

# Domain-Driven Modeling with Aspects and Ontologies

Pavel Hruby  
Microsoft  
Frydenlunds Allé 6  
2950 Vedbaek, Denmark  
+45 29229183  
phruby@acm.org

## 1. SUMMARY

A common task during developing software applications in specific domains is to identify application objects and their behavior. In this paper, we will illustrate in the domain-driven development of software applications, application designers can use domain ontologies as one of the sources for creating application models, in addition to traditional analysis based on user requirements. The conceptualized domain knowledge in the form of domain ontology can be used as a metamodel for application object models.

However, domain ontologies cannot describe specific functionality and the differences between different applications in a given domain, which originate in user requirements, because the concepts of the domain ontology must be applicable to all systems in the domain. Application objects often cannot be extended by additional functionality in a modular way, because the modules of functionality resulting from the user requirements are often not localizable into application objects that originate in the domain ontology.

To solve this conflict, we will illustrate that the domain-specific software applications lead to the architecture of a component with two orthogonal dimensions. The object dimension represents the categories originating in the domain ontology and the aspect dimension represents the functional modules that originate from user requirements.

## 2. APPLICATION OBJECTS

“An ontology is an explicit specification of a conceptualization” [Gruber 1993]. Ontological categories define the concepts that exist in the domain, as well as relationships between these concepts. For object-oriented applications, domain ontology defines the metamodel for application models in this domain. Ontological categories correspond to the metaclasses for application objects. For example, in the ontology for business systems called REA (Resources, Events, Objects), the economic agent is a metaclass for objects such as customer and vendor, the economic event is a metaclass for objects such as sale and payment receipt. Economic resource is a metaclass for objects such as money and item. Likewise, relationships between ontological categories become metarelations for the relationships between application object, such as stockflow is a metarelationship for relationships called outflow and inflow; duality between economic events is a metarelationship for the reconciliation relationship between Sale and Payment Receipt, see Figure 1.

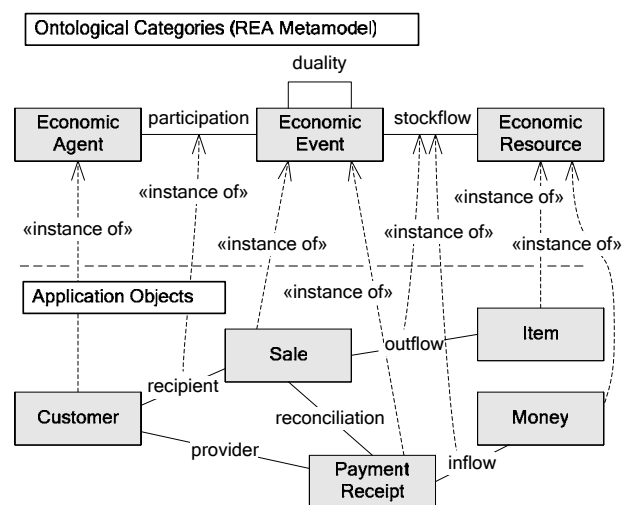


Figure 1. Application Model and its REA Metamodel

## 3. APPLICATION BEHAVIOR

### 3.1 Functionality of Domain Objects

The previous section illustrated that structure of a software application can be derived from ontological categories that apply to the application's domain. However, to build a useful software application, the mere structure of domain objects is not sufficient. Domain objects need functionality that is often not specified by the ontological categories, but is required by the application's users. For example, the REA ontology does not specify how to determine the identity of business objects, or how to collect financial data. However, functionality such as serial numbers and accounts are essential for the users of business applications.

Application functionality is not specified by domain ontologies for a good reason. Domain ontologies specify the structure of concepts that can be applied to all systems in the domain. Domain ontologies attempt to find the minimal, yet complete set of concepts covering the domain.

The functionality of application objects usually differs from one system to another because of specific user requirements, local conventions, as well as several other reasons. For example in business applications, some objects need human readable serial

numbers, such as customers and products; some do not, such as order lines. Financial reporting depends on local legislation, lines of business and reporting usually varies from one application to another, reflecting the fact that every company is somehow different than the other. A complete list of functionality of the domain objects probably cannot be specified in general for the whole domain; users of software applications would always need new features or new versions of existing features, which cannot be foreseen by those who create the ontology.

### 3.2 Cross-Cutting Domain Objects

Ontological categories determine one dimension of decomposition of the domain. The other dimension of decomposition is the application functionality. In this section we will show that in many cases the modules of application functionality are not localizable into a single application object.

In the business domain, for example, the serial number of an item is an attribute of the item object. The serial number is usually not a random number. The item serial number is chosen from a number series, which is an attribute of a group of the economic resources, to which the number series is applied. Thus, the object representing the item group contains rules specifying things such as the format of the serial number, whether a serial number should be unique, how does it depend on previous numbers of the series, rules determining whether serial numbers of deleted items can be reused, and other similar rules. The number series module cross-cuts two domain objects, the item object and the item group object, and the number is constructed by mutual collaboration between the part that resides on the item and the part that resides on the item group. It is useful to think of the number series as a single module, but this module cross-cuts two application objects.

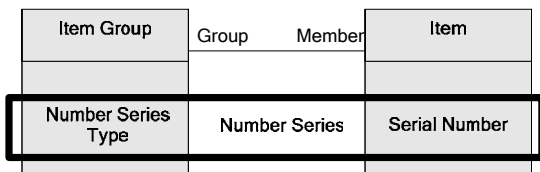


Figure 2. Number Series Cross-Cuts Application Objects

Aspect-oriented programming is one of the approaches, and an additive convention of thought on how to deal with cross-cutting concerns in a modular way. In the scope of domain-driven development, it is useful to think about entities derived from ontological categories as objects and about functionality of the software applications as aspects.

### 3.3 Aspect Categories

In the section about domain objects we have shown that ontological categories correspond to metaclasses of the application objects. A similar approach can be also applied to the entities in aspect dimension.

For example, we have shown that the number series is an aspect in the application model. However, the fundamental purpose of the number series is to give the application object unique identity. Therefore, we can think of the number series as a specific instance of a more general aspect category called **identification**.

Other instances of the identification aspect category are the name, phone number, e-mail address, URL (Uniform Resource Locator), GUID (Globally Unique Identifier) and ISBN (International Standard Book Number). More examples of the aspects categories are illustrated in Figure 5; instances of **address** aspect category are Billing Address or Shipping Address, instances of **posting** aspect category are G/L entry and Inventory entry.

## 4. DOMAIN-DRIVEN DEVELOPMENT

### 4.1 Object and Aspect Dimension

In this section we illustrate how these concepts can be used to develop a software application in a specific domain.

We have shown that a domain-specific model can be decomposed along two dimensions: the object dimension that reflects the ontological categories of the domain and the aspect dimension that reflects the behavior, which the software application must have in order to be useful, see Figure 3. We have also shown that the components both in the object dimension and the aspect dimension can be specified at two levels of abstraction; the level of ontological categories and aspect categories, and the level of application objects and application aspects.

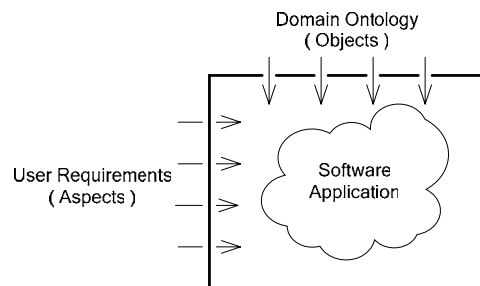


Figure 3. Objects, Aspects and Domain-Driven Development

### 4.2 Application Configuration

The configured software application is an application that conforms to the ontology for the particular domain and also contains specific functionality that meets user's needs. As the application objects are determined by the domain ontology, the process of creating an application model consists of assigning application aspects to application objects. This architecture is outlined in Figure 4.

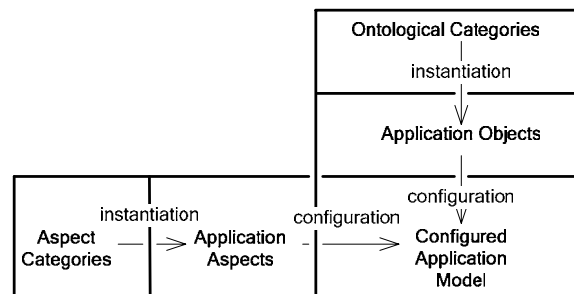


Figure 4. Application Configuration

Figure 3 illustrates the key message of this paper, which is, using domain ontologies to determine the application object model

leads to a software architecture with two orthogonal dimensions. An example of a business application configured in this way is illustrated in Figure 5. This application is a model of a simple sales module.

The **ontological categories** in this application are Economic Agent, Economic Event and Economic Resource; their instances are application objects Customer, Sale, Payment, Item and Cash.

The **aspect categories** in this application are Identification, Account, Address and Posting. Instances of the Identification aspect are Name, Item Number, Customer Number and Transaction ID. Instances of the Account aspect are Inventory Account, Bank Account, Customer Account and Cash Account. Instances of the Address aspect are Billing Address and Shipping Address. Instances of the Posting aspect are G/L (General Ledger) Entry and Inventory Entry.

The choice of the aspect categories is determined by user's needs. Other configurations of the sales process in the software applications for different users would contain a different set of aspect categories.

The **configured application model** illustrated in Figure 5 contains the Application Objects with Application Aspects. The Customer object contains the aspects Name and Number (identification aspects), Customer Account (account aspect), Billing Address and Shipping Address (address aspects). The Sales object contains the aspects Transaction ID (identification aspect) and G/L Entry (posting aspect). Payment Receipt contains the aspects Transaction ID and G/L Entry (posting aspect). The Item object contains the Item Number (identification aspect), and the Money object contains the Bank Account and Cash Account aspects.

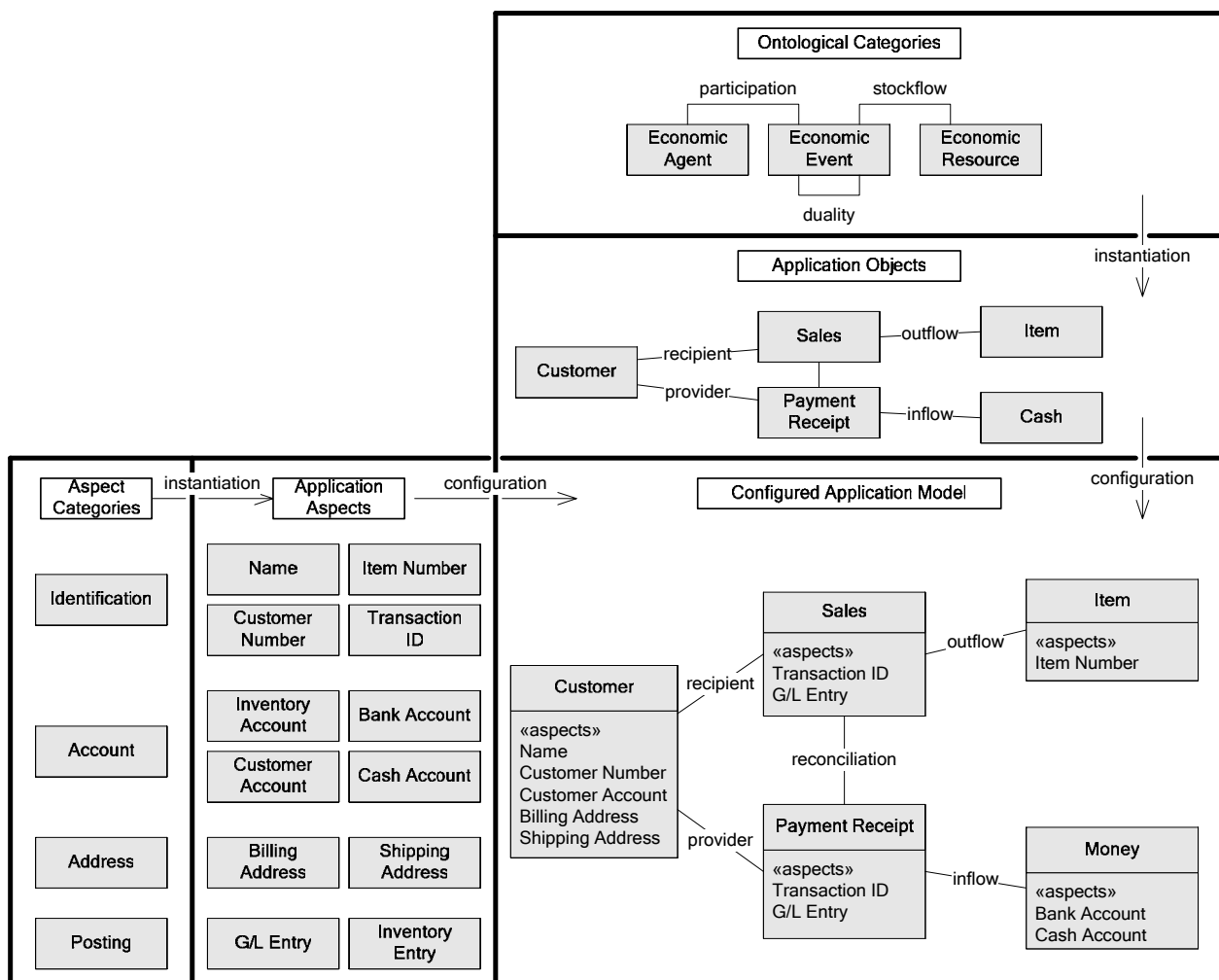


Figure 5. Example of Application Configuration

## 5. REFERENCES

[1] Geerts, G. McCarthy, W. *The Ontological Foundation of REA Enterprise Information Systems*, Michigan State University, August 2000

[2] Hruby, P.: *Ontologies in Aspect-Oriented Domain-Driven Development*, ECOOP 2005 workshop on Views, Aspects and Roles, Glasgow 2005.