# Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences

Edited by
HUGH J. WATSON

1998

Vol VI

## ORGANIZATIONAL SYSTEMS and TECHNOLOGY TRACK

IEEE COMPUTER SOCIETY

IEEE

UNIVERSITY OF HAWAII

UNIVERSITY OF HAWAII AT MANOA · COLLEGE OF BUSINESS ADMINISTRATION

# Building and Using Object-Oriented Frameworks for Semi-Structured Domains: The Sales Promotion Domain as Example

Arco Dalebout
*Center for Information Technology in Marketing (C/IT/M)*
*Department of Marketing Management, RSM*
*Erasmus University*
adalebout@fac.fbk.eur.nl

Jos van Hillegersberg
*Department of Decision- and Information Sciences, RSM*
*Erasmus University*
jhillegersberg@fac.fbk.eur.nl

Berend Wierenga
*Center for Information Technology in Marketing (C/IT/M)*
*Department of Marketing Management, RSM*
*Erasmus University*
bwierenga@fac.fbk.eur.nl

## Abstract

*Object-oriented (OO) frameworks are considered an important step forward in developing software applications efficiently. Success of frameworks has however predominantly been limited to structured domains. This paper describes a method for developing OO domain frameworks for semi-structured domains. The complexity of such domains requires the use of more elaborate analysis and design techniques than those normally used in OO analysis and design. In the method described here, the knowledge of domain experts forms the basis for developing the framework. The OO framework is constructed on the design level using a CASE-tool. Using the design model, the framework can be customized and extended to develop a variety of applications for the domain. The approach is illustrated by developing a framework for the sales-promotions domain, and using this framework to build an application for product managers. It is concluded that the approach described here is beneficial to build and use OO frameworks for semi-structured domains.*

## 1. Introduction

Application frameworks receive growing interest in both literature and practice [4]. A framework is a set of prefabricated software building blocks that programmers can use, extend, and customize for specific computing solutions [8]. Object-Oriented (OO) frameworks consist of classes and interacting software objects. An OO framework can be customized by creating new subclasses through inheritance.

Successful development of OO frameworks has mostly been limited to application frameworks. These frameworks "encompass a horizontal slice of functionality that can be applied across client domains" [8]. Such frameworks contain for example file-access and user interface design functionality. Less attention has been paid to the development of domain frameworks. These frameworks "encompass a vertical slice of functionality for a particular client domain" [8]. The few examples of domain frameworks that exist are concerned with relatively structured domains. An example of an OO framework for building accounting applications is given in [5]. IBM's Shareable Frameworks initiative also focuses on the development of frameworks supporting well known core-business processes like ledgers, warehouse management and order management [3].

In this paper it is demonstrated how a framework was built and used for application development for the semi-structured domain of sales promotions. We argue that the presented approach is applicable to this type of domains in general. *It is the objective of the research described in this paper to demonstrate a method for the development and use of OO frameworks for semi-structured domains.*

We found that building frameworks for semi-structured domains requires more elaborate OO analysis and design techniques than those typically available in current methods. Most OO methods suggest that analysts should mark all the nouns they find in domain-related texts as candidate classes. In semi-structured domains, this easily leads to unmanageable numbers of classes. The intended use of frameworks for building different applications also inhibits using prototyping techniques to elicit domain objects. Next to that, the analysis and design is hindered by the long experience needed to comprehend a semi-structured domain. While software developers can analyze and understand structured domains like accounting within reasonable time, the gap between domain experts and software developers in semi-structured domains like sales

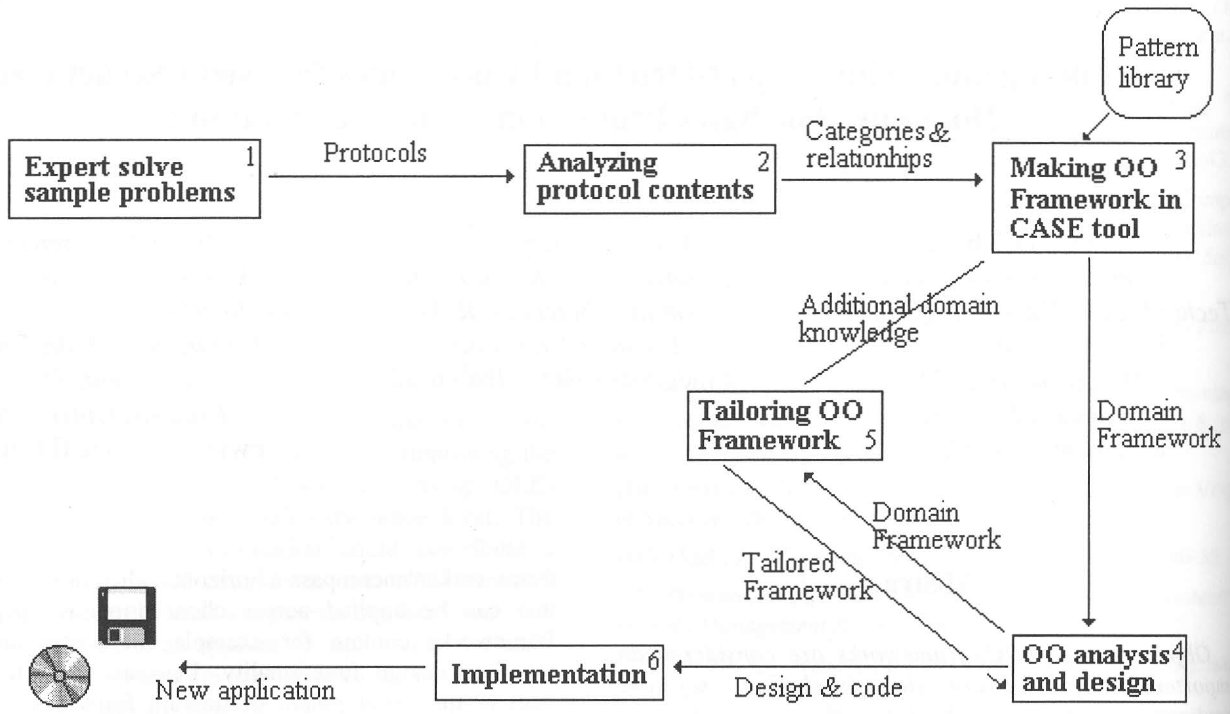


**Figure 1. Framework and application development method**

promotions, legal decision making or medical diagnosis is harder to bridge.

We propose to build OO frameworks based on the cognition of domain experts. Mental models of domain experts consist of building blocks that typically correspond to important concepts in the domain. Using content analysis, a technique from the social sciences, these concepts can be elicited. Figure 1 depicts the development method of the framework and the client application exhibited in this paper.

To capture the properties and operations of objects in a domain, a number of sample problems representative for the experts' job are devised. The think-aloud protocols of experts solving the sample problems form the basis for the OO analysis and design of the framework. Using content analysis, the protocols are transformed to an OO framework of the domain. The framework is constructed on the design level using a CASE-tool. For development of a client application, the framework can be customized and extended.

The framework we present focuses on sales promotion decision making, which is a prominent task of product managers. Sales promotions are marketing events aimed at having a direct impact on the buying behavior of consumers. Examples of sales promotions are price-offs, premiums, cash refunds, samplings, and coupons.

Sales promotion decision making is a typical example of a semi-structured, complex domain. Product managers act in a dynamic environment where products and actors quickly appear, disappear, and change position. Product managers frequently make semi-structured decisions under heavy time pressure. Mental models of product managers consist of building blocks that are typical marketing concepts: brands, prices, ads, sales, market share, or typical marketing actors: customers, competitors, producers, retailers. In this mental counterpart of the real world, the product manager replays, simulates, and forecasts things that happen in order to be able to respond to the opportunities and threats that occur.

The framework developed is a basis for the creation of various client applications, tailored to the specific characteristics of certain tasks, markets, or decision makers that we come across in reality.

The main part of this paper is dedicated to showing how we applied our method for framework development to the domain of sales promotion. The paper concludes by illustrating the use of an OO framework for the development of SPIDER, a case-based reasoning application for sales promotion decision makers.

## 2. A framework for sales promotions

### 2.1 Framework development step by step

In this section we will present the three steps of the framework development method (Figure 1):

*Step 1. Experts solve sample problems:* We propose an indirect but thorough way to utilize domain expertise for framework development. In this approach, think-aloud protocols of domain experts are used to construct a domain framework. A set of sample problems that can, more or less, be considered representative for the domain are collected or constructed and presented to domain experts. The solutions the experts produce are recorded on tape and transcribed so they can be used for further analysis. The output of this first step are the transcribed think-aloud protocols.

*Step 2. Analyze protocol contents:* The methodological framework of content analysis is used for the analysis of the protocol data we gathered in the previous step. The purpose of the analysis is to find out 'how domain experts conceptualize (make mental representations of) their domain.' In order to achieve this, the protocols gathered previously are coded by applying two procedures. The first procedure is meant to discover the so-called content categories. Nouns (or groups of words denoting nouns) that are related to the domain are looked up in the protocols. The nouns that have equal semantic meaning are put into one category. Each category receives a label or name that covers the semantics of the units it contains.

During the second procedure, all relationships that refer to one or more content categories, are extracted from the protocols. We need these relationships among the content categories, to be able to establish the relational and dynamic properties of the framework in the next step.

*Step 3. Creating the OO framework in a CASE-tool:* The framework should be modeled in a CASE environment capable of translating the model into different programming languages. In this way, the framework can be used in different projects, regardless of specific platform or language needs. If supported by the CASE-tool, the Unified Modeling Language (UML) [1] can be used to do the modeling.

The actual construction of the framework starts by mapping the top 20 or 30 content categories (ranking based on frequency of occurrence in the protocols) to object classes. This initial object model forms the basis for the rest of the modeling process.

Further refinement of the object model and the establishment of the relational and dynamic properties of the framework is done by using the relationships that were found in the protocols (step 2, second procedure). Within relationships, verbs relate one word or word sense to another, implying the way classes are related to one another. The text of a relationship is mapped to the OO model by a) checking whether the classes a relationship refers to are correctly represented in the initial object model and by b) modeling relevant verbs either as relationships between objects or as behavior.

### 2.2 A framework for the sales promotion domain

This section shows the application of the framework development method for development of an OO framework for sales promotions.

#### 2.2.1 Experts solve sample problems (step 1)

We devised two sample problems representative for the domain of sales promotions. Each interviewee (ten product managers and ten sales promotions authorities) had to solve a problem on the design of a sales promotion plan for the introduction of a new salad dressing and another one in which a sales promotion plan had to be devised for a troubled fashion magazine. The experts were asked to think aloud so their solutions could be recorded and transcribed.

The "Presto-Dress" problem (salad dressing) is briefly described in Exhibit 1. The "Marie-France" problem (fashion magazine) is not shown here, but the experts' solutions in response to that problem, are also analyzed and used for the modeling of the framework.

**The 'Presto-Dress' case in brief:** A salad dressing ('Presto-Dress') in a newly developed package concept, a transparent box with a handy spout, is going to be introduced The experts had to design a sales promotion plan for this introduction. The budget was Dfl.300,000.-

**Part of an expert protocol:** I'm going to campaign because what I really want is to drive my consumers to make trial purchases ... no added value promotions. That would distract attention from the product since added value is already in the product. Suppose I would give a CD that would, ..., no ..., I want to focus on my product and drive people to buy the product. If I want consumers to pay attention to my product, I could promote by giving an introduction discount. Or I would do a refund promotion. In other words: give the consumer cash money back immediately, or ...., I could promote by refunding the entire product price if people send in the bar-code.

**Exhibit 1. Presto-Dress Sample Problem with part of an expert protocol**

#### 2.2.2 Analyze protocol contents (step 2)

We used the theory of content analysis [6,9] for the analysis of the protocol data. In content analysis the many words of a text are classified into fewer content categories that are presumed to have similar meaning. On the basis of this categorization, the researcher tries to draw valid inferences from the text.

Exhibit 1 displays a small piece of a protocol transcript collected in step one. The twenty transcripts were coded by business school students enrolled in a course on sales promotion decision making. Each interview was coded by two students to assess coding reliability.

To subdivide the content of a text into separate elements, which can be classified, we need to delineate these elements. There are several ways to do that. We used two different coding units: the *first* was meant to discover important marketing concepts. We used plain - *words* (e.g., budget) or *word senses* (e.g., "the money that is available for the promotion") as coding units. After having extracted all words or word senses that had anything to do with sales promotion, the coders classified the units in categories according to equal semantic meaning. For example, "our budget," "the available 300,000," "the money we can spent," can be attributed the same semantic meaning (budget) and be put into one category. Each category finally received a name. Table 1 shows the top 30 content categories, indicating the number of units in each category. Brand, product, and promotion name appeared so frequently that counting them would have distracted the coders ("n.a." in Table 1).

The *second* coding procedure was performed to find relationships among the concepts. As coding units we used themes [9], which are components of relationships. For the purpose of this writing we will only consider the actual relationships. The relationships were noted down in tables with references to their locations in the actual text and the categories found in the previous step. Table 2 shows the relationships and their references to content categories that were extracted from the protocol fragment displayed in Exhibit 1.

**Table 1. Ranked average nr. of occurrences of categories in protocols (nr. protocols = 20)**

| *Rank* | *Category* | *Presto -Dress* | *Marie-France* | *Avg.* | *Rank* | *Category* | *Presto -Dress* | *Marie-France* | *Avg.* |
|---|---|---|---|---|---|---|---|---|---|
| 1. | Brand | n.a. | n.a. | n.a. | 16. | Market | 24 | 12 | 18 |
| 2. | Product | n.a. | n.a. | n.a. | 17. | Shape of Package | 27 | 0 | 14 |
| 3. | Promotion Name | n.a. | n.a. | n.a. | 18. | Promotion Strategy | 27 | 0 | 14 |
| 4. | Promotion Budget | 68 | 86 | 77 | 19. | Character | 0 | 17 | 9 |
| 5. | Sales | 3 | 102 | 53 | 20. | Proposition | 16 | 0 | 8 |
| 6. | Promotion Costs | 40 | 52 | 46 | 21. | Producer | 11 | 2 | 7 |
| 7. | Promotion Goal | 61 | 29 | 45 | 22. | Distribution | 11 | 0 | 6 |
| 8. | Consumer | 19 | 56 | 38 | 23. | Added Value | 10 | 0 | 5 |
| 9. | Advertiser | 0 | 71 | 36 | 24. | Reach | 5 | 5 | 5 |
| 10. | Price | 30 | 29 | 30 | 25. | Promotion Comm. | 10 | 0 | 5 |
| 11. | Package | 58 | 0 | 29 | 26. | Awareness | 7 | 2 | 5 |
| 12. | Trial | 44 | 4 | 24 | 27. | Promotion Prospect | 0 | 9 | 5 |
| 13. | Ad | 24 | 20 | 22 | 28. | Product Content | 4 | 4 | 4 |
| 14. | Market Share | 35 | 3 | 19 | 29. | Profit | 8 | 0 | 4 |
| 15. | Target Market | 3 | 34 | 19 | 30. | Promotion Location | 0 | 8 | 4 |

**Table 2. Sample relationships**

| *Ref.* | *Relationships* | *Categories referred to* |
|---|---|---|
| ns1-1 | I'm going to do a promotion campaign since I want consumers to make trial purchases | Promotion, Consumer, 'Trial purchase' |
| ns1-2 | No added value promotions because the added value is already in the product | 'Added value promotion', 'Product added value', Product |
| ns1-3 | If I want consumers to pay attention to my product I could promote by giving an introduction-discount | Consumer, Attention, Product, Promotion, 'Introduction discount' |
| ns1-4 | If I want consumers to pay attention to my product I could do a refund promotion | Consumer, Attention, Product, 'Refund promotion' |
| ns1-5 | If I want consumers to pay attention to my product I could promote by giving an immediate refund | Consumer, Attention, Product, Refund |
| ns1-6 | I could promote by refunding the entire product price if people send in the bar-code of the product | Refund, 'Product price', Bar-code |

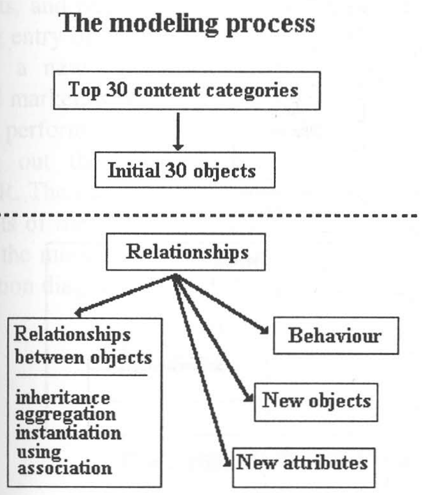


**Figure 2. The modeling process**

### 2.2.3 Creating the object-oriented framework in a CASE-tool (step 3)

After we reduced the set of nouns by putting them into content categories, we started building the sales promotions framework by mapping only the top 30 content categories (see Table 1) to object classes. These classes form the basis for the rest of the modeling process. The Unified Modeling Language (UML) [1] was used to model the framework.

The modeling process proceeded by refining this initial framework using the relationships that were found in the protocols (Table 2). Within relationships, verbs relate one word or word sense to another, implying the way classes are related to one another. As we saw in 2.1, the text of a relationship is mapped to the OO model by checking whether the classes the relationship refers to are correctly represented in the initial object model and by modeling relevant verbs either as relationships between objects or as behavior.

Figure 2, gives an overview of how the results of a content analysis were mapped onto an object model. The sales promotion domain framework model is too complex to remain comprehensible while being displayed in one diagram. The model is decomposed into six smaller units (see Figure 3), so-called packages, that contain all kinds of modeling elements [1]. Arrows between the packages indicate visibility relationships: if a client object in the Product package wants to make use of a server object in the Consumer package, then a visibility relationship must exist between both packages [4]. We will highlight some of the details of the contents of the Promotion package.

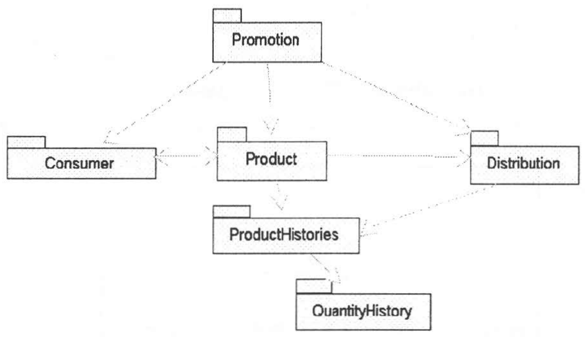


**Figure 3. The packages and their dependencies**

Figure 4 shows an explosion of the Promotion package and amongst others displays the sales promotion types the managers commented on in the interviews (for comprehensives many details are left out). The types were grouped based on similarity in characteristics and behavior. Promotions based on price discounts, for example, were put into one group (area surrounded by solid line), since they all share a mechanism that influences the product price. The other sales promotion types influence the added value of the product. A (virtual) pricing method was defined on the level of the abstract class and was overridden at the subclass level: Discount directly lowers the price, Coupon discounts the price when a coupon is handed over, Product plus promotions give an extra quantity of the product for the same price, and Refund gives a discount on the sale directly or after sending in a proof of purchase.

A special type of promotion often used by our interviewees was the joint promotion (area surrounded by dotted line). A joint promotion can be any type of promotion and is a coordinated effort among several different producers. The solutions managers proposed to the ‘Presto-Dress’ sample problem (see Exhibit 1) often had a joint character: giving away samples of the salad dressing when consumers buy lettuce or gluing coupons on packages of other products of the producer’s product range (WithOwnProduct).

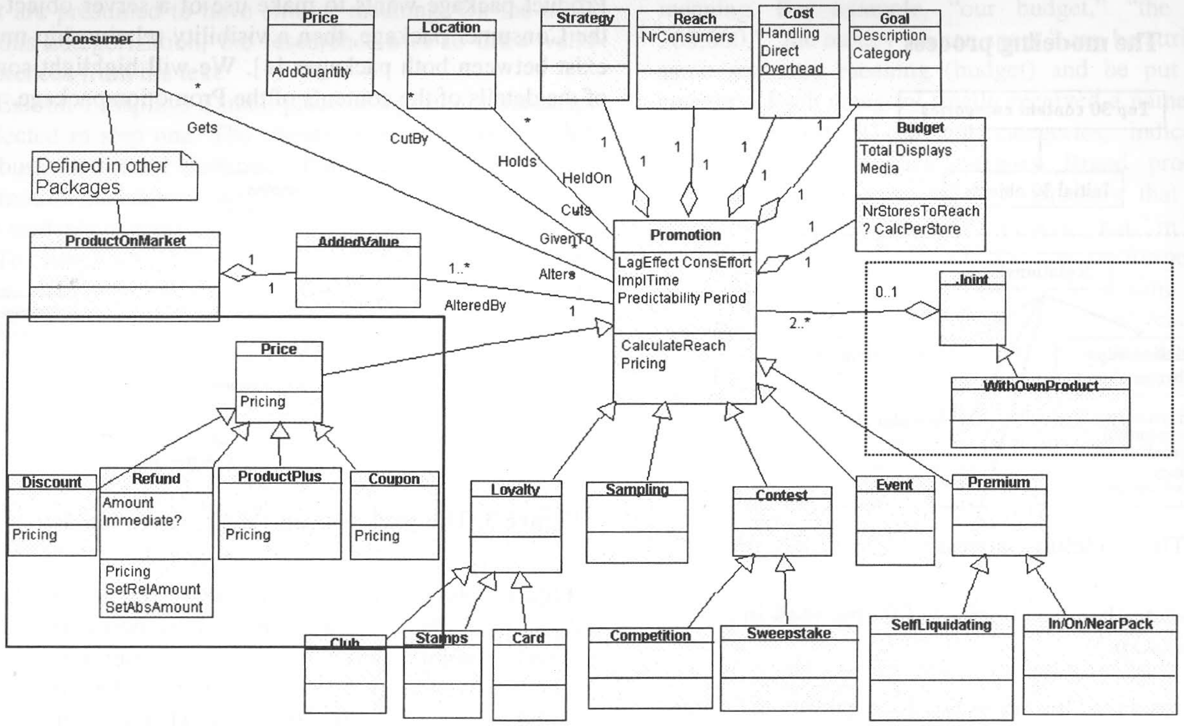


**Figure 4. The Promotion package**

The framework was modeled in MetaEdit, a CASE tool that supports the UML, which allows methodology engineering and adaptation, and generates source code in different OO-programming languages like C++, Smalltalk, Java and Delphi [7]. We found that using this tool has a number of advantages. First, models can easily be converted from one method to another. Second, the meta-level adaptability permits extra documentation fields (like extra comments, references to protocol relationships) to be added to the UML concepts. This domain information is crucial to enable future users (developers) to understand the framework. Third, source code generation facilitates adaptation of the framework on the design level whenever a new application is being created.

## 3. Application development

The sales promotion domain framework contains a basic application structure that developers of specific applications commonly have to develop themselves. Building applications by using the framework is faster; programmers can dedicate their efforts to the specific needs of application related issues. The extent to which the framework is reused by developers depends on the specific needs of the application and can vary from the reuse of one class definition to the reuse of a whole set of interacting objects. The use of the framework will now be illustrated by describing the development of SPIDER (Sales Promotions Intelligence Data EngineeR).

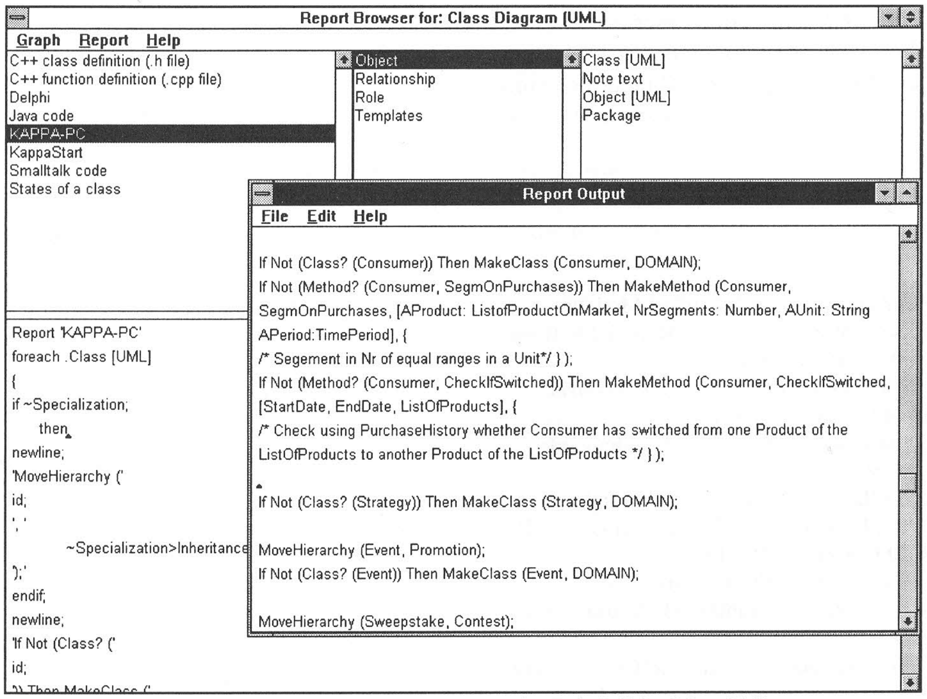
For SPIDER, we needed both a technique to represent domain and its knowledge and a reasoning mechanism for using that knowledge in the proper way. The sales promotion framework was used for the representation of the domain. This provided the developers, for instance, with the semantics for storing data, organizing it on the basis of the domain knowledge. Promotion Package objects are used to store data on the promotion itself (e.g., Type, Budget, Media), Consumer Package objects store data on consumers who participate in the promotion (e.g., Name, Address, AmountPurchased), Distribution objects store data on the retailers that adopt the promotion (e.g., Allowance, NrOfStores), and so on.

SPIDER is based on the problem solving technique known as case-based reasoning. Put simply, case-based reasoning is the computerized equivalent of what we refer to as reasoning on the basis of analogy when it is performed by humans. New problems are solved by using previously designed solutions to similar problems. A case-based reasoning system can serve as a central repository in which experiences can be collected. Algorithms are available for retrieval of the right cases at the right time. Certain case-based reasoning systems have a functionality for adapting cases to the new problem situations, which is useful since problems are seldom identical.

The domain framework is used by SPIDER by having its data stored in, and retrieved from, the framework's bjects. Objects once defined (as specific consumers, roducts, and retailers) can remain stored in the system, naking entry of new cases an easy job. Most data can be eused: a new sales promotion occurs in an already lefined market, was held by a known competitor, etc.

We performed a UML-use scenario analysis [1] for inding out the scope and required functionality of SPIDER. The use scenarios comprise the basic functional lements of the system. The scenarios were input for the est of the modeling process: drawing class diagrams and nteraction diagrams (step 4, Figure 1). After this stage an assessment was made on how the framework could be tailored for the specific application (step 5, Figure 1). Subsequent to this modeling, we generated code and currently we are addressing final implementation issues (step 6, Figure 1).

The case-based reasoning part of the system was implemented in KAPPA-PC, a hybrid environment combining multiple representation paradigms, including objects, rules, and procedures. MetaEdit could not generate code for this environment, but provides the possibility to customize code generation.


Report Browser for: Class Diagram (UML)

Graph Report Help

```
C++ class definition (.h file)
C++ function definition (.cpp file)
Delphi
Java code
KAPPA-PC
KappaStart
Smalltalk code
States of a class
```

```
Object
Relationship
Role
Templates
```

```
Class [UML]
Note text
Object [UML]
Package
```

```
Report 'KAPPA-PC'
foreach .Class [UML]
{
if ~Specialization;
     then
newline;
'MoveHierarchy ('
id;
','
          ~Specialization>Inheritance
');'
endif;
newline;
'If Not (Class? ('
id;
')) Then MakeClass ('
```

Report Output

File Edit Help

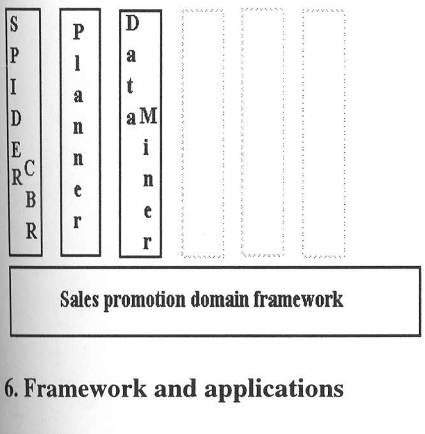
```
If Not (Class? (Consumer)) Then MakeClass (Consumer, DOMAIN);
If Not (Method? (Consumer, SegmOnPurchases)) Then MakeMethod (Consumer,
SegmOnPurchases, [AProduct: ListofProductOnMarket, NrSegments: Number, AUnit: String
APeriod:TimePeriod], {
/* Segement in Nr of equal ranges in a Unit*/ } );
If Not (Method? (Consumer, CheckIfSwitched)) Then MakeMethod (Consumer, CheckIfSwitched,
[StartDate, EndDate, ListOfProducts], {
/* Check using PurchaseHistory whether Consumer has switched from one Product of the
ListOfProducts to another Product of the ListOfProducts */ } );

If Not (Class? (Strategy)) Then MakeClass (Strategy, DOMAIN);

MoveHierarchy (Event, Promotion);
If Not (Class? (Event)) Then MakeClass (Event, DOMAIN);

MoveHierarchy (Sweepstake, Contest);
```

**Figure 5. Code generation for KAPPA-PC in MetaEdit**



**igure 6. Framework and applications**

In Figure 5 a screendump is shown of how MetaEdit generates KAPPA-PC code based on custom-made specifications. In the top left area of the screen you see the languages for which code can be generated. In the lower left area you can see part of the custom-made instructions that MetaEdit uses to generate KAPPA-PC code. The screen on top, finally, shows code that was generated, and that was used to build the case-based reasoning system for the sales promotion domain.

Figure 6 depicts a situation in which multiple applications are based on the framework. Next to SPIDER, a sales promotion planner and a data mining system subclass from the domain framework. Changes made to the domain model can be inherited by all applications.

## 4. Conclusions

The research described in the paper introduced a method for the development and use of OO frameworks for semi-structured domains. To illustrate the approach, an OO domain framework for sales promotions was constructed and used. The development method starts from protocols of experts solving sample problems. Then, using content analysis, the vocabulary of the domain is mapped onto an object model. In building the OO framework, content analysis provides a much more powerful vehicle for finding classes and relationships than many of the guidelines given in OO literature. The use of the framework was demonstrated by showing how the platform and language independent domain framework was used to generate code for a specific application (case-based reasoning for sales promotion) in a specific environment (KAPPA-PC). The work described here can be applied to other decision domains with a semi-structured character, for example investment decisions, management development, personnel selection, and legal problem solving.

## 5. References

[1] Booch, G. and Rumbaugh, J. (1997) Unified Modeling Language : UML Semantics Version 1.0. Rational Software Corporation. WWW: www.rational.com.

[2] Fowler, M. (1997) *Analysis Patterns: Reusable Object Models*, Reading, MA: Addison Wesley.

[3] IBM (1997) The San Francisco Project (IBM Shareable Frameworks). IBM. WWW: http://www.ibm.com/Java/Sanfrancisco/index.html.

[4] Johnson, R.E. (1993) How to design Frameworks. OOPSLA 1993 Tutorial, Washington: WWW: http://st-www.cs.uiuc.edu/users/ johnson/frameworks.html.

[5] Keefer, P. (1994) The Accounts framework. Master Thesis, University of Illinois: WWW:ftp://st.cs.uiuc.edu/pub/Smalltalk/st80_vw/accounts/thesis.ps.

[6] Krippendorff, K. (1980) *Content analysis, an Introduction to its Methodology*, California: Saga Publications.

[7] Smolander, K., Lyytinen, K., Veli-Pekka, T. and Marttiin, P. (1991) MetaEdit --- A flexible graphical environment for Methodology Modelling. In: Andersen, R., Bubenko jr., J.A. and Solvberg, A. (Eds.) *Advanced Information Systems Engineering, Third International Conference CAiSE'91: Proceedings*, pp. 168-193. Berlin: Springer-Verlag.

[8] Taligent, I. (1997) Building Object-Oriented Frameworks. Taligent. WWW: http://www.taligent.com/.

[9] Weber, R.P. (1985) *Basic Content Analysis*, Beverly Hills: Sage Publications.