information. To provide means to cache the tags/labels and attribute/value pairs in order to create service registries.

8.1.1 Creating Canonical Service Definitions and Interfaces

Step1: Simplify and standardise the interface of the services

BML derived DBE services were bound to the service endpoint interfaces as defined by their SSL model. For each service there would be one or more interfaces representing the service interface. In order to create workflows, the consuming clients also required in practical terms to have all the classes stored locally used as interface proxies or passed as parameters, exposing service details to the endpoints and making the system not very flexible with a medium to high SDO and a low SP.

A potential solution to this design issue (low SP) is to create a canonical definition for all services that represent simple input to the service and output while the service handle state via a protocol rather than the endpoint, similar to the way that standard internet protocols are designed. In DBE terms, this could be implemented in two different ways at the interface level.

The first way is to implement the standard approach of servENT's *Adapter* by creating an interface (CanonicalService.java) that gets implemented using a simple DBE Adapter (CanonicalServiceImpl.java) as illustrated on Table 8 (below). The interface could also include the method name of the part of the service (state) that we want to execute, assuming we would implement that method inside the service.

CanonicalServiceImpl.java

```
package org.dbe.services;
import org.dbe.servent.*;


public class CanonicalServiceImpl implements CanonicalService,
Adapter{
      public Object execute(String methodName, Object input)
      {
            Object output = myProcess(methodName ,input);
            return output;
      }
      public void destroy() {}
      public void init(ServiceContext arg0) {   }
}
```

CanonicalService.java

```
package org.dbe.services;
public interface CanonicalService {
      public Object execute(String method, Object input);
}
```

Table 8. Canonical service implementation as simple adapter

Another way to implement our service is to use the servENT's RPC where, rather than directly exposing the objects on our interface as before, we simply expose a message-passing interface based on java RPC holders.

As illustrated in the earlier part of this Part, this approach offers less coupling (and higher SP) than the previous one. Now our service adapter and implementation will look like:

CanonicalServiceImpl.java

```
package org.dbe.services;
import java.rmi.RemoteException;
import org.dbe.servent.Adapter;
```

```
import org.dbe.servent. ServiceContext;

import javax.xml.rpc.holders.ObjectHolder;

public class CanonicalServiceImpl implements Adapter,
CanonicalService{
      public void process(String method, ObjectHolder input,
ObjectHolder returnMessage) throws RemoteException {}
      public void destroy() {}
      public void init(ServiceContext arg0) {}
}
```

CanonicalService.java

```
package org.dbe.services;
import javax.xml.rpc.holders.ObjectHolder;

public interface CanonicalService{
      public void process(String method, ObjectHolder input,
ObjectHolder returnMessage) throws java.rmi.RemoteException;
}
```

Table 9. Canonical service implementation as RPC adapter

By having a canonical implementation, we choose to standardise how the endpoints look to the clients, hopefully ease development time and the learning curve as all the endpoints look the same.

Now developers would have to implement their services in a message passing/protocol fashion as the only endpoint of the service will have to read the input, decode it and call the correct internal methods that implement the service. Also the service endpoint is stateless, in the sense that state is encoded on the protocol and not on the infrastructure. This canonical approach to service endpoint design is also important for the rest of the implementation.

8.1.2 DBE Services as REST services

Step 2: Make your service reachable/executable using a REST-like approach

In order to decouple even further the services from the endpoint, the DBE architecture allows calling DBE services as simple URL Read/Write using the servENT's implementation Service

Handlers interfaces[32]. Table 10 illustrates the canonical implementation of the Service Handler.

```
package org.dbe.services;


import java.io.IOException;
import javax.xml.rpc.holders.ObjectHolder;
import org.dbe.servent.ServiceContext;
import org.dbe.servent.http.*;
import java.io.OutputStreamWriter;
public class CanonicalServiceHandler implements ServiceHandler {

      private CanonicalService service;

      public void handle(ServentRequest request, ServentResponse
response) throws IOException
      {
            // Execute the service and get the output
            String method = request.getParameter("method"));
            javax.xml.rpc.holders.ObjectHolder input = new
ObjectHolder(request.getParameter("input"));
            javax.xml.rpc.holders.ObjectHolder output = new
ObjectHolder();
            service. process(input, output);

            // We get the response writer so we write back to the
client
            // the results of executing the service
            OutputStreamWriter responseWriter = new
OutputStreamWriter(response.getOutputStream());
            String serialisedOutput = output.value.toString();
            responseWriter.write(serialisedOutput);
            responseWriter.flush();
            responseWriter.close();

            // We are finished
```

[32] http://opensoa.blogspot.com/2007/01/service-as-webpage-information.html

```
            response.setStatus(ServentResponse.SC_OK);
      }

      public void init(Object service, ServiceContext serviceContext)
      {
            this.service = (CanonicalService)service;
      }
}
```

Table 10. The service's ServiceHandler Implementation

DBE services could be extending to support from an object model only to REST style interactions with some constraints by simply creating a java class that extended the ServiceHandler interface.

By adding the *ServiceHandler* interface, when the class is initialised (*init* method, see Table 10) by the servent's *ClassLoader*, the service interface is passed as an argument.

Service requests coming via HTTP could then be captured and redirected to our service for execution and then return the result of the execution in a transparent manner via http.

With this design service endpoints and the addresses of the ServiceHandler could still be found via FADA while keeping the S&D and execution infrastructure free from service dependent interfaces such as the BML derived endpoints, reducing SDO while increasing SP when using the canonical service implementation.

The REST approach is feasible and easy to implement as our service interfaces have been defined to support a standardised input/output. Also another benefit from this approach is that we move from moving objects around to just moving data around.

Our approach to REST is based on the principle of passing what method to execute and the parameters as input encoded on the URI, which often is not considered pure REST. More formal REST semantics (such as PUT,GET,UPDATE,DELETE) could be implemented by extending the servENT's ServiceHandler capabilities. Currently this approach does not adhere to the REST semantics and this should be considered.

8.1.3 FADA as a Folksonomies and Attribute/Value framework

Step 3: Add simple tags/labels and attribute/values to identify your services

A critical decision for a developer of services on the DBE architecture is whether to go the "BML way" or the "FADA way" in order to make the services "Findable"[33].

[33] http://en.wikipedia.org/wiki/Findability

As illustrated earlier, deriving services from BML descriptions was a rather complex process. FADA also had the trade-off that it only allowed very simple keywords (tags) to make the services "findable". It is only obvious to think that a more expressive framework for describing will allow better S&D, however, the price to pay is complexity of the tool. But it is also natural to think that a developer will choose the easiest way to develop a service.

With BML the same dilemma could be seen when developing a webpage that search engines will index. As most commercial search engines (i.e. Google, Yahoo) do not support the direct processing of anything beyond standard XHML (i.e. no support of micro formats, or RDF and so on), web developers do not see an immediate value on adding rich semantics to them[34].

The DBE Portal implemented a similar concept. When the DBE portal was first installed a form was presented to the user that allowed him/her to enter basic details of the business such as location details, etc and business description/domain. These details were then deployed as BML on the DBE Semantic Registry (SR). Figure 12 illustrates the user form during the node installation's process and the data that would make the DBE Portal findable.

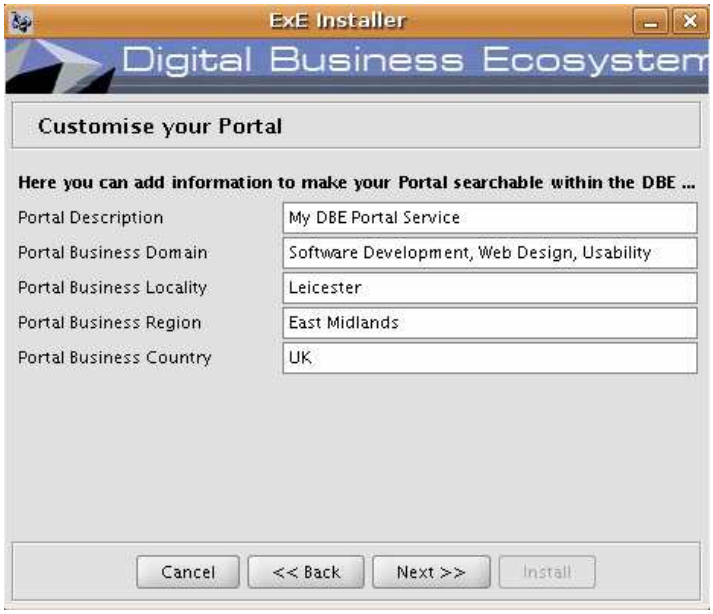


Figure 12. DBE Portal specification

[34] Something like GRRDL service (http://www.w3.org/2004/01/rdxh/spec) could be the way forward whereby rich semantic data can be embedded in webpages and processed automatically by webservices specialised in such data formats and a similar concept could be used on DBE style architectures, where the semantics are embedded inside the service (as in GRRDL webpage).

The lesson learned from the DBE (and the same for many other systems) is that a complex architecture is not required to build systems that behave like complex systems. A simple architecture that can be built upon easily should be preferred, as the main objective should be to gain critical mass of users in order to bootstrap complex behaviour with many users rather than provide a full (complex) architecture that only very advanced developers will be able to take advantage off.

FADA allowed the services to register tags on the P2P by simply adding entries to the **deployment.props** text file of the service as illustrated on the table below. FADA also registered as entries by default the name of the deployment directory (physical name on the disk), the SMID (smid) and the application name (applicationName).

deployment.props

applicationName=FADAWorkflow

smid= 309470978233474
adapter=org.dbe.workflow.fada.ServiceAdapter

entries=**activity**, **process**, **participant**, **workflow**

activity=send-payment-form
process=internet-purchase
participant=seller
workflow=FADAWorkflow

Table 11. Service configuration: deployment.props service's configuration file

Once our service is deployed on the servENT, we could inspect its entries using the FADA web based interface as illustrated in the figure below:

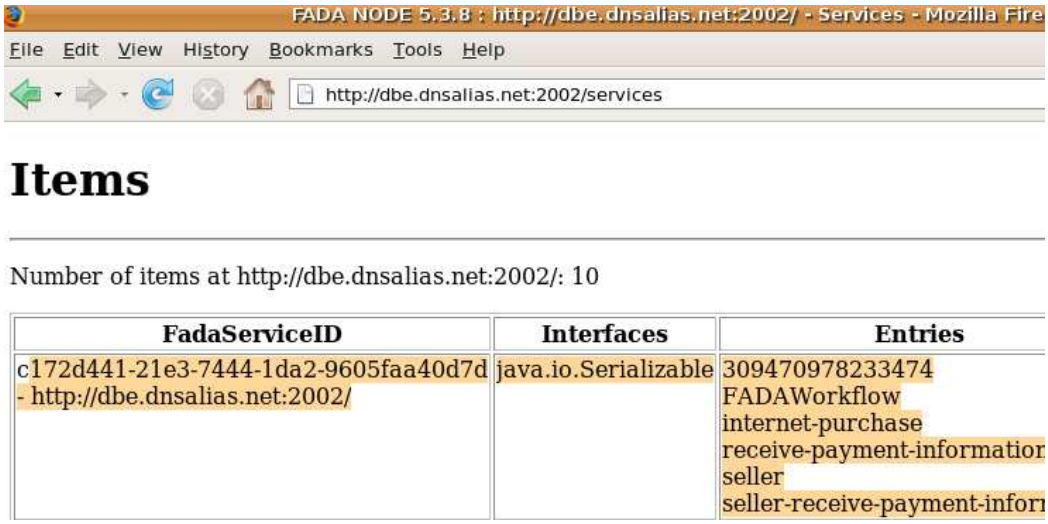


Figure 13. FADA entries listed on the FADA services node page

FADA entries are similar to tags as in "Folksonomies" tagging. They associate the service with a series of keywords. However, with the help of a configuration "trick" the user could assume that the tags could also be attribute/value pairs rather than just tags.

User assigned FADA entries took the format as described in **Error! Reference source not found.**. In our example we defined a series of entries (**activity**, **process**, **participant**, **workflow**) that can be associated to values that are then deployed as FADA entries.

Entries = entryName1, EntryName2, …., EntryNameN
EntryName1 = value1
EntryName2 = value2
……
EntryNameN = valueN

Table 12. deployment.props metaconfiguration

There is an important shortcoming with this approach. FADA, as well as with other P2P networks, can only offer on P2P limited search capabilities to one parameter such as a hashkey (such as DHT's[35]) or entries in the case of FADA , whereas to search for the attribute names associated for those values is not supported.

One way to overcome this limitation is to use a similar approach to the DBE's SR with two important modifications. The first modification will be that the service implements a HashMap

[35] http://en.wikipedia.org/wiki/Distributed_hash_table

with key/value pairs and the second one is that the service is responsible for updating the registry with this information. With this approach it is important not to make the bootstrap
Table 13 illustrates a potential implementation in Java of the HashMap. At service startup (*init* method), some of the first key value pairs can be added to the hashmap. Some of the possible and obvious that could be included are:

- Service description
- Service usage/runtime information
- Provider
- Provider URL
- Download/upload URL
- Contact details (i.e. vCard string)
- Location (Postcode/Country, Lat/Lon)
- etc

```
private void put(String key, String value)
{
            hashMap.put(key, value);
}
private String get(String key)
{
            return hashMap.get(key).toString();
}
private String getAttributeNames()
{
            return hashMap.keySet().toString();
}
public void init(ServiceContext serviceContext)
{
      hashMap.put("activity",serviceContext.getParameter("activity"));
      hashMap.put("process",serviceContext.getParameter("process");
      hashMap.put("participant",
                  serviceContext.getParameter("participant");
      hashMap.put("smid",serviceContext.getParameter("smid"));
}
```

Table 13. Key/Value pair HashMap initialisation

Defining these attributes should be part of the lowest common denominator of what constitutes a basic service. The service implementation also allows methods to change those parameters at runtime.
As shown in the reference implementation, more attributes could be added/changed and accessed at runtime via the *put/get* interface, making the attributes easily accessible, cacheable and transformable to data formats such as RSS/Atom[36] or replicated in distributed and shareable in attribute/value engines attribute/value storage engines such as the forthcoming CouchDB[37].

8.1.4 Enabling Caching for Service Information

Step 4: Don't call me, I'll call you

One way to achieve performance and scalability on web architectures is to provide means to cache web content. This relies on the notion the content is static for at least some period of time and it is not necessary to continuously query the original source of information.
As the service and its SM were separated on the DBE, it was the responsibility of the user and not the service to keep the SM up-to-date creating difficulties when an SM was updated (i.e. different BML data) or the service was updated (new service interface) or both. Two more consequences of this architectural design was how quickly the SR could become full of stale SMs that were no longer valid such as older versions or SMs of services that were no longer active.

In our simpler architecture, we delegate the responsibility of publishing the service relevant information to registry services (i.e. RSS storages, Attribute/Value storage engines etc) to the service itself, and also to remove it and keep it "active" on the registries. For example, a policy could be established that the per-service information on the registry is deleted every 7 days if the service does not re-register. This is similar to the FADA implementation of the JINI[38] inspired lease-protocol[39].
Our implementation is based on registering/deregistering the service information every time the service is started/stopped, and also refreshing the information every time an attribute is updated and also by having a thread that waits a specific period of time. It is possible to also have a thread that updates the registry every so often.

---

[36] If we output service information as RSS/Atom there are many already existing news RSS/Atom aggregators that could be used to enable them as service registries.

[37] http://couchdb.org/CouchDB/CouchDBWeb.nsf/Home?OpenForm.

[38] http://www.sun.com/software/jini/

[39] http://fada.sourceforge.net/outside-maven/FADA_5.0.pdf

If the service does not update itself then the registry service should remove the hashtable from its records. **Error! Reference source not found.** illustrates a possible implementation of registry update input operations. If an existing key is updated, the method will simply update it. The implementation should also have a configurable thread that automatically renews the registry.

```
private void put(String key, String value)
{
            hashMap.put(key, value);

            updateInRegistry(hashMap);
}
private String get(String key)
{
            return hashMap.get(key).toString();
}
private String getAttributeNames()
{
            return hashMap.keySet().toString();
}
public void init(ServiceContext serviceContext)
{
      hashMap.put("activity",serviceContext.getParameter("activity"));
      hashMap.put("process",serviceContext.getParameter("process");
      hashMap.put("participant",
                  serviceContext.getParameter("participant");
      hashMap.put("smid",serviceContext.getParameter("smid"));

      startUpdateRegistryThread(hashMap);

      updateInRegistry(hashMap);
}
public void destroy()
{
        deleteInRegistry();
}
```

Table 14. HashMap update and RegistryUpdate thread

9 Implementing Dynamic/Adaptable Workflows

With the architecture defined previously, there are several potential properties that services deployed can have. Taking into account some potential restrictions around the services such as application deployment constraints, specific hardware/software requirements per service or application context and semantics, in general terms the properties are:

- Services can be deployed on any node; Services could potentially be "hot swapped" across nodes (see below).
- Services are searchable and discoverable via tags P2P or key/value pairs via registries.
- Services have canonical service definition; Clients also have a canonical interface definition too.
- Services are not framework/programming language dependent: Java clients could S&D and Execute, other clients (i.e. PHP) can execute the services and they could also search for services[40].
- Loosely coupling of endpoint with service's functionality that could be easily reconfigured at runtime[41].
- Inside the service, the code should be implemented to find dynamically what method and what arguments to execute and convert the requests dynamically to match the objects inside the service via java reflection, helping the clients to be loosely coupled and more flexible at runtime.

Conceptually, another feature of the architecture is that services can be seen as computational units that can be configured, including executable classes, at runtime. Using the *put* method (Table 15), for example, we could "inject" our service with the functionality (java class) that will be in charge of doing the computation, decoupling the service from its implementation and using the infrastructure as a service bus, where even the services functionality could be configured at runtime.

Figure 14 illustrates the traditional way of having our client executing the service (top part, section (A)) while section (B), bottom part illustrates how the *ServiceImpl.class*[42] could be injected at runtime before the client invokes the service, with capabilities that even the client injects the functionality.

[40] Although not illustrated, a service that can be used by non-java clients to search could easily be implemented.

[41] This could be achieved by extending Servent's CoreService interface in our service.

[42] In practice for this example Java POJOs that can be serialised should work but care should be taken if implemented with servENT's class loaders.

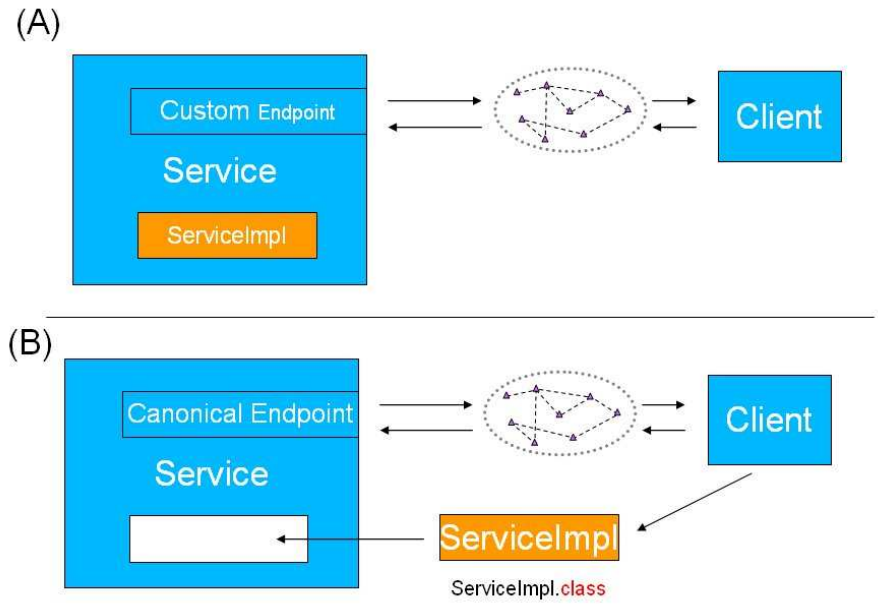


Figure 14. Service implementation injection at runtime

Table 15 illustrates how this could be achieved in java by injecting the *ServiceImpl.class* to our service via the canonical endpoint[43]:

```
public void put(String method, Object computational Class)
{
            objectshashMap.put(method, computationalClass);
            updateInComputationalRegistry(objectshashMap);
}
```

Table 15. Service implementation injection

When the service implementation is injected, then via java reflection the service could read the class and create the necessary mappings for the service to be able to execute that functionality via the canonical endpoint and the information handler.

[43] While this method breaks our SDO/SP architecture, it is shown as a concept.

For our workflow implementation we will use SBVR as a simple soft-systems specification language to extract roles and activities leaving the logical temporal aspects and if-then else constructs to be specified manually. The following sections will illustrate how the XPDL workflow diagram can be implemented using the adaptable architecture and how it could then be executed.

One last important property from this architectural design is that the service implementation can be independent from the services themselves and each service/activity dependency is related to host capabilities. For example, the services can be independent of business objects that are defined in SBVR but are dependent on the host that stores those services on a database.

9.1 Implementing and Executing the Prototype Workflow

We have implemented and executed the XPDL/SBVR-inspired workflow using the adaptable service oriented architecture. For the implementation we have used the three roles (Buyer, Seller and Bank) specified on the SBVR and then 3 different nodes were used to execute the workflow in a fully distributed configuration (although more configurations were used - see below), whereby a client would run on a fourth computer. All nodes were to be connected on P2P network using FADA.

In order to install the full workflow, download the previous compressed .zip files and unzip them in one or several working DBE nodes under the ***<$servent home>/deploy*** directory.

The implementation of the canonical service that has been used on the workflow is provided here:

http://dbe.dnsalias.net/opaals/CanonicalWorkflow.zip

The implementation of the Buyer part of the workflow is located here:

http://dbe.dnsalias.net/opaals/BuyerWorkflow.zip

The implementation of the Seller part of the workflow can be found here:

http://dbe.dnsalias.net/opaals/SellerWorkflow.zip

The implementation of the Bank part of the workflow can be downloaded it from here:

http://dbe.dnsalias.net/opaals/BankWorkflow.zip

The execution of the first 4 activities of the workflow (Client side) is illustrated on Table 16.

```
public static void runWorkFlow()
      {

            ObjectHolder input = new ObjectHolder("My input will be
this");
```

```
            ObjectHolder buyer2seller = null;
            ObjectHolder seller2buyer = null;
            ObjectHolder seller2bank = null;
            ObjectHolder bank2seller = null;

            // Buyer
            {

                  Log.getInstance().write("BUYER --------------");
                  ObjectHolder startOutput = execMethod("execute",
                              new String[] {"FADAWorkflow",
                              "buyer",
                              "start",
                  "internet-purchase"}, input);
                  Log.getInstance().write("start   -   executed   "   +
startOutput.value.toString());


                  ObjectHolder      buyerBrowseCatalogeOutput      =
execMethod("execute",
                              new String[] {"FADAWorkflow",
                              "buyer",
                              "browse-catalog",
                  "internet-purchase"}, startOutput);
                  Log.getInstance().write("buyerBrowseCatalogeOutput
- " + buyerBrowseCatalogeOutput.value.toString());


                  ObjectHolder         chooseProductOutput         =
execMethod("execute",
                              new String[] {"FADAWorkflow",
                              "buyer",
                              "choose-product",
                  "internet-purchase"}, startOutput);
                  Log.getInstance().write("chooseProductOutput  -  " +
chooseProductOutput.value.toString());
```

```
                ObjectHolder    insertProductShopingCartOutput    =
execMethod("execute",
                              new String[] {"FADAWorkflow",
                              "buyer",
                              "insert-product-shopping-cart",
                "internet-purchase"}, startOutput);

      Log.getInstance().write("insertProductShopingCartOutput  -  "  +
buyerBrowseCatalogeOutput.value.toString());

                buyer2seller = buyerBrowseCatalogeOutput;

            }

....
}
```

Table 16. Workflow execution client class

For our implementation, each one of the (SBVR/XPDL) activities is instantiated on the prototype to a call to the method *execMethod.* Table 17 shows the implementation of the method that calls our canonical service. Which activity to execute is determined by the tags used.

```
public static ObjectHolder execMethod(String methodName, String []
tags, ObjectHolder input)
      {
            // We obtain the local CSS
            ClientHelper            css            =            new
ClientHelper(ServentInfo.getInstance().getPrivateURL());
            try {
                  // We get a Workspace to the remote proxy object
according to the tags
                  Workspace  ws  =    (Workspace)  css.getProxy(null,
tags);

                  // Construct our canonical class signatures
```

```
                Class[] paramTypes = new Class[] {String.class,
ObjectHolder.class, ObjectHolder.class};
                ObjectHolder output = new ObjectHolder();
                Object[] paramValues = new Object[]
{methodName,input,output};
                ws.invoke("execute", paramTypes, paramValues);
                return output;
            }
            catch (Exception ex)
            {
                ex.printStackTrace();
                return new ObjectHolder();
            }
      }
```

Table 17. Executing the canonical method

execMethod takes 3 arguments, the method we want to execute within the canonical service, the S&D FADA entries used to locate the service on the P2P networks and the input data. The *ws.invoke* is the actual method that executes the method on the remote service, returning by value the result of the execution.

As our workflow execution is separated from the service implementation, we have not implemented any of the functionality of the activities or what is inside the services. In our examples, each activity returns a random string (via java.util.UUID). The full client workflow can be found on the following link (below).

http://dbe.dnsalias.net/opaals/FADAWorkflowClient.zip

In order to install and run the client, unzip and import the file with Eclipse, updating the external *Java Build Path/Libraries/External Jars* to the location of the ***<$servent home>/client*** and ***<$servent home>/common*** on the machine that is going to be the client.

To successfully execute the All FADA nodes among the roles (Bank, Seller, Buyer) it must be networked. We have deployed and executed successfully the workflow without changing a single line of code or configuration (except the deployment nodes) with the following physical node configurations:

- All Roles/Activities deployed on one node. Client deployed on a different host.
- Roles deployed on 3 different hosts (LAN). Client deployed on a different (same LAN) host.
- Roles deployed on 3 hosts (WAN). Client deployed on a different (WAN) host.

10 Conclusions

We have designed, implemented and deployed the SBVR specified workflow using the adaptable/dynamic architecture. For the work we have used the term “SBVR inspired” rather than SBVR implemented due to the shortcomings of SBVR when used for delivering practical results to SMEs.

While it was hoped at the beginning of the OPAALS project that SBVR was going to provide an interesting approach for designing workflows and as an extension to BML, there are currently important limitations that need to be overcome before the toolsets become relevant to current SMEs. Currently there are no open source or commercial SBVR tools available except perhaps consulting firms providing SBVR related services.

Rather than start from scratch in our implementation for the workflows, we have used the DBE platform for the foundation of the implementation of this work package. The DBE platform is not perfect[44] and we have taken advantage of their strengths (i.e. P2P directory, simple service orientation), while minimising some of its weaknesses (BML complexities).

This architecture has been designed using the experience gained from implementing and deploying DBE services in real scenarios. The recurring pattern of deciding ***how to interface the endpoints*** over and over again for each service via BML led to the decision of creating a canonical service definition, as illustrated by the 4 point service design strategy.

In terms of design, the canonical approach let the developer have the service “worrying” about its implementation and creating separate concerns over how to talk to the service and service implementation. In our opinion service endpoints and service implementations must be separate aspects of the system.

The search and discovery of services has been implemented using FADA entries (tags) which is a powerful yet limiting approach. On the one hand this approach allows greater flexibility. On the other hand far more complex definitions (i.e. RSS, MicroFormats, RDF etc) are not supported on the P2P directory and centralised registry approaches need to be adopted

The approach of having the service update its own information to a registry service can also allow more flexibility in terms of how complex the data can be, and it should be possible to design an RSS registry, an RDF registry and so on as all data will point to the same URI that represents the service endpoint. The service should support different types of representation.

What we gain in flexibility we lose in terms of management facilities that provide the top-down approaches. For example, the DBE supported BPEL and with it the usage of BPEL workflow engines on top of the BML/SSL. This support could be extended to our framework by creating the BML/SSL definition of our canonical service and deploy them to SRs, but then we would

[44] A perfect platform that can satisfy will be very difficult (even impossible) to achieve.

lose the flexibility and granularity of interaction of our services. Any approach will have its advantages/disadvantages.

However, as our system can be considered bottom/up, it can be possible to plug other frameworks/ systems (such as Semantic Web stack, Oasis stacks, OMG specs)on top of it to provide more advanced management and discovery facilities.

With the design we have focused on keeping the architecture as simple as possible to allow top-down frameworks to be plugged into it, while allowing low level interactions to integrate with current systems.

From our perspective it is better to support dynamic languages that use common standards that support higher level specifications such as BPEL. There are plenty of web developers that use PHP and that do not need BPEL to create their highly dynamic composite applications. The image below illustrates a very simple script to our adaptable DBE service from a Ruby script:

get.rb (/home/bayon/ruby/)

```
# Script to execute DBE services with Handler Implementation using ruby.
require 'net/http'.
require 'rexml/document'.
.
# url example http://dbe.dnsalias.net:2727/CanonicalHandler?input=MyInput.
#                                                    &message=MyMessage.
url= ARGV[0].
# get the XML data as a string.
xml_data = Net::HTTP.get_response(URI.parse(url)).body.
.
# output the data that we got.
puts xml_data.
.
```

We could extend S&D to other platforms by implementing a ServiceFinder service (not implemented, just shown as a concept) which basically is a REST-like call to the service.

http://dbe.dnsalias.net:2727/CanonicalHander?
method=**ServiceFinder**&
input=**FadaEntry1**,**FadaEntry2**,**FadaEntry3**

This service would return the endpoint of such a service, which the new one could use to execute the service that we were looking for. The answer to the question *"Can a PHP developer do something meaningful with the DBE in less than 10 minutes?*" with this architectural design starts to look like a **yes**.

This dynamic/adaptable architecture was not designed just as a concept. We have also validated the design in the field looking at the business advantages that the architecture can provide for its users. Currently this architecture has been deployed at one of the DBE SME participants EMNET to provide DBE backend capabilities to a PHP application, using the flexibility that the architecture provides. Tim Miller, the Managing Director of EMNET has said about the architecture:

"*The different approach to service orientation and the opportunities that this presents to the market place has been significant. The integration of systems and the beneficial impact on working practices that this can bring to clients and also internal users offers significant commercial opportunities from increase reach of services, improved efficiency and reduced costs. Use and deployment of services on the DBE has therefore offered a number of real benefits to the company and the construction of new services for clients*." Tim Miller, MD East Midlands Network (EMNET[45]).

[45] http://www.emnet.co.uk/

11 Part 3

Contributor – Nagaraj Konda, Birmingham City University

12 Methodology

The management part of the research has primarily involved the SME software developers towards an exploratory research on business model concepts their sub-concepts and their relationships to key business processes. This research was conducted in 2 phases. Phase 1 involved the selection of business model concepts by reviewing key publications related to this area. This phase supported the development of a framework for data collection and analysis that was done in phase 2. The data collection helped in creating a business model ontology based on the concepts chosen through the review of key publications and developing a list the commonly used sub-concepts with their definitions through combination of unstructured and structured interviews with 3 SME software developers. The interviews also facilitated in the identification of a sample list of business processes that supported the identified business model concepts. The final outcome of this research presents business model ontology, business processes and key vocabularies/terms used in the context of business models.

13 Autopoiesis – Links to Business Models and Process Re-engineering for Enterprises

The definitions and the semantic perspectives on enterprises being discussed needs to be linked to the theory of self-organisation from the autopoietic theory (Maturana and Varela, 1980). This however is expected to be discussed more in detail during the subsequent phases of the OPAALS project and how they link to the developments of Open Knowledge Space (OKS). Here in this report the challenges that are posed in doing so are being discussed briefly keeping in view the importance of this aspect to the enterprises. The challenges are primarily due to the presence of two contrasting approaches presented by two German sociologists i.e. Niklas Luhmann and Peter Hejl (Whitaker, 1995). There has been no consensus on these two views and the debate continues. The two views of enterprises have been presented briefly here.

Enterprises are viewed as social systems comprising of a coherent network of objects and processes. The application of the concept of autopoiesis is viewed to be problematic on social systems where people are seen as the enabling participants of this system (Whitaker, 1995). According to the theory, social systems need to be capable of self-reproducing and in this case reproducing people. The theory fails in this view as this reproduction is not possible. In order to present a possible view for applying autopoiesis for the analysis of an enterprise, Luhmann (1995) came up with a redefined view of the social systems where he replaced 'people' with 'communications' and making it a constituent element of the social system. In this new definition social systems can reproduce 'communications'. Though this definition makes the theory applicable, the approach presented by Luhmann (1995) has been criticized by Mingers (1994) and others as it does not consider the space in which the communications within a social system manifests.

Hejl (1984) on the contrary starts his discussion from a different understanding of social systems where he states that social systems are not self-maintaining or self-organising systems. According to Hejl (1984), social systems are "constituted by components, i.e., living systems, that interact with respect to a social domain. Thus the components of a syn-referential system are necessarily individual living systems, but they are components only inasmuch as they modulate one another's parallelized states through their interactions in an operationally closed way." This view from Hejl (1984) presents a new outlook on the enterprises. Whitaker has analysed these two approaches and highlights key practical examples of application of autopoietic theory to enterprises. These are in terms of business process re-engineering and implementation of ICT infrastructures. According to this analysis the approaches that have attempted to follow Luhmann have had less success than those based on the approach presented by Hejl (Whitaker, 1995). Whitaker also concludes his

analysis by saying that research in the area of enterprise evolution is moving towards 3 key aspects. According to him these are:

- Configuring groupware for mutual orientation and self-organising knowledge base accretion.
- Configuring hardware for mutual orientation.
- Configuring enterprise (re-)engineering practices for mutual orientation and self-organisation.

Business Models as a concept in the recent years has gained prominence. This has become a focus of research after the dotcom bubble has burst. Researchers have been keen to analyse what went wrong and what has made some of the dotcom firms succeed even though most of them failed. Business models have been used to explain the business logic of the firm for creating revenue and for supporting the re-configuration of enterprise logic, the 3rd aspect mentioned above by Whitaker (1995). This relationship where changes in the industry structures influences changes to the enterprise structures has been shown by Malone et al in the diagram below.

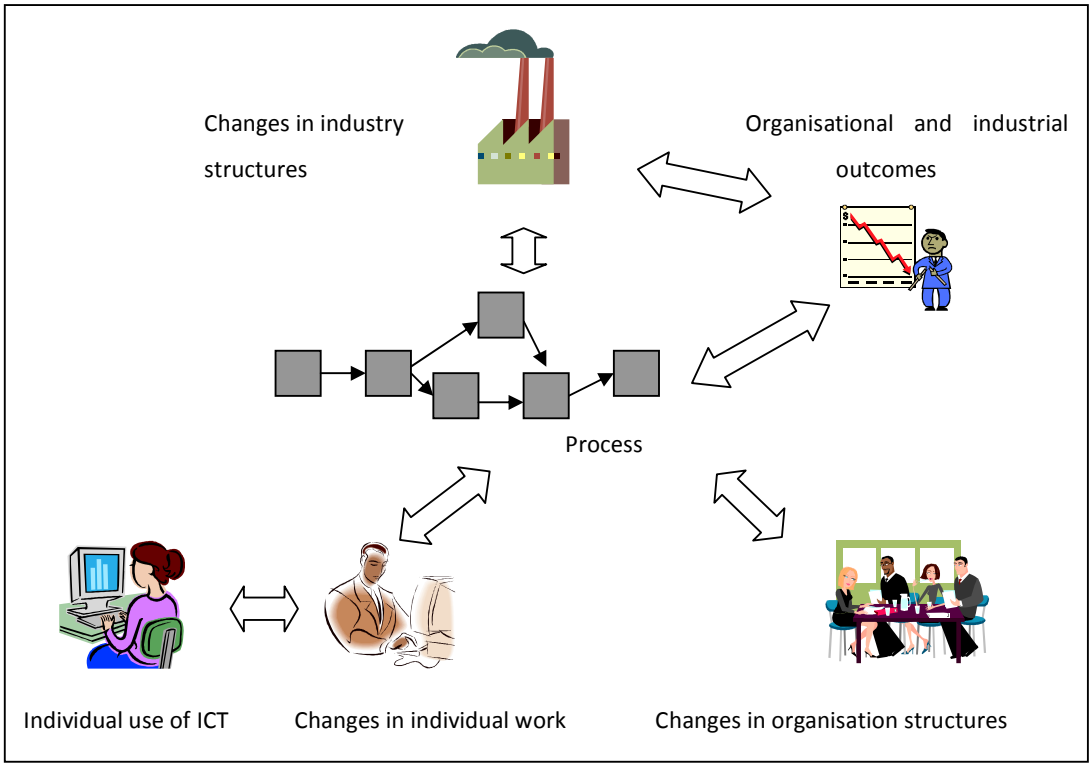


Figure 15. Changes to industry structures (Malone, 2003)

According to Ovans (2000), a business model is defined as “the way to do business.” Also a business model is seen as a value chain architecture of the participants and revenue models

(Timmers, 1998, 2000); and as a combination of revenue model, customer value concept and logistic model for a particular participant (Mahadevan, 2000). Business models provide details of the relationship existing between different actors participating in a commercial activity, the reasons for such relationships in terms of the benefits, details of the cost structure and the revenue flows (Elliot, 2002).

Changes to business models takes place under the influence of many factors. For example new technology inventions and innovations influence the emergence of new business models (Prahalad and Ramaswamy, 2000 and Venkatesh, 1999). As stated by Evans and Wurster (1997), “existing value chains will fragment into multiple businesses, each of which will have its own sources of competitive advantage.” Also, operating business models and value chain partnerships undergo changes as time progresses. The revenue generation possibility of a successful model reduces over a period of time, as the value proposition declines and no longer remains distinctive (Westland and Clark, 1999). Hence it is important for businesses to keep a watch on new business opportunities that come along and position themselves to adopt a suitable change model to sustain successful performance for growth (Linder and Cantrell, 2000). Indeed development of strategies for sustainable performance and growth would require the understanding of such value plane relationships, innovation and knowledge sharing capabilities, customer needs, a technological roadmap, competitive and cooperative landscape as well as the alignment of business strategies for the construction of emergent and new business models.

An understanding of the business model of an organisation helps in discovering business requirements in the forms of people, assets which includes IT systems, etc. According to Nijssen (2007), “the Business knowledge can be described using:

- a structured business concepts catalogue,
- an associated fact types catalogue, together with fact type readings, and
- the associated (business) rules catalogue”

The same author also states that “This is the beginning of a new era in which business requirements are properly specified, making extensive use of semantics (structured concept definitions), fact types, fact type readings, and business rules. The foundation of business knowledge is formed by the structured business catalogue, consisting of concept definitions.” Such an ontological view is being developed using the definitions and concepts presented by the following researchers.

Alt and Zimmerman (2001) have presented a working definition of a business model and have identified six generic constituent elements that are the constituents of a business model. These are Mission, Structure, Processes, Revenues, Legal Issues and Technology.

- Mission provides a high level understanding of the nature of the business including the description of the product or services.
- Processes provide a detailed view on the value creation process.

- Structure determines the nature of the industry and the roles of the players. Process provides a detailed view on the value creation process.
- Revenues provide a view of the investment and financial needs. Legal issues in the form of regulations could influence the value creation process.
- Legal issues, in the form of regulations, could influence the value creation process.
- Technology influences the design of the business model.

Chesbrough and Rosenbloom (2002) have presented a similar definition along with the differences between a business model and strategy including the limitations of business models. The key constituents of the business model according to this definition are the:

- Value proposition that is offered,
- The market segments served and the revenue generation mechanism adopted,
- Structure of the value chain,
- The cost structure and the profit potential of providing the service,
- Description of the value network and
- The competitive strategy.

Osterwalder (2004) has provided a slightly different definition to business model. According to this definition, "A business model is a conceptual tool that contains a set of elements and their relationships and allows expressing a company's logic of earning money. It is a description of the value a company offers to one or several segments of customers and the architecture of the firm and its network of partners for creating, marketing and delivering this value and relationship capital, in order to generate profitable and sustainable revenue streams". Osterwalder (2004) has linked his definition of business model to the balanced scorecard approach presented by Kaplan and Norton (1992) to develop a set of business model constituents. The balanced scorecard approach provides a mechanism to monitor and evaluate the performance of a business by looking at four critical aspects of a business. These are: financial perspective, customer perspective, business process perspective and people perspective looking at innovation and learning capabilities. The key constituents according to this work are:

Value proposition	People perspective
Target customers	
Distribution channel	Customer perspective
Relationship	
Value configuration	
Capability	
Partnership	Business process pespective
Cost structure and	
Revenue model	Financial perspective

The definitions for the business model provided by the above mentioned researchers identifies concepts such as value proposition, cost structure, etc. and also links to interesting analytical frameworks such as the balanced scorecard approach. Balanced Scorecard approach presented by Kaplan and Norton (1992) is an analytical framework for evaluating the strategy adopted by an organisation. The framework uses four focus areas to evaluate the performance. These are based on the following perspectives: People, Customer, Business Process and Financial.

The business model definitions provided by Osterwalder (2004) and Chesbrough and Rosenbloom (2002) have been combined so as to distinctively include all the key concepts in the business model. Based on this choice the key concepts used in this analysis are: Value proposition, target market, value chain comprising of distribution channel and relationship, capability, cost structure, revenue model, capability, value network comprising of value configuration and partnership. These terms are being individually defined below.

- Value proposition is how a product or service (solution) is distinguished from all others in its domain so that target customers consciously select it as a superior option.
- Target market - defined as the group of the population sharing a common set of traits, which distinguish them from everyone else.
- Value chain is a sequence of activities during which value is added to a new product or service as it makes its way from invention to final distribution.
- The value network is a graphic representation of all of the organizations, groups, and individuals that are or could be involved in the development, marketing, and use of a technology.
- Capability that is used by the enterprise in creating and offering value to the market.
- Cost structure is the relative proportion of fixed, variable, and mixed costs found within an organization.
- Revenue model – Provides a view on the different streams of revenue generation.
- Competitive strategy – Porter has presented three generic competitive strategies. These are based on cost, differentiation and niche market focus.
- Profit potential – The capacity to realise higher commercial values based on tangible and intangible benefits.

14 Data Collection

As mentioned in the methodology, the data collection was done in 2 steps. The 1st step involved in the expansion of the business model concepts defined above to further sub-concepts. The 2nd step identified the key business processes that are part of the three chosen businesses.

Towards step 1, the research team used an unstructured approach initially as no suitable approach was found but later developed a series of questions based on the initial experience to discover the commonly used business terms to associate with each business model concepts.

A list of questions is presented below:

1. What does this concept mean to your organisation?
2. What benefit does this offer to your organisation?
3. Give us an example as to how this benefits your organisation?
4. What is the intention of your organisation?

(Meta, Meta-meta levels)

5. How specifically do you apply this concept?
6. To whom do you apply this to?
7. When do you apply this?
8. How do you apply this?

(Model level)

Based on the interviews with the 3 SMEs using the questions above and more specifically question 5, the following sub-concepts or business terms were discovered:

1. Concept: Value Proposition

Description: The value proposition that a business offers is based on its resources i.e. the things that a business has and capabilities i.e. the things that a business could do.

Key sub-concepts or terms: Price, feature, trust, quality, delivery

2. Concept: Target Market

Description: Target market addresses two key questions – Who are our customers? Where are they and how many of them?

Key sub-concepts or terms: Market segment, market size, market drivers, consumers, customers, clients, youth, high income group, enterprise, business, specialised users, local, national, international

3. Concept: Value Chain

Description: Value chain is composed of several linked stages where each stage adds value to the product or service.

Key sub-concept or terms: In-house, outsourcing, off-shoring, global delivery

4. Concept: Value Network
Description: A value network generates economic value through complex dynamic exchanges between one or more enterprises, customers, suppliers, strategic partners and the community.
Key sub-concepts or terms: Enterprise, business, customers, consumers, clients, suppliers, partners, community, collaborator, contractor, consultant
5. Concept: Capability
Description: Capability identifies the potential to generate value.
Key sub-concepts or terms: Product, service, research, consultancy
6. Concept: Cost Structure
Description: Identifies the costs involved in the organisation.
Key sub-concepts or terms: Assets, human capital, intellectual capital, upgrades, licensing, royalties, legal, tax, duties, salaries, bonus, liabilities, insurance
7. Concept: Revenue Model
Description: Indicates the sources of revenue.
Key sub-concepts or terms: Direct sales, indirect sales, products, services, consultancy, research and development, accessories, complementary products, licensing, spin-offs, royalties
8. Concept: Competitive Strategy
Description: This is the approach taken by the business to compete in the market place.
Key sub-concepts or terms: Cost leadership, differentiation, niche focus, collaboration, exclusivity
9. Concept: Profit Potential
Description: Informs the pricing approach adopted by the business to realise the targeted profit levels.
Key sub-concepts or terms: Differential pricing, competitive pricing, product-line pricing, fixed price turnkey, time and material.

14.1 Business processes and terms used by an ISP in Midlands, UK

Products/Services: These are the product and service categories managed by the ISP.

- B2B
- B2C
- Web/Software Hosting
- Supply Chain

Business Processes: Three key processes are being managed.

- Administrative
- Stores
- Solutions

Administrative Processes

- Business Relationship Management
- Catalogue Management
- Marketing Management
- Order Management
- Business Intelligence
- Customer Service
- Customer Management
- Negotiations
- Merchandise Management
- Reseller
- Merchant Management
- Supplier Management
- Business Relationships Management
- Store Operations

Stores Processes

- Direct
- Reseller
- Merchant

Solutions Processes

- Shared Marketing
- Self-Registration
- Hosted Sites
- Evaluation of Business Results
- Catalogue
- Resellers

Detailed business processes for Sales and Order Management of web hosting service performed by the ISP is being used as an example. This is understood, from the discussion with the SME, to have been developed through consultations with Business Process Consultants for adopting industry standard best practices and mirrors the process developed

and used with most industry players. This business process is seen as a derivative of a generic order management process defined by Malone et al (2003). This process links to the BPMN representation made in this report.

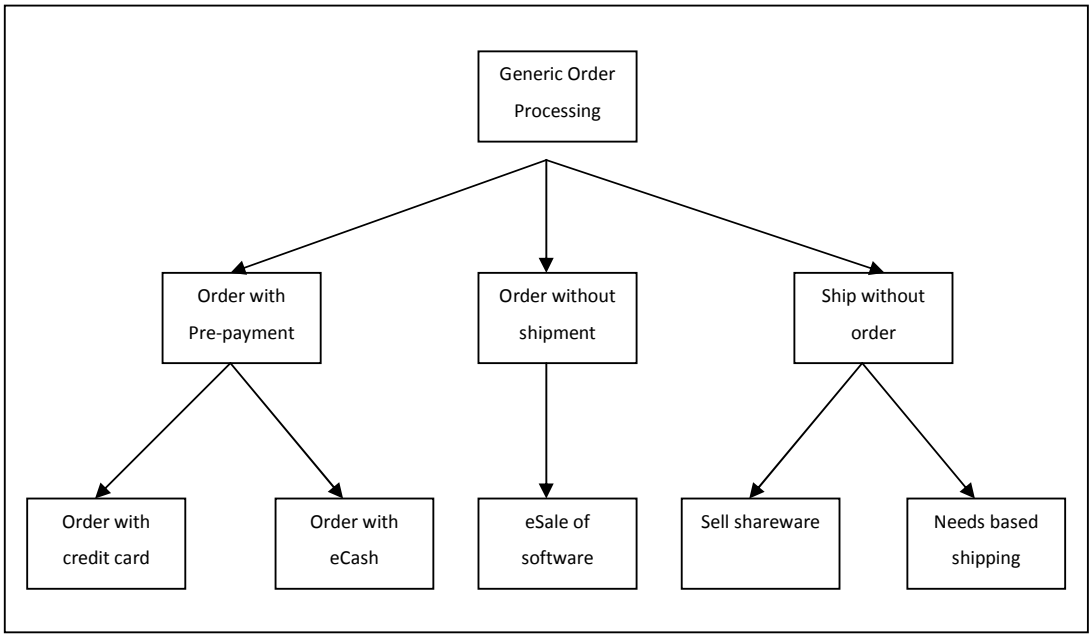


Figure 16. Ordering Process (Malone et al, 2003)

Business Processes for Web Hosting

- Sales and Marketing Processes
  - Create Customer Segments
  - Campaigns
    - Create Web Activity
      - Create Promotion
      - Create Creative Content
    - Create Email Activity
- Order Management
  - Process Order end-to-end
    - Capture Order Activity
      - Capture Order
    - Process Order
      - Process Order Activity
        - Approve Order
        - Try to release order to fulfilment

        - Release Capacity to fulfilment
        - Check Payment Status
    - Pre-process Order
        - Handle Prepared Order Expiry
        - Allocate Web Capacity
    - Request Payment Authorisation
        - Check Payment Status
    - Handle Rejected Order
    - Transfer Order
        - Update Order Status
- Process Pending Payment Authorisations
- Process Completed Orders
    - Capture Payment Activity

The business processes for sales and order management shown above becomes useful to search and discover executable software processes to support them. The business process in this case has been designed and implemented for a specific purpose and is based on certain assumptions. However, the interesting challenge is when there is a need for a change to these business processes i.e. when these requirements and assumptions change. In practice, businesses anticipate minor changes and to a large extent manage them as part of the business process. However, major changes are the area of concern. The changes could be influenced by many factors as stated earlier in this part. This could be due to the changes in the market environment, technical innovations, abnormal or unforeseen requirements, etc. For businesses to sustain their business model, such flexibility to manage changes is a key requirement. In most instances the decision to make changes are managed by the process manager and is seen as a manually driven process. In terms of making this dynamic, it will be interesting to see if any of the structural vocabulary approaches such as taxonomy, folksonomy or ontologies will be helpful.

On our opinion, we see that the folksonomies should not be mixed up with ontologies or taxonomies as a folksonomy is not a structural vocabulary. Vander Wal (2005) has published the famous folksonomy definition: "Folksonomy is the result of personal free tagging of information and objects (anything with a URL) for one's own retrival. The tagging is done in a social environment (shared and open to others). The act of tagging is done by the person consuming the information." (Vander Wal, 2005).

In addition, whereas folksonomic tagging is indeed a very dynamic process, ontologies and taxonomies could be described as "formal" rather than "dynamic". Moreover, as Gruber (2007) mentions, the results of folksonomic tagging can be represented with ontologies.

15 Conclusions

An interesting aspect from the business process presented above is its inflexibility to change. There is a set pattern that needs to be followed for every process and within every process a set pattern of activities need to be carried out. The aspect that is being covered is more towards already knowing about the business models, their associated business processes and designing a workflow that is executable. The representation of the runtime workflow is not a fixed structure from a computing perspective (e.g. static BPEL) but could be influenced to manage changes at the runtime-workflow level that can be achieved with simple representations such as the folksonomic-tagging approach illustrated in Part 2. At one point in time, our fixed (and desired) workflow structure is a simple abstraction or a map of a combination of tags.

However taking into consideration, the concept of the bazaar and the cathedral models presented in the earlier part, the argument would continue as to the nature of triggers that could influence the aspects of dynamism and flexibility that is a key to business model sustainability.

References

Alt, R. & Zimmermann, H., 2001. Business Models. *Electronic Markets*, 11 (1), p. 3-9.

Altman, R., Bada, M., Chai, X.J., Whirl Carillo, M., Chen, R.O., & Abernethy, N.F., 1999. RiboWeb: An Ontology-Based System for Collaborative Molecular Biology Biology. *IEEE Intelligent Systems*, 14(5), p. 68-76.
Available at: http://citeseer.ist.psu.edu/altman99riboweb.html [accessed 10 June 2007]

Barabási, A.-L., Bonabeau, E., 2003. Scale-free networks. *Scientific American*, 288, p. 60-69.

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J., Slaughter, S. (2003). Is Internet-Speed Software Development Different?, IEEE Software, 20 (6), pp. 70-77, November/December, 2003.

Biskupski, B., Dowling, J., Sacha, J., 2007 Properties and mechanisms of self-organizing MANET and P2P systems. *TAAS*, 2(1).

BRG, Business Rules Group 2003. Business Rules Manifesto - The Principles of Rule Independence.

Chesbrough, H. & Rosenbloom, R.S., 2002. The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies. *Industrial and Corporate Change*, 11 (3), p. 529-555.

Dowling, J., Sacha, J., Haridi, S., 2007. Improving ICE Service Selection in a P2P System using the Gradient Topology. *SASO 2007*, p. 285-288.

Elliot, S. 2002. *Electronic Commerce: B2C Strategies and Models.* Wiley: Chinchester.

Evans, P. B. & Wurster, T. S., 1997. Strategy and the economics of information. *Harvard Business Review.* September-October.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral Dissertation, University of California, Irvine, 2000). Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Glass, R. L. 2005. *It failure rates–70 percent or 10-15 percent?*, IEEE Software, 22(3), p. 112, 110–111

Glass, R. L. 2006. *The standish report: does it really describe a software crisis?*, Commun. ACM, 49(8), p.15–16.

Gruber, T. R. 1993. *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition, 5(2), p. 199-220. Available at: http://tomgruber.org/writing/ontolingua-kaj-1993.htm

Gruber, T. R. 2007. *Ontology of Folksonomy: A Mash-up of Apples and Oranges*. International Journal on Semantic Web and Information Systems, 3(2). Available at: http://tomgruber.org/writing/ontology-of-folksonomy.htm

Hejl, P.M., 1984. Towards a theory of social systems: Self-organisation and self-maintenance, self-reference and syn-reference. In Ulrich and Probst, ed. *Self-Organisation and*

Management of Social Systems: Insights, Promises, Doubts, and Questions. Berlin: Springer-Verlag, 1984, p. 60-78.

Jorgensen, M. and Molokken-Ostvold, K. J. 2006. *How large are software cost overruns? Critical comments on the Standish Group's chaos reports*, Information and Software Technology, 48(4), p. 297–301

Kleppe, J. W. A. and Bast, W. 2003. *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison Wesley.

Linder, J. & Cantrell, S., 2000. *Changing Business Models: Surveying the landscape*. Available at: http://www.accenture.com [accessed 13 July 2007].

List B. and Korherr B. 2006. *An evaluation of conceptual business process modelling languages Symposium on Applied Computing*, in Proceedings of ACM symposium on Applied computing.

Luhmann, N., 1995. *Social Systems.* Stanford CA: Stanford University Press.

Mahadevan, B., 2000. Business Models for Internet based E-Commerce: An anatomy. *California Management Review*, 42 (2).

Malone, T.W., Crowston, K., Herman, G.A., 2003. *Organizing Business Knowledge: The MIT Process Handbook*. The MIT Press: Cambridge, Massachusetts.

Maturana, H. & Varela, F., 1980. *Autopoiesis and Cognition: The Realization of the Living.* Dordecht: Reidel.

Mingers, J., 1994. *Self-Producing Systems: Implications and Applications of Autopoiesis.* New York: Plenum Publishing.

Nijssen, S., 2007. *SBVR: The Common Knowledge Language and The Most Promising Alternative in IT.* Available at: http://www.brcommunity.com/b377.php [accessed 5 October 2007].

OMG 2004. *Business Semantics of Business Rules - Request For Proposal.*

OMG 2007. *Semantics of Business Vocabulary and Business Rules* (SBVR).

Osterwalder, A. 2004. *The Business Model Ontology, A Proposition in a Design Science Approach.* Univertsite de Lausanne Ecole des Hautes Etudes Commerciales. Available at: http://www.businessmodeldesign.com/publications/The%20Business%20Model%20Ontology%20a%20proposition%20in%20a%20design%20science%20approach.pdf [accessed 7 June 2007].

Ovans, A., 2000. Can You Patent Your Business Model?. *Telektronikk*, 2, p. 8-19.

Prahalad, C.K. & Ramaswamy, V., 2000. "Co-opting customer competence," *Harvard Business Review*, January-February, p. 79-87.

SGI, Standish Group International 1994. *The chaos report.*

Swenson K., 2006. *The BPMN-XPDL-BPEL value chain.*

Timmers, P., 1998. Business models for electronic commerce. *Electronic Markets*, 8 (2), p. 3-8.

Timmers, P., 2000. *Electronic Commerce: Strategies and models for business-to-business trading.* Wiley: Chinchester.

Vander Wal, T. 2005. Folksonomy Definition and Wikipedia, vanderwal.net, November 2. Available at: http://www.vanderwal.net/random/entrysel.php?blog=1750

Venkatesh, A., 1999. Postmodernism perspectives for macromarketing: an inquiry into the global information and sign economy. *Journal of Macromarketing*, 19 (2), p. 153-169.

Westland, J.C. & Clark, T.H.K., 1999. *Global Electronic Commerce: Theory and Case Studies*. MIT Press: Cambridge, MA.

Whitaker, R., 1995. Self-Organization, Autopoiesis, and Enterprises. Available at: http://bat710.univ-lyon1.fr/~jmathon/autopoesis/Main.html [accessed 01 October 2007]

White S. A., 2006. *Introduction to BPMN*, IBM Software Group.

Appendix A: Relation with other standards

The discussion about languages for business processes modelling is often focused on the main standards emerged in the market. From one side, the OMG's family of standards, that includes BPMN and the related XPDL (*XML Process Description Language*); on the other side, BPEL (*Business Process Execution Language*) developed by the Workflow Management Coalition.

BPMN and BPEL provide a formal model for expressing executable processes that address all aspects of enterprise business processes. It is widely recognised that BPMN and BPEL can be used to address different needs and in different contexts. As shown in Fig. **Error! Reference source not found.**, the former is closer to the business environment, is aimed at modelling business processes and is more oriented towards business consultants and analysts. On the contrary, BPEL is thought to be closer to technology and it is thus aimed at the execution of processes.

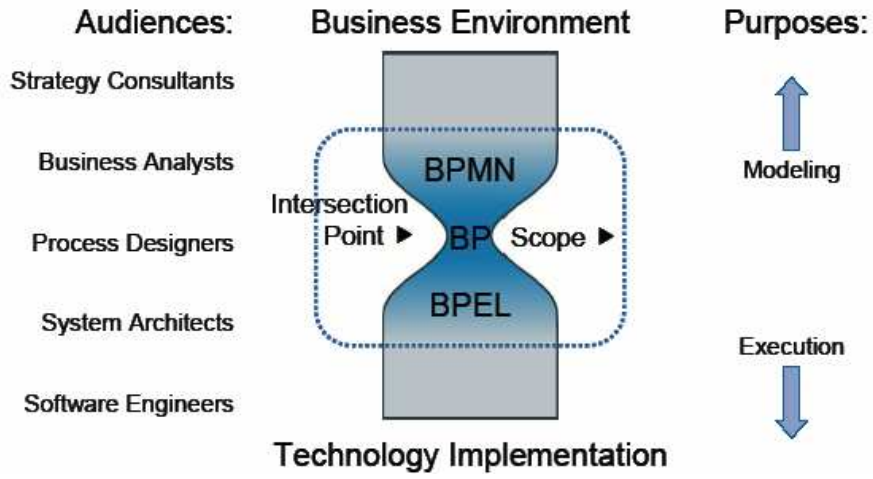


Figure 17. Business Process Modelling Hourglass (White, 2006)

A more interesting discussion is about the choice of XPDL rather than BPEL, based on the assumption that these two standards are direct competitors. Indeed, BPEL and XPDL are entirely different things for entirely different purposes. The main differences are summarised below (Swenson, 2006).

BPEL is an *execution language*. It is a programming language that has variables and operations, that can send and receive SOAP messages, and there is strong support for XML and XML transformation. It has constructs that make it easy to call multiple web services at the same time, and synchronize the results. On the contrary, XPDL is a *process design format*. It is a file format that represents the "drawing" of the process definition. It has X & Y

coordinates and node size. It has a concept of lines, and points along the line that give it a particular path. The nodes and lines have attributes which can specify executable information such as roles, activity descriptions, timers, web service calls, etc. XPDL 2.0 contains extensions in order to be able to represent all aspects of BPMN. The goal is to be able to save and exchange the process diagram (Swenson, 2006).

In other words, the goal of BPEL is to provide a definition of web service orchestration, the underlying sequence of interactions, the flow of data from point to point. Ultimately BPEL is all about bits and bytes being moved from place to place and manipulated. The goal of XPDL is to store and exchange the process diagram. It allows one process design tool to write out the diagram, and another to read it, and for the picture that you see to be as similar as possible. It does not, however, guarantee the precise execution semantics.

Figure 18 describes this different usage. At the top are various design level tools. At the bottom are execution environments. XPDL can be used to carry the design from design tool to design tool. BPEL, XPDL, or other formats might be used to communicate the executable process to the engine. Generally, a vendor specific design tool is necessary to translate the design into an engine specific format (Swenson, 2006).

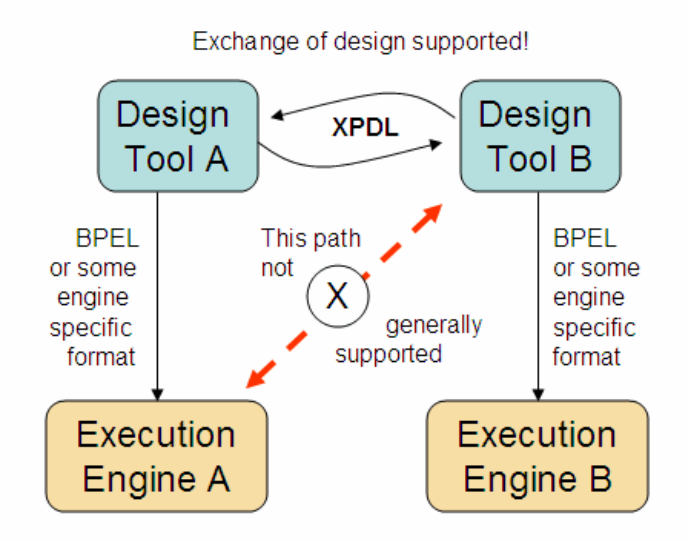


Figure 18. BPEL and XPDL