

database systems, etc. The split also increases the inter-operability among business processes. Traditionally, each business process is built as an application, which encapsulates a specific task. However, business processes must often interact. When business objects are hidden inside applications, it is hard to inter-operate without a deep understanding of the implementations of each application. This makes business process interaction difficult and complicated [7]. By separating business objects from underlying systems, the interaction of business processes is simplified. The business logic is separated from the lower level system implementation.

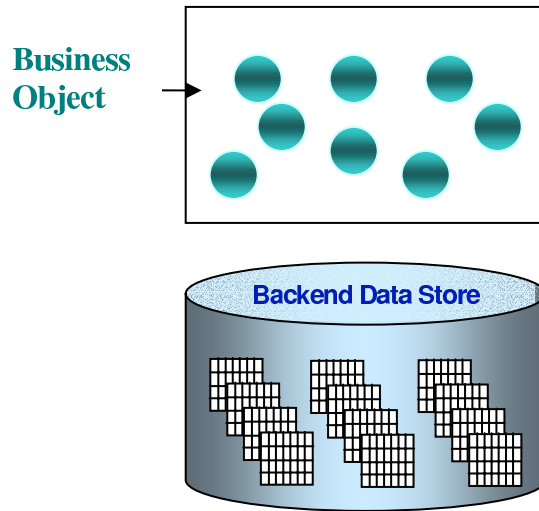


Figure 1: Business concepts and system implementations are separated levels.

However, the concept of business objects is unique to the business application world. Current IT technology does not support business object concepts as part of other functionality, such as data transformation, analysis, and integration, etc. In particular, business intelligence (BI) is an increasingly important building block in today’s enterprise, because it directly aids decision making. But current BI technology deals with traditional database systems – it does not handle business objects directly. Adding business objects into business intelligence tools requires knowledge of both the business and software domains; this requirement slows down BI analysis and therefore makes the use of BI tools for time-sensitive business decisions less effective.

The ETL tools are a standard technology that eases the pain of data transformation. The user of ETL tools can focus on the semantic mapping from a data source to a data target and let the ETL tool manage the underlying transformation details. This idea is exactly what is needed in business object management systems. But current ETL technology only supports lower level software data (e.g. data inside dbms). In short, there is an “impedance mismatch” between business object aware software and conventional systems involved in business processes.

To maintain the encapsulation of business objects and to make business objects transformation easier and smoother, it is imperative to introduce business objects as components into ETL tools.

3 Callisto, a Pluggable Component for IBM ETL Technology

In this section, we describe the Callisto¹ project and show how to add business objects in the IBM WebSphere Product Center (WPC) to an ETL tool as a pluggable component.

3.1 WPC Business Objects

The WebSphere Product Center (WPC) is a product information management (PIM) system that provides a centralized repository for an enterprise’s master data. This information is maintained in a relational database but is presented to the user as business objects with a retail flavor: *Items* belong to *Categories*, and *Categories* are organized into *Hierarchies*. Hierarchies are an especially useful feature of the WPC, and are a great example of how business objects simplify user interaction: a user can organize and view the categories in different hierarchies and can create different catalogs for the same item set.

¹Callisto originated as an IBM Extreme Blue Project. <http://www.ibm.com/extremeblue>

More specifically, categories are defined by a set of attributes, like 'UPC', 'description', etc. Items that are in a category have values for each of the attributes that the category provides. Items may belong to multiple categories, and Categories may form parent-child relationships so that one category may be more specific than another; this forms a hierarchy. For instance, the Category 'Chips' may be a child of a Category 'Snack Foods'. Because attributes are inherited, any item in the Chip category also has values for attributes in the Snack Foods category. So, in a sense, Categories provide meta-data for collections of items. This meta-data about the items is also the Category's data. Thus even with these simple business objects there is a non-trivial relation between data and meta-data. It is this type of mixing that makes these business objects difficult to model.

We chose to use the WebSphere Product Center as our proof of concept project because the benefits of using ETL tools with the WPC are immediate. Consider the following scenarios:

1. *Business intelligence:* Because the WPC stores valuable master data, we would like to be able to use this information for BI-based analysis. For instance, combining transactional data with categorical information from the WPC could allow for analysis of optimal item categorizations. The ETL technology is essential for this type of scenario because it provides the link into the business intelligence tools: once information is exposed to the ETL with our custom operators, the ETL itself is used to manipulate the data to place it into a data warehouse for analysis.
2. *Trading partner collaboration:* Trading partners wishing to collaborate need access to each other's information, but if one partner is using the WPC, and the other partner a relational database, there is no easy way to transfer information between the two systems. Through an ETL tool, the business using the WPC can expose its business objects in a way the other business can use, and the ETL itself can be used to massage this relational data so that it can be exchanged easily.
3. *Global data synchronization:* It is common for businesses to use a single WPC instance as a master information repository and also have redundant information stored in relational systems throughout the enterprise. To synchronize the WPC and such systems, it is necessary to expose the business objects in an ETL tool so that they can be mapped to schemas suitable for use by the other systems (and vice versa).
4. *Catalog construction:* Catalog construction is similar to data synchronization in the sense that the initial construction of a WPC instance requires the use of other various types of datastores. However, in this use of the ETL and custom operators, the goal is to speed deployment of a WPC instance by using the ETL technology to extract, rather than synchronize, information from various data sources.

3.2 The IBM ETL Tool

The IBM ETL tools provide a set of *operators* that are used to transform data. Operators themselves are simply logical components that produce and consume tables. For instance, a join operator consumes two tables and produces one. Data sources and targets are also operators than only produce or only consume tables. Data source operators, data target operators, and operators that transform tables from one schema to another are all logically operators, and are all used the same way (and are implemented similarly and inside the ETL). This uniformity allows for a great deal of flexibility in how operators can be used. To transform data, operators are connected to each other in a producer/consumer fashion and the ETL system takes care of moving data from operator to operator during transformation.

The ETL also provides an operator customization mechanism that allows users of the tool to add user-defined operators representing any type of data source or target, or even any type of relational transform (i.e. inner join), provided that the operators present information to the tool using sets of relational tables. This mechanism is extremely useful because it allows the user to focus on the semantics aspects of the operator: the user simply determines how tables are to be used by the operator, and the actual implementation of all the data management is provided by the ETL using the same strategies as built-in operators.

The goal of our project is to express WPC business objects in the ETL tool through customized operators, i.e. represent these business objects as data sources and targets.

3.3 Integration Challenges

Current ETL technology supports relational formats, such as relational databases tables, CSV files etc. To represent business objects inside the ETL, we must find a way to describe business objects in a relational format without resorting to examining how the business objects are implemented and stored. In essence, we must create custom ETL operators that expose the required information. This is not an easy task because business objects are usually semi-structured or unstructured; WPC business objects are semi-structured. The following are the key challenges: (1) In many business objected based systems, there

is no clear boundary between data and meta-data. An ETL system requires operators to expose meta-data while a dataflow is designed, and then manipulate data during runtime. For a relational table, data are rows and meta-data columns; for other business objects, the distinction between data and meta-data is much less distinct. One business object could be the meta-data for another business object, which complicates the problem. Often, the more complex the business objects, the less distinct the data/meta-data distinction is. (2) The relational presentation of a business object must be as rich as the original object. That is, information about the business object should not be lost when the object is represented in a relational way. In addition, the information presented in the relational view must be presented in a way that is useful. (3) There is a trade-off when handling optional attributes of business objects. For instance, one Category business object may be represented as a table with columns for 'name' and 'price', but another Category object may require 'UPC' and 'description'. Both are category objects; if we represent them as one relational table, there will be many null fields; and if we use multiple tables, foreign key references will slow down the reconstruction of these objects. And (4) business objects and their relational views must relate to each other in a consistent, complete, and useful way. That is the references among business objects should be preserved after the transformation. In this paper, we mainly focus on the first challenge since it is the most difficult one.

3.4 An ER Model Approach

A first naive attempt to solve this problem would be to construct an ER model of the business objects. In the following, we explain this approach in detail, partly because this modeling process is very similar to the UML approach described in the next section. At the end of this discussion we explain why a direct ER model approach is not feasible.

To represent business objects in relational format, one can construct a model of the business objects and map the objects (or parts of objects, or sets of objects) onto ETL constructs (or parts of constructs, or sets of constructs. ETL constructs include tables, data types, operators, and other concepts in an ETL system). We need to decide which part of the business objects to model; not all business objects can be represented as ETL constructs, efficiently or otherwise. In general, the model of the business objects may not resemble how the backend of the application itself is storing the objects. This is because the view of the object presented to the user of the object is necessarily much different from how the object is represented internally. (Of course, because the higher level applications use relational databases as backing stores, the application does have a way of mapping business objects to tables, but this mapping is not constrained in the same way that an ETL representation of the objects is. Using the backing store directly also violates the abstraction provided by the object, which we wish to avoid. In addition, external systems normally do not have access to the backend store for reasons, such as security, privacy etc.)

Unfortunately, there are some limitations with this model. This approach requires the schema of each business object to be known at the modeling time. That is, it requires the meta-data to be static. However, this is not always possible for business objects. For example, as we have seen in Section 3.1, in WPC, categories can be business objects. So categories are dynamic in the sense that different WPC instances contain different category objects. However, categories are also meta-data for items. To model an item object, its meta-data information needs to be fixed and be known at modeling time. In other words, such with such an ER approach all WPC instances would need have same meta-data for item objects, i.e. same categories. Therefore, the ER model approach is not inadequate. So instead, we decided to move to a richer model, using the ER model as a starting point.

3.5 A Dynamic Approach using UML and EMF

Because an ER model can only capture static meta-data (e.g. it requires all Category objects in WPC to share a single schema), a more dynamic approach is needed. That is, we need some way to model business objects so that different instances of the objects could have different schemas. We need to be able to model objects in such a way that the meta-data presented by business objects can change dynamically and need not be known to create the model.

To allow for this dynamic modeling approach, we had to turn to a modeling framework. We chose the java-based Eclipse Modeling Framework (EMF) [2] as a base and used UML to actually create the EMF model. Our model does not always specify the schemas for the representations of the objects it contains; rather, business objects were modeled as 'virtual table producers', where the actual schema of the table to be produced as a relational representation was generated when the actual modeled object was used. In this way, we are able to statically define schemas for certain objects if convenient, and are able to delay determining a meta-data representation of a business object until a particular model is instantiated by examining a WPC instance. Thus, our objects can present different schemas in the ETL tool, and we can overcome the first challenge described in Section 3.3.

In addition to being modeled as virtual table producers, business objects in our model hold references to other business objects; we were able to use our earlier ER model to help find these references. One advantage to this approach is that when a

business object in our model needs to present a relational representation of itself, it can follow references to other objects and use their relational representations as part of its own. In our particular model of WPC objects, we didn't need to expose all business objects as virtual table producers; we modeled many objects, but only exposed some of them. This is possible because the references between business objects are used extensively.

Furthermore, by mapping our model to java objects, we are able to add some of the associated semantics and constraints to the java code of the object. For instance, a java object representing a category can raise an exception indicating it is in an illegal state, and this logic can be placed directly in the class; in the ER model, this type of verification requires additional attention from the database side.

To access WPC business objects, the generated java objects in our model communicate with a WPC instance using the WPC scripting language mechanism. WPC scripts are programs that are run inside of a WPC instance to transfer data and perform other functions. We used the Apache Velocity Engine [1] to generate WPC scripts based on a set of script templates that are customized using the state of the relational representation of the modeled objects. For instance, given a Category object, Callisto invokes the Velocity Engine to customize a WPC script, using that Category's state, and the customized script is then used to read and write actual items from and to a WPC instance.

In general, our approach can be summarized into the following steps:

1. Construct a UML model of the business objects that captures the key relationships between the objects. Keeping in mind a set of custom operators that you would like to deploy can help constrain the model (see 4 below).
2. Then, create a way to access business objects and extract necessary information needed by the UML model. For example, in the Callisto project, this step is to generate java code for objects that can be populated by examining a WPC instance. Also write code to write an object back to the target system (i.e. our templated WPC scripts). Logic can be added to the objects to make sure they are consistent.
3. Then, pick out objects, parts of objects, or sets of objects that need be exposed in the ETL tool. These 'slices' need not represent anything originally in the target system; they need only be useful for step 4. Such objects should be modeled as virtual table producers in EMF. Write code that dynamically generates the schemas for the relational representation based on the state of the object. Also write code that propagates a change in the relational representation back into the java object.
4. Finally, create customized ETL operators that expose the objects from step 3. Simple business objects that map to single or multiple tables can be expressed as operators directly (i.e. a Category object used as an item source operator). Other operators can combine the relational tables of multiple objects (i.e. by joining) to present a clean representation to the user.

4 Discussion

Probably the most interesting question about this method of modeling is the extent to which it is required. That is, is there some 'meta-model' of generic business objects that can be deployed for specific models, such as the WPC/retail models? If there exists such a model, then once it is added to ETL tools, all business objects can be connected to the ETL tools.

However, such a meta-model, even if it exists, is not necessarily elegant. By definition, a business object contains domain-specific meaning. Thus, any meta-model would need to have some mechanism to allow specific instances of the model (i.e. WPC/retail) to be able to represent domain-specific knowledge. In short, the meta-model of business object would really need to (1) be generic enough to be deployed for different business object schemas and (2) contain rich semantics that can be mapped to business objects in different domains. Either of these can be solved separately, but it becomes a hard problem when these two requirements are combined together. Therefore, we use customized operators as our solution to bring business objects into ETL tools.

5 Conclusion

We have seen that business objects – abstractions of objects that businesses use in their day-to-day operations – are a powerful way of representing an enterprise's data. To use business objects with other systems, including business intelligence tools and relational database systems, the objects need to be exposed to ETL systems without breaking the business object abstraction. Our approach to providing this integration is to use a UML model, which captures the key constraints between objects, and

then generate a modeling framework. This modeling framework then produces relational presentations of business object instances based on the object's state. Finally, custom operators use these relational representations to present clean relational table schemas to the rest of the ETL system.

6 Acknowledgments

The authors thank the IBM Software Group, especially the Business Intelligence ETL and the WebSphere Product Center development group for providing resources to this project.

References

- [1] Apache Velocity Engine. <http://jakarta.apache.org/velocity/>.
- [2] Eclipse Modeling Framework (EMF). <http://www.eclipse.org/emf/>.
- [3] Peter Eeles and Oliver Sims. *Building Business Objects*. Wiley Computer Publishing, 1998.
- [4] David Gillibrand. Essential Business Object Design. *Communications of the ACM*, 43, 2 2000.
- [5] Albert Maier, Bernhard Mitschang, Frank Leymann, and Dan Wolfson. On Combining Business Process Intergration and ETL Technologies. In *BTW*, 2005.
- [6] Alkis Simitsis, Panos Vassiliadis, and Timos Sellis. Optimizing ETL Processes in Data Warehouses. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, 2005.
- [7] Oliver Sims. *Business Objects, Delivering Cooperative Objects For Client-Server*. The IBM McGraw-Hill Series. McGraw-Hill Book Company, 1994.
- [8] Jeff Sutherland. Business Objects in Corporate Information Systems. *ACM Computing Survey*, 27, 1995.
- [9] Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. Conceptual Modeling for ETL Processes. In *DOLAP*, 2002.