An ORM based Meta-Schema to Model Business Processes

Harindra Wijesekera
Hewlett Packard Australia

James Sykes
Swinburne University of Technology

An ORM based Meta-Schema to Model Business Processes

Harindra Wijesekera[a], James A. Sykes[b]

[a]Enterprise Integration Practice
Hewlett Packard
Sydney, Australia
e-mail:harindra.wijesekera@hp.com

[b]School of Information Technology
Swinburne University of Technology
Melbourne, Australia
e-mail:jsykes@swin.edu.au

Abstract

An ORM based meta-schema, to represent various kinds of models of business processes, is discussed in this paper. Modeling of an enterprise as a series of business processes is gaining prominence with the emergence of new technologies such as the Internet. Enterprises are moving towards establishing well-described business processes within and across enterprises to include their customers and suppliers. Such integrations should deliver economic efficiencies to all stakeholders involved in a set of common objectives. The suggested meta-schema can be used in the early phases of requirements elicitation to specify, communicate, comprehend and refine various artifacts. This will encourage domain experts and knowledge analysts to work together in describing vital information of each business process and how they interact with each other. This should allow existing business processes to be better documented and also increase the business efficiency of each process by refining them, if required. This meta-schema, in comparison with some of the available techniques, will provide a comprehensive set of concepts to capture various aspects of business processes in a single notation system.

Keywords

Object role modeling, Aspect modeling, View modeling, Business Process Modeling, Meta Modeling, Unified Modeling Language (UML), Event-Driven Process Chains (EPC), Data Flow Diagramming (DFD), IDEF-3

Introduction

Business process modeling (Hammer & Champy 1994, Hammer 1996) has gained popularity in many organizations due to technological developments such as the Internet. The Internet (Berners-Lee 1999, W3C 2001) can be used to achieve advances in organization wide processes and improve economic efficiency by providing seamless integration. When developing such business modules, it is important that software development activity is considered as meeting a set of requirements. The need to clearly conceptualize business processes is such a requirement.

The meta-schema introduced in this paper is defined as a collection of meta-data types, their associations and constraints. It is designed to allow business domains to be defined at the type level and forms the underlying language. This schema allows both knowledge analyst and domain expert to jointly create a conceptual model of the Universe of Discourse (UoD). The primary aim is to provide constructs that are useful in capturing all requirements. It conforms to the 100% principle described in (ISO 1982, ISO 1996). The 100% principle requires representing all aspects conceptually. This schema also adheres to the conceptualization principle (ISO 1982, ISO 1996) by recording only non-implementation details. It strongly encourages the synchronization of mental models of knowledge analyst and domain experts at a conceptual level.

Gupta and Sykes [2001] have developed a framework which draws on theories from philosophy, linguistics, cognitive science and conceptual modeling. The framework focuses on developing conceptual models at an ontological level thereby allowing domain experts and analysts to explicitly describe concepts. The framework consists of 7 processes. They include; (1). User's concrete system, (2). User's conceptual system, (3). Informal symbolic system shared by user and analyst, (4). Analyst's conceptualization of the informal symbolic system, (5). Analyst's interpretation of the UoD as a concrete system, (6). Analyst's re-conceptualization in terms of CSMF constructs and (7). Analyst's formal symbolic system.

In such a system of processes, when moving from one process to another there could be a loss of knowledge, hence creating gaps (with possible varying interpretations) between them. This loss will be most significant if the analyst is not familiar with the UoD of concern or similar UoDs or has no significant experience in the subject matter. As mentioned earlier, the proposed meta-schema is intended to allow domain experts and analysts to synchronize their mental models directly into the meta-schema, thereby eliminating the need for creating an informal symbolic form, and to focus on developing a formal symbolic system (process 7 of the framework) .

The meta-schema has borrowed various concepts from well-developed techniques such as the Unified Modeling Language (UML) (Fowler 2000, Booch et al 1999, Quatrani 1998, Alhir 1998), Data Flow Diagramming (DFD) (Gane 1989), Event Driven Process Chains (EPC) (Scheer 1998, 1999), and Integrated Definition Language (IDEF-3) (Menzel et al 1995, IDEF3 Process Flow). These concepts are brought together into a single formalism using natural language statements. Achieving a single formalism is the main motivation for development of the schema. It will result in the creation of multiple views, including: business processes, business process hierarchies, flow of information among and within business processes, events, event-driven function hierarchies, formation of rules and the flow of rules between business processes. The schema will allow analysts to import models that have been developed using the above techniques, provided the models belong to the same problem domain. Because an integrated formalism is being used to store and maintain models, possibilities exist for developing interactions among concepts imported from multiple techniques. Once imported the Analyst could reverse engineer the source model from the meta-schema provided the model is mapped as-is, meaning that no additional interactions have been defined in the meta-schema. Even though separation of models is possible within the meta-schema, it is not encouraged, as this schema is geared towards maintaining a single formalism. A notation system is not suggested as various presentation formats could be used.

In section 2, a business process modeling review is conducted with the emphasis on UML, DFD, EPC and IDEF-3. Also, requirements for a business process modeling activity are discussed. The ORM based meta-schema is then presented in section 3, followed by a mapping process in section 4. Concluding remarks are then discussed in Section 5.

Process Modeling Review

A variety of process-oriented techniques exist in the form of UML (activity diagrams and use-case diagrams are part of this technique), DFD, EPC and IDEF-3. Many variations of the above techniques are also available. However, only the main techniques are being discussed here.

In the authors' opinion, none of the above techniques on their own provide required concepts for comprehensive process modeling. This has been one of the catalysts for efforts towards creating a new meta_schema for process modeling, others being the need for a single formalism and the use of natural language.

UML provides various diagrams to describe business detail. These diagrams tend to focus on areas such as class definitions, interaction descriptions etc. However, a lack of integration in the areas of events, their interactions and possible flow of information is apparent. Also, concepts should be available for better integration among business details in order to represent the domain as a whole. The difficulties associated with linking business rules and their interactions with activity diagrams are such examples. The use of various diagramming techniques can also reduce the comprehensibility by increasing a level of complexity for the user. A common notation system and a natural language based technique should provide a better foundation for a good modeling environment.

The DFD technique allows domain experts and analysts to form well-described processes, decompositions and flows of data. However, it does not provide concepts to describe business events, concurrency rules, etc. that are relevant to describe business behavior.

The EPC technique (Bancroft et al 1998, Scheer 1999), in comparison with UML and DFD provide concepts to form business events and functions. It also allows the use of conditions that describe business behavior. However, it does not provide facilities to specify business rules (Wijesekera & Sykes 2001) that may be discovered within higher and lower level business processes that describe the scope and boundary of the system.

Modeling of business processes is not a trivial activity. Modeling provides a foundation to the system by describing the scope and boundary coupled with the required functionality as specified by the domain experts.

A business process modeling activity should facilitate the following requirements:
- Ability to declare Business Processes.
- Ability to declare Information Stores.
- Ability to declare Entity Types that represent business entities.
- Ability to specify Rule Types.
- Ability to describe Links, Connections or associations between above stated concepts.

Business process modeling (BPM) should be based on a simple and flexible notation system. BPM should allow business facts to be described clearly in a coherent and cohesive manner. It should also encourage the use of natural language in order to increase expressiveness and reduce ambiguity. Other requirements that need to be satisfied in addition to basic concepts and structure are:

- Ability to describe fact types and rule types that support associations between Business Processes, Information Stores and Entity Types.
- Ability to describe business process de-composition.
- Ability to further decompose business processes into required activity or functions.
- Ability to declare a direction for each link or connection. A direction should only contain the constraints of “From” or “To” when leaving or entering a concept. A direction provides the ‘flow of control’.
- The ability to provide various views, for the business process model, thus increasing comprehensibility among stakeholders who are involved in the process of modeling.

ORM based Business Process Meta-Model

The ORM based business process meta-model consists of various fact types, as shown in Figure- 1. The activity of modelling should commence by providing high-level process descriptions that determine the scope and boundary of the system. The required decompositions and interactions such as business events, functions and rules are also to be described.

Initially a process will be defined. A process can be defined as a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer (Hammer & Champy 1994). An enterprise, irrespective of its size, may contain ten or less principal business processes (Hammer & Champy 1994, Hammer 1996). These processes are set towards converting various inputs to outputs in an efficient and effective manner. When more than one principal process is involved, interactions among them should be specified with the use of Links.

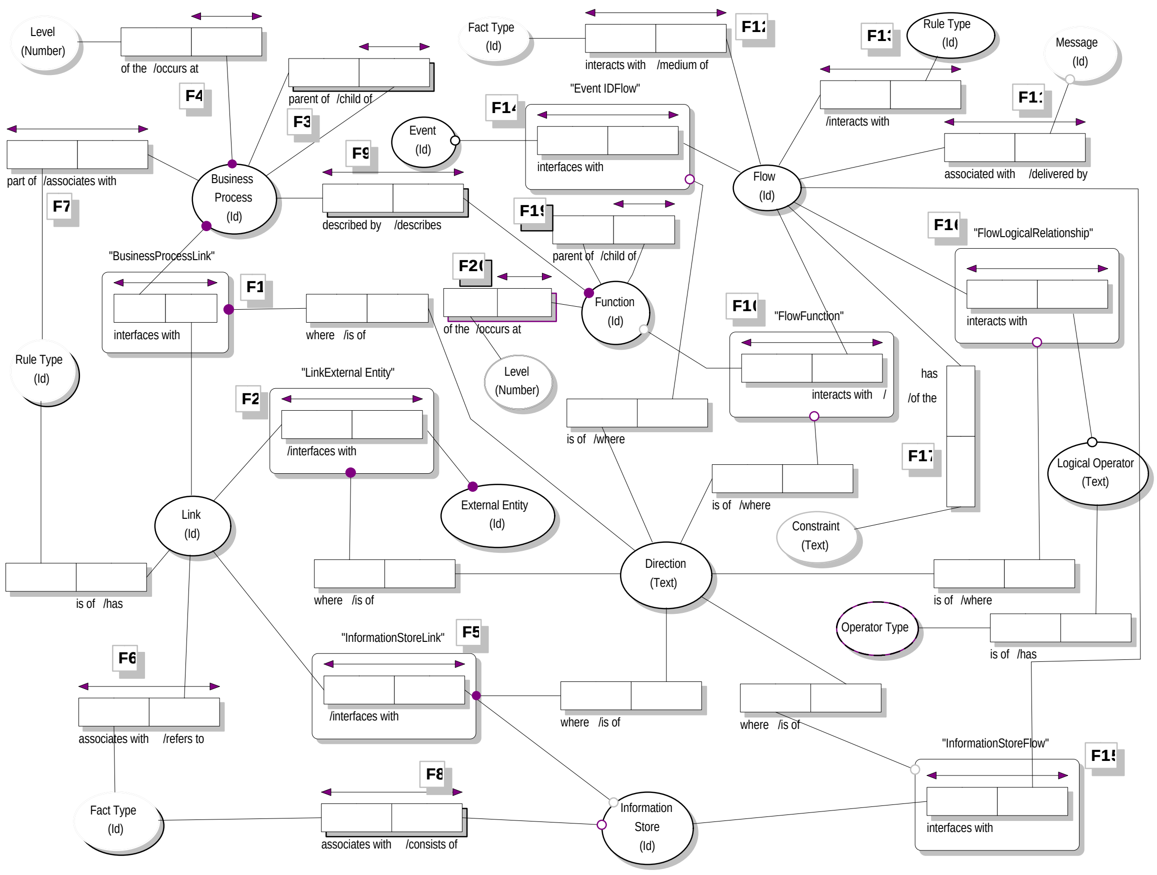


Figure- 1 Business Process Meta-Model

Links connect various business processes together and describe the flows of business activity that occur between them. With Links the control flow is assumed, i.e. one business process is required to be completed before entering the next. Links also require directions to indicate the flow of inputs and outputs among them.

Meta Fact type (F1):
Business Process (Id) ‘’ …interfaces with Link (Id) ‘’ …where Direction {To Link, From Link}

Example:
Business Process (Id) ‘P1’ …interfaces with Link (Id) ‘L1’ …where Direction {To Link}
Business Process (Id) ‘P2’ …interfaces with Link (Id) ‘L1’ …where Direction {From Link}

Business processes may interact with entities that are external to the boundary of the system. They are commonly known as ‘external entities’ (Gane 1989, Jacobson et al., 1995). It could be a person, organization or process that is not managed by the system which has influence on the business to determine its success or failure to a degree. Customer (E1) interacting with order-to-payment process (P2) is an example.

Meta Fact type (F2):
External Entity (Id) ‘’ …interfaces with Link (Id) ‘’ …where Direction {To Link, From Link}

Example:
External Entity (Id) ‘E1’ …interfaces with Link (Id) ‘L2’ …where Direction {To Link}

A principal business process is a large collection of many other business processes. It is, therefore, necessary to decompose it into subsidiary business processes. Business process decomposition is required to discover the composition of business, reduce complexity, provide clear communication and increase comprehensibility among domain experts and knowledge analysts.

Meta Fact type (F3):
Business Process (Id) '' …parent of …child of Business Process (Id) ''

Example:
Business Process (Id) 'P2' …parent of …child of Business Process (Id) 'P4'

Each sub-process will occur at a certain Level. Even though a level can be derived, it would be a good practice for stakeholders to identify, discuss and specify the level of each business process in the hierarchy. Thus allowing stakeholders to formally recognise processes and the dependencies among them is important.

Meta Fact type (F4):
Business Process (Id) '' …occurs at …of the Level (Number) ''

Example:
Business Process (Id) 'P2' …occurs at …of the Level (Number) '0'

Business processes may also require interactions with Information Stores. They can either receive items from Information Stores and/or send items to it. Information store is defined as an object type that stores relationships of various other object types. They may include fact types (Halpin 1995, 2002) and sentence types (Wijesekera & Sykes 2001). For example an information store can be used to maintain customer level details such as name and address.

Meta Fact type (F5):
Information Store (Id) '' …interfaces with Link (Id) '' …where Direction {To Link, From Link}

Example:
Information Store (Id) 'I1' …interfaces with Link (Id) 'L3' …where Direction {From Link}

Business Process, Information Store and Link, as specified above, can be associated with other entity types, values, fact types, sentence types and rule types. They will help describe valuable information inherent with business processes and transformations that occur inside them. The following are some examples of its usage.

Meta Fact type (F6):
Link (Id) '' …refers to …associates with Fact Type (Id) ''

Example:
Link (Id) 'L3' …refers to …associates with Fact Type (Id) 'F1'

Links may refer to various other object types to provide information about items being transported, either being sent or received.

It may be necessary for domain experts to describe business rule types that are being used and/or discovered within processes. Such rules can be specified as and when they are discovered during the analysis and requirements elicitation phase. Discovery of credit rating rules in a financial application is an example.

Meta Fact type (F7):
Business Process (Id) ‘’ …associates with …part of Rule Type (Id) ‘’

Example:
Business Process (Id) ‘P2’ …associates with …part of Rule Type (Id) ‘R1’

Information stores may consists of one or more fact types, sentence types and rule types. Such requirements can be described with the use of following fact type.

Meta Fact type (F8):
Information Store (Id) ‘’ …consist of …associates with Fact Type (Id) ‘’

Example:
Information Store (Id) ‘I2’ …consist of …associates with Fact Type (Id) ‘F2’

Further decomposition of business processes may result in lower level details. Such lower level details should be encapsulated into functions in order to improve clarity. These functions interact with various other object types as described in the text following. A ‘function’ can therefore be described as consisting of an action(s) that is needed to describe business requirements at a lower level. Some functions may consists of a single algorithmic step, where as others may be formed to represent a series (bundle) of steps. It is also possible to decompose functions. Function decomposition primarily results in simplification of algorithmic functions for ease of use, understanding and extensibility.

Meta Fact type (F9):
Business Process (Id) ‘’ …described by …describes Function (Id) ‘’

Example:
Business Process (Id) ‘P6’ …described by ...describes Function (Id) ‘Fn1’

Each Function will be associated with flows in describing how they interact with other object types. The object type of ‘Flow’ (Meta Fact type F10 below) is being used to describe interactions that occur at function level, in contrast to the use of ‘Link’ object that describe interactions between processes. However, ‘Flow’ and ‘Link’ objects are similar in nature, with respect to control flow and transition.

Functions are decomposed to increase clarity and reduce clutter, even though their use is entirely dependent on stakeholders.

Meta Fact type (F19):
Function (Id) ‘’ …parent of …child of Function (Id) ‘’

Meta Fact type (F20):
Function (Id) ‘’ …occurs at …of the Level (Number) ‘’

Meta Fact type (F10):
Function (Id) ‘’ …interacts with Flow (Id) ‘’ …where Direction {To Flow, From Flow}

Example:

Function (Id) ‘Fn1’ …interacts with Flow (Id) ‘O1’ …where Direction {To Flow}

The above fact type is similar to what has been described for business processes. Also, object types that a function would participate with may be in the form of events, messages, rules etc.

Flow(s) may be associated with Message types. Message types can be used to describe information that may be sent to or received from objects. All messages should be well defined in order to clearly communicate the underlying meaning.

Meta Fact type (F11):
Flow (Id) ‘’ …associated with …delivered by Message (Id) ‘’

Example:
Flow (Id) ‘O3’ …associated with …delivered by Message (Id) ‘M1’

The associations between flow and other object types such as fact types, sentence types and rule types can be described as follows.

Meta Fact type (F12):
Fact Type (Id) ‘’ …interacts with …medium of Flow (Id) ‘’

Example:
Fact Type (Id) ‘F3’ …interacts with …medium of Flow (Id) ‘O2’

Meta Fact type (F13):
Rule Type (Id) ‘’ …interacts with Flow (Id) ‘’

Example:
Rule Type (Id) ‘R2’ …interacts with Flow (Id) ‘O6’

Events are associated with Flows. Flows provide the link between Events and Functions.

Meta Fact type (F14):
Event (Id) ‘’ ...interfaces with Flow (Id) ‘’ …where Direction {To Flow, From Flow}

Example:
Event (Id) ‘EV1’ ...interfaces with Flow (Id) ‘O1’ …where Direction {From Flow}
Event (Id) ‘EV1’ ...interfaces with Flow (Id) ‘O2’ …where Direction {To Flow}

It is also possible to associate Information Stores with various Flows, as described below, to communicate Function level requirements.

Meta Fact type (F15):
Information Store (Id) ‘’ …interfaces with Flow (Id) ‘’ …where Direction {To Flow, From Flow}

Example:
Information Store (Id) ‘I2’ …interfaces with Flow (Id) ‘O5’ …where Direction {From Flow}

The control flow of a process may require to be branched in order to describe the underlying business requirements. A ‘ Logical Operator’ can be used to describe the convergence and divergence of process flows. Convergence allows multiple paths to become one and divergence allows one path to break into multiples. All logical operators should belong to one of the following types.

(1). AND, (2). inclusive-OR, (3). exclusive-OR, (4). synchronize-AND, (5). synchronize-OR

When a process is branched the 'AND' operator can be used to ensure that converging items or sub processes are complete.

The inclusive-OR logical operator can be used when one or more activity is required for processing. This may be useful when processes are converged or diverged. The exclusive – OR logical operator can only be used to describe mutually exclusive conditions.

Synchronize–AND and OR conditions are similar to AND, inclusive-OR operators, difference being the synchronize condition. The synchronize condition requires all activities to synchronize by commencing or concluding at the same time.

Meta Fact type (F16):
Logical Operator (Text) '' …interacts with Flow (Id) '' …where Direction {To Flow, From Flow}

Example:
Logical Operator (Text) {L1} …interacts with Flow (Id) 'O2' …where Direction {From Flow}

Meta Fact type (F17):
Logical Operator (Text) '' ...has ... is of Operator Type {AND, IOR, XOR, SAND, SOR}

Example:
Logical Operator (Text) {L1} ...has ... is of Operator Type 'AND'

It is also possible to specify constraints on flows. These constraints are in addition to rule types that could be specified on flows. Constraints may be in the form of time restrictions (e.g., "should be completed within 3 minutes"), directional constraints (e.g., "source function to be completed prior to destination") etc. Fact type F18 allows stakeholders involved in requirements elicitation to specify constraints as a piece of text.

Meta Fact type (F18):
Flow (Id) '' …has ...of the Constraints (Text) ''

Guidelines for using the meta-schema

A set of guidelines for using the meta-schema to describe a domain of interest is as follows:

Step 1: Knowledge analyst and domain expert need to discuss the domain in general terms. This will provide a forum to understand parties involved, their roles, profitability criteria of the enterprise, goals, objectives and business processes.

Step 2: Capture business processes. Each business process once discovered should be captured as an object type. When multiple processes are discovered their interactions should be considered and described using links (Fact type F1). Even though a top-down like procedure is being described here, it is not necessary to force such a discipline. It is also possible to discover details at any level first and then scope such details with the use of business processes thereafter. Step 2 usually follows Step 1.

Step 3: Do business processes require further simplification? Further simplification of processes should provide clarity to the model. Use fact types F3 and F4 to describe them. The interactions among newly created processes should be described using fact type F1. Step 3 follows Step 2. However, if further decomposition result in functions (lower level actions that are needed to describe business requirements), go to Step 11.

Step 4: Identify external entities. Typical external entities are customers, taxman, shareholders, etc. Describe external entities and their interactions with the use of fact type F2. External entities typically interact with business processes. Step 4 usually follows Step 1.

Step 5: Further describe external entities by specifying their attributes through facts. This step may follow Step 4.

Step 6: Continue to capture facts of the business, as and when discovered. Even though described here this may co-exist with any other step(s), as fact gathering is a continuous process.

Step 7: Do relationships of various object types require to be grouped? Think of grouping them for clarity and ease of reference. Information stores describing customer and product details are such examples. Describe them using fact type F5. Also, specify fact types that are contained by information stores using F8. Follows step 1.

Step 8: Describe the ways in which information stores may interact with other object types such as business processes. Use fact types F5 and F6 for this purpose. Follows step 7.

Step 9: Describe rules that are available in the domain. Use the procedure documented in (Wijesekera & Sykes, 2001) to capture them. This step may occur any time during the process. However, following Steps 1, 2, 3 is likely.

Step 10: Do rule types associate (used by and/or generated by) with business processes? If so use fact type F7 to specify them. This follows Step 9.

Step 11: Use fact type F9 to describe functions that exist in the domain. This step follows Step 3. Like business processes, and for similar reasons, further decomposition is applicable for functions. Use fact types F19 and F20, if applicable.

Step 12: Do functions described above interact with other object types, such as functions? Usually they do. Specify such interactions using Flows (Fact type F10). If flows are constrained, use F18 to describe them. This step follows Step 11.

Step 13: Are there messages (information sent or received) to be associated with flows? Describe using fact type F11. This step usually follows Step 12.

Step 14: Describe fact types that interact with flows using F12. Usually follows Steps 12 and 13.

Step 15: Rules may be discovered at the function level (similar to Step 9). Describe them using F13. This step follows Step 11.

Step 16: Identify events that are associated with functions, event ‘Order Received’ prior to function ‘Process Order’, for example. Describe them using F14. This step follows Step 11.

Step 17: Do information stores require interactions with functions? Describe them using fact type F15. Information stores should be described at Step 7.

Step 18: Is flow control (convergence and/or divergence of functions) of a process required to describe business functionality. Use fact types F16 and F17. This step may follow Step 11.

The Mapping Process

The process that has to be adopted when transforming constructs from the source to the target will now be described.

A process of this nature should provide the following benefits:

- Ability to map various techniques into a single environment.
- Once assimilated the ability to extend the model by refining and gathering further requirements through a common formalism.
- The ability to combine requirements that are gathered through various techniques in a seamless fashion.
- The use of natural language to define and extend the model.
- Ability to re-create the source from the target. This is only possible if information from source is represented as-is.

UML to Meta Schema Mapping

The UML provides various kinds of diagrams. Activity diagrams are recognized as useful for describing processes (Quatrani 1998, Alhir 1998, Fowler 2000). Additionally use-case diagrams are identified by various authors as useful for providing contextual level requirements. Use-case diagrams render the users’ view of a system. Also, a textual description should be provided for each use-case. Both use-case and activity diagrams are considered in this work due to its orientation towards specifying business functionality.

UML activity diagrams are scoped by high-level business processes, thus encouraging process hierarchies and stakeholder interactions at the process level. Use-cases only focus on user functionality of a system and therefore should be mapped as functions and flows. Actors can be mapped as External Entity Types. The connections that exist between actors and use cases are drawn as associations. A textual description for each Flow, if required, will be used to provide details such as “include”, “extend” and “generalization”. The meta-schema is extensible and therefore will allow stakeholders to create new fact types as and when required to represent business requirements.

Activity diagrams comprise various constructs. They are described in the following table. Also, the ways in which they are mapped into the meta-schema is described.

No	**Construct**	**Specification**	**Meta-schema Mapping**
1	Activity State Sub-activity State	UML 1.3 UML 1.4	Function (Function Type can be used as an attribute to further differentiate states, if required)
2	Call State	UML 1.4	Function, (Function Type can be used as an attribute to further differentiate states, if required)
3	Action State	UML 1.3, 1.4	Function, (Function Type can be used as an attribute to further differentiate states, if required)
4	Decisions Branching	UML 1.4 UML 1.3	Logical Operator (the use of AND, OR operators), Rule
5	Swimlanes	UML 1.3, 1.4	Business Process
6	Action-object Flow	UML 1.4	Fact types and Flows
7	Object Flow	UML 1.3	Fact types and Flows
8	Control Icons	UML 1.4	Event and Flow
9	Transitions	UML 1.3, 1.4	Flow
10	Synchronized States Forking and Joining	UML 1.4 UML 1.3	Logical Operator and Flow
11	Dynamic Invocation Dynamic Concurrency	UML 1.4 UML 1.3	Function Type and Flow
12	Conditional Forks Conditional Threads	UML 1.4 UML 1.3	Event
13	Initial State	UML 1.3, 1.4	Event (Start)
14	Final State	UML 1.3, 1.4	Event (End)
15	Guard Expressions	UML 1.3, 1.4	Rule

An Action state can be represented by a Function. For example; ‘Receiving an Order’ has been viewed as an action state in (Fowler 2000, figure 9.1). A call state will also be described as a function. Activity and sub activity states will be represented by functions that could be further decomposed into functions representing activity states, actions states or call states. A function is commonly used to describe lower level details.

In the authors’ opinion, a function where possible should be described as following an event or resulting in an event. Events should enhance comprehensibility by highlighting states being achieved or pre-condition(s) that are required to be satisfied prior to executing a function. For example, describing an ‘Order Received’ event prior to describing the function ‘Process Order’.

A Fork has one incoming transition and several (simultaneously executed) outgoing transitions. It can be represented with the use of ‘AND’ Logical Operator. The use of ‘AND’, in comparison with Fork is explicit and improves verbalization and comprehensibility.

A Branch can be represented by an “exclusive OR” Logical Operator. It describes mutually exclusive activities. In general, it is possible to precede an exclusive-or condition with an Event to provide high clarity to the process. If order total is greater than 5000, ‘Process Order’ otherwise ‘require Supervisor Signature’ should be represented with the use of an Event, “exclusive OR” condition and Functions.

Event – Order Total is known
Condition, “exclusive or” – Order total greater than 5000
If “yes”- Function to ‘Process Order’
If “no” – Function for ‘require Supervisor Signature’

The suggested meta-schema, in comparison with Activity Diagrams, will allow users to define events that occur in business in a clear and concise manner. The completion of ‘Overnight Delivery’ and ‘Regular Delivery’ activities to be followed by events such as ‘Overnight Delivery Completed’ and ‘Regular Delivery Completed’ are such examples. Join allows merging of several activities that are in progress, hence will be represented by “inclusive-OR” or “AND” Logical Operators.

A conditional thread associates a condition to a function. It can be represented by an Event followed by a Function, where the event represents the condition and the process represents a function. Conditional threads require a piece of text to be specified in connecting line.

Synchronization states are used in activity diagrams to ensure that parallel activities are concluded. The “and” logical operator or the “and” Logical operator combined with Event(s) can be used to represent such requirements.

An ‘asterisk’ has been used to describe dynamic concurrency. It allows for iterative activity to be described without constructing a loop. The suggested approach recommends the use of function types. Functions can be of type; call state, action state, decomposed activity and repetitive functions. From a modeling viewpoint, it is beneficial to describe repetitive functions as activities in order to improve clarity. It is possible for such activities to be decomposed.

Swim lanes are used to describe people, divisions, departments, etc. that are responsible for activities. As described in (Fowler 2000), if swim lanes are to be used it is necessary to group activity diagrams into vertical zones separated by lines. Also, the swim lane approach tends to organize the enterprise as a hierarchy, hence creating “silos”. A business-process-centric approach is recommended in this paper. Enterprises are recognized as consisting of groups of processes (Hammer & Champy 1994, Hammer 1996).

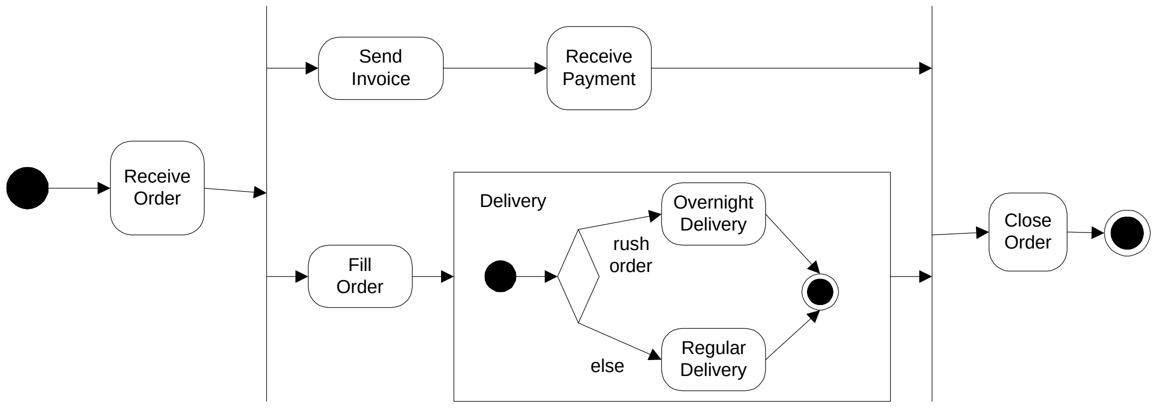


Figure- 3 Sample UML Activity Diagram (9.1 of Fowler 2000)

The sample UML activity diagram, described above, is mapped onto equivalent meta types from the proposed meta-schema.

The activity 'Receive Order' is mapped as a Function. When an order is received, this function will be followed by other functions such as to 'Fill Order' and 'Send Invoice'. Above functions can be performed on a parallel basis and independent of each other, hence 'and' logical operator is used to describe the requirement.

Function (Id) 'Receive Order' ...interacts with Flow (Id) 'FI1' …where Direction {To Flow}
Flow (Id) 'FI1' …interacts with Logical Operator (Text) {L1} …where Direction {From Flow}
Flow (Id) 'FI2' …interacts with Logical Operator (Text) {L1} …where Direction {To Flow}
Flow (Id) 'FI3' …interacts with Logical Operator (Text) {L1} …where Direction {To Flow}
Logical Operator (Text) {L1} ...has ... is of Operator Type 'AND'
Function (Id) 'FillOrder' …interacts with Flow (Id) 'FI2' …where Direction {From Flow}
Function (Id) 'SendInvoice' …interacts with Flow (Id) 'FI3' …where Direction {From Flow}

The orders can be fulfilled as an 'Overnight Delivery' or as a 'Regular Delivery' activity. It is possible to describe overnight and regular delivery activities as part of a decomposition of a function. However, in this example they are expanded onto the same level.
An 'exclusive or' logical operator is used to describe the mutually exclusiveness of these activities

Function (Id) 'FillOrder' …interacts with Flow (Id) 'FI4' …where Direction {To Flow}
Flow (Id) 'FI4' …interacts with Logical Operator (Text) {L2} …where Direction {From Flow}
Logical Operator (Text) {L2} ...has ... is of Operator Type 'XOR'
Flow (Id) 'FI5' …interacts with Logical Operator (Text) {L2} …where Direction {To Flow}
Flow (Id) 'FI6' …interacts with Logical Operator (Text) {L2} …where Direction {To Flow}
Function (Id) 'OvernightDelivery' …interacts with Flow (Id) 'FI5' …where Direction {From Flow}
Function (Id) 'RegularDelivery' …interacts with Flow (Id) 'FI6' …where Direction {From Flow}

The completion of the above activities will be notified with events such as the following.

Function (Id) 'OvernightDelivery' …interacts with Flow (Id) 'FI7' …where Direction {To Flow}
Function (Id) 'RegularDelivery' …interacts with Flow (Id) 'FI8' …where Direction {To Flow}

Overnight delivery and regular delivery functions are mutually exclusive. The logical operator L3 is used to describe this fact.

Logical Operator (Text) {L3} …interacts with Flow (Id) 'FI7' …where Direction {From Flow}
Logical Operator (Text) {L3} …interacts with Flow (Id) 'FI8' …where Direction {From Flow}
Logical Operator (Text) {L3} ...has ... is of Operator Type 'OR'

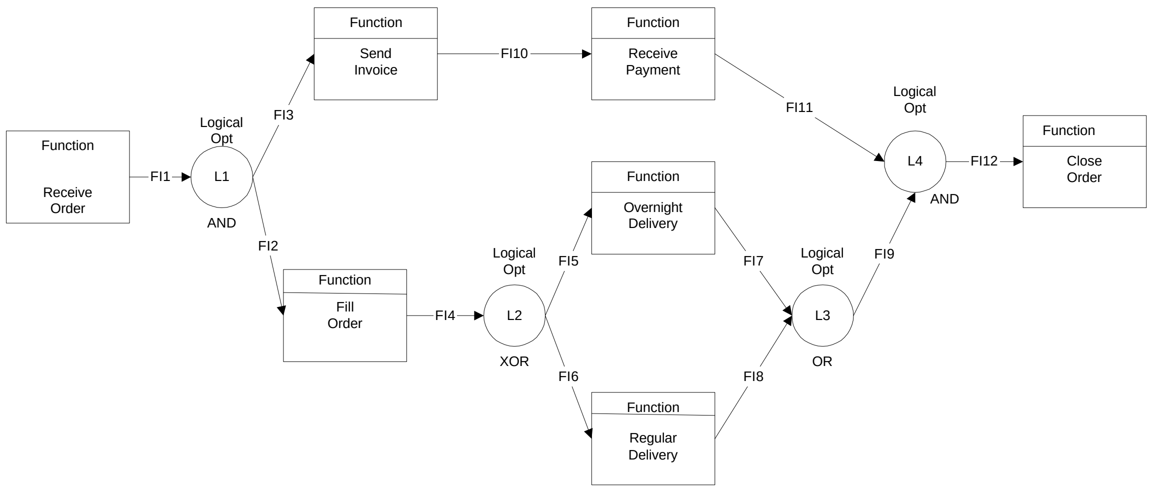
The order will be closed when goods are delivered and payment is received. The logical operator L4 ensures that parallel activities are complete.

Logical Operator (Text) {L3} …interacts with Flow (Id) 'FI9' …where Direction {To Flow}
Logical Operator (Text) {L4} …interacts with Flow (Id) 'FI9' …where Direction {From Flow}
Logical Operator (Text) {L4} ...has ... is of Operator Type 'AND'

Function (Id) 'SendInvoice' …interacts with Flow (Id) 'FI10' …where Direction {To Flow}
Function (Id) 'ReceivePayment' …interacts with Flow (Id) 'FI10' …where Direction {From Flow}
Function (Id) 'ReceivePayment' …interacts with Flow (Id) 'FI11' …where Direction {To Flow}
Logical Operator (Text) {L4} …interacts with Flow (Id) 'FI11' …where Direction {From Flow}

Logical Operator (Text) {L4} …interacts with Flow (Id) 'FI12' …where Direction {To Flow}
Function (Id) 'CloseOrder' ...interacts with Flow (Id) 'FI12' …where Direction {From Flow}

A diagrammatic representation of the above fact types is shown below. The symbols used in this diagram are simple, hence easily comprehensible. A circle represents a logical operator. Functions are described using boxes and arrows indicate flows.



Conclusions

A uniform, natural language based, ORM meta-schema for business process modeling is introduced in this paper. Various concepts are combined, including ones from well-defined techniques, to provide the required facilities to model business processes. The meta-schema, as described in Section 3.0, provides various views of the system. They include business process views, rule view, function view and event views.

The proposed meta-schema is extensible due to the nature of its underlying architecture. This means new concepts can be introduced and their associations with existing or new concepts can be clearly defined. Also, new concepts when required can be introduced into the schema without impacting on constraints already in place.

Section 4.0 described the process to be adopted when transforming models that have been developed with UML activity diagrams and use cases. As stated, such models can be successfully transformed into an integrated view. Even though it is now possible to enhance such models with events, rules etc., for better representation, in this paper they have been transformed as-is.

The new meta-schema will assist both domain experts and knowledge analysts to co-operatively create models by contributing new facts and refining existing ones. It should also provide a clear description of the model, thereby improving communication among stakeholders and increasing comprehensibility.

The inability to cater for various business concepts and their interactions in a unified manner has been recognized as a major deficiency in most modeling languages during the investigation process. This work is an attempt towards reducing this deficiency, by integrating various concepts from other modeling languages into a single formalism.

Requirements, as described in Section 2.0, are met by providing concepts to define Processes, Information Stores, Entity Types and associations among them. It also provides concepts to describe fact types and rule types that are required to support businesses. Decomposition of processes is supported with a view to clearly describe the scope and boundary of the system. At a certain level in the decomposition hierarchy business processes will be defined as a series of events and functions in order to describe internal processing.

The meta-schema described here requires tool support to further its potential in the requirements elicitation process.

References

Alhir, S. S (1998), ‘UML in a NUTSHELL’, O’Reilly & Associates

Bancroft, N. H, Seip, H., Sprengel, A (1998), ‘Implementing SAP R/3 – 2nd edition’, Manning Publications Co

Berners-Lee, Tim (1999), ‘Weaving the Web’, Orion Business

Booch, G, Rumbaugh, J, Jacobson, I (1999), ‘The Unified Modeling Language User Guide’, Addison Wesley

Fowler, M (2000), ‘UML Distilled: a brief guide to the standard object modeling language – 2nd Edition with Kendall Scott’, Addison Wesley

Gane, C, (1989), ‘Rapid System Development – using structured techniques and relational technology’, Prentice Hall

Gupta, P. & Sykes, J. A. (2001). , ‘The conceptual modelling process and the notion of a concept’, In Rossi, M. & Siau, K. (Eds.), Information modelling for the new millennium, Idea Group Publishing

Halpin, T (1995), ‘Conceptual Schema & Relational Database Design 2nd Edition’, Prentice Hall

Halpin, Terry (2002), 'An ORM metamodel of Information Engineering', Journal of Conceptual Modeling, February 2002, Issue 18, www.inconcept.com/jcm

Hammer, M., Champy, J (1994), 'Reengineering the Corporation: a manifesto for business revolution', Allen & Unwin,

Hammer, M (1996), 'Beyond Reengineering', Harper Collins Publishers

IDEF3 Process Flow and Object State Description Capture Method Overview, http://www.idef.com/idef3.html

ISO (1982), Concept and Terminology for the Conceptual Schema and Information Base, International Organisation for Standardisation, Publication no. ISO / TC97 / SC5 - NIAM 695, van Griethuysen, J. J. (ed.) Switzerland

ISO (1996), Information Technology (IT) Conceptual Schema Modelling Facilities (CSMF), International Organisation for Standardisation, Publication no. ISO / IEC / SC21 WG3 N2039 (first edition)

Jacabson, Ivar., Ericsson, Maria., Jacobson, Agneta (1995), 'The Object Advantage Business Process Reengineering with Object Technology' Addison Wesley

Menzel, Christopher P., Painter, Painter, Michael K., de Witte, Paula S., Blinn, Thomas., Perakath, Benjamin (1995), 'Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report', Knowledge Based Systems Incorporated, One KBSI Place, 1500 University Drive East, College Station, Texas 77840-2335

Quatrani, Terry (1998), 'Visual Modeling with Rational Rose and UML', Addison Wesley

Scheer, A, W (1998), 'ARIS – Business Process Frameworks, 2nd Edition', Springer-Verlag

Scheer, A. W (1998), 'Business Process Engineering – Study Edition, Reference Models for Industrial Enterprise', Springer-Verlag

Scheer, A, W (1999), 'ARIS – Business Process Modeling', Springer-Verlag

W3C (2001), W3C XML Schema Part 0- Primer, http://www.w3.org/TR/xmlschema-0/

Wijesekera, H, Sykes, J (2001), 'An Integration of Rule Types and XML', 12th Australasian Conference on Information Systems, Coffs Harbor, Sydney, Australia