BORM – overview of the methodology and case study of the agrarian information system

BORM – přehled metodolologie a případová studie informačního systému pro agrární komoru

V. MERUNKA

Czech University of Agriculture, Prague, Czech Republic

Abstract: BORM (Business Object Relationship Modelling) is a methodology developed to capture the knowledge of process-based business systems. It has been in development since 1993 and has proved an effective method, which is popular with both users and analysts. This paper presents BORM, its tools, and methods via a case study of the agrarian information system. BORM is based on the combination of object-oriented approach and process-based modelling. Also, an advantage of BORM is the small number of concepts required combined with a considerable expressiveness. In this way, BORM is in the tradition of pure approach established over the past years by structured modeling techniques.

Keywords: Business Object Relation Modelling (BORM), object-oriented approach, process-based analysis, information system development, business processes, requirement engineering

Abstrakt: BORM je metodologie pro vývoj business systémů, která je založena na získávání znalostí o procesech. Od roku 1993, kdy je metodologie vyvíjena, byla prokázána jako efektivní a oblíbený přístup jak mezi uživateli, tak i mezi analytiky. Tento článek stručně prezentuje vlastní metodologii, její nástroje a postupy na příkladě případové studie informačního systému pro agrární komoru. BORM je založen na kombinaci objektově orientovaného přístupu a procesního modelování. Jednou z jeho dalších z výhod je malý počet požadovaných pojmů, které však mají bohaté vyjadřovací možnosti. Z tohoto pohledu BORM navazuje svým čistým a jednoduchým přístupem na tradici zavedenou před lety technikami strukturované analýzy informačních systémů.

Klíčová slova: BORM, objektově orientovaný přístup, procesně založená analýza, tvorba informačních systémů, podnikové procesy, získávání a modelování požadavků

INTRODUCTION

The attitude of business towards Information Technology (IT) is constantly changing towards being more and more sophisticated. Systems and tools become available. Additionally, there is a constant exchange of ideas between the IT and the business communities arising out of the development of knowledge-based systems. One such example is the method presented here, which was originally developed to capture the knowledge necessary for the development of IT systems but which has revealed an increasing potential for more general knowledge based system development. The work described here originally started in 1993 and was intended to provide seamless support for the building of object oriented software systems based on pure object-oriented languages, such as 'Smalltalk' (Hunt 1997; Hopkins, Horan 1995; Beck 1997), together with pure object databases, such as 'Gemstone'. It is now realised that this method also has a significant potential in capturing knowledge of business processes, business data and business issues (Bahrami 1999, Gray 1992, Catell 1994).

In our experience in the period of 90', during the work on major projects, IS analysts face the problem when not all system requirements are known at the start of the project and the customer expects that discovery and refinement thereof will be part of the project. The problem is even more complicated because the function of the major systems built has impact on the very organizational and management structure of a company or organization where the system is implemented – such as new or modified job positions, management changes, new positions, new departments, etc. Therefore it is desirable to address also the change of these related structures during the work on information systems (Blaha 1998; Shriver, Wegner 1987).

Process models composed from business objects represent a proven and actually used method of analysis, design and implementation of organizational changes with active participation of the customers and with relat-

Table 1. BORN project

Project	NSF	NS	NPD	NO	ANS	ANA
National Agrarian Chamber (analysis and design of software for commodity market public information system)	7	7	7	6	4	4
Hospital complex (BPR of organization structure)	6	12	12	8	10	12
TV and radio broadcasting company (BPR and company transformation for open market)	4	9	9	14	8	8
Regional electricity distribution company (customer information system analysis)	12	19	19	23	12	12
Regional electricity distribution company (failure handling information system analysis and prototype implementation)	19	31	34	27	13	14
Regional gas distribution company (BPR of all company)	28	81	97	210	11	12
Regional gas distribution company (BPR of all company)	23	60	63	120	12	12

Key

NSF = Number of system functions	NO = Number of objects[1] (participants)
NS = Number of scenarios	ANS = Average number of states per object
NPD = Number of process diagrams	ANA = Average number of activities[2] per object

[1] Only objects having activities, objects realizing data flows in processes are not included here
[2] in BPR projects, each activity includes about 4-6 additional business related entities (goal, job position, success factor, …)

ed development of the information system (Taylor 1995; Darnton, Darnton 1997; Partridge 1996).

The methods described here have been used to successfully develop a wide range of systems of diverse sizes. In this paper, the case study of analysis and design of software for fruit market public information system for National Agrarian Chamber of the Czech Republic is presented.

Experience with practical projects performed in BORM by Deloitte&Touche Czech Republic suggests that the description of business environments necessary to prompt new activities involves significant knowledge based content. The systems mentioned above range through all sizes of software development and the selection of them can be seen from the Table 1.

The method presented in this paper has proved to be effective and beneficiary in the process of describing and subsequently understanding how real business systems evolve (Belin, Suchman 1997; Mellor, Shlaer 1997). Such knowledge is the key for the success of any business and is especially crucial for those employees whose responsibility is business development.

The *Business Object Relation Modelling* (BORM) method provides a tool to capture knowledge and to present it in a such way that, the authors believe, is far more effective than other business processes, data, or functional modelling methods. This increase in effectiveness is due largely to the use of a unified and simple method for presenting all aspects of the relevant model.

To fully understand how a business works requires a significant detailed knowledge, which can be captured in a knowledge base. BORM provides an effective tool to facilitate knowledge elucidation and an effective graphical language for describing the structures and interactions within the knowledge so elicitated.

One problem, common for both the designers of information systems and knowledge elucidation, is the 'conceptual gap' or 'representational mismatch', which occurs when communicating with a user or knowledge expert.

Goldberg (1995) uses the term 'Concept Space' to describe what the user/experts believe, assume or know to be the case. The 'Articulation Space' is what the expert/user communicates in response to the analyst's questions. The analyst then constructs a model to feed back to the user/expert their mental model of the concept space, which they construct out of the information presented in the articulation space. This model is known as the 'Analyst Space'. The difference between this analyst's model and the user space is the concept gap.

To a certain extent, part of this gap is unbridgeable; we cannot easily reduce the gap between concept and articulation space, as these exist in the user/expert's head. It is true, however, that the languages, natural and graphical, used by the analyst in representing this model are a vital component in the user/expert's ability to validate this model against its own concept space. The subset of BORM methods introduced here is, we believe, a useful tool in closing this gap.

WHAT ARE THE PROBLEMS WITH OBJECT ORIENTED DESIGN METHODOLOGIES?

The first and we think the major problem with Object Oriented methodologies arises in the initial stages of

system development cycle (Coterrell, Hughes 1995; Cantor 1998; Royce 1998; Davis 1993; Kotonya, Sommerville 1999). The initial stage of any Object Oriented design methodologies should be concerned with two tasks. The first is the specification of the requirements for the system. The second is the construction of an initial object model, often called an *essential object* or *conceptual model* built out of a set of domain specific objects known as *essential objects.* Both these tasks should be carried out with the active participation of the stakeholders, in order to ensure that the correct system is being developed. Consequently, any tools or diagrams used at these early stages should be meaningful to the stakeholders, many of whom are not 'computer system literate'.

The most common technique for requirements specification in current object oriented methodologies is Use Case Modelling. Indeed Use Cases are often the foundation of most Object Oriented development methods (Jacobson 1992). Use Case Modelling is concerned with the identification of actors which are external entities, who interact with the system. This means that in order to employ Use Case Modelling, it is necessary for developers to already know the system boundary and to distinguish between entities, which are internal and external to that boundary. It is our experience that the correct identification of the system boundary is a non-trivial task and can only take place at the end of the requirements specification stage.

Use Case Modelling is essentially a text-based system, any diagrams employed do not contain any significant information but only identify the actors involved in each use case. Neither is it an object-oriented process as the Use Cases determined could be subsequently developed in any programming paradigm. In addition, Use Case Modelling is often insufficient by itself to fully support the depths required for initial system specification. Fowler (1999) highlights some deficiencies in the Use Case approach, suggesting that Use Case diagrams, if they are to convey all the necessary information, need supplementation by Sequence and Activity diagrams. These modifications to Use Case Analysis would result in a number of different diagrams that are used initially to define the interaction between any proposed system and its users. There are many views on the effectiveness of Use Cases as a first stage in System Design. Simons and Graham (1999) for example describe a situation where Use Case Modelling obscures the true business logic of a system.

The BORM approach is based on the fundamental concept of process modelling. This follows from the belief that it in necessary, for the deployment of a new system, not to view that system in isolation, but to view it in the context of the companies total organizational environment. A new system, when introduced into an organisation, will normally totally change the way that the organisation operates. In addition, a BORM process model is object oriented from the beginning and is defined in an easy to understand graphical notation. From the process model, scenarios can be developed. Scenarios were originally developed in OBA to capture, similar information to that presented in Use Cases. A Scenario, however, is an *instance* of a User Interaction whereas a Use Case is more like a procedural description of a *type* of user interaction. Our experiences on the projects listed above suggest that the process way of thinking is more natural to business employee. Consequently, stakeholders in the proposed system can more easily understand BORM models and consequently make a greater contribution to the correctness of the system design.

In BORM, any initial diagram supports only the problem domain specific concepts; any software-orientated concepts are left until later in the modelling process. In addition, in the early stages BORM uses a single diagram that embodies the same information as the numerous diagrams used by other methodologies, This is an attempt to make it easier for the user to form a complete understanding of the interaction of the various system components.

In BORM, concepts and their notation change as the development process proceeds. This is in sharp contrast with UML, which claims to be a universal system in that the same notation is used for analysis, design and documenting the implementation. Our reasons for changing notation are based on the observation that this universality of the UML's notation hinders the design process. In this, we are in broad agreement with the criticism of this aspect of UML expressed by Simons and Graham (1999).

The second problem that we find with most development methodologies is that, during subsequent stages, they require a number of different diagrams to fully describe the system. Each diagram is used to model an independent aspect of the system. Thus, we have one diagram for the objects static structure and a second one for the state changes of particular objects. One diagram showing the message passing between objects and a further diagram to model the activities the system must perform. The fundamental principle of Object oriented systems is that of encapsulation. This means that all object's data values are stored in the same place as the functions (methods) that manipulate them. The synergy created by this unification of data and functionality leads to greater understanding of the situation being modelled, and to a correctly designed solution being developed.

Each diagram is a visual representation of an abstract model, which exists in the brain of the analyst. Developers use diagrams to communicate with customer and other designers. They are the basis of the *Analyst Space* defined earlier. If this model is object oriented in nature, its representation should reflect it and must not require the viewer to deduce this fact from a number of different diagrams, each of which reveals one particular aspect of the object nature of the model.

Finally, the modelling concepts used in most development methodologies are used throughout the system development cycle. Moreover, these notations tend to be specifically designed to represent, at an abstract level, concepts from object oriented programming languages.

For example, the models used in OMT (Rumbaugh et al. 1999, Derr 1995) or UML (Booch et al. 1998, Rumbaugh

at al. 1999) can use quantifiers, link classes, external relations, aggregations, etc. While many of these concepts are necessary for software implementation in hybrid object-oriented programming languages such as Java, C++ or C#, they are too 'computer-oriented' to be useful in capturing the primary system information. Such diagrams are often 'conceptually rich' and difficult for the customer and other 'non computer' people to fully understand. There is of course no compulsion to use these features, however, in our experience, software designers will often use all the facilities provided without any consideration as to their appropriateness. The use of complex concepts too early in the design process often compromises the requirements determined for the system, since users find themselves constricted by the programming nature of the models and consequently are unable to fully articulate their needs. In Simons and Graham (1999), speaking of UML, it is stated that "Developers often take the most concrete example of notational element in use ... and retrofit these interpretations higher up in the analysis process".

If we compare standard Entity-Relation Diagram (ERD) (Carteret, Vidgen 1995; Date 1995) with 'object-class diagram' used in OMT (Derr 1995) or UML (Booch et al. 1998), we find that ERD only uses three basic, but powerful concepts. Object-class diagram, on the other hand, generally uses about twenty different concepts, spread over a number of different levels of abstraction.

In the analysis phase, we need to acquire the best possible knowledge of the problem formulation, and there the implementation details may cause trouble. On the other hand, in the design phase we need to focus on implementing the outputs from the analysis but we do not need to know some aspects of the reality modeled.

Underestimation of the model differences in the individual phases of development of an information system results – in some instances of "real programmers" – in such a simplification where the analysis using UML is viewed as mere graphic representation of the future software code – typically in C++. Analytic models are then used not to specify the problem formulation with the potential users of the system who are also stressed by the complexity of the models that are presented to them. Projects in UML very often suffer from this problem.

THE BORM APPROACH

BORM, like other OOA&D methodologies, is based on the spiral model for the development life cycle (Eriksson, Penker 2000). One loop of the object-oriented spiral model contains stages of *strategic analysis, initial analysis, advance analysis, initial design, advanced design, implementation and testing* (see Figure 1).

The first three stages are collectively refereed to as the *expansion stages*. Expansion ends with the finalising of the detailed analysis conceptual model, which fully describes the solution to the problem from the requirements point of view (Yourdon 1995).

The remaining stages are called the *consolidation stages.* They are concerned with the process of developing from "expanded ideas" to a working application. During these stages the previously completed conceptual model is transformed, step-by-step, refined, reduced and finalised into a software design. In contrast to other methodologies, in BORM the object-oriented design stage is fluently connected to implementation without any sharp discontinuity, similar to the smooth transition between Object Oriented Analysis and Object Oriented Design. As a consequence, we can consider the program coding as at the last and the most detailed phase of the design process.

During the progress around the loop, the developer may undertake small excursions (little spirals out from the main spiral) into the implementation of smaller partial applications, used to develop, test and tune the pro-

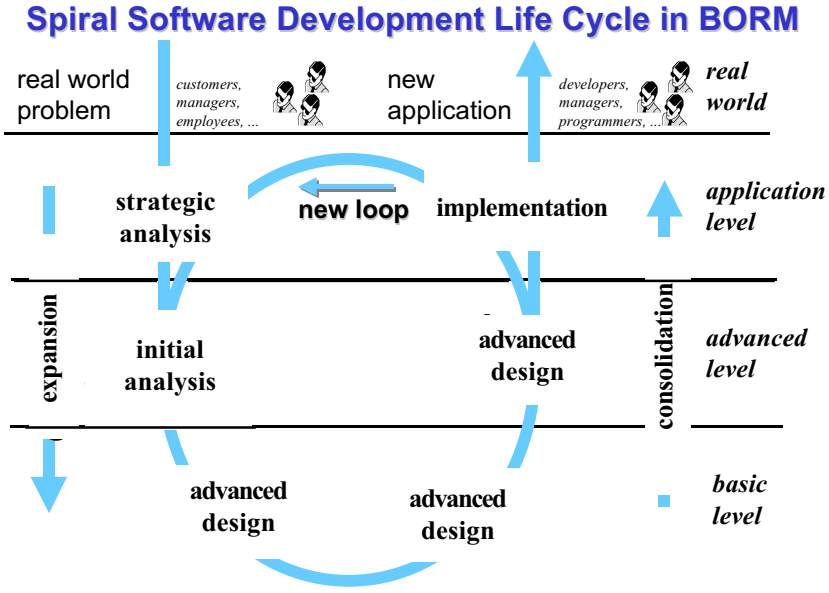


Figure 1. BORM stages

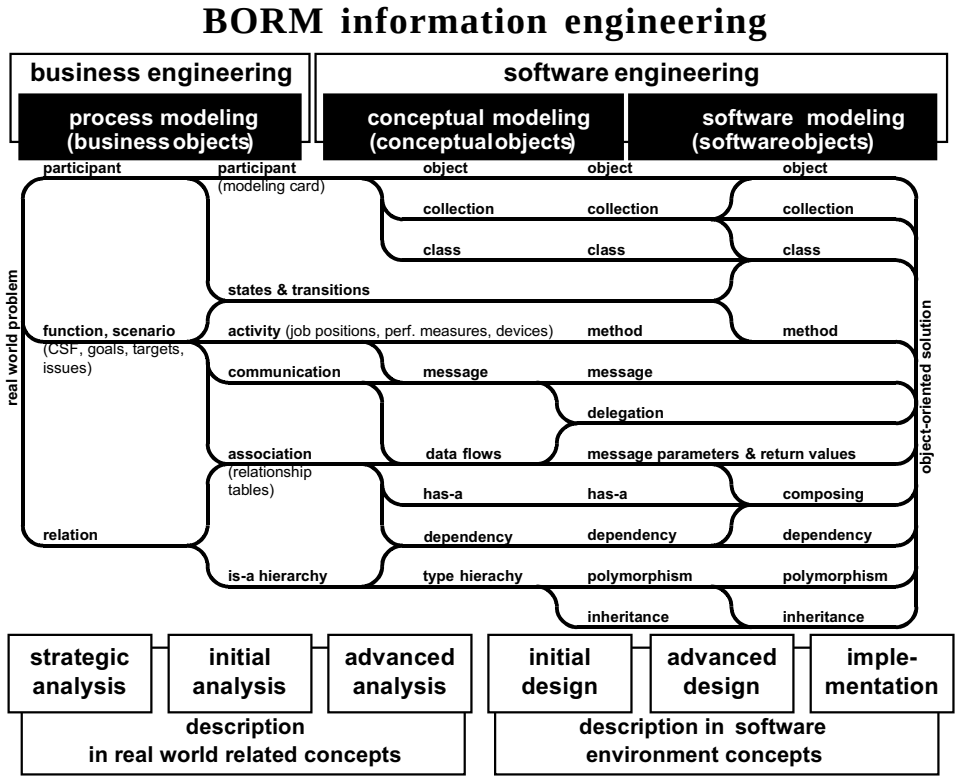


Figure 2. BORM evolution of concepts

gram incrementally by using previously completed modules.

The behaviour of any *prototype* (in BORM we prefer the more accurate name "*deliverable*") is also interesting. Every finalised application is also a deliverable and may be used as a basis for generating further requirements, which, in turn, leads to a new development loop.

Every object is viewed as a machine with *states* and *transitions* dependent on the behaviour of other objects. Each state is defined by its semantic rule over object data associations and each transition is defined by its behaviour, necessary to transform the object from its initial to its terminal state. Consequently BORM objects have the characteristics of Mealy-type automaton (Coterrell, Hughes 1995). Business object diagram accents the *mutual relationships* (communications and associations) of states and transitions of objects in the modelled system.

In BORM (see Figure 2), it is possible for each concept to have some of the following:

1. *A Set of Predecessor Concepts* from which it could be derived by an appropriate technique and a *Set of successor concepts*, which could be derived from it by an appropriate technique. For example a conceptual object composition from a business object association.
2. *A Validity Range* – the phases (of the development process) where it is appropriate. State –Transition diagrams for example are used extensively used in business conceptual modelling but are not supported by any current programming language.
3. *A Set of Techniques and Ru*les, which guide the step-by-step transformation and the concept revisions between the system development phases. These are the following:

a) *Object Behaviour Analysis*; which is a technique for transforming the initial informal problem description into the first object-oriented representation.

b) *Behavioural Constraints*; which is a set of determining rules which describes the set of possible transformation of the initial model into more detailed form with precisely specified object hierarchies like inheritance, dependency, aggregation etc.

c) *Pattern Application* which helps to synthesise the analysis object model by the inclusion of object patterns.

d) *Set of Structural Transformations* (Class Refactoring, Hierarchies Conversions and Substitutions, solving legacy problems solving programming environment constraints.) which are aimed at the final transformation of the detailed design model into a form acceptable in the implementation environment (programming language, database, user interface, operating system, etc.).

CASE STUDY – COMMODITY MARKET PUBLIC INFORMATION SYSTEM

In this part, we present the selected subset of project documentation of this information system. All diagrams were drawn in BORM extended standard of UML. Program code is written in the simplified Smalltalk-80 programming language.

System Processes - Objectives of Agrarian Chamber

1. to obtain governmental financial support to enable tailoring commodity production more closely to market demands
2. to convince customers that all actions are fundamentally for customer benefit
3. to obtain data and money from the government to support consultancy activities for farmers
4. to obtain reliable data from government sources to inform the decision making process
5. to increase overall sectorial revenue
6. to encourage home production of key commodities
7. to monitor conditions check to loans for farmers and to lobby to improve such conditions

OBJECT BEHAVIOR ANALYSIS OF BUSINESS OBJECTS

MODELING CARDS

Farmer
---
produce food
make profit
make decisions
report compulsory data

Loan Guarantee Board
Intervention Board
Bank
Agrarian Chamber
Wholesale
Retail
Commodity Exchange
Statistical Office

Agrarian Chamber
---
lobby
represent farmers
perform surveys

Parliament
Government
Statistical Office
Research Institutions & Universities

Commodity Exchange
---
trade

Farmer
Food Industry
Wholesale

Bank
---
distribute loans

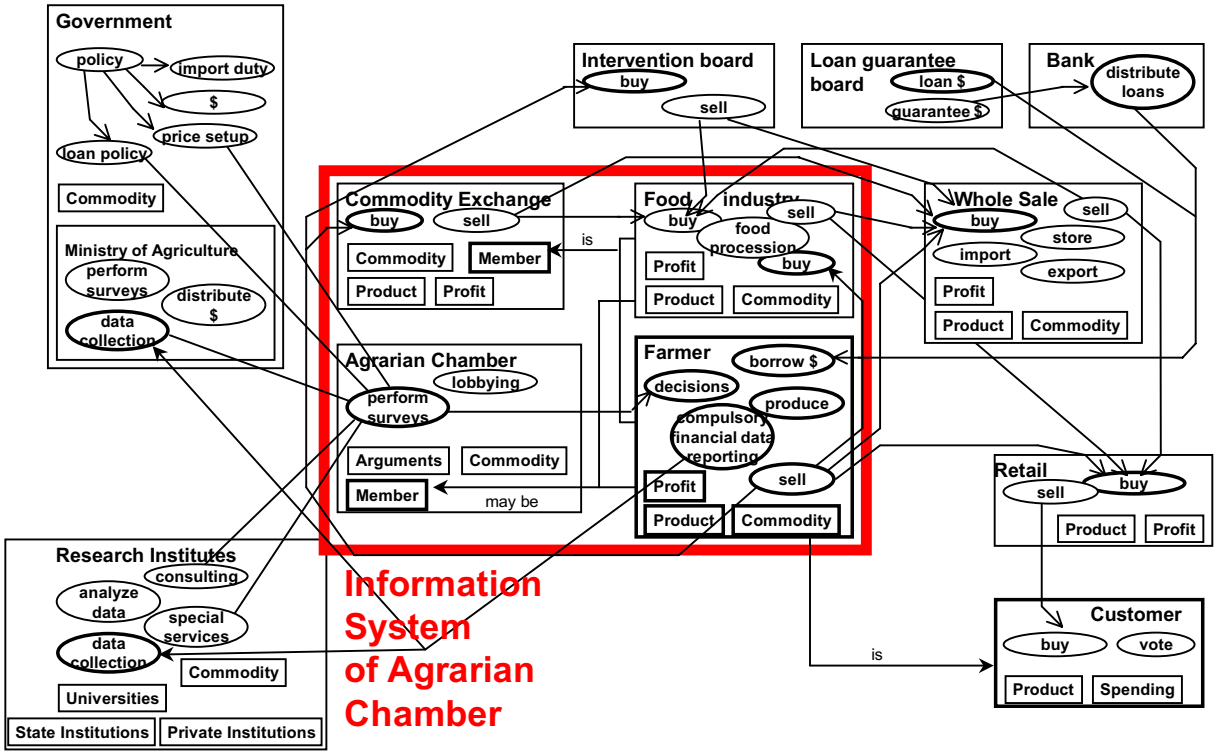
Government

STRATEGIC ANALYSIS OBJECT BEHAVIOR ANALYSIS OF BUSINESS OBJECTS

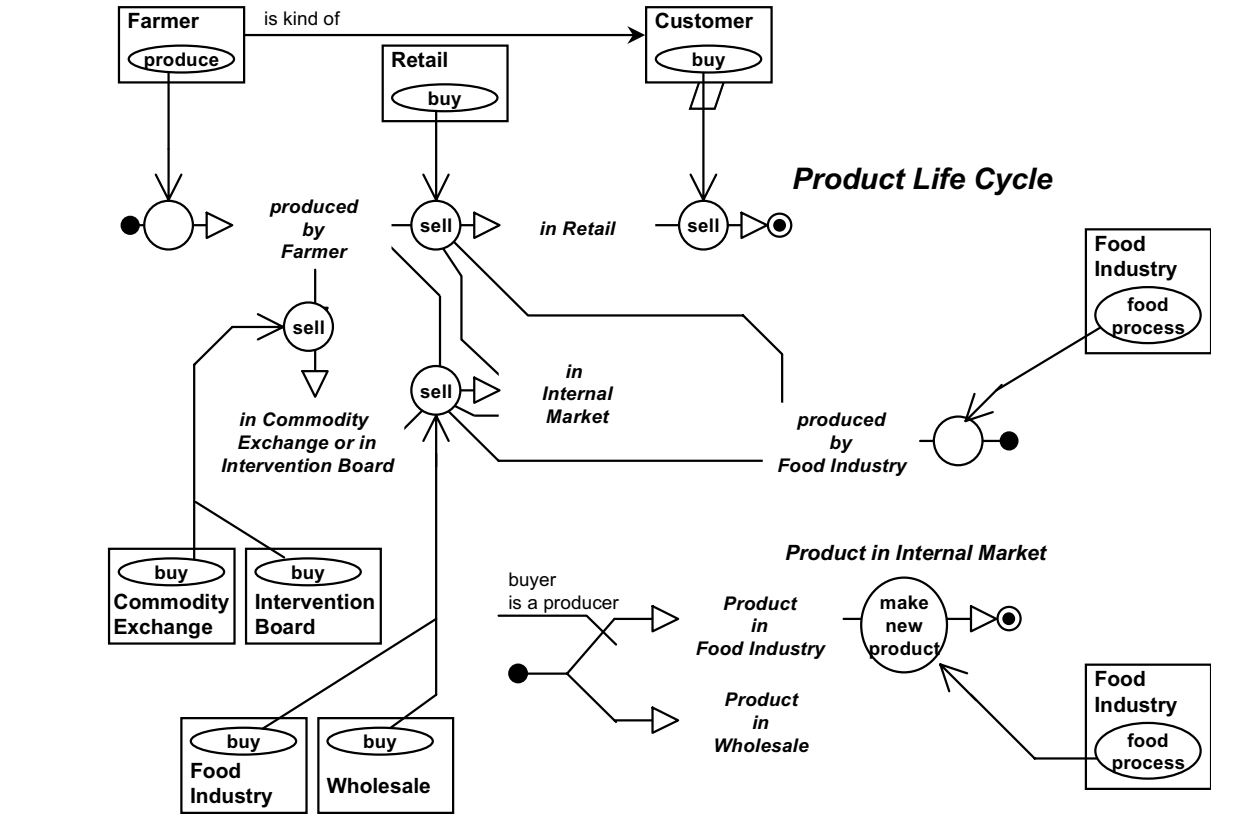
OBJECT RELATIONSHIPS

object	relationship	associated with
Farmer	may be a member of	Agrarian Chamber
Farmer	produce	Product
Farmer	sells	Product
Farmer	decides to produce	Product
Farmer	makes	Profit
Farmer	may sell product to	Commodity Exchange
Farmer	may sell product to	Food Industry
Farmer	may sell product to	Wholesale
Farmer	may sell product to	Retail
Farmer	may sell product to	Intervention Board
Farmer	borrow the money to produce from	Bank
Farmer	borrow the money to produce from	Loan Guarantee Board
Farmer	uses info for decisions making from	Agrarian Chamber
Farmer	reports compulsory financial data to	Statistical Office
Farmer	reports compulsory financial data to	Ministry of Agriculture
Agrarian Chamber	has members from	Farmer
Agrarian Chamber	performs surveys of	Farmer

BUSINESS OBJECTS IN AGRARIAN SECTOR
"FARMER" AND ITS RELATIONSHIP AND INFORMATION SYSTEM BOUNDARIES

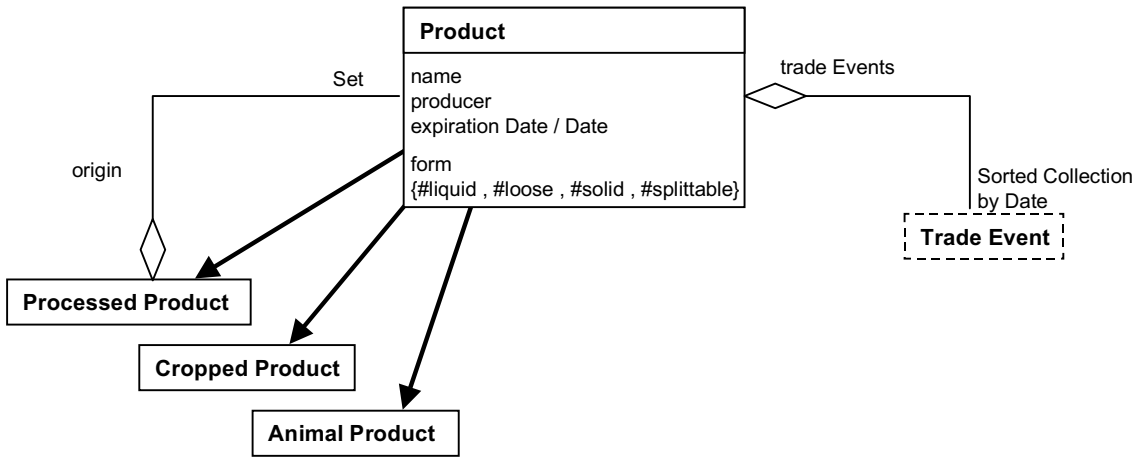


BUSINESS OBJECTS - INITIAL ANALYSIS
OBJECT LIFE CYCLES (RULES & EVENTS)

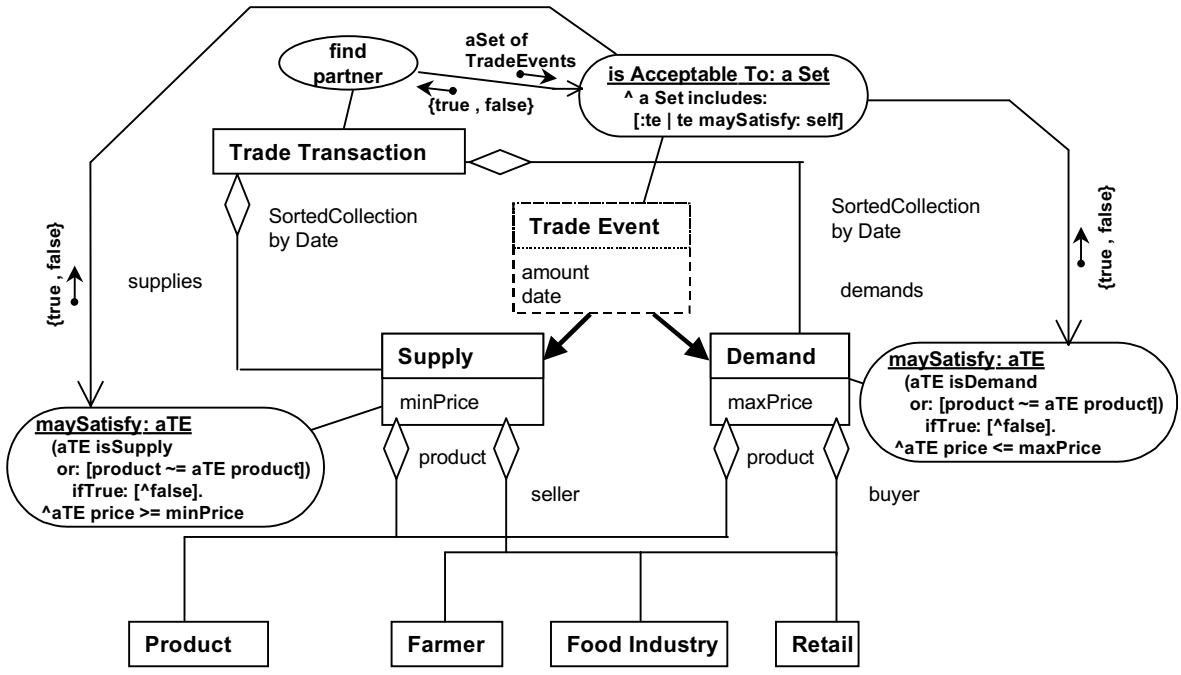


produced by farmer	in retail	in commodity exchange or in intervention board	in internal market	produced by food industry	product in food industry	product in wholesale

PRODUCT - DESIGN
OBJECT RELATIONSHIP DIAGRAM
OBJECT HIERARCHY



TRADE TRANSACTION - DESIGN
OBJECT RELATIONSHIP DIAGRAM
BEHAVIORAL VIEW



THE ADVANTAGES OF BORM

1. BORM follows the Process-oriented approach, which has proved to be beneficial in software development. Generally, the process-oriented approach lead to a faster and more comprehensive analysis of the problem being solved.
2. In our experience, stakeholders from the problem domain are able to understand the BORM approach very quickly, normally a one hour introduction at the start of analysis is enough)
3. In Deloitte & Touche (Prague office) a business consulting team has worked for the past three to four years using the BORM system as well as ARIS and Ratio-

nal's Objectory/Unified method. They have found BORM to be in average 3–4 times faster in carrying out the analysis phase compared to other methods.

4. The methodology is easily acceptable to domain experts, analysis consultants and developers. Because BORM is based on a step-by-step transformation of the model and in each phase only a limited and consistent subset of BORM concepts are used.
5. BORM has been used enthusiastically by Smalltalk and Java programmers and by non-relational object database programmers. One feature of BORM they find attractive is the way it exploits collection concepts not just classes and the way that these collection classes are seamlessly integrated into the development environment. (Compare with multi-objects in UML).
6. BORM has more object hierarchies (polymorphism, is-a, dependency …) than other methods which usually only provide concepts supported by programming languages. Usually only object inheritance is supported.

These last two features provide a much richer language with which to express modelling ideas.

FINAL COMMENTS

Today, when improved visual programming tools combined with the support of rapid application development environments are available, it would appear that the whole software development process is becoming easier. This statement is true, however, only for those cases where the complexity of the solution and of users' requirements is relatively simple. Business systems developed for real companies often have a much higher level of complexity which make development much more difficult. Consequently, it is essential (*from the software developer's viewpoint*) to improve the initial phases of software development.

Until recently, it was correctly assumed that conceptual modelling tools and techniques were used through all stages of project development, from the initial phase to the eventual implementation. However, the position of conceptual modelling is currently being used solely in the implementation phase, as a result of the evolution of software development tools. The analysis is now being performed using newly developed techniques and "business" objects modelling tools.

We believe that Object-oriented programming has changed not only the system development but also all of computer science (Booch 1994, Cox 1986). In software development, any team must be well organised, with clear and common goals. Managers of such projects must be clear about the potential benefits as well as understanding the management Object-oriented development.

Object-oriented programming can help in the development of a large system by significantly reducing the developing and maintenance time. But the adoption of Object-oriented programming requires considerable developments not only in technical knowledge but also managerial and cultural realignment; such changes can only be achieved by suitable training combined with the use of well designed and easy to use software development tools like those described here.

CONCLUSION

Currently there is not a 'standard solution' to the problem of gathering and representing business knowledge. Our approach, described here, developed out of business experience and enhanced by graphic models with clear connection towards the system development seems to be a promising candidate for such a standard. The notation we propose may serve not only as a tool for formal representation of modelled information, but also, as we have demonstrated, as a useful tool for communicating with developers and experts from the problem domain (managers, employees, etc.). The key advantages of BORM are its graphic models of knowledge representation, which provides easy and effective feedback. There are also clear rules how to progress through the system development process using this knowledge representation.

REFERENCES

Bahrami (1999): Object Oriented System Development. McGraw-Hill; ISBN: 0-071-16090-6.

Beck K. (1997): Smalltalk Best Practice Patterns. Prentice Hall; ISBN 0-13-476904-X.

Bellin D., Suchman S.S. (1997): The CRC Card Book. Addison-Wesley; ISBN 0-201-89535-8.

Blaha M., Premerlani W. (1998): Object Oriented Modelling and Design for Database Applications. Prentice Hall; ISBN 0-13-123829-9.

Booch G. (1994): Object-Oriented Analysis and Design with Applications, Second Edition. Benjamin Cummings; ISBN 0-8053-5340-2.

Booch G., Rumbaugh J., Jacobson I. (1998): The Unified Modelling Language User Guide. Addison-Wesley; ISBN 0-201-57168-4.

Cantor M. (1998): Object-Oriented Project Management with UML. J. Wiley and Sons; ISBN 0-471-25303-0.

Carteret C., Vidgen R. (1995): Data Modelling for Information Systems. Pitman Publishing; ISBN 0-273-60262-4.

Catell R.G.G. (1994): The Object Database Standard – ODMG93. Morgan Kaufman Publishers; ISBN 1-55860-302-6.

Coterrell M., Hughes B. (1995): Software Project Management. Thomson Computer Press; ISBN 1-850-32190-6.

Cox B.J. (1986): Object Oriented Programming – An Evolutionary Approach. Addison-Wesley; ISBN 0-201-10393-1.

Darnton G., Darnton M. (1997): Business Process Analysis. International Thomson Publishing; ISBN 1-861-52039-5.

Date C.J. (1995): An Introduction to Database Systems (6th Edition). Addison-Wesley; ISBN: 0-201-82458-2.

Davis A. (1993): Software Reqiurements – Objects, Functions and States. Prentice Hall; ISBN 0-13-562174-7.

Derr K.W. (1995): Applying OMT – A Practical Guide to Using the Object Modelling Technique. Sigs Books; ISBN 1-884842-10-0, Prentice Hall; ISBN 0-13-231390-1.

Eriksson H.E., Penker M. (2000): Business Modelling with UML. J. Wiley and Sons; ISBN 0-471-29551-5.

Fowler M., Scott K. (1999): UML Distilled (2nd Edition). Addison-Wesley; ISBN 0-201-65783-X.

Goldberg A., Kenneth R.S (1995): Succeeding with Objects – Decision Frameworks for Project Management. Addison Wesley; ISBN 0-201-62878-3.

Gray P.M.D, Kulkarni Krishnarao G., Paton N.W. (1992): Object-Oriented Databases – A Semantic Data Model Approach. Prentice Hall; ISBN 0-13-630203-3.

Henderson-Sellers B. (1991): A Book of Object-Oriented Knowledge. Prentice Hall; ISBN 0-13-059445-8.

Hopkins T., Horan B. (1995): Smalltalk – an introduction to application development using VisualWorks. Prentice Hall; ISBN 0-13-318387-4.

Hunt J. (1997): Smalltalk and Object Orientation. Springer; ISBN 3540761152.

Ince D. (1991): Object-Oriented Software Engineering. McGraw Hill; ISBN 0-07-707402-5.

Jacobson I. (1992): Object-Oriented Software Engineering – A Use Case Driven Approach. Addison-Wesley; ISBN 0-201-54435-0.

Knott R.P., Merunka V., Polak J. (2000): Process Modelling for Object Oriented Analysis using BORM Object Behavioural Analysis. In: Proceedings of Fourth International Conference on Requirements Engineering ICRE 2000. Chicago, IEEE Computer Society Press; ISBN 0-7695-0565-1.

Kotonya G., Sommerville I. (1999): Requirements Engineering: Processes and Techniques. J. Wiley and Sons.

Mellor S., Shlaer S. (1997): Object Lifecycles: Modelling the World in States. ISBN 0136299407.

Meyer B. (1988): Object-Oriented Software Construction. Prentice Hall; ISBN 0-13-629049-3.

Partridge C. (1996): Business Objects – Reengineering for Reuse. Butterworth-Heinemann; ISBN 0-7506-2082-X.

Royce W. (1998): Software Project Management: A Unified Framework. Addison-Wesley; ISBN 0-201-30958-0.

Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. (1991): Object-Oriented Modelling and Design. Prentice Hall; ISBN 0-13-630054-5.

Rumbaugh J., Jacobson I., Booch G. (1999): The Unified Modelling Language Reference Manual. Addison-Wesley; ISBN 0-201-30998-X.

Satzinger J.W, Orvik T.U. (1996): The Object-Oriented Approach – Concepts, Modelling and System Development. Boyd & Fraser; ISBN 0-7895-0110-4.

Shriver B., Wegner P. (1987): Research Directions in OOP. MIT Press; ISBN 0-262-19264-0.

Simone A. J. H., Graham I. (1999): 30 Things that go wrong in Object Modelling with UML 1.3, chapter 17. In: Kilov H., Rumpe B., Simmonds I. (eds.): *Behavioral Specifications of Businesses and Systems.* Kluwer Academic Publishers: 237–57.

Taylor D.A. (1995): Business Engineering with Object Technology. John Wiley; ISBN 0-471-04521-7.

Yourdon E. (1995): Mainstream Objects – An Analysis and Design Approach for Business. Prentice Hall; ISBN 0-13-209156-9.

Arrived on 29th August 2002

Contact address:

Ing. Vojtěch Merunka, M.Sc., PhD., Česká zemědělská univerzita v Praze, Kamýcká 129, 165 21 Praha 6-Suchdol, Česká republika
e-mail: merunka@pef.czu.cz