A Domain Engineering Approach to Specifying and Applying Reference Models

Iris Reinhartz-Berger[1], Pnina Soffer[1], Arnon Sturm[2]

[1]Department of Management Information Systems
University of Haifa, Haifa 31905, Israel
iris@mis.hevra.haifa.ac.il, spnina@is.haifa.ac.il

[2]Department of Information Systems Engineering,
Ben-Gurion University of the Negev, Beer Sheva 84105, Israel
sturm@bgu.ac.il

Abstract: Business process modeling and design, as an essential part of business process management, has gained much attention in recent years. An important tool for this purpose is reference models, whose aim is to capture domain knowledge and assist in the design of enterprise specific business processes. However, while much attention has been given to the content of these models, the actual process of reusing this knowledge has not been extensively addressed. In order to address this lack, we propose to utilize a domain engineering approach, called Application-based Domain Modeling (ADOM), for the purpose of specifying and applying reference models. We demonstrate the approach by specifying a sell process reference model and instantiating it for a chocolate manufacturer. The benefits of utilizing the ADOM approach for specifying business models are the provisioning of validation templates by the reference models and the ability to apply the approach to various modeling languages and business process views.

1 Introduction

Business process modeling and design, as an essential part of business process management, has gained much attention in recent years. Business process design is a complicated task due to the increasing complexity of organizations and market forces that drive organizations to continuously improve in order to sustain their competitive position. Business processes entail a network of related activities, both within an organization and in collaboration with its environment. While diversity of business processes among organizations is high, there are many common aspects that apply to the majority of the organizations which share common characteristics (e.g., market segment, size, logistic typology, etc.). Moreover, knowing some of these commonalities can be of help when inter-organizational processes are designed.

This fact has been widely recognized, and motivated the emergence of a number of reference models, whose aim is to provide generic knowledge about business processes in order to assist in their design in specific enterprises. To this end, reference models prescribe what is sometimes termed "best practice" processes for a specific business segment. Reference models were promoted by Enterprise Resource Planning (ERP) vendors, who used them as a prescription for processes that should be adopted as part of the implementation of the ERP system. Some of these models deal with business processes only (e.g., [St01]) and some appear as part of a set of views in an enterprise model (e.g., [Sc98], [Sc99]).

However, our observation is that while much attention has been given to the construction of reference models and to the knowledge that is captured in them, the process of reusing this knowledge through process design in a specific organization is somewhat neglected. In particular, the reference models themselves provide little support (if any) to their actual implementation.

In this paper we rely on a well-established discipline of domain engineering, applied in software engineering for reusing various types of artifacts in the process of software design, and introduce its principles to business process reference models. Specifically, we adopt the Application-based DOmain Modeling (ADOM) ([RS04], [SR04]) approach to reference models and utilize its embedded mechanism to facilitate the specialization of a generic reference model to the specific needs of an enterprise. ADOM is based on a three layered architecture: application, domain, and language. The domain layer consists of specifications of various domains, while the application layer consists of particular systems or business processes. The language layer includes metamodels of (modeling) languages. ADOM enforces constraints among the different layers, or more precisely, the domain layer enforces constraints on the application layer, while the language layer enforces constraints on both the application and domain layers.

When adopting ADOM to reference models, the reference model itself serves as the domain model, specifying and enforcing constraints on the application model, which is the implementation of the business process in a specific enterprise. Thus, the contribution of this paper is the establishment of an approach that provides guidelines and validation templates when utilizing reference models. Due to the popularity UML [OU03] gained in the software engineering and enterprise modeling communities, we chose to apply the ADOM approach to reference models using UML activity diagram notation.

The remainder of the paper is organized as follows. Section 2 is a literature review of the two main areas that this paper integrates: business process reference models and domain engineering. Section 3 introduces the ADOM approach, while Sections 4 and 5 explain the construction of a domain (reference) model in ADOM and its instantiation in a particular enterprise, respectively. We accompany these explanations with examples of a Sell process (as a reference model) and a chocolate manufacturer (as a particular enterprise). Finally, Section 6 summarizes the benefits of the ADOM approach when dealing with process reference models and refers to future research plans.

2 Literature Review

This section reviews existing literature about business process reference models, demonstrating our claim that model reuse has received relatively little attention. It discusses types of reuse processes found in existing reference modeling approaches. Next, the domain engineering literature explains how reuse support is emphasized and an analogy to reference model reuse process approaches is presented.

2.1 Business Process Reference Models

Reference models have been discussed, classified, and evaluated using a number of evaluation frameworks and criteria (e.g., [FL03], [FL05], [MZ00], [SR98]). However, most of these attempts address mainly the model itself in terms of expressiveness and adequateness, structure, compatibility, and other factors. When the application of the model is addressed, the properties that are discussed are flexibility, extensibility, and cost-effectiveness. The intended reuse process has not been addressed in these works as a classification or an evaluation criterion. Nevertheless, reference models can be classified based on their intended reuse process, which can be reuse by adoption, reuse by assembly, reuse by specialization, or reuse by customization. We shall discuss each of these reuse types and indicate examples when such exist.

Reuse by adoption: Reference models based on this approach (e.g., SAP [CL99] and Scheer [Sc98]) are very detailed models, whose intended use is to be taken "as is" and serve for the enterprise under consideration. The knowledge they provide is at the lowest level of abstraction, aimed at being reused without modifications. This is consistent with the perception that an enterprise has to adapt itself to the model (or the software it underlies) rather than the other way around. Nevertheless, studies reported in the literature hardly show evidence that such full adoption is applied in practice. Daneva [Da99] measured the level of reuse of the SAP reference models in a number of case studies belonging to different market segments, and indicated that full reuse was not achieved in any of them, although in some cases the level of reuse was remarkably high. The parts that were not fully reused were either modified or designed from scratch. However, no guidance is provided in the model for performing such operations.

An example of this category is Scheer's model [Sc99], which includes a number of views besides the process model depicted as Event-driven Process Chains (EPC). It aims at representing the entire set of possible solutions. It implicitly indicates mandatory and optional parts by using logical relations (and, or, exclusive or). However, as indicated by Rosemann and Van der Aalst [RA05], these relations may represent run-time choices as well as design decisions.

Reuse by assembly: Some reference models (e.g., DEM [Va98]) provide detailed "building blocks", whose intended use is by selection and assembly. The knowledge captured in these models is, similarly to the former type, at the lowest abstraction level. However, they provide the enterprise with some degree of flexibility, facilitated by the possibility of selecting the appropriate building blocks from a variety of such available. As suggested by the reuse by adoption approach, in this approach the enterprise is free to modify parts or design parts if none of the existing satisfies its need, but these actions are not guided by the model. The DEM model, similarly to Scheer's model, includes a number of views, but here processes are specified as Petri-Nets. It was constructed to be used within the Baan ERP system, and accompanied by an infrastructure that enables its reuse as part of the ERP implementation process. The infrastructure includes a repository in which the model parts reside, and a rule base aimed at completeness verification and at keeping consistency among the different views of the model. However, the rules that address the composition of the model parts are not applicable to changes made in the details of these parts.

Reuse by specialization: In contrast to the first two types of reuse, models whose intended use is through specialization provide knowledge at a high level of abstraction. The main advantages of reuse by specialization are twofold. First, the knowledge captured in the model serves as a basis for constructing the specific model of an enterprise without imposing a detailed solution. Second, generic models may not be completely domain-specific, thus they allow reusing knowledge across domains that share common characteristics. An example of a model based on this approach is the Supply Chain Operations Reference (SCOR) [St01] model, which outlines supply chain operations and management processes, grouped according to logistic typologies and main process types involved (e.g., source, make, deliver). SCOR states that in order to become operational, the processes need to be specialized and a model of a lower level of abstraction must be constructed, according to the practices of the specific enterprise. However, the model does not include any mechanism that supports or guides such specialization (e.g., by indication about mandatory or optional steps).

Reuse by customization: This approach has been recently presented by Rosemann and Van der Aalst [RA05], and is not applied by an existing reference model yet. It is specifically targeted at reference models attached to enterprise systems, whose application is part of the implementation of the system in an enterprise. The approach is motivated by the limitations of the reuse-by-adoption approach, where a low-level model specifies all the possible options and variants (or some separate models need to be consolidated). Such model does not distinguish design decisions from run-time decisions, mandatory from optional activities, and possible dependencies among design decisions. The model suggested by [RA05] is at a low abstraction level, employing an extended EPC (Configurable-EPC), where configuration possibilities as well as dependencies are explicitly specified. Specifying configuration possibilities facilitates the adoption of parts of the detailed model without altering its level of abstraction. As this is a recent development, the approach still does not meet many configuration challenges, such as mandatory vs. optional decisions, different levels of configuration decisions, inter-process dependencies, and more.

The configuration possibilities are derived from the functionality of the enterprise system to be implemented. The same rationale may be applicable to reference models that are not software-related, but configuration possibilities are less straightforward to identify in this situation.

2.2 Domain Engineering

As the variability of information and software systems has increased, the need for an engineering discipline concerned with building reusable assets (such as specification sets, reference models, software patterns and components) on one hand and representing and managing knowledge in specific domains on the other hand, has become crucial. This discipline, called *domain engineering* [Cl02], supports the notion of a domain, defined as a set of applications that use common concepts for describing requirements, problems, and capabilities. The purpose of domain engineering is to identify, model, construct, catalog, and disseminate a set of software or business artifacts that can be applied to existing and future systems in a particular domain. As such, it is considered an important type of software reuse, verification, validation, and knowledge representation [Me97].

A sub field of domain engineering is *Domain analysis* which identifies a domain and captures its ontology. It should specify the basic elements of the domain, organize an understanding of the relationships among these elements, and represent this understanding in a useful way. Similarly to the process reference models area, domain analysis relates to different types of reuse. We classify the domain analysis methods and techniques into two categories: single-level and two-level domain analysis approaches.

In the ***single level domain analysis approaches***, the domain knowledge is defined by domain components, libraries, or architectures. These domain artifacts are reused in an application as they are, but can be modified to support the particular requirements at hand. The Draco approach [Ne89], for example, organizes software construction knowledge into several related domains, each of which encapsulates the requirements and different implementations of a collection of similar systems. Meekel et al. [Me97] propose a domain analysis process that is based on multiple views. They used Object Modeling Technique (OMT) [Ru91] to produce a domain-specific framework and components. The feature-oriented approach as applied by Gomaa and Kerschberg [GK95] and FODA [Ka90] suggests that a system specification will be derived by tailoring the domain model according to the features desired in the specific system. That is, a specific system uses the reusable architecture and instantiates a sub-set of features from the domain model.

In the ***two-level domain analysis approaches***, connection is made between a domain model and its usage in an application model. Contrary to the single-level domain analysis approaches, the domain and application models in the two-level domain analysis approaches remain separate, while validation rules between them are defined. These validation rules enable avoiding syntactic and semantic mistakes during the initial stages of application development, reducing development time and improving system quality.

These approaches mainly utilize the metamodel concepts in which the domain model is the metamodel and the application model is the derived model that correspond to the metamodel. Examples for such approaches are the studies by Schleicher and Westfechtel [SW01], Gomma and Eonsuk-Shin [GE02], and the Generic Modeling Environment (GME) [No99], which also uses the Object Constraint Language (OCL) [WK98] to specify additional constraints. These approaches lack the support for dynamic constraint specifications and are limited in their accessibility as they use different jargons within the two layers.

Considering the reuse approaches of business process reference models and the domain analysis approaches, the following analogy can be made. The reuse by adoption, by assembly, and by customization approaches are analogous to the single level domain analysis approach, in which an existing element can be reused or modified when applied to a specific application. The reuse by specialization is akin to the two-level domain analysis approach. Yet, the reuse by specialization approach provides only the high-level model, and does not entail mechanisms for validating the specialized model that is created. In this paper we explicitly address that issue.

3 The Application-based Domain Modeling Approach

Being influenced by the classical framework for metamodeling presented in [OM03], the Application-based DOmain Modeling (ADOM) approach is based on a three layered architecture: application, domain, and language. The *application layer*, which corresponds to the model layer (M1), consists of models of particular enterprises, including their structure (data) and behavior (business processes). The *language layer*, which corresponds to the metamodel layer (M2), includes metamodels of modeling languages, such as UML, EPC, OMT, etc. The intermediate *domain layer*, which can be labeled M1.5, consists of specifications of various domains; in particular it can include reference models. The ADOM approach enforces constraints among the different layers; in particular, the domain layer (the reference models) enforces constraints on the application layer (the specific enterprise process models).

Figure 1 depicts the architecture of the ADOM approach in a specific case study. The application layer in this figure includes process models of three organizations: a chocolate manufacturer, a computer store, and a software development company. These applications belong to different logistic typologies. The chocolate manufacturer, which sells special kinds of chocolate from its stock and produces chocolate to renew its stock, is classified as a make-to-stock typology. The computer store, which composes different off-the-shelf parts into computers that meet customer needs, is classified as an assemble-to-order typology. Finally, the software development company, which analyzes, designs, implements, tests, and maintains software products according to a customer's Request For Proposal (RFP), is classified as an engineer-to-order typology. Although different, the aforementioned applications deal with similar business processes, such as sell products and buy raw materials.

The domain layer in Figure 1 includes two generic process models, one for selling and the other for buying. These process reference models provide guidelines for instantiating the business processes in particular enterprise applications, such as the chocolate manufacturer, the computer store, and the software development company. The language layer in this example will be limited to *UML*. In this paper we chose UML activity diagrams as a language for process modeling in ADOM for the following reasons. First, UML is the de-facto standard modeling language. Second, in order to specify process reference models we need tools for expressing activities, triggers, sequences, synchronization points, conditions, etc. UML activity diagrams include these capabilities. Third, using a stable notation, such as UML, benefits from the maturity of its development environment, including its CASE tools.

In order to support the variability of enterprise business processes, we utilize the UML built-in stereotype mechanism. As defined in [OU03], a ***stereotype*** is a kind of a model element whose information content and form are the same as the basic model element, but its meaning and usage are different. In the domain layer, a new "multiplicity" stereotype family is introduced to represent how many times a model element can appear in a particular application. In the application layer, any element (class, association, activity, state, etc.) is stereotyped according to the elements declared and constrained in the domain layer. A model element in an application model must preserve the relations of its stereotypes in the relevant domain model.

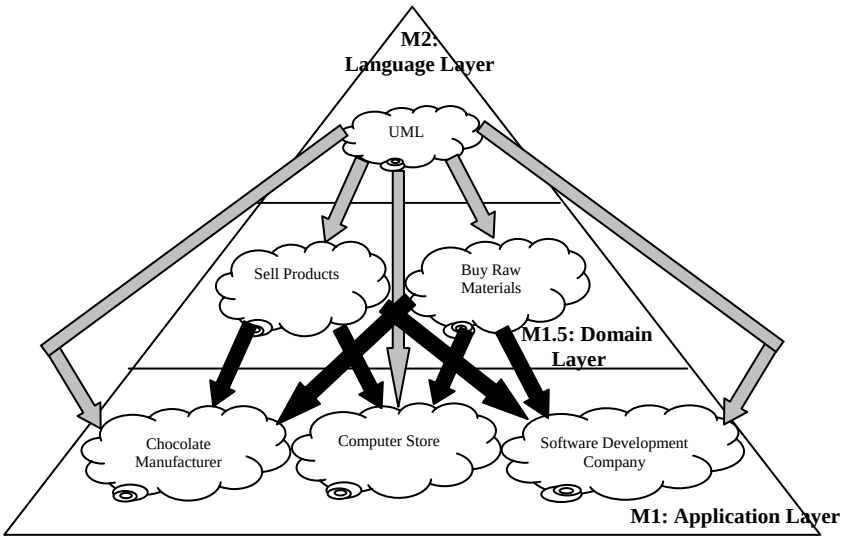

Figure 1. The Application-based DOmain Modeling (ADOM) architecture

Section 4 explains and demonstrates the use of ADOM for constructing a process reference model, while reusing it for creating enterprise-specific process models is explained and demonstrated in Section 5.

4 Constructing Reference Models in ADOM

The reference models are constructed in the domain layer using the language vocabulary and constraints. In our case, the vocabulary of activity diagrams includes activities, transitions, conditions, synchronization points, etc. The multiplicity of these elements can be constrained using stereotypes. The main multiplicity groups are: (1) an optional application element, denoted as <<0..n>>, meaning that this element can be "instantiated" any number of times in an application model, (2) an optional single application element, denoted as <<0..1>>, meaning that at most one application element can be classified as the domain element, (3) a mandatory application class, denoted as <<1..n>>, meaning that this element should be "instantiated" at least once in any application model, and (4) a mandatory single application element, which is the default (no multiplicity is indicated) and is equivalent to <<1..1>>, meaning that exactly one application element can be classified as the domain element. Other adjusted groups, denoted as <<min..max>>, are also legal.

Figure 2 is an activity diagram that depicts the Sell process at the reference model level. The process begins with an optional application activity, called **Quote Activity**. This activity can be further elaborated into one or more **Quote Preparation** activities, followed by an optional single **Quote Monitoring** activity. The reference model also allows dependencies among **Quote Preparation** activities, as indicated by the optional self link of **Quote Preparation**. A **Quote Preparation** activity may be followed by another **Quote Preparation** activity or a **Quote Monitoring** activity or can directly proceed to the branch denoting the evaluation of the quote. If the quote is not approved, the **Sell process failed**. If the quote is approved or a customer order is received as an external event, the mandatory single **Insert Order** activity is executed. Note that since the **Quote Activity** is optional the process may simply start by receiving a customer order. Upon completion of the **Insert Order** activity, an optional **Validate Configuration** activity is performed in order to check the product feasibility. This activity is very useful for an assemble-to-order typology. Then a one or more **Check Availability** activities are performed to check the availability of raw materials, products, resources, etc.

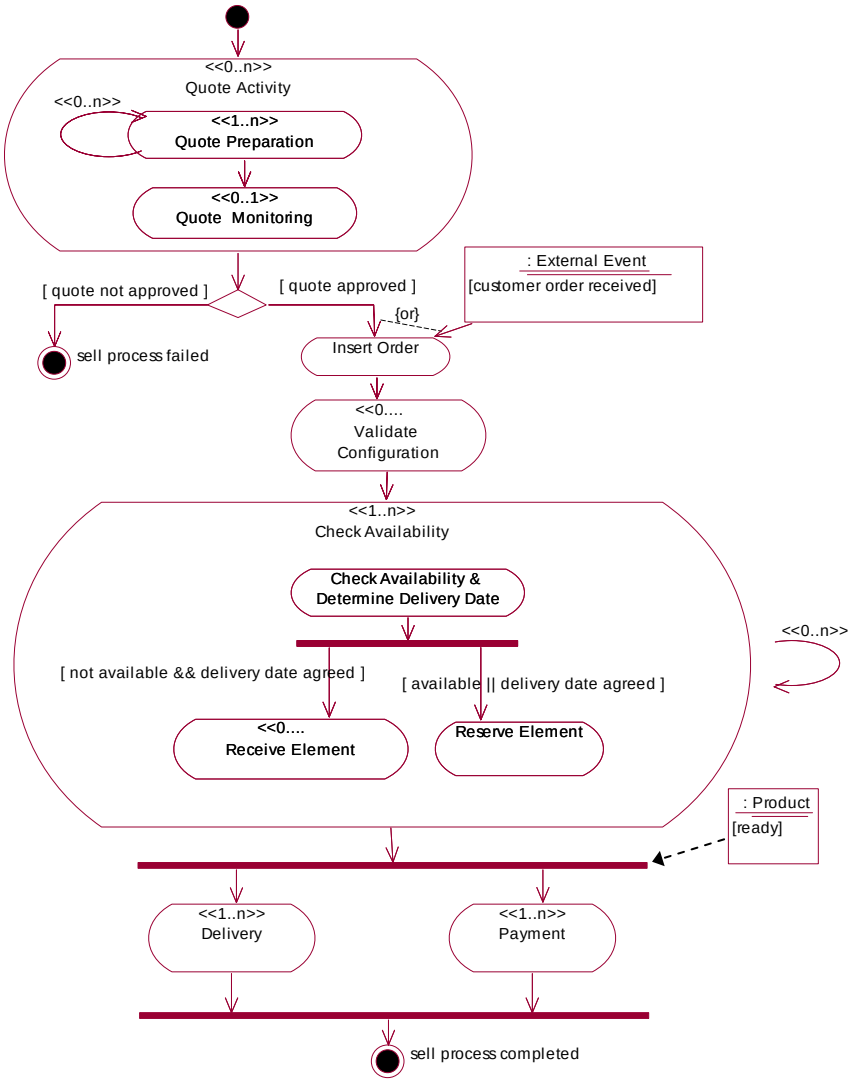

Figure 2. The reference (domain) model of the Sell process

The **Check Availability** activity consists of **Check Availability & Determine Delivery Date** activity, followed by a **Reserve Element** activity in case the item is available or the delivery date is agreed (but the element is not available), and optional concurrent **Receive Element** activities in case the item is not available, but the delivery date is agreed. Note that the reference model does not specify action when delivery date is not agreed. This illustrates the fact that the model includes only generic enough information. Cases that are not specified in the model can be added specifically when the model is specialized. The reference model also allows dependencies among **Check Availability** activities, as indicated by the optional self link of the **Check Availability** activity. Thus, a **Check Availability** activity may be followed by another **Check Availability** activity, or may continue to another stage (i.e., **Delivery** and **Payment**). The **Check Availability** activity should be followed by at least one **Delivery** activity and at least one **Payment** activity. Yet, the beginning of these activities is subject to the condition that the **Product** is **ready**. Upon completion of all **Delivery** and **Payment** activities the Sell process successfully completes.

5 Instantiating a Reference Model in ADOM

An enterprise-specific process model builds on the knowledge captured in a reference (domain) model and uses it as a validation template. All the constraints enforced by the reference model should be applied to any process (application) model of that domain. In order to achieve this goal, any element (class, association, activity, state, etc.) in the application model is classified and stereotyped according to the elements declared in the reference model. Particularly, when constructing a specialization of the reference model, one should create a mapping from the domain elements to the enterprise elements and backwards.

In the rest of this section we will show how the reference model of the Sell process shown in Figure 2 fits the chocolate manufacturer example.

As noted, the chocolate manufacturer sells special kinds of chocolate from stock. Figure 3 is a UML activity diagram that describes the Sell process in this organization. This process is triggered by an external event in which the customer order is received (either by phone, fax, email, or personally). As a consequence, an activity of inserting an order, called **Order Chocolate**, begins. Note that no quote activity is performed here, and no configuration validation, as this is a standard product supplied from stock. The next step is **Check Chocolate Availability**. This activity is zoomed into its sub-activities: first, the chocolate availability is checked, while the delivery date is determined. The delivery date might be immediately or any time in the future that is agreed between the customer and the seller. The possibility of future delivery dates enables the seller to receive chocolate unavailable in stock from production without canceling the customer order. If the required chocolate is available or the delivery date is agreed (in case the chocolate is not available immediately), the amount of chocolate ordered by the customer is reserved, in order to avoid "overbooking" of the chocolate in stock. If the chocolate is not available, but the delivery date was agreed, **Receive Chocolate for Order** should be executed for the process to proceed further. The process continues only if the product, i.e., the ordered chocolate, is ready. In this case two parallel activity sets occur: one handles the payment, while the other deals with the delivery. The payment activities include **Issue Invoice** and **Receive Payment** which are executed in this order. The delivery activities deal with three types of shipment. In the first shipment type, "shipment by customer", the only activity that is needed is **Prepare Delivery by Customer Documents**. In the second shipment type, "shipment by manufacturer for export", three activities are sequentially executed: **Reserve Overseas Carrier**, **Prepare Export Documents**, and **Load Containers**. In the third shipment type, "shipment by manufacturer for local needs", only two activities are needed: **Prepare Shipment Document** and **Load Trucks**. Finally, after the payment and delivery activities are finished, the Sell process is (successfully) completed.

Notice that the Sell process of the chocolate manufacturer follows the reference model of a general Sell process, shown in Figure 2. As noted, in this case, the quote activity and validate configuration processes are redundant and, hence, do not appear in the application model (Figure 3). On the other hand, the delivery and payment activities are refined as allowed in the reference model by the multiplicity constraints of these activities. The reference model also does not forbid possible dependencies among these activities, as actually exist in the chocolate manufacturer case study. The chocolate production, which is done in the background, is not part of the selling process and, hence, does not appear in the reference model neither in the application model of the chocolate manufacturer case study.

The reference model of the Sell process was also applied to the other two examples mentioned above, the computer store (assemble-to-order) and the software development company (engineer-to-order). Due to space limitations, we will not go into the details of these models. Nevertheless, it is important to note that despite the difference in the focus and nature of the Sell process in the three organizations, the three specific models that were created follow the reference model in Figure 2 and comply with it.

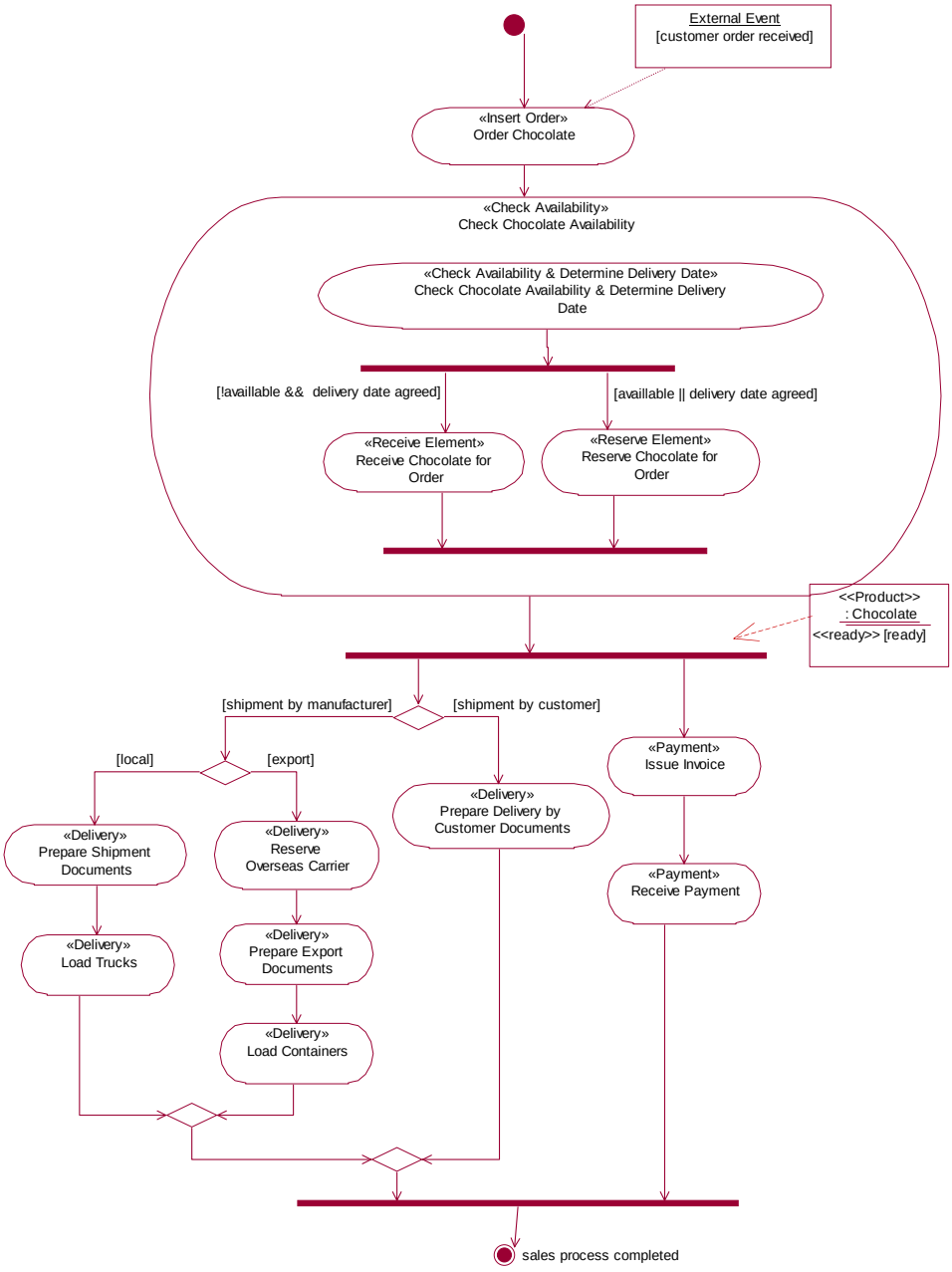

Figure 3. The Sell process in the chocolate manufacturer

6 Summary and Future Work

Reference models are models whose aim is to capture domain knowledge and assist in the design of enterprise specific processes. However, while much attention has been given to the content of these models, the actual process of reusing this knowledge has not been extensively addressed. This paper proposes to utilize the ADOM approach, whose roots are in the domain engineering discipline, as a platform for constructing reference models and instantiating them in a specific enterprise. The benefits of utilizing the ADOM approach for the purpose of specifying and applying reference models are twofold. First, while applying a specialization-based reuse approach, ADOM's reuse process is more powerful than the ones employed by existing reference models. The uniqueness of the proposed approach, as compared to other reuse-by-specialization approaches, is that the reference model provides a validation template for its instantiations. This can be achieved, for example, by utilizing the stereotype mechanism of UML. In the paper we discussed the possibilities of constructing different specializations of the reference model, where each has its own specific features, yet they are all based on the reference model and comply with it. Second, although demonstrated to UML, ADOM is a generic approach that can be applied with various modeling languages. This enables us to extend the model to other views of an enterprise, such as data structure, while maintaining the same logic as demonstrated on process models.

In the future, we plan to examine the integration of various enterprise model views and their counterparts in the related modeling language (e.g., UML). In addition, we intend to experiment the utilization of the ADOM approach to the business process reference models in real enterprises and explore the implications of using the proposed approach. For this purpose, we plan to develop a CASE tool which will guide the developers in constructing reference models and instantiating them in particular enterprises.

References

[Cl02] Cleaveland, C. "Domain Engineering", http://craigc.com/cs/de.html, 2002.

[CL99] Curran, T. A. and Ladd, A., SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1999.

[Da99] Daneva, M., Measuring Reuse in SAP Requirements: a Model-based Approach. In SSR'99, Proceedings of the Fifth Symposium on Software Reusability, ACM Press. New York, pp.141-150, 1999.

[FL03] Fettke, P. and Loos, P. "Classification of Reference Models – A Methodology and its Application", *Information Systems and e-Business Management*, 1(1), p. 35-53, 2003.

[FL05] Fettke, P. and Loos, P. "Ontological Analysis of Reference Models", In: Green, P. and Rosemann, M. (eds), *Business Systems Analysis with Ontologies*, p. 56-81, Idea Group, 2005.

[GK95] Gomaa, E. and Kerschberg, L. "Domain Modeling for Software Reuse and Evolution", Proceedings of Computer Assisted Software Engineering Workshop (CASE 95), 1995.

[GE02] Gomma, H. and Eonsuk-Shin, M. "Multiple-View Meta-Modeling of Software Product Lines", Proceedings of the Eighth IEEE International Confrerence on Engineering of Complex Computer Systems, 2002.

[Ka90] Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. “Feature-Oriented Domain Analysis (FODA) Feasibility Study”, CMU/SEI-90-TR-021 ADA235785, 1990.

[Me97] Meekel, J., Horton, T. B., France, R. B., Mellone, C., and Dalvi, S. “From domain models to architecture frameworks”, Proceedings of the 1997 symposium on Software reusability, pp. 75-80, 1997.

[MZ00] Misic, V. B. and Zhao, J. L. “Evaluating the Quality of Reference Models”, *Proceedings of ER 2000 – 19th Conference on Conceptual Modeling*, p. 484-498, Springer-Verlag, Berlin, 2000.

[Ne89] Neighbors, J. “Draco: A Method for Engineering Reusable Software Systems”, in T. Biggerstaff and A. Perlis. Software Reusability. Volume I: Concepts and Models. ACM Press, Frontier Series, Addison-Wesley, Reading, pp. 295-319, 1989.

[No99] Nordstrom, G., Sztipanovits, J., Karsai, G. and Ledeczi, A. “Metamodeling - Rapid Design and Evolution of Domain-Specific Modeling Environments”, Proceedings of the IEEE Sixth Symposium on Engineering Computer-Based Systems (ECBS), pp. 68-74, 1999.

[OM03] OMG-MOF, “Meta-Object Facility (MOF™)”, version 1.4, 2003.

[OU03] OMG-UML, “The Unified Modeling Language (UML™)”, version 1.5, 2003.

[RS04] Reinhartz-Berger, I. and Sturm, A., “Behavioral Domain Analysis – The Application-based Domain Modeling Approach”, the 7th International Conference on the Unified Modeling Language (UML'2004), Lecture Notes in Computer Science 3273, pp. 410-424, 2004.

[RA05] Rosemann, M. and Aalst van der, W. M P. “A Configurable Reference Modeling Language”, *Information Systems* (forthcoming).

[Ru91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. Object-Oriented Modeling and Design, Prentice-Hall International, Inc., Englewood Cliffs, New Jersey, 1991.

[Sc98] Scheer, A. W, Business Process Engineering: Reference Models for Industrial Enterprises, Springer Berlin, 1998.

[Sc99] Scheer, A. W., ARIS – Business Process Frameworks, Springer, Berlin, 1999.

[SW01] Schleicher, A. and Westfechtel, B. “Beyond Stereotyping: Metamodeling Approaches for the UML”, Proceedings of the 34th Annual Hawaii International Conference on System Sciences, pp. 1243-1252, 2001.

[SR98] Schuette, R., and Rotthowe, T. “The Guidelines of Modeling – an Approach to Enhance the Quality in Information Models”, *Proceedings of ER 1998 – 17th International Conference on Conceptual Modeling*, p. 240-254, Springer-Verlag, Berlin, 1998.

[St01] Stephens S., “Supply Chain Operations Reference Model Version 5.0: a New Tool to Improve Supply Chain Efficiency and Achieve Best Practice”, Information Systems Frontiers, Vol. 3 No. 4, pp. 471-476, 2001.

[SR04] Sturm, A. and Reinhartz-Berger, I., “Applying the Application-based Domain Modeling Approach to UML Structural Views”, the 23rd International Conference on Conceptual Modeling (ER'2004), Lecture Notes in Computer Science 3288, pp. 766-779, 2004.

[Va98] Van Es, R., Dynamic Enterprise Innovation, Baan Business Innovation B.V., The Netherlands, 1998.

[WK98] Warmer, J. and Kleppe, A. The Object Constraint Language: Precise Modeling with UML. Addison-Wesley, 1998.