A Log-based and Traceability-Oriented Business Object Association Model for Code Generation

Zhongjie Wang, Dechen Zhan, Xiaofei Xu

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

Summary
Building well-designed business models is a key step to implement agilely reconfigurable enterprise software and applications (ESA) to adapt to rapid changes in business environments. Traditional business modeling methods can not effectively deal with complex association relationships between business objects; therefore it has great influences on extendibility, adaptability, second-round development efficiency and traceability of ESA. To solve this limitation, in this paper we present a log-based and traceability-oriented business object association model, in which complex and volatile associations between objects are clearly separated from the inner structure of objects and three numerical association styles, i.e., 1 to 1, 1 to \( n \) and \( n \) to 1, are emphatically discussed. Business logics on objects are classified into two types of operations, i.e., simple operations that deal with inner logics in a simple object, and complex operations which deal with numerical associations between objects. Run-time states of business objects and association information between objects are separated as logs for further traceability.

Key words: Business models, business objects, association, logs

1. Introduction
Modern enterprises are in a rapidly and violently changing business environment, which leads to frequent changes on management patterns and business processes of enterprises. To support such changes, broadly applied enterprise software and applications (ESA) must have the capacity of agile reconfiguration. Building well-designed business models are considered as a key and prior activity to support agility of ESA. Although core of business models is the process view, each business process may be elaborately decomposed into three parts, i.e., a set of business objects to be manipulated (e.g., bills, resources, humans, reports, etc), a set of operations that take effects on these business objects, and then, a set of rules that describes the execution or triggering sequences of these operations. Therefore we may draw a conclusion that only if business object and the corresponding operation models are flexible enough, may the agility of ESA be ensured.

Traditional ESA business models are usually constructed with process-oriented way, i.e., modeling each business as a process and a set of activities. This approach has some obvious shortcomings, e.g., (1) if a modeler has not yet found all the business process in an enterprise, then the process models would be incomplete; (2) structures of some processes are comparatively complicated, therefore the quality and soundness of the model will be determined to a large extend by the ability and experience of the modeler, etc. If we start with another view, i.e., business object modeling, above limitations will be eliminated. This is because number of business objects in an enterprise is determinate and numerable, therefore if all the business objects are identified and their attributes, states, operations are obtained, and then completeness and soundness of business models are easily ensured.

Fig. 1 An example of associations between different business objects
Business objects do not isolatedly exist, and there are various associations between objects. Fig. 1 shows a simple example in Procurement domain of enterprises, e.g., Fig. 1 (a) describes the business “Gather/decompose a set of procurement requirement bills (Req. Bill) to draw the procurement plans (Plan Bill), then select proper suppliers according to the results of comparison on quality and price (SC bill), finally create the procurement orders (Order) with the chosen providers.” There are four business objects and three associations between them.

In object models, elements that usually change are mainly the associations, and the inner attributes of a business objects seldom changes. For example, in routine situations, an enterprise may do its procurement process with the style of Fig. 1(a), however in some exceptive circumstances (e.g., urgent procurement requirements), the final procurement orders may directly created from req. bills without planning, just as shown in Fig. 1(b)(c). Therefore, an ESA must have the ability to simultaneously support all possible situations (e.g., Fig. 1(d)).

In order to produce the final executing source codes according to business models, besides basic business logics in each object, those associations between objects must also be elaborately coded. One of the most familiar strategies is to design an association as one or several attributes in related objects, and when relationships between object instances occur at run-time, values of these attributes are written in. In such situation, the attribute set of each object will contain a mass of redundant association attributes (e.g., the attribute “quantity of planned procurement requirements” in req. bill expresses the association with plan bill) besides self-related information.

This way has very little to be recommended with the reasons that:

1. Cannot support “n to n” associations, e.g., a req. bill may be planed in multiple plan bills, while a plan bill may inversely satisfy multiple req. bill, therefore it will not realize good traceability.
2. Increase the complexity of programming and deteriorate agility, e.g., in design phase of ESA, it is impossible to clearly know all the possible associations, therefore it is not possible to design the complete association attribute set for each object, and when the product is applied in practice, such associations may change, a mass of database schemas (e.g., tables, views, foreign keys, index, triggers, etc) and source codes must be modified.
3. Increase the chaos of object structure, e.g., with the increasing of variable associations, the number of association attributes in an object will possibly increase linearly, therefore increase the difficulty to effective maintenance.

If we consider the principle “Separation of Concerns”, i.e., clearly separating stable business objects from unstable associations, then the two parts may be modeled and coded independently (e.g., the former is mainly done during design phase and the latter is mainly dealt with during implementation phase according to specific requirements) and are then composed together to form the integrated ESA. This way is briefly shown in Fig. 2.
The main purpose of this paper is to present an association/state log based business object association model. Under the help of a code generation tool, this kind of models may be automatically transformed into executable source codes, and during run-time, association and traceability between object instances will be flexibly and rapidly supported.

Rest of this paper is organized as follows. In section 2 we put forward some related works in literatures. In section 3, the log-based object association model are presented in details. In section 4, the detailed development pattern and process based on our model is briefly discussed. Finally is the conclusion.

2 Related works

Association is an important aspect in Object-Oriented modeling, and because of the characteristics of frequent changing[2], it is also considered as a puzzle in OO researches[3][4][5].

One of a key principles in software engineering domain is “Separation of Concerns”[6][7] with the purpose of improving flexibility of models and systems. This principle is initially derived from the idea of “separate data structure from program”, and in business process modeling there also appear some similar principles, e.g., “separate business process from program”, “separate business rules from business process”, etc.[1][6][8]. In a word, “separate those unstable elements from those stable ones”[9][10]. For example in [11], aiming at business variations, a model-driven and process-configuration based ESA development style is presented. The author of [2] also considered that associations between business are easy to change, therefore he separated these associations out and presented the definition of “Rule Object” with 22 typical patterns (i.e., how to compose business objects and related rule objects). In [12], a new modeling approach of how to separating business object models from business process models is elaborately discussed.

However, these methods mainly focus on associations between macro, process-centered business elements (e.g., activities), and there lacks of enough guidance for applying them to business object associations.

Since association is a key factor to deteriorate flexibility of ESA, “Simplify associations between elements” is an obvious approach in ESA modeling [4][7]. However, associations cannot be completely eliminated whatever happens, and for this reason, in literatures there are a lot of methods on how to design good associations, such as Open-Close Principle (OCP), Acyclic Dependence Principle (ADP), Stability Dependence Principle (SDP), etc[13]. Other approaches conclude extended Entity-Relation diagram[14], complex association approach[15], etc. One of the common features of these methods is: first defining object interfaces, then manually coding objects and their interfaces, finally control association behaviors between objects by complex calling methods[5].

Further, there appeared a classical solution for object association in [16], namely, Object Association Pattern, in which each association is considered as an independent class and this class is uniformly treated as general objects. In [17], based on the analysis of association’s three forms (i.e., 1 to 1, 1 to n and n to 1), the author presented three methods to implement associations (i.e., pointer-based, matrix-based and association class based approaches), and association class was also regarded as the best one.

Following list some other typical strategies in literatures on this problem.

In [18], Ontology is imported to build Business Object Model (BOMs), and associations are also expressed as ontology. However it is only conceptual modeling and is too coarse to be applied in ESA design.

In [5], object hook/flange and object assembling were presented. The former implements a dynamic, direct and rapid mechanism for object behavior associations, i.e., all the “flanges” are realized in run-time instead of build-time. The latter is based on object hook interfaces (rather than the ordinary object interfaces) to achieve domain-oriented dynamic manipulation of object association and automatic maintenance of association semantics.

In [19] the authors presented an Agent–Object-Relationship (AOR) modeling approach, in which E-R diagram and UML are extended to express some special association types. But AOR did not mention how to implement these associations.

In [20], Active Business Objects (ABOs) are used to depict the constraints, trigger relations and association operations between objects, with the purpose of message transformation between objects (which is a type of association, too).

In [3], aiming at the uncertainty of business object attributes and associations, the authors transformed the persistent business objects into the generalized table structure and generalized recursion structure, and variable business associations are designed from these abstract data models, i.e., an abstract business object is firstly constructed and bound dynamically with the concrete physical storage structure lately in the deployment phase.
3 Log-based business object association model

3.1 Elements in the model

First we present the basic structure of our model, as shown in Fig. 3. Our main contribution is to extend the traditional business object model\textsuperscript{[7]}, i.e., adding Association, State Logs, and Association Log into it, and classifying business operations into simple and complex operations.

![Diagram of business object association model]

Fig. 3 Basic elements in business object association models

3.2 Business object

Various bills, reports, resources (humans, equipments, fields) in enterprises are jointly called Business Objects in ESA, and may be denoted as:

\[BO := \langle ID, \text{name}, \text{Category}, \text{Attribute\_set}, \text{State\_set}, \text{StateTransition\_set}, \text{SubBO\_set}\rangle\]

In which,

- **ID** and **Name** are the unique identifier and name of the object;
- **Category** refers to the object class that this object belongs to;
- **Attribute\_set** is the attribute set contained in the object, and each attribute is defined as \(\text{Attribute} := \langle \text{attribute\_id}, \text{attribute\_name}, \text{data\_type}\rangle\) with specific ID and name. \(\text{State\_logic}\) is a condition expression composed of attributes in Attribute\_set to show when the object is in the state.
- **StateTransition\_set** is the state transition set of the object, and each transition is defined as \(\text{StateTransition} := \langle \text{from\_state}, \text{to\_state}\rangle\) to describe that the transform from from\_state to to\_state is allowed.
- **SubBO\_set** is the child object set contained in the object. Generally speaking, each object may have zone, one or multiple child objects. \(\forall BO, \exists BO_1\) makes \(BO \in \text{SubBO}\_set(BO_1)\), then \(\text{State\_set}(BO) = \emptyset\), \(\text{StateTransition\_set}(BO) = \emptyset\).

3.3 Business object class

Those business objects that have the same goals and similar attributes/states together constitute a business object class. For instance, a "req. bill" may be classified into "material req. bill", "labor insurance article req. bill", "equipment accessory req. bill", etc. The reason to present this concept is that, objects belonging to the same class may have a majority of similar features, and we may use the class to depict these similarities without modeling them repeatedly.

3.4 Business object state log

During the run-time of ESA, state of each object instance must be efficiently maintained. According to query the state information, operations that may be executed in next step on this object instance will be determined. Current approaches usually add an attribute in the Attribute\_set of each object to save its state information.

The deficiency of this approach is obvious, i.e., the state of an object at one time is not unique (e.g., "req. bill" object may be in "part planned" and "part ordered" simultaneously) due to concurrent associations with two or more other objects, and one attribute cannot save multiple state information at the same time; if multiple state attributes are adopted, it will then result in redundancy.

For this reason, in our model we separate the state information from a business object itself. State log is adopted to save the state of all the business object instances and states are not related to the attributes any longer.

During run-time, after each operation, the state of each impacted object instance will be written into the state log. By querying this log we may obtain current state of each object instance. Table 1 shows a segment of the log.
### 3.5 Business object association

In ESA, associations between business objects are briefly classified into two types, i.e.,

- **Key association**, which is the most familiar association in OO, e.g., the object “order” is key associated with object “customer” by the foreign key “CustomerID” (in “order”) and the primary key “CustomerID” (in “customer”).

- **Numerical association**, which describes the “Create from...” relationship between objects, e.g., “req. bill” is numerical associated with “plan bill” by the numerical attribute “req. quantity” (in req. bill) and “planned quantity” (in plan bill); or, the “Allocated to...” relationship between objects, e.g., “customer Payment Bill” is numerical associated with “Sale Order” to describe one payment bill is paid for what set of orders.

### 3.6 Association log

In order to separate variable parts (associations) from object itself, in our model, we would not save the association information in the attribute set of each object any longer, but distill them to form an independent association log.

Traditional object modeling technique, e.g., OO, E-R diagram, IDEF, usually emphasize on the former associations but ignore the latter ones (they model such associations in process models by arrows between two activities), and programmers have to manually control such numerical associations in source code.

As mentioned above, numerical associations are frequently changing, just like Fig. 1(d), there may exist various numerical associations between two objects.

According to the number of object instances of the two parties in one numerical association, it may be further classified into three forms, i.e., 1 to 1, 1 to n and n to 1 (n to n may be considered as the combination of 1 to n and n to 1). Fig. 4 shows a simple example, in which one “plan bill” may be simultaneously satisfy multiple “req. bill”, and vice versa.

Besides the numerical attributes, a numerical association should have some other information, just as shown in Fig. 5(a), and Fig. 5(b) gives a simple example.

**Table 1 A segment of state log**

<table>
<thead>
<tr>
<th>Object</th>
<th>Object Instance</th>
<th>Operation</th>
<th>State</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req. Bill</td>
<td>SO0201P001</td>
<td>Create</td>
<td>New</td>
<td>06-02-01</td>
</tr>
<tr>
<td>Order</td>
<td>PO0201P023</td>
<td>Audit</td>
<td>Auditing</td>
<td>06-02-01</td>
</tr>
<tr>
<td>Req. Bill</td>
<td>SO0201P001</td>
<td>Audit</td>
<td>Auditing</td>
<td>06-02-02</td>
</tr>
<tr>
<td>Req. Bill</td>
<td>SO0201P001</td>
<td>Audit</td>
<td>Audited</td>
<td>06-02-05</td>
</tr>
<tr>
<td>Req. Bill</td>
<td>SO0201P001</td>
<td>Planning</td>
<td>Part planned</td>
<td>06-02-05</td>
</tr>
<tr>
<td>Req. Bill</td>
<td>SO0201P001</td>
<td>Planning</td>
<td>Total planned</td>
<td>06-03-11</td>
</tr>
<tr>
<td>Req. Bill</td>
<td>SO0201P001</td>
<td>Payment</td>
<td>Part paid</td>
<td>06-03-11</td>
</tr>
<tr>
<td>Req. Bill</td>
<td>SO0201P001</td>
<td>Payment</td>
<td>Total paid</td>
<td>06-03-14</td>
</tr>
</tbody>
</table>

**Fig. 4 n to n numerical associations between objects**

**Fig. 5 Numerical association model between objects**

Association log actually is a representation of object association model at run-time and records the association between object instances. As long as one complex operation (will be defined in next sub-section) finishes, the concrete association information will be written into the log. By querying the log, flexible traceability between different object instances may be implemented. Table 2 shows a segment of this log (corresponding to the n to n associations in Fig. 4).
3.7 Simple operation

In our model, operations on business objects are classified into Simple Operation (SOP) and Complex Operation (COP).

A SOP is defined as an atomic operation on a single object, e.g., Create, Read, Update, Delete (CRUD), etc, and does not related to any other objects. It is also the finest action in business process (cannot be further decomposed) and is considered as the most stable business logic (this is because SOP are consequentially existing and does not need to change along with other logics). SOP has the following characteristics:

(1) Related to one object;

(2) Make the object occur 0 or 1 state transitions;

(3) May be related to state, i.e., only when the object is in some specific state can it be executed; or may be unrelated to states, i.e., it may be executed at any states;

(4) Objects belonging to the same class may possibly have different SOP sets, e.g., “Oversea Order” has a SOP “Check current exchange rate”, while “Home Order” does not have such SOP.

A SOP is formally defined as SOP :=<op_id, op_name, BO, state_related_flag, initial_state_set, final_state_set>, in which BO is the business object that the SOP operates on, state_related_flag is a flag to show whether SOP is related to states, i.e., state_related_flag=1 means that only when BO is in one of the states in initial_state_set can SOP be allowed to execute (e.g., the SOP “Req. Bill Planning” may only be allowed to execute when the BO “Order” is in the state “Audited”), and after the execution of SOP, BO should reach the states of final_state_set; state_related_flag=0 means that SOP is not related to states and may be executed at any states (e.g., “Create New Order”, “Query Order Information”, etc).

A SOP should complete the following tasks:

(1) Business logics (e.g., CRUD) on BO;

(2) Write BO’s state information (after execution of SOP) into the state log.

3.8 Complex operation

A complex operation (COP) is an operation dealing with an association between objects and come down to two objects in the association (called Upriver and Downriver objects), denoted as COP :=<op_id, op_name, association_id, type>, in which association_id refers to the association that COP operates on, type represents one of the following:

(1) Push: SOP is triggered/executed by the upriver object;

(2) Pull: SOP is triggered/executed by the downriver object;

(3) Push/Pull: SOP may be triggered/executed by the upriver or downriver objects arbitrarily.

For example, a COP related to the association between “req. bill” and “plan bill” is “Generating procurement plans from requirements”, which is a pull SOP, i.e., this operation should be called in the interface of “plan bill” to generate new plan bills according to the data of selected req. bill.

The codes of a COP should complete the following tasks:

(1) Query upriver objects instances;

(2) Generate new downriver objects according to the association value (by user’s input), e.g., the planned quantity in Fig. 5(b);

(3) Write the association information into association log;

(4) Write the state information of related object instances into state log.
4 ESA development pattern based on business object association model

One of the primary goals of our model is to improve the agility of ESA, i.e., when associations between objects change frequently, ESA may adapt to such changes with high efficiency and low scales of code modifications.

In Fig. 2 we have presented the basic idea of our development process based on our model, and Fig. 6 shows the detailed process.

Based on the model and process, we have designed a code generation tool to support automatic code generation. The input of this tool is the association model, and the output is executable source codes (we adopt J2EE-based software architecture, in which user interfaces are JSP files, background business logic are EJB files, and MVC structure is used to compose these files together. The final system is running on Weblogic application server platform).

5 Conclusions

Our approach (model and tool) has been applied in several Enterprise Resource Planning (ERP) projects and has gained significant results, which is mainly reflected on the improvement of developing efficiency and code quality. According to the statistical data, the average second-round developing period of each sub-system (average 10-15 business objects) of ERP is reduced from 1 month to 12 days, i.e., the developing efficiency increased by 60%.

Our approach still has some insufficiencies that need further researches, e.g.,

1. Because of complexity of enterprise business, state logic and operation’s business logic should be formally defined in models, therefore the code generation tool make directly transform them into source codes;

2. Our approach could only deal with bill and resource objects; for report objects, because their data are from multiple data sources and the relationships between these data are comparatively complex, therefore we temporarily cannot treat with such objects;

3. Our approach only aims at numerical association, but for other types of associations we have not considered them yet.

Acknowledgments

Research works in this paper are partial supported by the Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) in China (Grant No. 20030213027) and the National Natural Science Foundation (NSF) of China (Grant No. 60573086).
References


Zhongjie Wang received his B.S., M.S. and Ph.D. degrees in computer science from Harbin Institute of Technology in 2000, 2002 and 2005 respectively. He is now a lecture in computer science in School of Computer Science and Technology at Harbin Institute of Technology (HIT), PRC. His research interest includes software engineering, software reuse, software reconfiguration, software component related techniques.