Trakia Journal of Sciences, Vol. 8, Suppl. 3, pp 71-87, 2010

Available online at:
http://www.uni-sz.bg

ISSN 1313-7069 (print)
ISSN 1313-3551 (online)

OBJECT-ORIENTED PROCESS MODELING AND SIMULATION – BORM EXPERIENCE

V. Merunka*

Czech University of Life Sciences in Prague, dept. of Information Engineering, FEM
Czech Technical University in Prague, dept. of Software Engineering in Economy, FJFI

ABSTRACT
BORM (Business Object Relationship Modeling) is a development methodology used to store knowledge of process-based business systems. It has been in development since 1993 and has proved an effective method, popular with both users and analysts. BORM is based on the combination of object-oriented approach and process-based modeling. In this paper, we present BORM use as a tool for capturing process information required in the initial phases of information system development. The BORM effectiveness gained is largely due to an unified and simple method for presenting necessary aspects of the relevant business model, which can be simulated, verified and validated for subsequent software implementation. The BORM methodology makes extensive use of business process modeling towards the area of software engineering. This paper outlines BORM and presents it on an application example created in Craft.CASE analysis and modeling tool.

Key words: BORM, business modeling, conceptual modeling, business process, business process simulation, object-oriented approach, MDA, Craft.CASE.

INTRODUCTION

The attitude of business towards Information Technology (IT) is constantly changing, and increasingly sophisticated. New systems and tools are becoming 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. Today, when modern visual programming tools, combined with the support of rapid web-based application development environments and sophisticated end-user hardware technologies, are available, it would appear that the whole software development process is becoming easier. However, this statement can apply only in those cases where the system complexity of the solution and of the users' requirements is relatively small.

We started our professional career at the beginning of the 1990's as university teachers and software engineers, specializing in the new trend of object-oriented programming (OOP), object-oriented languages and object-oriented database systems. This evolution of OOP has been documented in many books and papers, for example (Goldberg and Rubin, 1995; Taylor, 1995; Yourdon, 1995; Darnton & Darnton, 1997).

As object technology gradually became the main course of our software production, our projects not only became larger, but also began to place considerable emphasis on integration with already existing business systems. More advanced techniques of OOP such as programming technologies and object-oriented databases are presented in (Ambler, 1997; Catell, 1994).

The aim of our projects was to analyze and suggest improvements to business processes, company structure, data flows, organizational structure; as well as providing IT support for them. We soon realized that we needed to carry out analyses of the problems that were supposed to be solved in order to be able to design the system and properly test their solution.

***Correspondence to:** *assoc. prof. Vojtěch Merunka, Ph.D., Czech University of Life Sciences in Prague, dept. of Information Engineering, FEM*
merunka@pef.czu.cz
Czech Technical University in Prague, dept. of Software Engineering in Economy, FJFI
vojtech.merunka@fjfi.cvut.cz

First, it is important to identify, document, and test a system in order to be able to analyze and design a more elaborated system. There are different methods and tools. On one hand, there are methods and tools oriented towards business modeling such as EPC (EPC, 2008) or varieties of Flow Charts. However, these tend to have paradigmatic gap in weak relationships with the software engineering. On the other hand, there are methods and tools of software engineering based on UML, which assume that the business requirements has already been correctly formulated and verified. (Eriksson and Penker, 2000; Kotonya and Sommerville, 1999). We used several methods for our projects, but none of them were able to smoothly combine these two worlds of modeling. This is why we began our own research.

The major problem here arises in the initial stages of the system development cycle. The initial stage of any methodologies today 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 from 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 specification of requirements in current object-oriented methodologies is Use Case modeling, and subsequent use of Sequence, Collaboration and State-Chart Diagrams. This is the foundation of most Object-Oriented development methods. However, this approach is often insufficient by itself to fully support the depths required for initial system specification. Business analyst highlights some deficiencies in this approach. There are many views on the effectiveness of Use Cases and related tools as a first stage in System Design. (Simons and Graham, 1999) for example describe a situation where Use Case Modeling obscures the true business logic of a system. Because of standard UML-based tools are too oriented at the world of programming concepts, other methods for business logic and process modeling appeared:

1. The basic grammar of other process modeling tools is based on Petri Nets. The strengths of this approach are that it is both graphical and has strong mathematical basis. A practical implementation of Petri Nets is EPC diagram of Aris methodology, for example.
2. Another techniques are based on miscellaneous varieties of flowchart diagrams. This approach is the oldest diagramming technique used in computer science. It was primarily user for visualizing the sequences of operations in computer programs. Today, flowcharts are frequently used to model business processes. A practical implementation of flowcharts is workflow diagram used in Proforma Workbench or FirstStep Business CASE Tools. Indisputably, it is also Activity Diagram of UML.
3. The third technique used here is the use of state machines. These have the theoretical background, as well as Petri Nets. A practical implementation of state machines is state-chart diagram in UML, for example. Indeed, the sequence diagram of UML has features of state machines as well.

The overview of all approaches for modeling business logic and processes described here is presented in **Table 1.**

METHOD BACKGROUND

BORM, Business Object Relation Modeling is in continuous development since 1993 when it originally was intended as a vehicle to provide support for building object-oriented software systems based on pure object-oriented languages such as Smalltalk and object-oriented databases. It has now evolved into a system development methodology that has been used successfully in about 30 projects. These systems range through all sizes of software development.

The most common technique for requirements specification in current software development methodologies is Use Case modeling as the start of UML documentation process. The main information about UML is (UML, 2009). Indeed Use Cases are often the foundation of most object-oriented development methods (Jacobson, 1992). It is concerned with the identification of external actors, which interact

with the software part of the system. This means that in order to employ Use Case modeling, it is necessary to already know the system boundary and 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, which often requires significant understanding of the proposed system and consequently can only successfully take place at the end of the requirements specification stage.

Tab. 1. *Business Modelling Paradigms*

approach	theory behind	advantages	disadvantages
EPC - Aris	Petri Nets	very popular in Europe, perfectly supported by Aris CASE Tool, easy and comprehensible method for domain experts	weak relation at subsequent software development techniques, slow analysis, low expressiveness of large models
UML Activity Diagram	flowchart	industry standard, supported by many CASE tools	too software-oriented, difficult to understand by domain experts
UML sequence and state-chart diagram	state machine	industry standard, supported by many CASE tools	too software-oriented, difficult to understand by domain experts
Workflow Diagrams	flow chart	easy and comprehensible method for domain experts, perfectly supported by many business CASE Tools	not very popular in Europe where Aris takes the dominant place, weak relation at subsequent software development techniques

Other option is usage of some business engineering method such as EPC (EPC, 2009), miscellaneous Flow Charts etc. They work fine for business system analysis and design, but have poor support for subsequent software engineering activities, because there is a big semantic and conceptual gap between them and recent software modeling methods. This is the reason of frequent misunderstanding among business and software engineering people.

Our experience in system modeling suggests that UML is not suitable for the first stages of analysis, where business processes need to be recognized. UML diagrams are too complex for the business community as they often contain too much detail concerning potential software implementations. This means classes, inheritance, public/private methods, attributes, link classes, etc. Here we got almost the same experience as documented in (Simone and Graham, 1999).

We believe that the business community needs a simple yet expressive tool for modeling; able to play an equivalent role to that played by the Entity-Relation Diagrams, Data-Flows Diagrams or Flow-Charts over the past decades. One of the strengths of all these approaches was that they contained only a limited set of concepts (about five) and were comprehensible by problem domain experts after a few minutes of study. Unfortunately the UML/BPML approach lost this advantage of simplicity.

That is why we developed the BORM process diagram and the way, how to start business system analysis in simple but precise method going smoothly from business analysis and simulation to detailed UML software design based on MDA principle. Our approach is to start with a small set of the business-level concepts, which can subsequently be transformed into more software-oriented concepts.

BORM fundaments

The BORM methodology has been developed on academic grounds since the 1990s. It unifies the MDA principle, using an object-oriented paradigm and a unified approach to business and IT system modeling. For more on the BORM method, see (Knott et al., 2000; Liu and Roussev, 2006).

MDA

MDA defines an approach that separates a specification of business system description (*CIM – Computation Independent Model*) from its computer implementation specification (*PIM – Platform Independent Model*); and this computer specification from the final solution on a concrete technological platform (*PSM – Platform Specific Model*). According to MDA, there is a mutual relationship between these three views, and the models should transform from one to another when a system is created. MDA is created and maintained by the Object Management Consortium (MDA, 2009).

Object-oriented approach

The OOP has its origins in the researching of GUI and programming languages, that took place in the 1970s. It differs from other software engineering approaches by incorporating non-traditional ways of thinking into the field of informatics. We look at systems by abstracting the real world in the same way as in ontological, philosophical streams. The basic element is an object that describes data structures and their behavior. In most other modeling approaches, data and behavior are described separately, and, to a certain extent, independently. OOP has been explained in many books, but we think that this one (Goldberg and Rubin, 1995) written by OOP pioneers belongs to the best.

Automata theory

The theory of automata is a study of abstract automatons and the problems they can solve. An automaton is a mathematical model for a device that reacts to its surroundings, gets an input, and provides an output. The automatons can be configured in a way that an output from one of them becomes an input for another of them. An automaton's behavior is defined by a combination of its internal state and its accepted input.

The automata theory is a basis for system behavior descriptions. Its usage for modeling and simulation in software engineering activities has been written in (Shlaer and Mellor, 1992) and many newer publications.

Business engineering – business models

The first part of the method covers the CIM field, i.e. business engineering. It transforms a project assignment into a model described by data hierarchies, process participants, process scenarios, various diagrams and generated reports. The main instrument of verification and validation is the process simulator.

For the following purposes, it is possible to use this part of BORM without any relation to a software engineering phase:

- Organizational consultancy projects. These are process analysis, organizational structure analysis, and drafts for processes or organizational structure improvement.
- Projects documenting processes and organizational structure. These are, for instance, projects whose aim is knowledge management, creating training materials, knowledge visualization, etc.
- Projects for preparing the groundwork for selection procedures for organizational consultancy, or other consultancy services.
- Projects for preparing the groundwork for selection procedures for the delivery of information systems, or other software engineering projects.

Software engineering – conceptual models

Second part of the BORM is PIM, i.e. conceptual modeling based on software engineering principles.

- It fluently relates to the preceding phase of business engineering, i.e. that the software system assignment is described as a business model, and is well tested and revised.
- It uses the same, or very similar terms and rules, as the preceding phase. It is unnecessary to learn any other method for either the first phase, or this phase.
- Apart from classical models in the UML standard, the Craft.CASE tool enables work to be carried out with concrete objects (class instances) and collections of concrete objects. An analyst can test a draft on a small prototype created from these concrete objects.

The main instrument of verification and validation of this part of the method is a relationship to preceding parts of the method and option to work with concrete objects (class instances).

Two layers of a model

Process approach and object orientation are the pillars of the BORM method. It is the application of principles that are successful in the field of modeling and software creation. The basis of the object approach is the notion that each action has an object that executes it; or, vice versa; that each object has a behavioral role in a model. It is impermissible to have an action without an object, or an object without an action.

With object-oriented orientation, various process situations can be modeled:

- An object executes an action that influences the action of another object.
- Actions that could be decomposed into a sequence request-solution-result with a particular extent of various object participation.
- Modeling and analysis of process and organizational structure relations.
- Descriptions (requests) of new systems.

The presence of information in a row (behavior, for example) indicates a necessity for new information in another row (subjects), and vice versa. It is advisable to proceed in both rows, step-by-step, whilst modeling. This interconnection of both rows prevents a "jamming" of the project after which an analyst would not know how to continue.

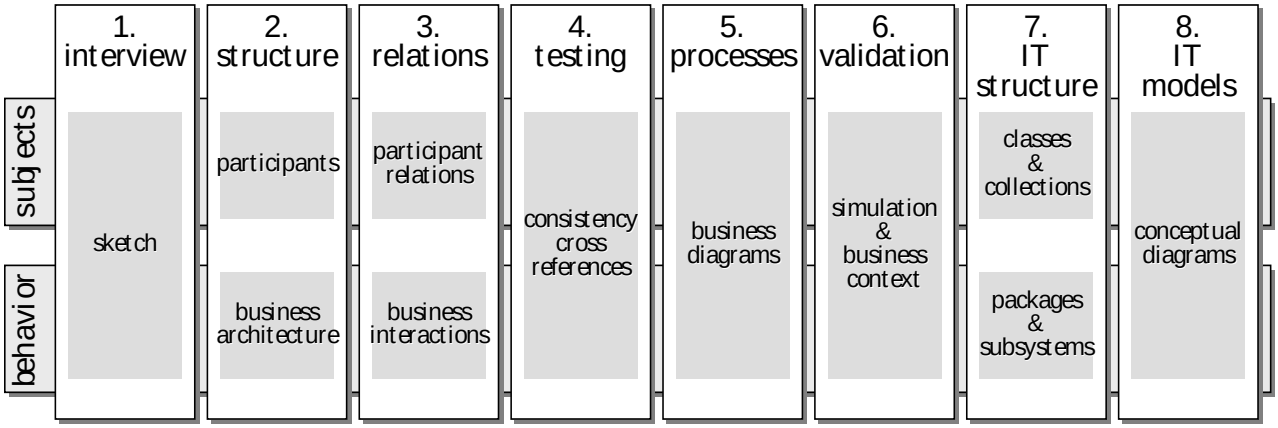

Figure 1. BORM Method

BORM interpretation of MDA

In BORM (**Figure 1 and 2)**, each concept has 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.
2. A validity Range - The phases where it is appropriate. In each phase of BORM modeling, only limited set of concepts is recommended.
3. A Set of techniques and rules, which guide the step-by-step transformation and the concept revisions between the system development phases. There are refactoring techniques, data normalization, design patterns and other programming-related techniques (Ambler, 1997) adopted for BORM concept transformations.

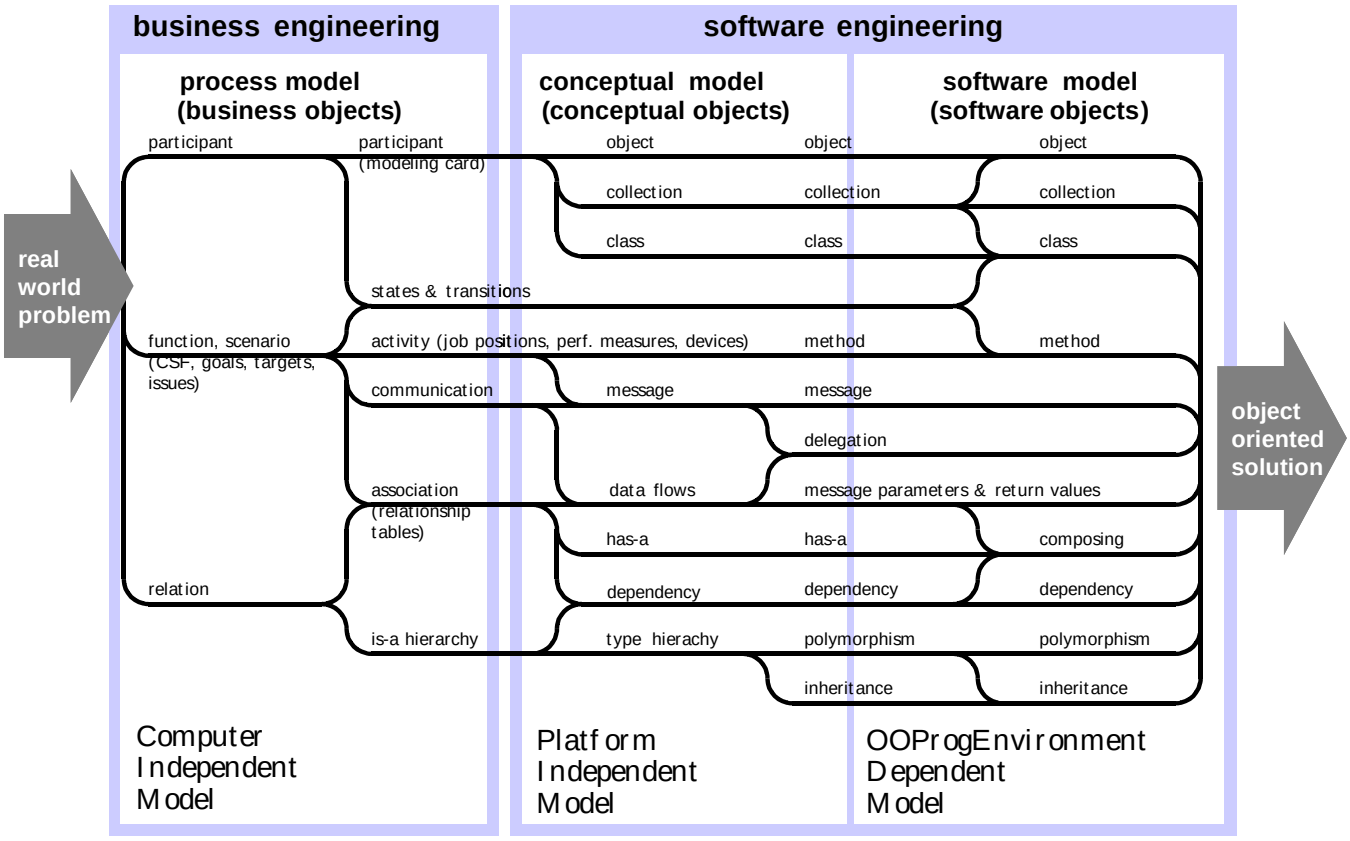

Figure 2. MDA approach in evolution of BORM terms – object-oriented software development example

BORM business diagram

BORM respects UML and MDA, but uses an original diagram for business process modeling. It conveys together information from three separate UML diagrams: *state*, *communication* and *sequence*. The BORM group has found that it is clearly understood by business stakeholders.

- Each subject participating in a process is displayed in its states and transitions.
- This diagram expresses all the possible process interactions between process participants. The business process itself consists of a sequence of particular communications and data flows among participating subjects.

More formally, BORM process diagrams are graphical representations of interconnected Mealy-type finite state machines of particular subjects. Visual simulation of a business process is based on market-graph Petri net. Almost the same approach is described in detail by (Barjis and Reichgett, 2006). Therefore we can show states, transitions and operations for all subjects playing a role in a business process. This is a very powerful, yet simple diagram.

There is an example of BORM business process diagram at **Figure 3.** It shows an *invoice processing*. There are six participating subjects in this process: *Supplier*, *Invoice Registration Department*, *Store Department*, *Account Payable Department*, *SAP* and *Bank*. The small rectangles within these subjects are their states. The ovals within subjects are their activities, which are conceivable as the transitions between the states as well. The big arrows between subjects are the subject communications, and the very small arrows are the data flows.

CRAFT. CASE MODELING TOOL

At first, we attempted to use the existing tools and adapted them for BORM. Our experience covers Microsoft Visio and MetaEdit. (Metacase, 2009) However, we found that none of them allowed us to model, test and simulate in the way we needed.

The Craft.CASE tool has been designed according to state-of-the-art principles of object-oriented software development, and uses a special Smalltalk programming language technology that has been evolving since the 1970's. The program consists of many user functions, has its own internal

object-oriented database, several graphic editors and a programmable interface. The Craft.CASE tool provides all instruments for CIM (as business models) and PIM (as conceptual models), including their mutual interconnection and the possibility to undertake thorough testing. (Craftcase, 2009)

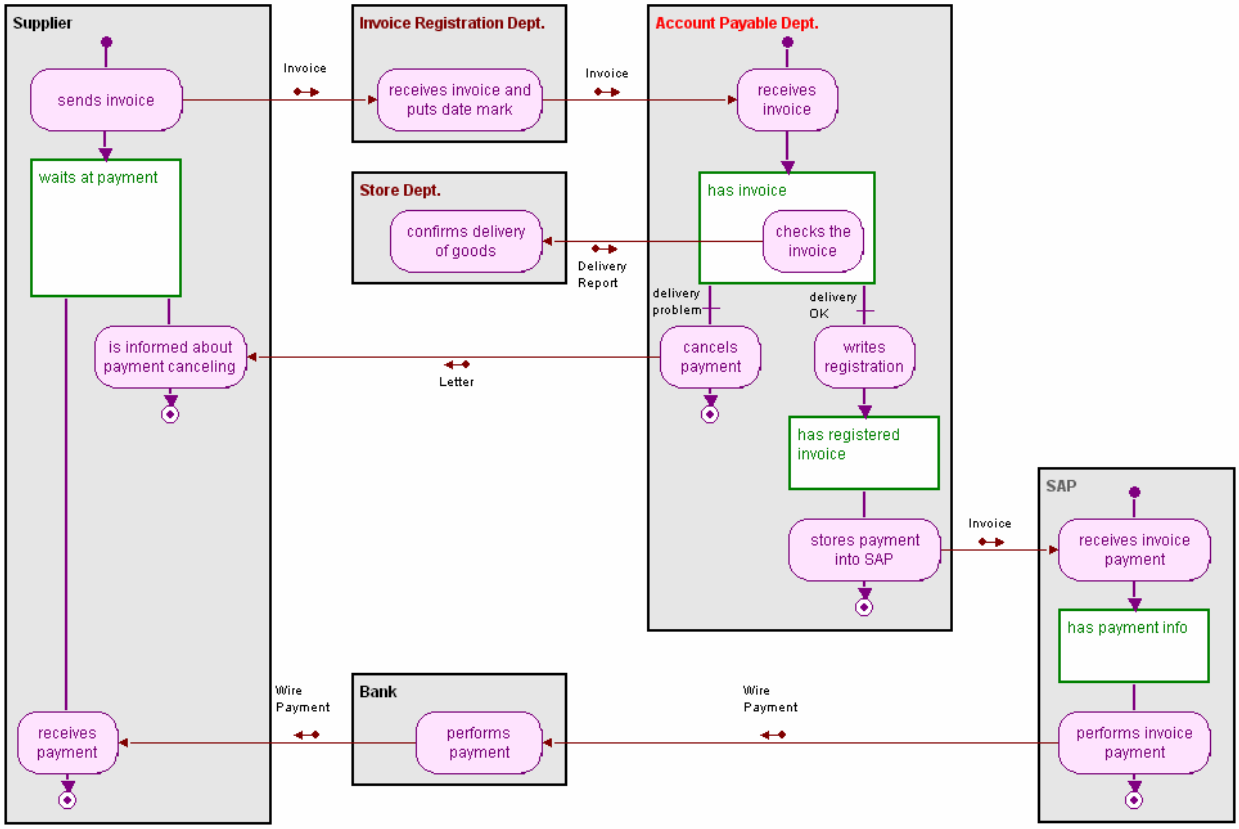

Figure 3. Business Process Example

The Craft.CASE can be used in process and organizational consultancy and in analytical projects and information system drafts while identifying requests on newly – designed systems; also in component modeling and service oriented architecture. It works with a system model and its processes in an original way that is based on:

- An ability not only to visualize processes and systems in the form of diagrams, but also to test them by means of cross-references and graphic simulations.
- An ability to model not only symbolic terms (for example, a customer, order, payment, etc.) as drawn symbols in a diagram, but also the possibility of working with real objects (several concrete customers, orders, payments, etc. that differ in their real values such as a date, name, price, etc.). Craft.CASE supports both class-level and instance-level of modeling.
- A method having diagrams as the result of a gradual deriving and checking.

Craft.CASE implements an original approach called C.C method. The C.C method combines majority of CIM and PIM parts of BORM with concept transformations via business process simulators, instance-level modeling and set of transformation rules describing how to derive subsequent concepts from previous ones. Moreover, in each step of the C.C method, Craft.CASE keeps consistency between two layers of a model; subjects and behaviors. Thanks to metamodel background and system internal procedures, there is rigidly checked, whether all subjects from the first layer (e.g. classes, object states, packages etc.) have corresponding behaviors from the second layer (e.g. scenarios, use-cases, operations etc.) and vice versa. More information about Craft.CASE such as its programming facilities, metamodel etc. is described in (Merunka et al., 2008).

Analysts would much rather hear "I don't want this" from a client or prospective user while projecting a system, than "I didn't want that" after having completed the project. This is why

during modeling, it is important to have an opportunity to test, verify and validate the system and its processes. Proper testing includes visualization and simulation - not only with abstract terms, but also with real objects of the modeled system at the instance level.

EXAMPLE

Our example of modeling using Craft.CASE is to model an information system for a company called FD (Food Delivery); dealing with food products delivered from suppliers to local customers. A typical example of such activity is shopping and delivering the goods to a home, or home delivery of pastries and milk and similar products. The information system is primary designed for communication between a company and its delivery employees. It also provides an information portal for customers to whom the products are to be delivered.

Project initiation

The first stage of business modeling is the initiation of work on a project. At the beginning of this phase there are interviews, and then an initial structure and relations are modeled. The testing of generated reports is executed as the final part of this stage.

Interview

Since kick-off meeting with the client, we have negotiated that the project should focus on ordering a meal through a customer process. The client expressed a preference for communication with customers via web pages. The client also wants work activity descriptions for the logistics manager and van drivers.

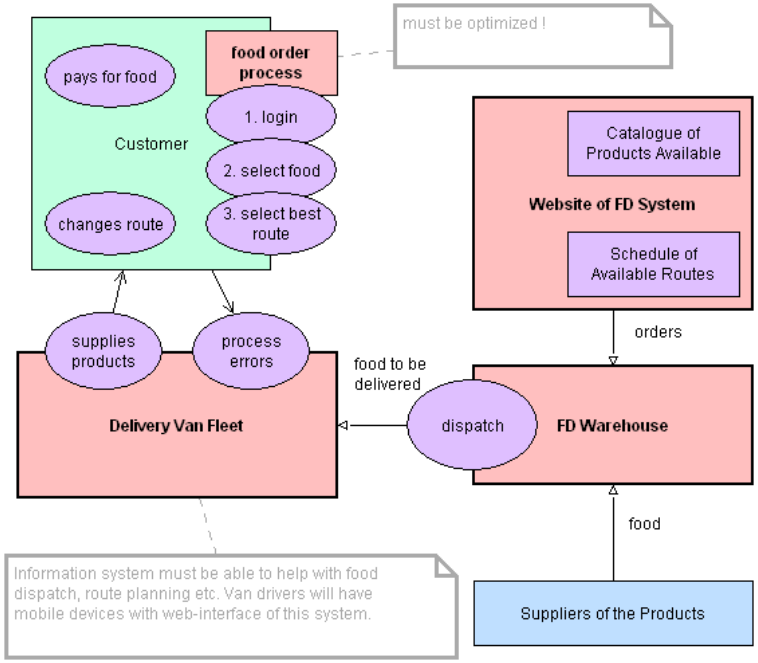

Figure 4. Record of kick-off meeting interview – free drawing in Craft.CASE

The client wishes to enable its customers to choose a delivery time based on a calendar of scheduled delivery routes that have been prepared in advance. Its advantage over the competition is in offering different prices for different deliveries that differ in date and hour; allowing customers to choose between either a cheaper, or a more convenient delivery time.

Business modeling (Computer-Independent Model)

Structure

Structure definition is the first formal step of the method. An analyst should be capable of finding key subjects and process fields on the basis of performed interviews.

Participants

Based on the scope of the project and the allowed project time, eight participants were found **(table 2).** These are the job positions of the *logistics manager* and the *van driver*, as well as future software components of the *web interface* and *database system*; and, of course, a *customer* and some *suppliers*, since they take part in the processes.

Table 2. *Participants*

Customer	Person who orders food or a person who wants to be a customer.
FD Database	Database of products, orders, routes, customers and suppliers.
Logistics Manager	Person responsible for the distribution of orders. This person uses information and scheduling system.
Supermarket Supplier	Department of Supermarket responsible for food delivery.
Supplier	Small company or individual farmer.
Supplier Ordering System	Automatic system for requesting suppliers for food.
Van Driver	Person who delivers food.
Website of FD	Communication software enabling orders.

Hierarchies

For the detailed mapping of problems to be solved and future use of information system creating, two hierarchies of concrete testing data were recognized and modeled. They are food delivery *product catalogue* and a list of *suppliers* for these products.

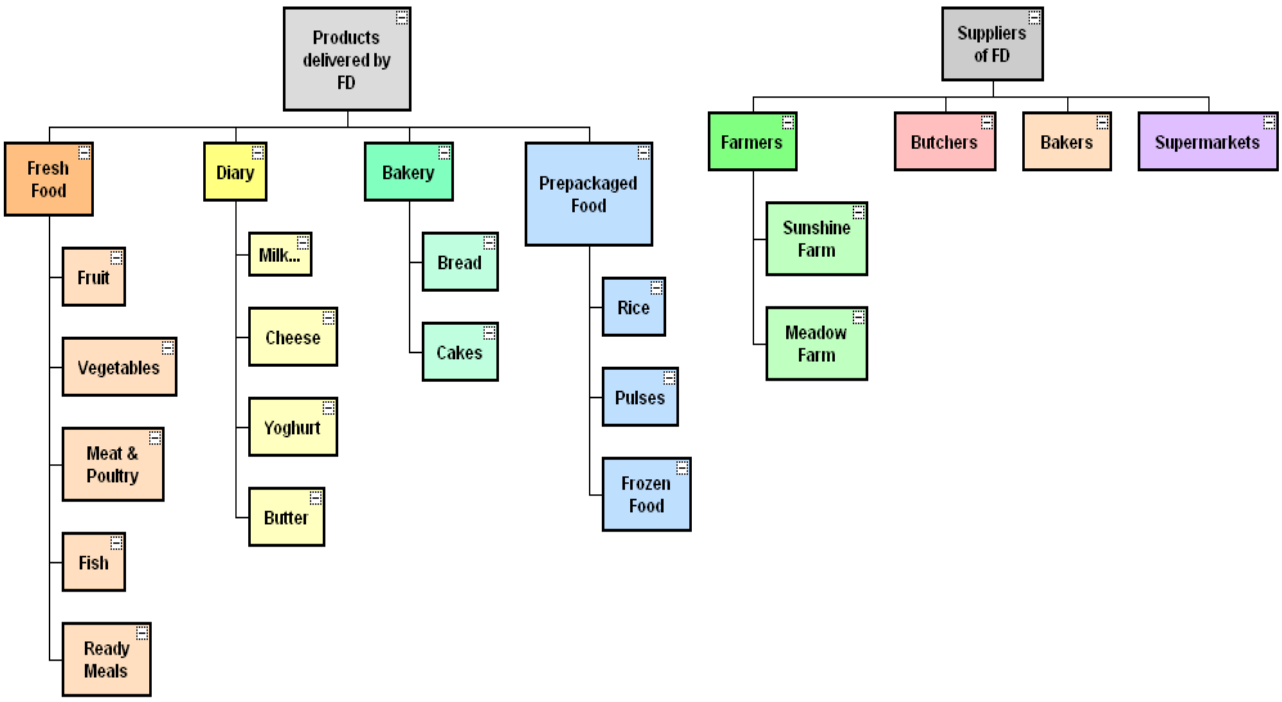

Figure 5. Products and supplier hierarchies

Business architecture

Five functional areas were developed. It was then decided that three functions (marked as external functions) related to care, and that virtual food products warehouse, advertising activity and web page maintenance will not be included in the concrete scope of our project. The project was limited to activities related to ordering and delivering (marked as internal functions).

Name	Description	Status
Advertize FD	Collection of processes promoting FD business (both to clients and food suppliers)	external
Dispatch Orders	Here is collection of processes performed by FD management. This is about route planning, orders dispatching etc.	internal
FD Website Maintenance	User management, database update, SW and HW maintenance.	external
Maintain Stock	FD company internal responsibility. Here is collection of processes with no direct interface to any client.	external
Sell Food	Core business of FD company. Here is collection of proceses performed by or performed for clients.	internal

Value	Owner type	Owner name
external	Function	FD Website Maintenance
		Maintain Stock
		Advertize FD
internal	Function	Sell Food
		Dispatch Orders

Figure 6. Business Functions (Formatted HTML report from Craft.CASE)

Hierarchy relations

Several bindings were modeled between elements of the product hierarchy and the product supplier hierarchy to finalize system requirements. These data relationships (elements of the Cartesian product, more formally) show a concrete supplier of concrete products that can be used later for instance-level testing.

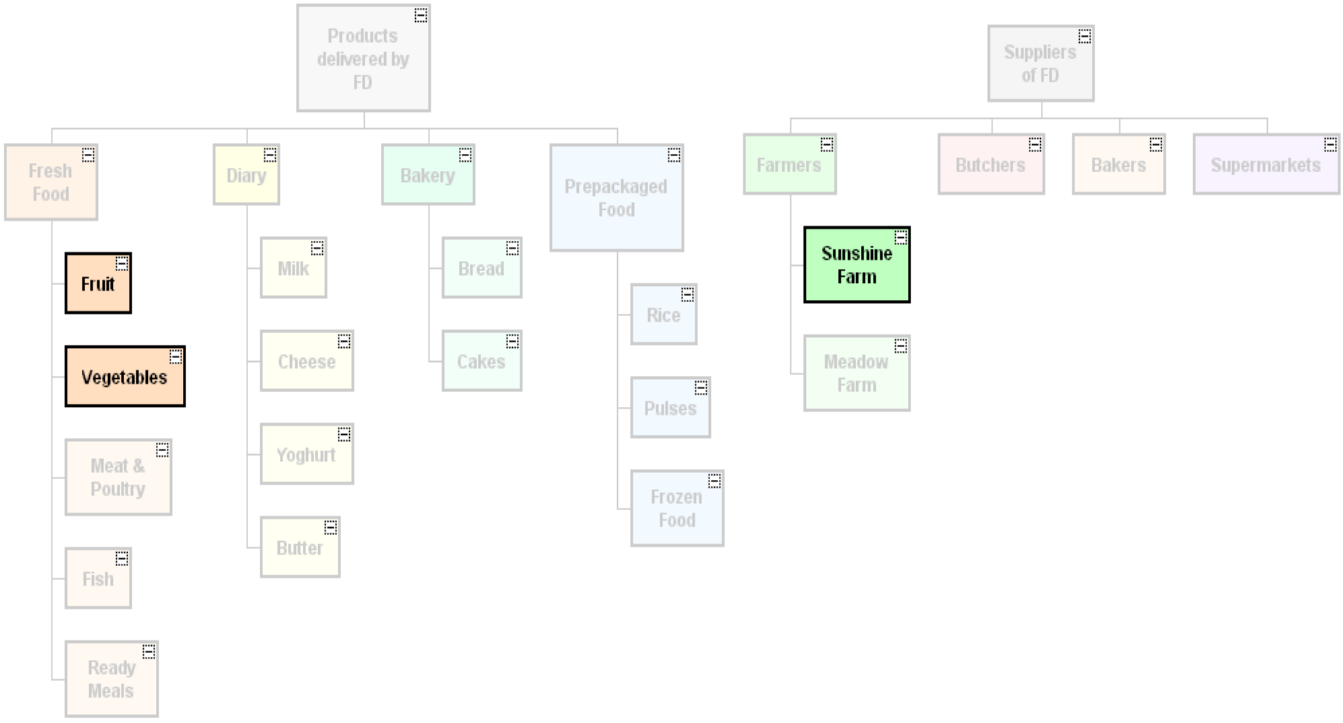

Figure 7. Fruit and vegetables delivery as an example of instance-level binding between hierarchies

Detailed scenarios and participant roles in scenarios

Six business process scenarios were analyzed in three functional areas – see **Figure 9.** Concept of a business scenario is described in (Taylor, 1995). Although this project does not deal with web page maintenance, a new customer registration scenario was modeled into the system because its content evocated problems that are being solved. (Note small icons in corners of rectangles indicating diagram decompositions.)

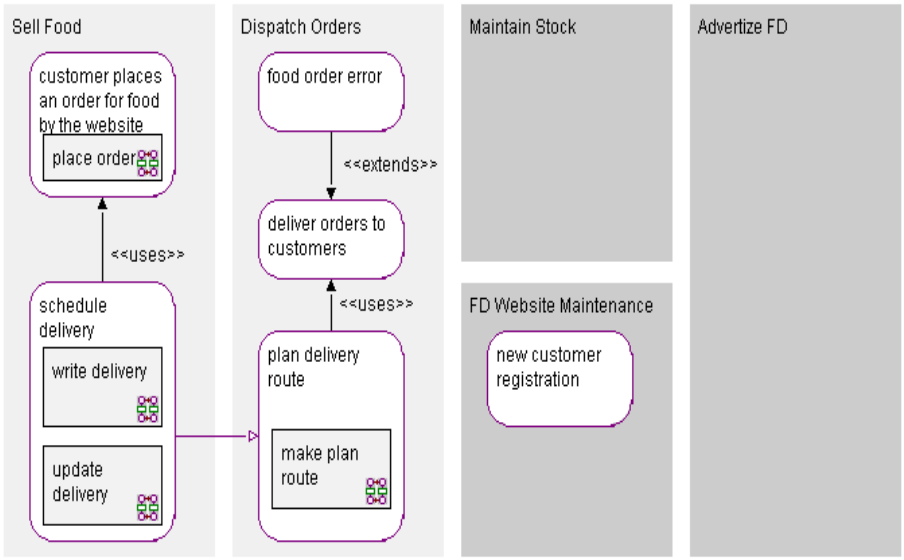

Figure 8. Business architecture diagram (Use-Case clone) of the whole system requirements

customer places an order for food by the website	
initiation: Customer want to buy food.	*roles:* **Customer** initiates **FD Database** provides information **Website of FD** cooperates
action: Customer logs into the website, selects food, and places his/her order.	
result: Order is accepted.	
Integrated functions: Sell Food *Derived diagrams:* place order *Is used by:* schedule delivery	

plan delivery route	
initiation: New day started.	*roles:* **FD Database** provides information **Logistics Manager** is responsible **Van Driver** cooperates **Website of FD** cooperates
action: Routes are planned from information on website for all orders requested for delivery on current day.	
result: Route for each van driver for the current day is produced.	
Integrated functions: Dispatch Orders *Follows:* schedule delivery *Derived diagrams:* make plan route *Uses:* deliver orders to customers	

schedule delivery	
initiation: Customer wants to arrange delivery on specific date.	*roles:* **Customer** initiates **FD Database** provides information **Website of FD** cooperates
action: Customer selects the best delivery date on website.	
result: Order delivery is planned on selected date.	
Integrated functions: Sell Food *Is followed by:* plan delivery route *Derived diagrams:* update delivery, write delivery *Uses:* customer places an order for food by the website	

deliver orders to customers	
initiation: Routes are planned and optimized.	*roles:* **Customer** receives **FD Database** provides information **Logistics Manager** initiates **Van Driver** is responsible
action: Van follows the predermined route to deliver its orders to customers.	
result: Food has been delivered. Order has been acepted. Payment done.	
Integrated functions: Dispatch Orders *Is extended by:* food order error *Is used by:* plan delivery route	

food order error	
initiation: Food is damaged of incorrect item has been delivered.	*roles:* **Customer** initiates **FD Database** provides information **Logistics Manager** approves **Van Driver** cooperates
action: Van driver and customer agreed that food should be returned and payment adjusted.	
result: The item is returned and the customer credited with his/her cost.	
Integrated functions: Dispatch Orders *Extends:* deliver orders to customers	

new customer registration	
initiation: A person wants to be a new customer.	*roles:* **Customer** initiates **FD Database** provides information **Website of FD** cooperates
action: Person supplies his/her name, address etc. into website.	
result: Customer is registered and ready to apply order or is refused.	
Integrated functions: FD Website Maintenance	

Figure 9. Business process scenarios in formatted HTML report from Craft.CASE

Testing

Testing was done using presenting participants' modeling cards. On the basis of this testing, it was decided to model both the ordering and distribution planning processes in detail. Our concept of the modeling card originates from CRC – Class-Reponsibility-Collaborator technique (Bellin and Simone, 1997).

Customer

Collaborators:	FD Database	Logistics Manager	Van Driver	Website of FD
customer places an order for food by the website (initiates)	provides information			cooperates
deliver orders to customers (receives)	provides information	initiates	is responsible	
food order error (initiates)	provides information	approves	cooperates	
new customer registration (initiates)	provides information			cooperates
schedule delivery (initiates)	provides information			cooperates

Logistics Manager

Collaborators:	Customer	FD Database	Van Driver	Website of FD
deliver orders to customers (initiates)	receives	provides information	is responsible	
food order error (approves)	initiates	provides information	cooperates	
plan delivery route (is responsible)		provides information	cooperates	cooperates

Van Driver

Collaborators:	Customer	FD Database	Logistics Manager	Website of FD
deliver orders to customers (is responsible)	receives	provides information	initiates	
food order error (cooperates)	initiates	provides information	approves	
plan delivery route (cooperates)		provides information	is responsible	cooperates

FD Database

Collaborators:	Customer	Logistics Manager	Van Driver	Website of FD
customer places an order for food by the website (provides information)	initiates			cooperates
deliver orders to customers (provides information)	receives	initiates	is responsible	
food order error (provides information)	initiates	approves	cooperates	
new customer registration (provides information)	initiates			cooperates
plan delivery route (provides information)		is responsible	cooperates	cooperates
schedule delivery (provides information)	initiates			cooperates

Website of FD

Collaborators:	Customer	FD Database	Logistics Manager	Van Driver
customer places an order for food by the website (cooperates)	initiates	provides information		
new customer registration (cooperates)	initiates	provides information		
plan delivery route (cooperates)		provides information	is responsible	cooperates
schedule delivery (cooperates)	initiates	provides information		

Figure 10. Modeling cards of participants in formatted HTML report from Craft.CASE

Detailed business modeling – business process simulation

Processes

First, the ordering process was modeled. Based on the project focus, a new property for communications was created, with „manual" or „automatic" values, with a relation to the graphic interface where, for automatic communication -- expecting a software realization -- a thick arrowed line between oval activities is drawn. Manual communications are visualized in the standard way of thin arrowed line between oval activities.

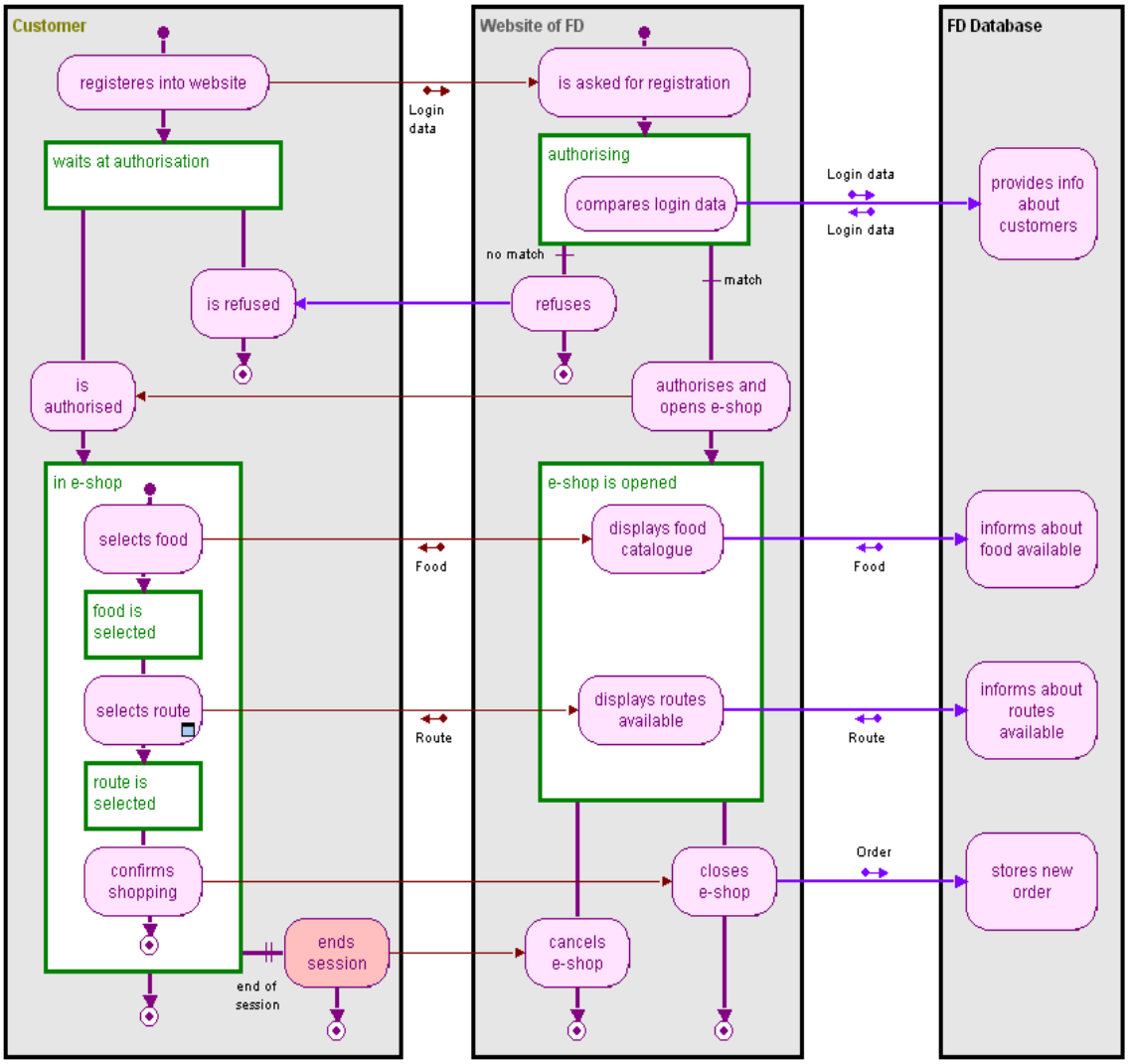

Figure 11. Business process of ordering goods

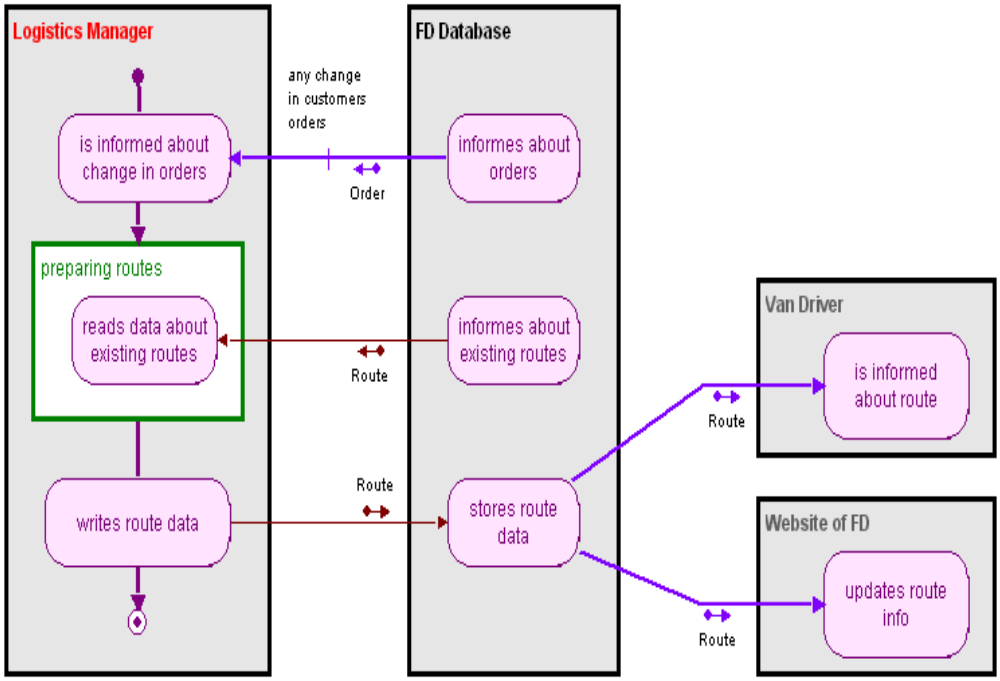

Figure 12. Delivery planning business process

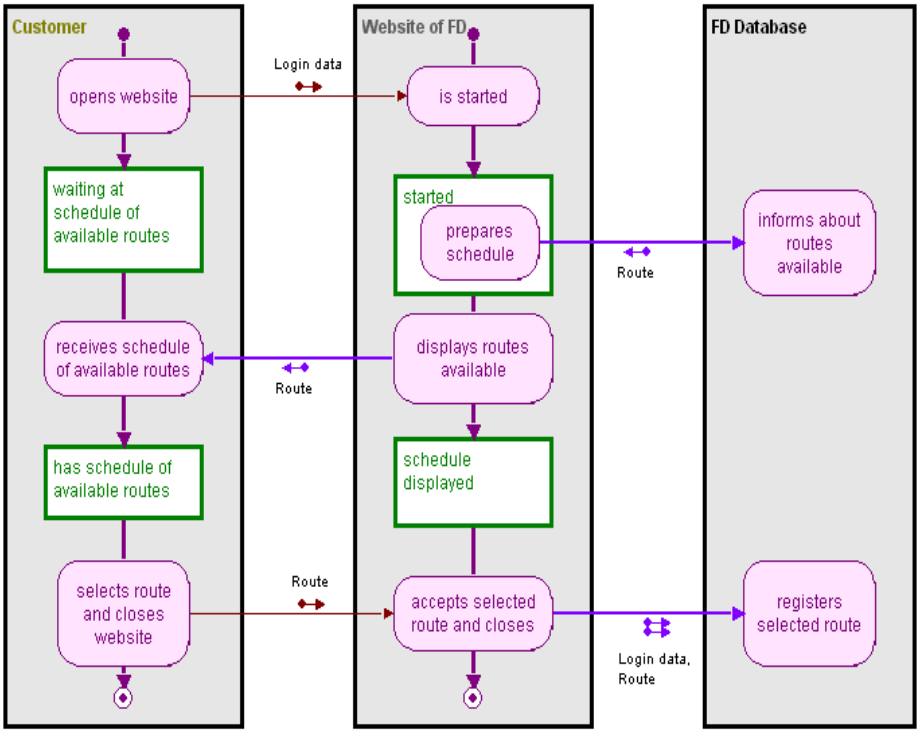

Figure 13. Sub-process of delivery ordering by a customer

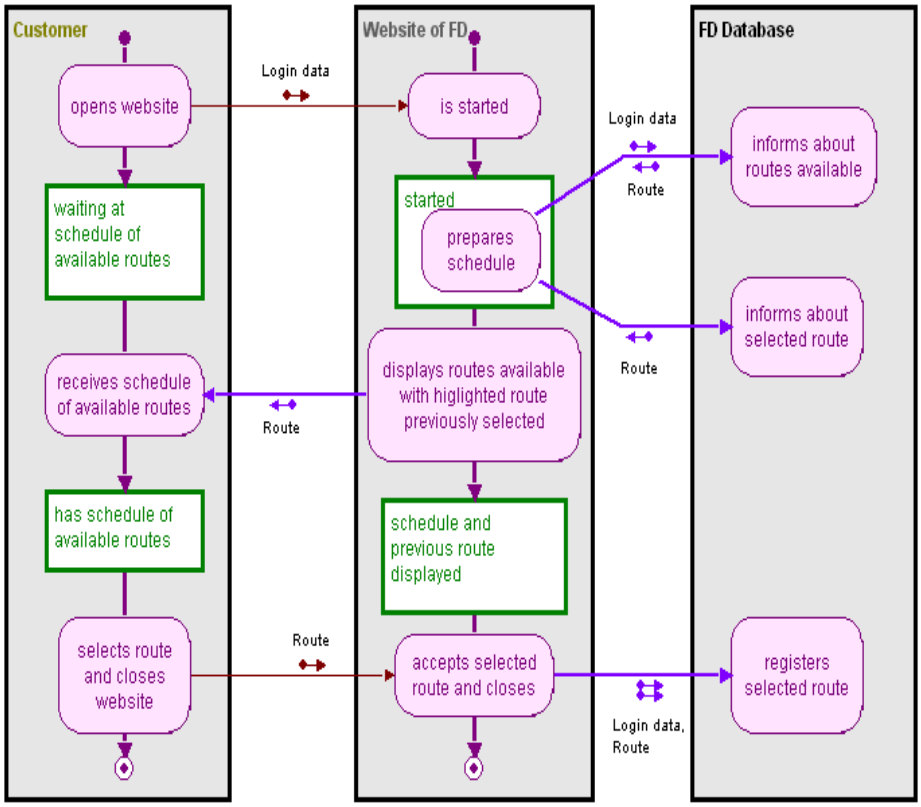

Figure 14. Sub-process of delivery adjustment by a customer

Process testing

Detailed modeling cards and a simulator were used for testing in this step. Craft.CASE simulator displays particular simulation steps and allows dialogue with the user.

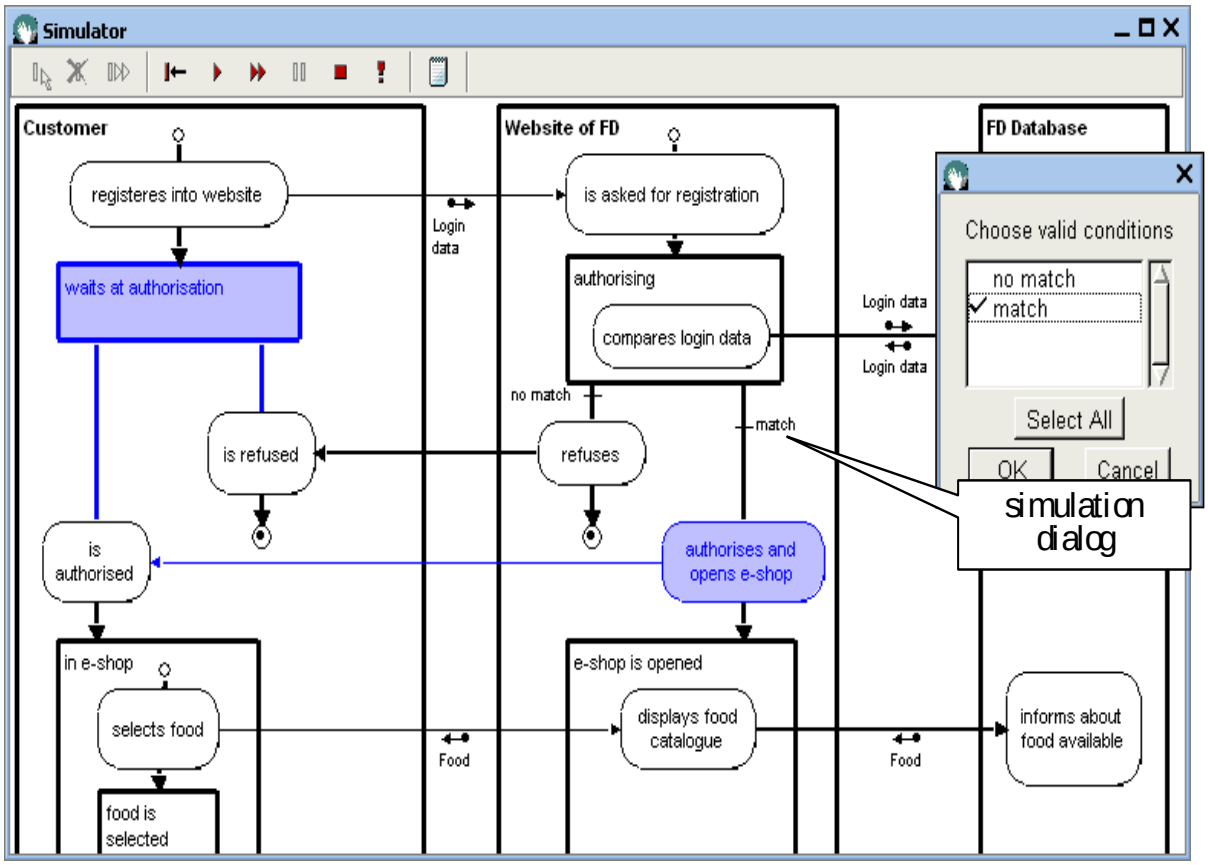

Figure 15. One concrete simulation step in Craft.CASE

Modeling cards

The modeling cards of selected objects were presented during a meeting with the client. Based on these cards (generated as a result of the simulation) a process model was finally confirmed by clients/stakeholders for subsequent software implementation.

Collaborators in diagram with name 'write delivery':	FD Database	Website of FD
start: opens website		>>
waiting at schedule of available routes: receives schedule of available routes		<<
has schedule of available routes: selects route and closes website		>>

Collaborators in diagram with name 'place order':	FD Database	Website of FD
route is selected: confirms shopping		>>
in e-shop: ends session		>>
waits at authorisation: is authorised		<<
waits at authorisation: is refused		<<
start: registeres into website		>>
start: selects food		
food is selected: selects route		

Collaborators in diagram with name 'update delivery':	FD Database	Website of FD
start: opens website		>>
waiting at schedule of available routes: receives schedule of available routes		<<
has schedule of available routes: selects route and closes website		>>

Figure 16. Cards of a customer's detailed behaviour (Customer)

'Route' from 'Website of FD'	Customer	FD Database
accepts selected route and closes		registers selected route
accepts selected route and closes		registers selected route
displays routes available	receives schedule of available routes	
displays routes available with higlighted route previously selected	receives schedule of available routes	
prepares schedule		informs about routes available, informs about selected route
prepares schedule		informs about routes available

'Route' from 'Customer'	Website of FD
selects route and closes website	accepts selected route and closes
selects route and closes website	accepts selected route and closes

'Route' from 'FD Database'	Logistics Manager	Van Driver	Website of FD
informes about existing routes	reads data about existing routes		
stores route data		is informed about route	updates route info

'Route' from 'Logistics Manager'	FD Database
writes route data	stores route data

Figure 17. Modeling cards of route data flow behavior (Route)

IT modeling (Platform-Independent Model)
After finishing the business model and confirming its business requirements, a conceptual model of the company's information system in the UML standard was initiated. All business-modeling outputs are also the most important part of the documentation for the software solutions supplier.

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 system development seems to be a promising candidate for such a standard. The approach we propose may serve not only as a tool for formal representation of modeled 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.

The number of projects executed in past 10 years gives us an important feedback. The clients say our analysis gives them a complex and context view of issues they did not see before. Clients appreciate BORM models having collection of business elements and their relationships being visualized and simulated together. Moreover, several clients use miscellaneous legacy Process Modeling Systems for historical reasons (e.g. EPC-based ARIS, for example). (EPC, 2009) However they prefer to analyze and design processes using BORM as well.

Our next work will concentrate on elaboration of the BORM, its concept transformations from and to other methodologies and research in the area of business process patterns.

ACKNOWLEDGMENTS

The author would like to acknowledge the support of the research grant project MSM6046070904 of the Czech Ministry of Education and the SGS project of FJFI CVUT.

REFERENCES

1. Ambler, S. (1997) *Building Object Applications That Work, Your Step–By–Step Handbook for Developing Robust Systems Using Object Technology*, Cambridge University Press/SIGS Books, ISBN 0521648262.
2. Barjis, J., & Reichgelt, H. (2006). A Petri Net Based Methodology for Business Process Modeling and Simulation. *In the proceedings of the Fourth International Workshop on Modeling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS)*, Paphos, Cyprus, May 23-24, 2006. ISBN: 972-8865-49-X

3. Bellin, D., Simone, S. S. (1997) *The CRC Card Book*, Addison–Wesley, ISBN 0-201-89535-8.
4. Catell, R. G. G. (1994) *The Object Database Standard – ODMG93*, Morgan Kaufman Publishers, ISBN 1-55860-302-6.
5. Craft.CASE (2009) - Business Process Modeling, http://www.craftcase.com
6. Darnton, G. and Darnton, M. (1997) *Business Process Analysis*, International Thomson Publishing, ISBN 1861520395.
7. Eriksson, H. and Penker, M. (2000) *Business Modeling with UML*, John Wiley and Sons, ISBN 0-471-29551-5.
8. EPC – Event-driven Process Chain (2009), http://en.wikipedia.org/wiki/Event-driven_process_chain
9. Goldberg, A. and Rubin, K. S. (1995) *Succeeding with Objects – Decision Frameworks for Project Management*, Addison Wesley, ISBN 0-201-62878-3.
10. Hall, J. A. (2008) *Accounting Information Systems*, Cengage Learning EMEA, ISBN 0324560931.
11. Jacobson, I. (1992) *Object–Oriented Software Engineering – A Use Case Driven Approach*, Addison Wesley, ISBN 0-201-54435-0.
12. Knott, R. P., Merunka, V., Polak, J. (2000) *Process Modeling for Object Oriented Analysis using BORM Object Behavioral Analysis,* Proceedings of Fourth International Conference on Requirements Engineering ICRE 2000, Chicago, USA, IEEE Computer Society Press, ISBN 0-7695-0565-1.
13. Kotonya, G. and Sommerville, I. (1999) *Requirements Engineering: Processes and Techniques*, John Wiley and Sons, ISBN 978-0-471-97208-2.
14. Liu, L. and Roussev, B. (2006) *Management of the Object–oriented Development Process*, Idea Group Inc, ISBN 1591406048.
15. MDA – The Model Driven Architecture, OMG – The Object Management Group (2009), http://www.omg.org.
16. Merunka, V., Brozek, J., Nouza, O. (2008) Automated Model Transformations Using the C.C Language, *Proceedings of the International conference EOMAS 2008*, June 16 – 17, Montpellier, France, Springer LNBIP 2008, ISSN 1865-1348.
17. MetaCase - Domain-Specific Modeling with MetaEdit+ (2009), http://www.metacase.com/
18. Simone A. J. H. and Graham, I. (1999) 30 Things that go wrong in Object Modeling with UML, chapter 17 in: *Behavioral Specifications of Businesses and Systems* eds. H Kilov, B Rumpe, I Simmonds (Kluwer Academic Publishers), 237 – 257.
19. Shlaer, S. Mellor, S (1992) Object Lifecycles: Modeling the World in States, Yourdon Press, ISBN 0136299407.
20. Taylor, D. A. (1995) *Business Engineering with Object Technology*, John Wiley and Sons, ISBN 0-471-04521-7.
21. The UML standard, OMG – The Object Management Group (2009), http://www.omg.org, ISO/IEC 19501.
22. Yourdon, E. (1995) *Mainstream Objects – An Analysis and Design Approach for Business*, Prentice Hall, ISBN 0-13-209156-9.