Proceedings of the 5th WSEAS Int. Conf. on APPLIED INFORMATICS and COMMUNICATIONS, Malta, September 15-17, 2005 (pp233-238)

From Business World to Software World: Deriving Class Diagrams from Business Process Models

WARARAT RUNGWORAWUT[1] AND TWITTIE SENIVONGSE[2]
Department of Computer Engineering, Chulalongkorn University
254 Phyathai Road, Pathumwan, Bangkok 10330 THAILAND

Abstract: - Business world and software world coexist and interface when there is a demand to implement an information system in business environment. People in business world have more knowledge in business processes and logics of the information system while people in software world have more expertise in building software artifacts. This paper proposes a guideline to help software designers to build a software model, i.e. a UML class diagram in this case, from a business process model, i.e. a BPMN diagram which is given by a business analyst. The guideline borrows the idea of object-oriented domain analysis to identify UML classes from a business process model and enhance by knowledge about the application domain such as domain-specific patterns and other kinds of semantics. At this stage, software designers can follow this guideline to manually create class diagrams, but it is expected that the guideline can be automated in parts and will complement the concept of OMG's MDA by supporting the building of its PIM.

Key-Words: - Business process model, Software Model, Business Process Modeling Notation, UML Class Diagram, Domain Analysis, Platform-Independent Model, Model Driven Architecture

1 Introduction

OMG's Model Driven Architecture (MDA) is the base architecture for software development with a number of modeling standards, which will be used to create software models, and turns the models into engineering artifacts [1]. Three steps are at the core of MDA. First, Platform-Independent Model (PIM) is created to model the business logic of the software. It is at the high level of abstraction and is independent of any implementation technology. Second, PIM is automatically or semi-automatically mapped into one or more Platform-Specific Model (PSM) which still models the business logics but with more details on a specific technology platform on which the software system will be implemented (such as CORBA, EJB, Web Services platforms, database models, and others). Finally, PSM is mapped, again automatically or semi-automatically, into implementation code of the target platform. MDA separates technology-independent concepts from technology-dependent concepts, and supports reuse of high level models for their implementation onto different platforms.

Since MDA starts from PIM which is a software model, the motivation of our work is how to complement MDA as the world of software still has to interface with the world of business. On top of PIM, there are requirements from users in the business world, the information from which should be analyzed into PIM. We can start the development process with some formal 'requirement' model from the business side, i.e. a business process model, before deriving into software models. Business process models describe flows of processes or activities within the boundary of the business functions [2] and are created by business analysts who have good understanding of the business requirements, but may not have knowledge about how to realize the business activities in terms of software modules. On the other hand, software models are constructed by software designers who know more about the techniques to model the composition and cooperation of software modules that collectively implement the business functions. As business process models and software models present different views of the business applications [2], bridging the gap between the two views can help software designers to derive PIM. However, it is generally difficult to derive a software model from a business process model since business process models do not represent directly the building blocks of the software to be designed and implemented [3].

Despite the difficulty, close consideration reveals that a process flow generally refers to concepts that can be building blocks for the corresponding software model. This paper reports an initial attempt to stretch MDA up to the business world by bridging business process models (BPMs) and PIMs (i.e. BPM2PIM). Here we propose an initial

guideline to help software designers to construct a software model (i.e. a UML class diagram [4]) from a business process model (i.e. a BPMN diagram [5]). The guideline borrows the idea of object-oriented domain analysis to identify UML classes from BPMN processes and adding details to the classes by semantic information of the application domain such as domain-specific software patterns and other knowledge about the application domain.

An overview of BPM2PIM approach is in Section 2. Section 3 explains the guideline under BPM2PIM to derive class diagrams from business process models in details. Discussion about the use of this guideline is in Section 4 and about related work in Section 5. Section 6 gives a conclusion and future work.

2 BPM2PIM

The BPM2PIM approach makes MDA reachable from the business world by stretching vertically the top of MDA to meet business process models (Fig. 1). Business process models become part of model-driven software development by application of a BPM2PIM guideline. This guideline would describe the process to derive a PIM software model from a business process. In this paper, we restrict the guideline only to a mapping between a business process, which is modeled by a process modeling notation (e.g. BPMN), and a PIM, which is modeled by a UML class diagram. The preliminary version of the guideline is hence called the Business-Process-Model-to-Class-Diagram (BPM2CD) guideline.

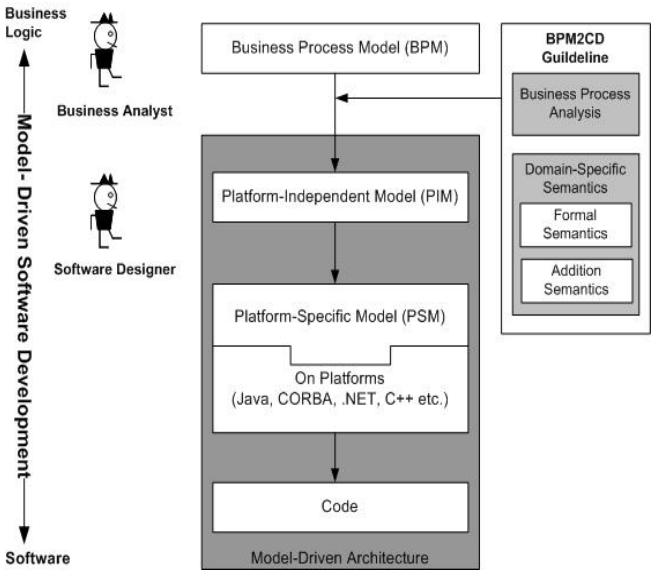


Fig. 1 BPM2PIM approach with BPM2CD guideline

The BPM2CD guideline comprises three parts: (1) Applying business process analysis (2) Applying formal domain-specific semantics (3) Applying additional semantics. Applying business process analysis is the part that analyzes the business process model in order to identify building block concepts of the software to be constructed. This is an analogy to domain analysis for object-oriented software design which derives a conceptual model of the software from a use case specification [6]. The resulting conceptual model of the software is augmented by domain-specific semantics either existing ones (such as software patterns) and other add-on semantics. The end result is a more detailed class diagram at PIM level. Details of these three parts are in the next section.

3 BPM2CD

The BPM2CD guideline describes the derivation process for designing a class diagram from a business process model. At this stage, software designers can follow the guideline to manually derive a PIM-level class diagram. Nevertheless, more automation is targeted as discussed in Section 6.

3.1 Applying Business Process Analysis

It is assumed that a business analyst would describe business requirements in terms of a business process model which is expressed by some process modeling language. A business process model describes, from start to finish, the flow of processes, events, or transactions within the business as well as the associated logics to produce or complete something of value to the business organization [2].

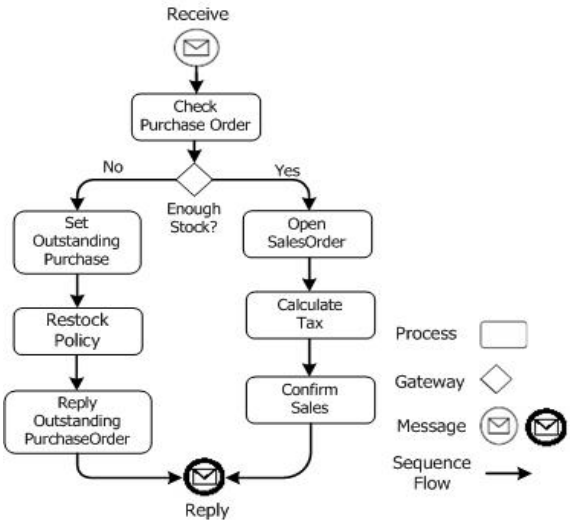


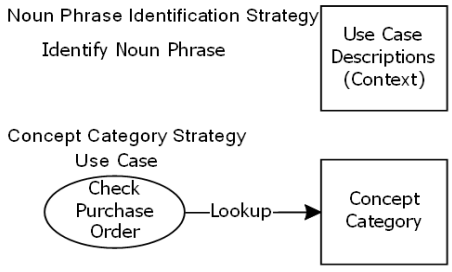
Fig. 2 Business process model for purchase order

Fig. 2 shows a business process of a vendor processing purchase orders of goods from its customers. After receiving a purchase order, checking is performed to see if there is enough goods in stock to complete the purchase order. If so, a sales order is opened, the tax is calculated, and the sale is confirmed. But if not, the restock process is performed to reorder goods from a supplier before the purchase order is responded as an outstanding order. The business process can in fact be modeled by any business process modeling language, but here it is represented by BPMN language [5].

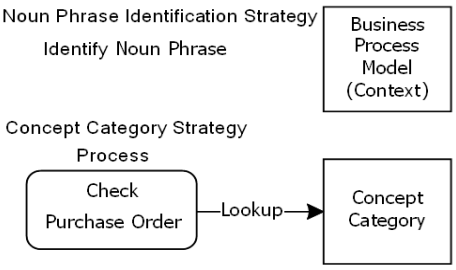
The business process model in a graphical notation as above is seen as a requirement from the business analyst. Therefore it is analogous to the traditional requirement obtained from interviewing users of the business organisation which may be written in descriptive text or formally developed by a system analyst in the form of a UML use case specification.

Business Analogy	Object-Oriented Analysis & Design (OOA&D)	Associated Document to OOA&D
Business Processes	Requirement Analysis	Use Cases
Roles in Organization	Domain Analysis	Conceptual Model
Responsibility/ Interaction	Responsibility Assignment, Interaction Design	Design Class Diagram, Collaboration Diagram

(a) Analogy between business and OOA&D terms



(b) Domain analysis strategies applied to use case



(c) Domain analysis strategies applied to business process

Fig. 3 Use-case-to-conceptual-model strategies and the analogous business-process-to-conceptual-model strategies

In [6], a comparison is given between terms in the business world and those in the world of software development (Fig. 3(a)). The comparison clearly shows the correspondence between a business process in a business process model and a use case in a use case specification. There are also established strategies for domain analysis, i.e. noun phrase strategy and concept category strategy, which will analyze the use case specification in order to identify classes that model the software (i.e. to build a conceptual model of the software) [6] (Fig. 3(b)). In the noun phrase identification strategy, use case descriptions will be examined in order to identify important noun phrases that may become the object-oriented classes of the software. This can be enhanced by the concept category strategy which uses the use case name to lookup a repository of concepts (or keywords), namely the concept category, within the application domain. The concepts found will be identified as the classes of the software. The BPM2CD guideline therefore defines its business process analysis part by adopting such use-case-to-conceptual-model strategies to identify software classes from a business process model (Fig. 3(c)).

3.1.1 Noun Phrase Identification Strategy

With the noun phrase identification strategy, a software designer will examine the business process model in order to identify important noun phrases that would become classes at PIM level. Fig. 4 highlights the important noun phrases found in the business process model of Fig. 2.

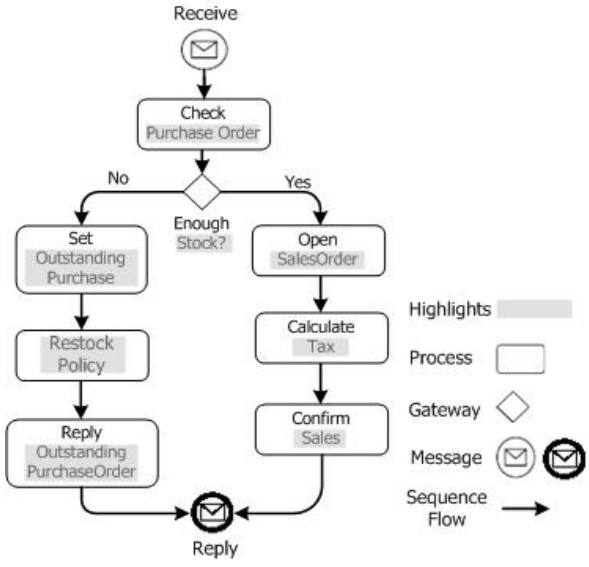


Fig. 4 Noun phrase identification

3.1.2 Concept Category Strategy

Concept category is a repository that maintains concepts or key vocabularies for application domains. The concepts are listed by categories that are generally important for any domains. It is assumed that domain experts define this knowledge base for a common use by software designers. Each process name in the business process model (such as *check Purchase Order* in Fig. 3 (c)) is used to lookup in the concept category in order to find concepts that relate to the process name. The result is a number of concepts, some of which may already be identified by the noun phrase identification strategy, and some of which may not be present in the business process model but they relate to the business domain. Fig. 5 gives an example of the concepts, related to the processing of a purchase order, which are identified by looking up the concept category.

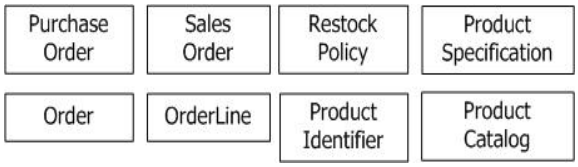
Concept Category	Examples
Physical or tangible objects	Point-of-Sale Teminal
Specification, designs, or descriptions of things	ProductSpecification
places	Store
transaction	Order
transaction line items	OrderLine
Roles of people	Buyer, Seller
Containers of other things	Store
Things in a container	ProductIdentifier
Other computer of electo-mechanical system external to our system	-
Abstract noun concepts	-
organization	SalesDepartment
events	SalesOrder, PurchaseOrder
Processes(often not represented as a concept, but may be)	CheckPurchaseOrder
Rules and policies	RestockPolicy
catalogs	ProductCatalog
Records of finance, work, contracts, legal matters	Receipt
Financial instruments and services	-
Manuals, books	-

Fig. 5 Concepts in concept category

3.1.3 Conceptual Model

The software designer examines the concepts resulting from applying the two strategies and decides on the concepts that would constitute the conceptual model of the class diagram at PIM level. Fig. 6 (a) shows the conceptual model derived by the noun phrase identification strategy. This can be added by the concepts from the concept category strategy in Fig. 6 (b).

(a) From noun phrase identification strategy



(b) From concept category strategy

Fig. 6 Conceptual model

3.2 Applying Formal Domain-Specific Semantics

The conceptual model roughly forms the structure of PIM. Details such as attributes, methods, and relationships between classes can be added by studying existing formal domain-specific semantics. Such semantics can be in several forms, e.g. descriptive text, ontology, or the formal model such as software patterns.

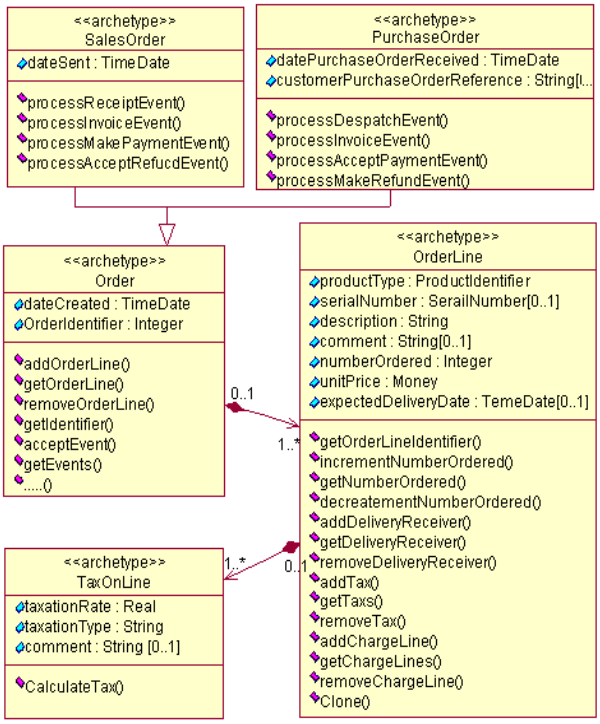
Software patterns provide a solution for understanding and modeling a specific part for a business software system. Patterns are always at a higher level of abstraction than normal analysis classes. There are several kinds of patterns that can be applied to modeling software. In this paper, the focus is on archetype patterns which describe possible PIMs that can be adopted by specific business domains [7]. We can adopt parts of the order archetype pattern (Fig. 7 (a)) and the product archetype pattern (Fig. 7 (b)), cataloged in [7], to our purchase order example. The two archetype patterns can refine the conceptual model with details such as attributes, operations, composition, aggregation, and inheritance of the classes. The result of the merging between the archetype patterns and the conceptual model is depicted in Fig. 8.

3.3 Applying Additional Semantics

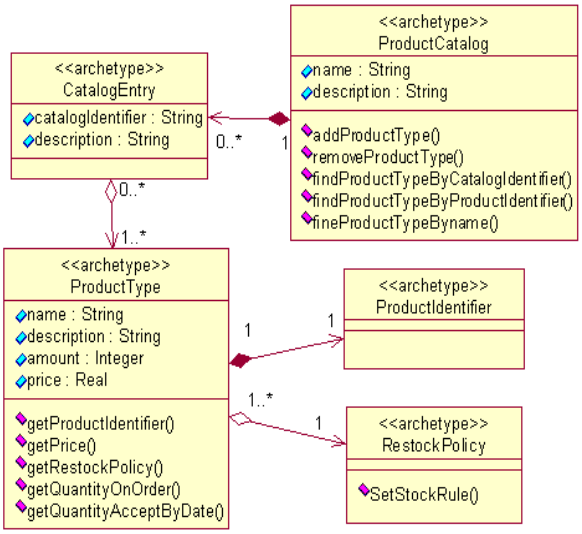
Additional semantics, unlike formal semantics in Section 3.2, refers to other knowledge about the business domain which may not be cataloged formally. For our example, suppose that there is no archetype pattern for the detail of *RestockPolicy*. The software designer can add information about the approaches to reordering of goods and formulate into a class diagram as in Fig. 9.

RestockOnHand refers to reordering of goods when insufficient stock is on hand or the inventory level has reached a reorder point. *RestockEconomic*

refers to reordering of goods based on the Economic Quantity Order model which considers the quantity to order that minimizes the total variable costs required to order and hold inventory [8]. In both approaches, the quantity to order can be optimal quantity. The final PIM resulting from the BPM2CD guideline can be obtained by attaching Fig. 9 to Fig. 8 on the *RestockPolicy* class.



(a) Order archetype pattern



(b) Product archetype pattern

Fig. 7 Part of order and product archetype patterns

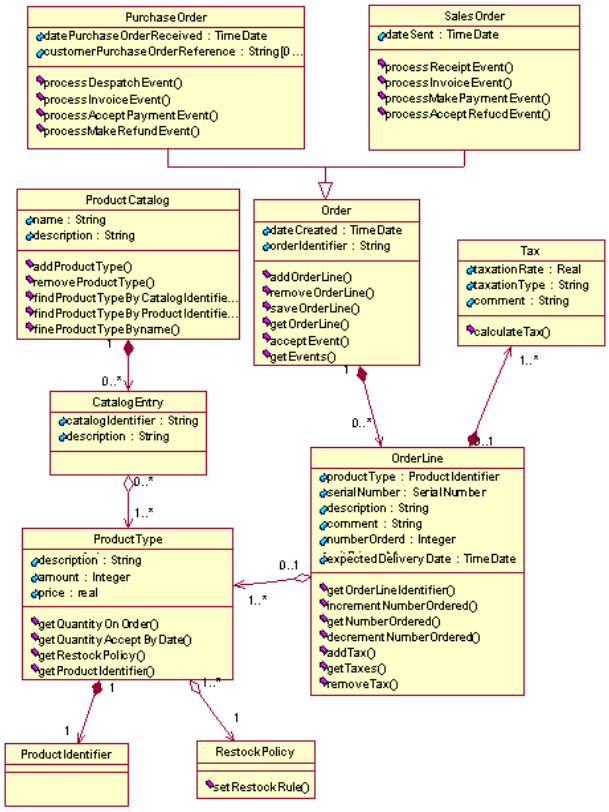


Fig. 8 Applying archetype patterns to conceptual model

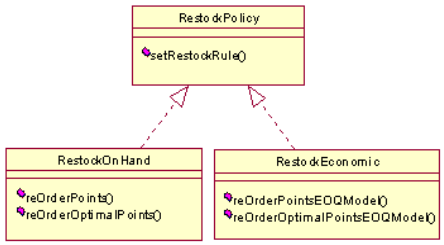


Fig. 9 Additional semantics

4 Related Work

Mapping between business process models and UML has been targeted by a number of researches, and most of the time, it is manual or semi-automatic. The obvious case is the straightforward mapping between UML activity diagrams and process flow languages such as in [9], [10]. The less obvious case is the mapping to other UML diagrams. In [11], dependencies between the Event Driven Process Chain (EPC) model and several kinds of UML diagrams, including class diagrams, have been established. Their approach identifies UML classes based on the information from the EPC model only and identifies an event in the EPC model as a UML

class. Our approach, on the other hand, considers both the information in the business process model and other external semantic information, and also adopts the object-oriented 'lexical analysis' approach to identify classes. In [12], the work focuses on the use of patterns for business processes and also on deriving UML classes from the process patterns. Similar to our approach, some semantic information is added to complete the resulting class diagrams but no clear guideline has been given on how to identify the classes and where the additional semantics come from. Also their approach adopt UML profile for Web application architecture to organize the resulting class diagrams so their class diagrams would exhibit platform information, i.e. 'PSMer' than the class diagrams from our approach. In [13], an analysis on business operations results in solution artifacts (e.g. business objects, process flows, user interfaces, application connectors) that altogether will form a architectural model of the solution to the business problems. Their solution model is a PIM, but it is more of an architectural model, not a software model as in our approach.

5 Conclusion

The paper has presented the BPM2CD guideline that attempts to bridge a business process model and a PIM-level class diagram. The techniques to apply a business process analysis and domain-specific semantics respectively to the business process model and the conceptual model of the software have been discussed.

As mentioned before, the guideline can be used manually by software designers. Nevertheless, more degree of automation is foreseen to make the guideline goes along better with the philosophy of automatic or semi-automatic mapping between models in MDA. Automated analysis of noun phrases and verbs together with the use of ontology for the concept category is possible. The resulting candidate concepts can be recommended to software designers for further class selection through a supporting design tool. Moreover, formulation of domain-specific semantics into class diagrams can be automated such as a formulation from ontology-based domain semantics. Also, the derivation process could be enhanced for MDA by a formal mapping between the metamodel of the business process modeling language and UML metamodel.

A closer look at the application of domain-specific semantics is also expected. There should be a classification of such semantics, i.e. what kinds of formal and additional semantics can be applied in the guideline.

References:

[1] MDA Resources. Object Management Group. Available: www.omg.org/mda/index.htm

[2] H. Smith, *BPM and MDA: Competitors, Alternatives or Complementary*, *WhitePaper*, Available: www.BPtrends.com , July 2003.

[3] A. Kleppe, J. Warmer and W. Bast, *MDA Explained: The Model Driven Architecture Practice and Promise*, Addition-Wesley, 2003.

[4] Object Management Group (OMG), *UML Specification version 2.0*, October 2, 2004 Available: www.uml.org

[5] Business Process Management Initiative (BPMI), *Business Process Modeling Notation (BPMN) version 1.0*, May 3, 2004. Available: www.bpmi.org

[6] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*, Prentice Hall, Inc., 1997.

[7] J. Arlow and I. Neustadt, *Enterprise Pattern and MDA: Building Better Software with Archetype Patterns and UML*, Pearson Education, Inc., 2003.

[8] R. H. Wison, *A Scientific Routine for Stock Control, Harvard Business Review*, vol. 13, 1934, pp. 116-128.

[9] P. Jiang, Q. Mair, J. Newman, *Using UML to Design Distributed Collaborative Workflows: from UML to XPDL*, Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), 2003.

[10] S. A. White, *Process Modeling Notations and Workflow Patterns*, *White Paper*, Available: www.BPtrends.com, March 2004.

[11] P. Loos and T. Allweyer, *Object-Orientation in Business Process Modeling through Applying Event Driven Process Chains (EPC) in UML*, Proceedings of 2nd International Enterprise Distributed Object Computing Workshop (EDOC'98), 1998, pp.102-112.

[12] O. H. Barros, *Business Information System Design Base on Process Pattern and Frameworks*, Industrial Engineering Department, University of Chile, September 2004. Available: www.BPtrends.com

[13] Y. Huang, S. Kumaran and K. Bhaskaran, *Platform-Independent Model Templates for Business Process Integration and Management Solutions*, Proceedings of IEEE International Conference on Information Reuse and Integration (IRI'03), 2003, pp.617-622.