UDC: 007.5
Original scientific paper

STRUCTURED AND OBJECT-ORIENTED METHODS IN A COMPLEX IS PROJECT

Josip Brumec, Vesna Dušak, Neven Vrček
University of Zagreb, Faculty of organization and informatics, Varaždin
E-mail: jbrumec@foi.hr or vdusak@foi.hr or nvrcek@foi.hr

Object-oriented and structured methods are not mutually exclusive, although each of them has their advantages when used in developing a specific type of information system. For that reason, some characteristics of real systems and their ISs, that determine the choice of methods, are analysed in this paper. It has been concluded that for the development of complex ISs, especially those that integrate business and technological processes, a combined use of structured and object-oriented methods gives the best results. The way structural and object methods are integrated into a consistent methodology for the development of ISs is also shown in the paper. The results are illustrated with examples from a large-scale project, which was carried out by the authors.

Keywords: IS development,object-oriented methods,structured methods, BMS-methodology.

1. PROBLEM DEFINITION

When object-oriented methods appeared [20], [9], [4] or [3], some practicians wrongly concluded that object methods will completely displace the structured methods of IS development. James Martin, who was at first a great advocate and promoter of structural methods in IS development [16], in his later books speaks even of "object revolution" [17]. Nevertheless, at the same time with theoretical explanation and the practical employment of object methods, certain authors, for example [21], try with analytical consideration to determine those areas of use within which object-oriented methods are better used than structured ones and vice versa. In that sense a very important analysis was made by Ian Graham in [12] where he concluded that the use of an object approach in developing application software gives poor results unless the object-oriented system analysis and design have not been carried out before. In their new book [2] authors M. Blaha and W. Premerlani give a detailed analysis of various kinds of data models and differentiate between those for which object-oriented databases (OO-DB) are better used and those for which it is better to use relational databases (DB). Since their conclusions agree with the basic thesis of this paper, we have included a summary of their results in Table 1.

Table 1. Areas of object and relational database use [2]

Relational DBMS	OO-DBMS
* Business information systems with a large DB (>20 tables, > 1000 records per table) * Decision support systems (*ad hoc questions*) * Application using 4GLs (a lot of screens, forms and reports) * New applications, relying upon the existing relational data bases * Systems with great safety demands (authority, security, DB-redesign)	* Engineering applications (CAD/CAM, CIM, CASE) * Multimedia applications (picture, text, sound) * Knowledge bases for expert systems * Extremely distributed applications used with various technical equipment * Applications with specific requests (user defined data types, inheritance, long transactions, multiple program versions, etc.) * Programmable electronic devices

Areas of particular interest, when considering design methods, are information systems for production systems. These ISs contain various types of applications enumerated in Table 1, because to manage the entire business process, it is necessary to connect business and technical activities that complement and substitute each other. In modern information systems of that kind large databases are used, *ad hoc* questions are created, a lot of screen pictures and printed reports are used and top security is demanded. All of this leads to the using of relational databases. At the same time, connection with CAD/CAM is required, work is done distributively and with various hardware, picture and sound are connected, the concept of inheritance is applied (a customer demand is turned into an order, a customer's request and bill of material "becomes" an order which, after being executed, "turns into" a dispatch list and an invoice), long transactions (EDI) are used and a great deal of equipment of different types in the technological process is operated directly (NC-machines, the management of work activities in the technological process, automatic dispatching, the robots and manipulators in a storehouse, etc.) [15]. For all of these things to function object-oriented methods and OO-DB need to be used. Designing of such systems is a real challenge for a computer expert in which structural and object-oriented methods are not mutually exclusive, but they must be integrated in order to complement each other. Problems arising with the designing of such information systems, as well as examples of methodological processes that can be used in solving them, are the topics discussed within this paper. Practical use of this suggested approach is illustrated by a project designed for a chemical company, which is referred to as the "case study" here.

2. THE SELECTION OF STRUCTURAL AND OBJECT-ORIENTED METHODS FOR DESIGNING A COMPLEX BUSINESS IS

A great number of methods and techniques used in designing information systems are discussed theoretically and applied practically. It is important for a designer, in the process of designing a particular IS, to fully recognise the goals, characteristics and principles of a real, functioning business-manufacturing system and to choose and integrate into a consistent whole those methods and techniques that will provide the best effect. A group of methods and techniques connected in a particular way to solve a particular problem are called a methodology. We will consider a methodology appropriate to the information system development of a **b**usiness **m**anufacturing **s**ystem and will, therefore, call it *BMS-methodology*[1].

The result of each methodology application is an **information system model** consisting of: *the organisation model, the process model, the data model and the resource model*[2]. In this paper we will discuss only a part of this BMS-methodology, namely the part dealing with process and data modelling.

The suggested BMS-methodology, shown in Table 2, consists of specially chosen and specifically interrelated methods and techniques, that are gradually applied to perform individual steps while modelling the entire information system. Steps 1 - 5 refer to the phase of IS/IT planning, steps 6 and 7 refer to the analysis of a functioning production system, while steps 8 - 12 refer to the design and the development of the application software.

The sequence of steps or problems in IS designing are shown on the left of the table, and the methods used for designing each of them are shown on the right-hand column of Table 2. Depending on the type of problem they are trying to solve and the manner in which that solution is carried out, these methods can be divided into the following: strategic (sign §), structured (#) and object-oriented (¤). Some of the methods (for example DFDs or an action diagram) can be used in several of these steps. Table 2 shows the names of the methods, as well as a number of important reference books that give us a great deal more information about these various methods.

The critical points in the above process, i.e. those which computer experts usually pay too little attention or disregard completely, are steps 1, 2, 5 and 8. Additional difficulties arise from the fact that entities and concepts in different methods are not the same. For example, a business technology matrix works with data classes, DFDs with data flows and storages, an ERA-model works with objects and relations, a relational model with relational schemes and keys, and an object scheme works with classes. These constructs are interrelated, but their characteristics and meanings are different. It is, therefore, impossible to formalise the transfer of one concept to another. An IS designer must have the necessary knowledge, experience and inventiveness. And that is why, as we stated in [5], we still do not have ICASE.

[1] This name comes from the idea that there is no general methodology appropriate for designing each type of IS, but that there are methods which can be part of several specific methodologies. It is difficult to imagine that those *same methodologies* could be used in the designing of ISs for the engineering industry, banks, hospitals, the army or for sale on the Internet, although the *same methods* can be used in designing all of them (e.g. action diagrams or ERA). That problem deserves special consideration and is not a part of this paper.

[2] The concepts "*information system design*" and "*information system model*" are not synonyms as is often suggested colloquially. IS design must provide a complete plan for the improvement of BMS and deals with terms like: the application strategy of information technologies, BMS restructuring, an estimate of the effects, the design and implementation of a new IS, communications, end-user education, etc. Therefore, "IS designing" has a different meaning from "IS modelling".

Table 2. Steps, methods and techniques of BMS-methodology

Problem/step in designing or modelling	**Methods and techniques:** § -strategic; # - structural; ¤ -object oriented
1. Co-ordinate the strategic plan of business system development with the available information technologies	§ BCG-matrix [24] § 5F model [19] § Porter's "value chain" model [19]
2. Adjust the system to the business strategy	§ BPR [18] # Business system scheme (rich picture) [25]
3. Define the processes in the business and manufacturing system	# BSP-decomposition [18] # BSP-analysis of the basic resources life cycle
4. Define CSF and any information necessary for system management	§ Rockart's CFS analysis [19] # Ends-Means analysis [19]
5. Define the optimal architecture of the information system	# Business technology matrix [16], [5], [6], [7] # Affinity analysis [16], [14], [8]
6. Describe the basic business processes	# Data flow diagram (DFD) [16]
7. Describe in detail the business processes and the prerequisites for their realization	# Data flow diagram (detailed) # Action diagram [16] # Decision trees and tables [13]
8. Design the general structure and functions of the application software (procedures, events, object classes, inheritance and transition of objects)	# HIPO-diagram ¤ Event scheme [10] ¤ Designing objects and classes [2] ¤ Transition diagram [10]
9. Develop a data model	# ERA-model [1], [11] ¤ Object-model [10]
10. Design a detailed structure and the functions for specific programs and procedures	# Action diagram [16] ¤ Object scenario [10]
11. Develop a relational data model	# Relational model [11] # Normalisation [11]
12. Develop the software	# CASE and 4GL [16], [23] ¤ OO-CASE and C++ [2]

As the methods mentioned in Table 2 are generally well-known, we won't give a description and explanation of their use. We will only describe briefly the *Coad*-notation [10] for applied object methodologies.

3. STRUCTURAL-OBJECT INTERACTION IN DESIGNING A PROGRAM SYSTEM

Step 8 in Table 2 is a new IS and it is the transition from the **analysis** of a business-manufacturing system to the **design** of the software for its information system. This is also where we start talking about the interaction and the integration of structural and object methods. At this stage, it is necessary to make a general structure for the application software that will fully support the business technology developed in the previous steps. We can consider the HIPO-diagram to be the best practical notation for describing the general software structure *with regards to the amount of knowledge a designer has at that moment.*

INPUT	PROCESS	OUTPUT
	AP 1 -A description of the *1st* application	

`Input 1`
`Input 2`
:
`<Input RSh 1>`
`<Input RSh 2>`
:
	Program 1.1
	A short description of the process and its algorithm. Refer to supplement if a detailed description or object model is needed.
	Program 1.2
	Program 1.n
		`Output 1`
		`Output 2`
		:
		`<Output RSh 1>`
		`<Output RSh 2>`
		:

	AP m -A description of the *m-th* application	
	Program m.1 **Program m.n**	

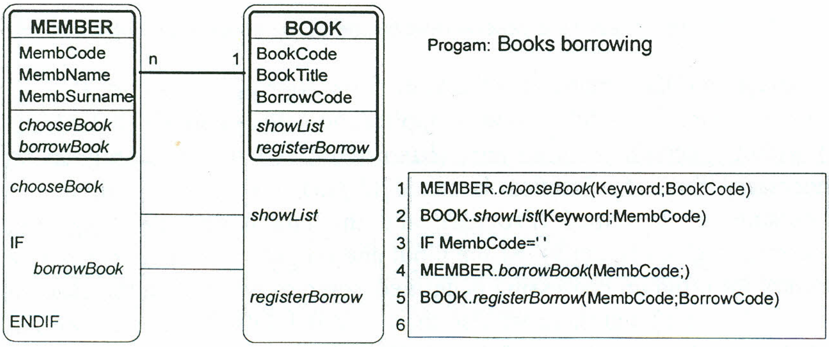
Figure 1. A HIPO-diagram giving a general description of the application software

The HIPO-diagram (HIPO being an abbreviation for *Hierarchy* + *Input, Process, Output*), shown in Figure 1 is a description of application software in which programs are arranged within a certain program area (*Hierarchy*), and for each program the input data necessary for it to function is stated (*Input*), the basic algorithms for program processing are described (*Process*), and the output data resulting from program processing is given (*Output*). At the input and output stage of each program, the data necessary for program processing is defined, some of which is in the classical media (`Input, Output`), but there are also those that will later be transformed into relational schemes and realised as relational DBs (`<Input Rsh>`, `<Output Rsh>`). When describing the process, which can't simply be explained in one sentence, a supplement is added in which the program algorithm is elaborated in more detail. This elaboration will be done within step 10 of the BMS-methodology either by using structural methods (an action diagram) or object-oriented methods (an object model and object scenario). Thus the applied HIPO-diagram is a summarised view of the entire application software, and it is also the starting point for a detailed development of specific programs, either by structural or object methods.

Programs supporting the standard business activities and depending upon relational data models will be generated on the basis of an action diagram by using standard CASE tools and will be completed with the appropriate 4GL. Programs connecting business activities with equipment for automated manufacturing, production control management or raw material and product transportation will be made using object methods. The use of these object-oriented methods is necessary in those cases where the process in the business-manufacturing system depends on the events that influence the condition of that system or when business and technological processes proceed in parallell, in *real time*, and have an influence on each other.

There are several notations for object-oriented analysis and object-oriented software development because object technologies are still being developed and standardized. For the object modelling *Coad's* notation [10] was used here because it is, according to the author's experience, the most appropriate for this phase of IS development.

The basic construction of object methodologies is an **object**, representing the abstract unity of *data* (the data describing a certain thing or phenomenon that must be taken into consideration) and *processes* (the procedures, the methods, the operations, the services) that can be performed by the object or on the object. These processes describe the behaviour of an object over the course of time. Any number of objects described by the same attributes, and on which the same processes can be performed, make up a **class** or a type of objects. *Coad* notation represents these classes of objects with a rectangle (with bold lines and rounded corners) within which three separate parts are written: the name of the class, its attributes and its processes. All the individual objects of the same type are represented by a rectangle with thin lines Figure 2.



```
MEMBER.chooseBook(Keyword;BookCode)
BOOK.showList(Keyword;MembCode)
IF MembCode=' '
MEMBER.borrowBook(MembCode;)
BOOK.registerBorrow(MembCode;BorrowCode)
```

Figure 2. Object notation - the object model and the object scenario

Objects are connected in their activities, so their relations are represented by lines above which two characteristics are usually written: *connection* (this is symbolised by a directed triangle) and *cardinality* (marked *0*, *1* or *n*). Cardinality shows how many objects of a certain type enter into a connection with objects of a second type. A model of such relations is called an *object model*, and this is shown in the upper part of Figure 2. The behaviour of the objects in a real system and the appropriate program, with all the important factors having been taken into consideration, is described in the *object scenario* which is shown in the lower part of Figure 2. In addition the object scenario contains a sequence of statements in the following form:

```
OBJECT_TYPE.process(request;answer)
```

In order to illustrate the object modelling technique, we will give a simplified example in Figure 2 of an easily understandable program for borrowing books as part of a library information system. Two objects are used in the program (MEMBER and BOOK) whose cardinality is *0,n* on the part of MEMBER (i.e. one member can borrow at least *0* or at most *n* books), and *0,1* for the BOOK (i.e. a book can either be lent to no one or to only one member at that time). Three attributes are defined for each object (MembCode, ..., BorrowCode) and two processes (***chooseBook***, ..., ***registerBorrow***). The object scenario is, according the statements on the right side of Figure 2, read in the following way:

1 object MEMBER starts a process ***borrowBook*** and sends a request to the object BOOK by giving the parameter `KeyWord`, and expecting data BookCode for the chosen book;

2 object BOOK, activated by object MEMBER, starts a process ***showList*** and displays the present value of the data BorrowCode;

3 IF BorrowCode = ' ' (empty) it means that the chosen book has not been borrowed yet, so the MEMBER who started the process can borrow it;

4 if the chosen book is available, object MEMBER starts a process ***borrowABook*** and puts on display his/her data MembCode;

5 the previous process starts a process ***registerBorrow*** of the object BOOK in which input data MembCode is recorded as BorrowCode;

6 ENDIF is the end of the loop that begins in statement 3.

If a greater number of object types participate in a relation than in the above mentioned simple example, and if the data flows between them in both directions, then the object model and the object scenario are drawn separately. Using methods mentioned earlier the following relations, that are important for the program development, are modelled: the *problem domain* (i.e. which objects and their relations are solved in the program), any *interaction with the system* (i.e. which object begins a certain process), the *user interface* (i.e. what the data is and in what form it will be presented to the user) and any *data management* (i.e. what data about certain objects will be saved and also how it will be saved).

4. THE INTEGRATION OF STRUCTURAL AND OBJECT METHODS IN A REAL IS PROJECT

The business-manufacturing system, whose entire effectiveness needs to be improved by using new information technology, is a firm that could be described as the *continuous manufacturing of standard products* of a process type [22]. Generally this type of firm can be, in a general taxonomy of the production systems, marked (described) by the abbreviation CMSP. Throughout the paper this abbreviation will also stand for the firm name. A real production system called CMSP is shown in Figure 3. Its basic characteristics are:

a) Product assortment consists of nearly 20 products, each of them manufactured in 20-100 varieties. Product documentation and the manufacturing technology are made in the form of a prescription, called the *formula*. Whenever a new variety of a product is needed, a new formula is made and analysed. The production process is automated, with the packing of products into packages for sale being an integral part of the manufacturing process. The biggest problem is the necessary adjustment of equipment whenever a new variety of product is made. For that reason the firm produces larger quantities of products than currently needed, according to the estimate for the customers' future demands. Finished products are stored, and later dispatched upon demand. The storehouse also receives unpacked products, so that manufacturing process can sometimes become simply a matter of putting the products into their packages and then they're ready to be sold.

b) The firm has about 2000 regular customers and receives daily about 50 customer orders, with approximately 10 items on each order. These orders are usually collected by the sales agents, and the customers expect the the goods to be delivered in the shortest possible time (maximum: three days). The products are delivered to customers all over the country in small vans owned by the firm.

c) The firm has a small number of suppliers they use most of the time. The basic raw materials are chemical substances but the packages used for the finished product are also very important.

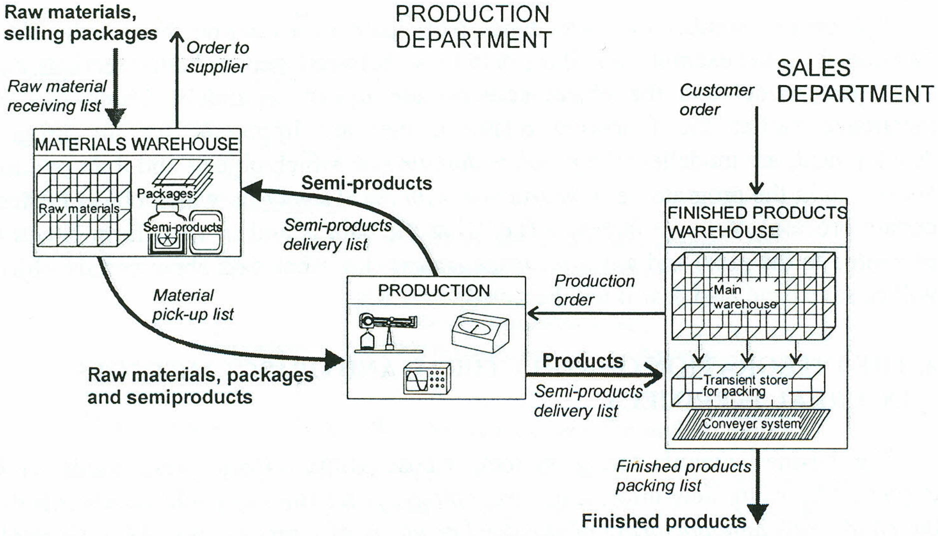

Raw materials, selling packages
Order to supplier
Raw material receiving list
PRODUCTION DEPARTMENT
SALES DEPARTMENT
Customer order
MATERIALS WAREHOUSE
Raw materials
Packages
Semi-products
Semi-products
Semi-products delivery list
FINISHED PRODUCTS WAREHOUSE
Main warehouse
Material pick-up list
PRODUCTION
Production order
Products
Semi-products delivery list
Transient store for packing
Conveyer system
Raw materials, packages and semiproducts
Finished products packing list
Finished products

Figure 3. The basic manufacturing and business flows of a CMSP

The firm's management, along with its business consultants, determined the strategic goals for the firm and how to improve its business in the following ways:

a) Accelerate the collection of customers' orders and make sure that all that needs to be ready for a given day is ready, so that at least 90% of orders received during any working day can be delivered the next day.

b) Adjust the quantity of the stored, finished products so as to meet customers' demands and to meet the expenses of the production resources adjustment.

c) As recycling is impossible (because chemical processes are irreversible), it is necessary to reduce the shrinkage to the level of the world's best firms in that industrial branch.

In defining the above list of goals the principles of Porter's "Value chain" analysis were used. The main business flows were also reengineered, as shown in Figure 3. During reengineering, the effect of new IT was estimated in some of the important business processes (e.g. in order collecting, in delivery automatization, and in production process management). The matrix of a *new* business technology was made after that and the IS architecture for CMSP was optimized. It was concluded that the entire IS can be divided into seven subsystems, within which the shortest business flows were defined. Using *BCG*-analysis and *Porter's* "Value chain", the priority for the design of specific IS subsystems was established. For each of these subsystems a detailed process model was made using DFD techniques, with a special emphasis on those processes in which the new information technology would be applied. An example of the process model for the subsystem SALES, considered to be one of the most important ones, is shown in Figure 4.

How using IT to improve a certain process and exactly how it improves it cannot be fully measured without a complete process analysis. In order to make such an analysis, it is necessary to use modelling and description techniques not ordinarily used by computer experts. Figure 5 shows the analysis and the final results of the development process for **Product Shipment** (as defined on the right side of the DFD in Figure 4). To produce the information the "rich picture" technique was used.

The dispatching process is a critical point in CMSP management for two reasons: each customers' order must be fulfilled within 24 hours after has been received, and in the process of packing and dispatching there must be no mistakes (which are possible with manual work, epecially when considering the similarities between the products and the packages). In order to avoid making these mistakes, the store functions were separated from the dispatch functions, throughout BPR. The system shown was named *transient warehouse*, and it consists of a wall unit with 200 cells sized 50x50x80 cm. The cells are emptied from the front, when the products are dispatched to the customers, and then filled in from behind, when the quantity of products is at a minimum. The system of storing and filling the cells is a separate process, and this proces is not shown in Figure 5.

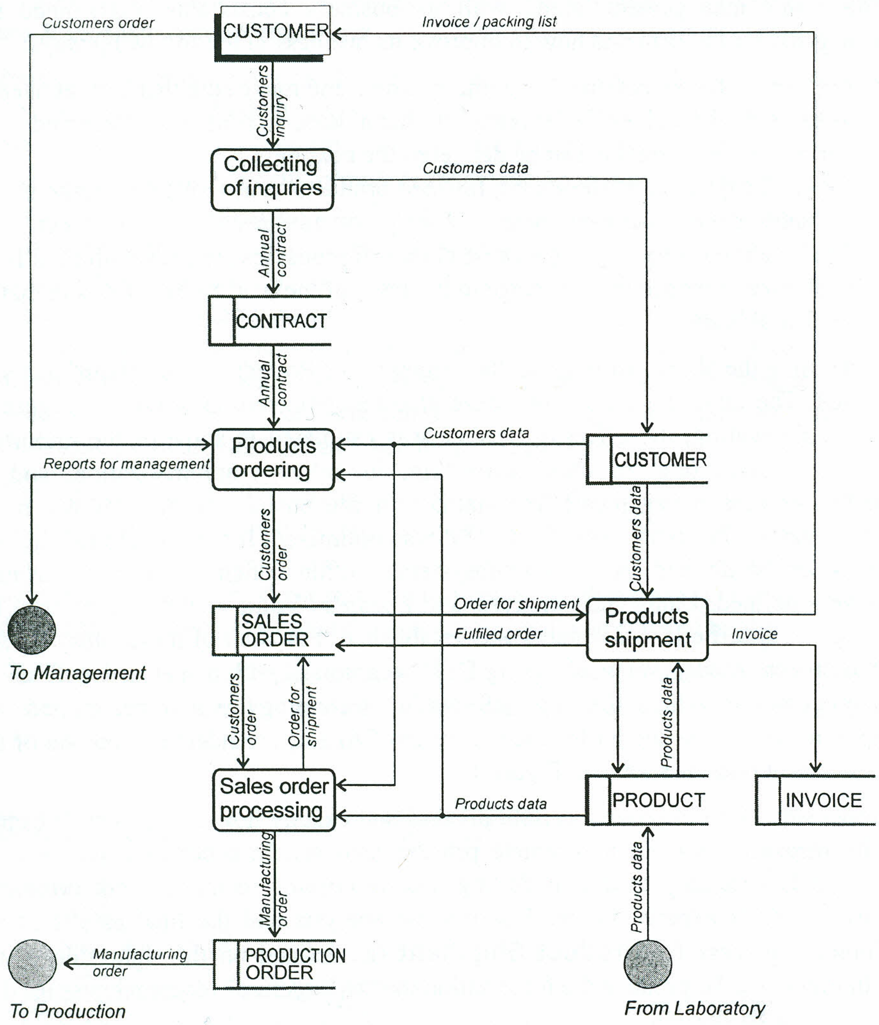


Figure 4. A process model for the subsystem SALE

In order to further emphasise that only with a total understanding of all the details in a certain process, can the use of appropriate information technologies be considered (including the choice of hardware, software modelling and work procedure definitions), we will describe the process **Product shipment** by quoting a part of the text from the project documentation made while working on this particular part of the project.

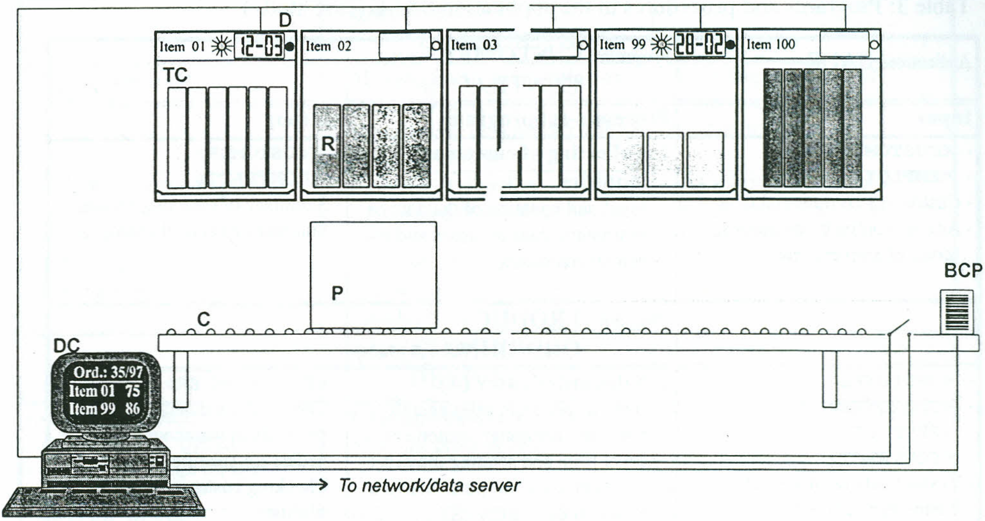


Figure 5: The scheme for the *Product Shipment* process

> "For all orders, depending on their priority that's determined by a delivery deadline, a semi-automated dispatch procedure is carried out (shown schematically in Figure 5). After an employee chooses an order on the dispatch computer (**DC**), all its items are automatically shown above the transient warehouse cells (**TC**). Above every cell (from which the products must be taken to fulfil a chosen order), a small lamp switches itself on, and the quantity of the products needed is written on on a small LCD display (**D**). The employee takes the ordered product item (**I**) and packs it into a package (**P**), then that goes down the conveyer belt (**C**). After he takes out the ordered quantity of these chosen products, he presses the key next to the screen (notifying that the order was executed) and the lamp switches itself off. When all the items of the order are fulfilled, packets of these products then go to an electronic scale with a switch (**S**), where the correctly fulfilled order is additionally checked by comparing the weight of the packet with the calculated weight of the ordered products. If the weights correspond, a *packing list* and *invoice* are printed (the bar code printer **BCP** is at the end of the conveyer belt) twice (one of them goes to the customer, the other one goes to the subsystem ACCOUNTING), as well as the customer's address and post-code. Both of which are written in bar code (and then taped to the appropriate place on the package). The package is then automatically sent to the loading conveyer belt (the process known as **Loading and transportation of products**). If the actual and calculated weights do not correspond, the package is put on a separate line for further checking. A copy of the packing list, with the customer's receipt for products received is returned after delivery to the process **Orders processing** in order to register the completing of that order. A detailed outline of this process, as well as the procedures used when actual and calculated weights do not correspond, are described in the program **PRODUCT SHIPMENT**."

The above text shows that structural methods alone are not enough when you want to develop a software to support the described work procedures and connect a business information system with the automated production equipment. Therefore, it is necessary at this stage to integrate object and structural methods. In accordance with section 2 of this paper, a part of the HIPO-diagram for this particular case study is shown in Table 3.

Table 3: Programs and procedures in the application SALE (case study)

Aplication: **SALE**	Program: **CUSTOMERS and CONTRACTS**	
Input	**Processes-subprograms**	**Output**
- <CUSTOMER> - <EMPLOYEE> - Customer's commercial data - Annual contracts: quantities, terms of delivery, etc.	1. **Collecting of customers' inquiries** Input and updating of the data on customers, their inquiries and any annual contractsx	- <CUSTOMER> - <CONTRACT> - Summary of customer groups - Summary of annual contracts
	Program: **PRODUCT ORDERING (A)+(S)**	
- <EMPLOYEE> - <CUSTOMER> - <PRODUCT> - <CONTRACT> - Textual data from annual customer's contract - Customer's order	1. **Sales order entry (A)** Input of orders in sales agent's notebook computer, customer's order print and sending the order to a firm via a computer network 2. **Sales order entry (S)** Receiving and input of customer's order, print them and send them to the customer	- <SALES ORDER> - Checking the quantity of products in warehouses and the orders for specific products - Checking customers' paying abilities
	Program: **SALES ORDER PROCESSING**	
- <CONTRACT> - <CUSTOMER> - <PRODUCT> - Product type and quantity in stock	1. **Sales order processing** Check the received orders, calculate the discounts. If the quantity of product in stock is minimal, launch a production order	- <SALES ORDER> - <PRODUCTION ORDER>
	Program: **PRODUCT SHIPMENT**	
- <SALES ORDER> - <PRODUCT>	1. **Distribution of the ordered products in the warehouse cells** A detailed description of process analysis in an object model	- Signal on the front side of the warehouse cells: signal lamp switches on, the number of packages and items is printed out
- <SALES ORDER> - <PRODUCT>	2. **Management of product delivery from warehouse cells** A detailed description of the process in an object model	- Delete data, feedback after issuing. - Priority list of warehouse cells that need refilling
- <SALES ORDER> - <PRODUCT> - <CUSTOMER>	3. **Preparing of invoices** Compare the weight of the ordered and delivered products, print the invoice, charge the buyer, lower the stocks. Make a denial of invoices possible.	- <INVOICE> - <PRODUCT> - Print out the bill/packing list - Customer's post-code, printed in bar code
- <SALES ORDER> - <PRODUCT> - <CUSTOMER>	4. **Export** Additionally make the export invoice and the accompanying documentation	- <INVOICE> - <PRODUCT> - Export invoice and - documentation

	Program: **CARGO DISTRIBUTION**	
- Moving hand data	1. **Moving hand manipulation** Starting, stopping and manipulating its functions using a built-in *firmware*	- Signalling moving hand position and the efficiency of product transfer
- <SALES ORDER> - <CUSTOMER> - <EMPLOYEE> - <VEHICLE - Moving hand data	2. **Cargo distribution** - A detailed description of the process in the object model - Records of vehicles and employees that transport products	- <TRANSPORT ORDER> - Printed transportation order
	Program: **CLOSING DELIVERY**	
- <SALES ORDER> - <TRANSPORT ORDER> - Customer's receipt for the delivered products	1. **Closing customer orders** Mark the order as fulfilled	- Summary of the fulfilled orders
- <SALES ORDER> - <INVOICE> - <PRODUCT> - Customer's complaint	2. **Processing complaints** Record a complaint, a denial of the invoice with the items, change the amount of the customer's debt	- Summary of complaints

Certain programs were described in more detail in the form of an action diagram, but since this design method is generally well known, it will not be described in this paper.

Programs integrating business and technical processes were modelled by the previously mentioned object methods in *Coad* notation. For example, Figure 6 shows the object scheme, and Figure 7 shows the object scenario for **Product shipment** in this case study.

When designing such programs, some of the object oriented CASE-tools will be used. A great number of specific program procedures for control of technical equipment in a transient warehouse (e.g. electronic control of the transient warehouse cells, the LCD which displays the data for the quantity of ordered products, light signalling, feedback etc.) will be written in **C++** and will be linked up with the *firmware* equipment.

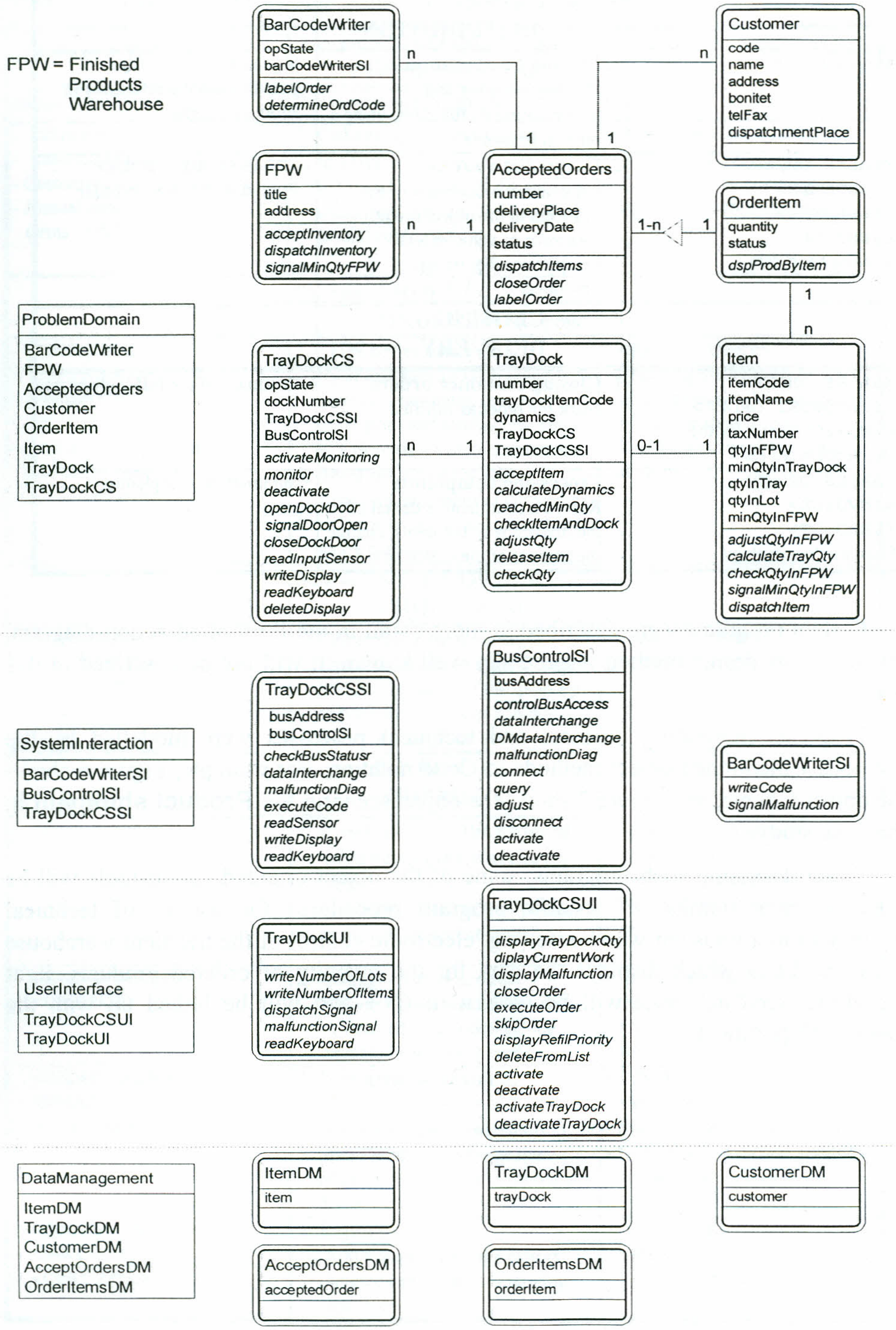
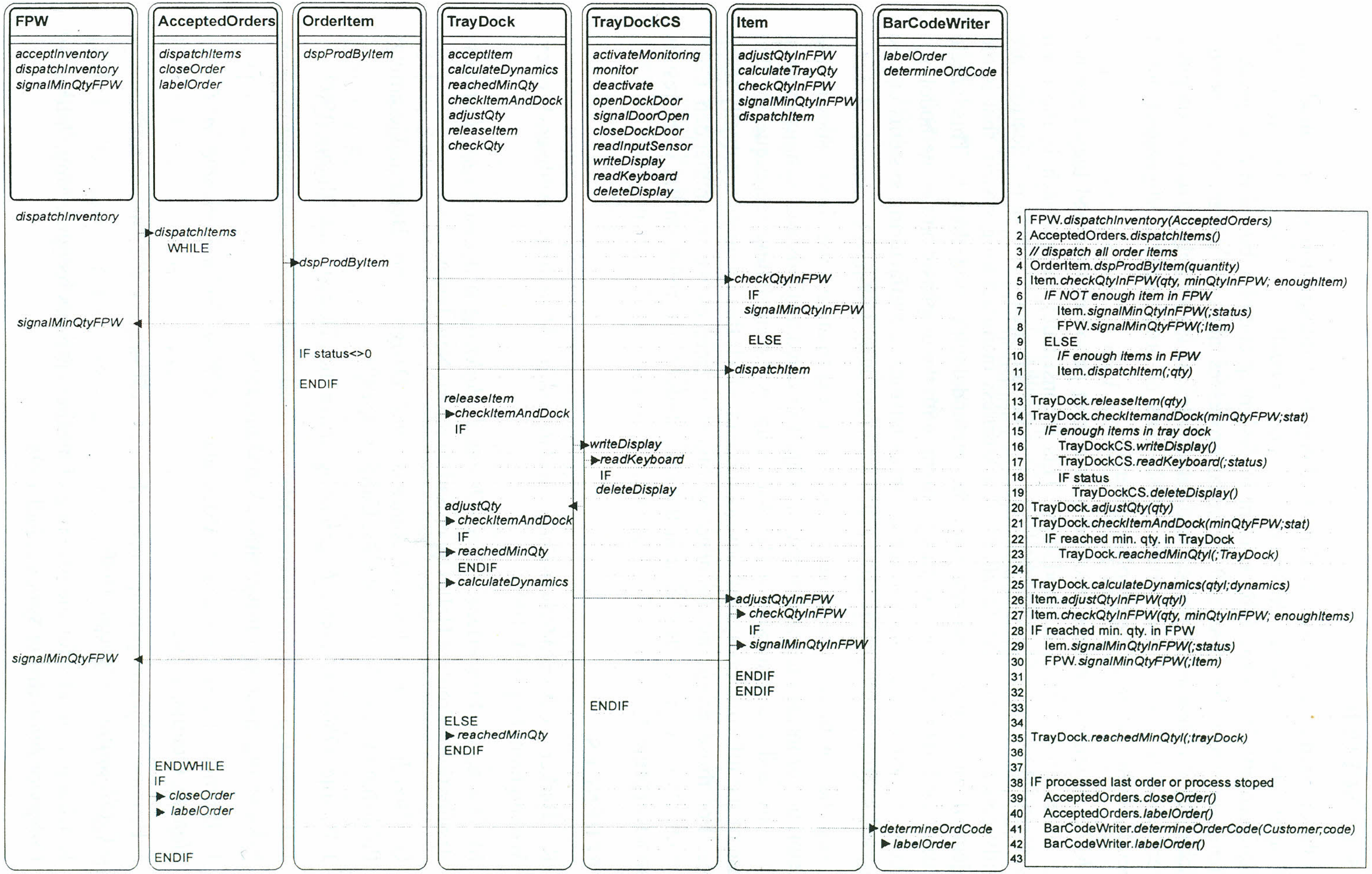


Figure 6. An object model for part of a real project (*case study*)



```
FPW.dispatchInventory(AcceptedOrders)
AcceptedOrders.dispatchItems()
// dispatch all order items
OrderItem.dspProdByItem(quantity)
Item.checkQtyInFPW(qty, minQtyInFPW; enoughItem)
  IF NOT enough item in FPW
    Item.signalMinQtyInFPW(;status)
    FPW.signalMinQtyFPW(;Item)
  ELSE
    IF enough items in FPW
    Item.dispatchItem(;qty)

TrayDock.releaseItem(qty)
TrayDock.checkItemandDock(minQtyFPW;stat)
  IF enough items in tray dock
    TrayDockCS.writeDisplay()
    TrayDockCS.readKeyboard(;status)
    IF status
      TrayDockCS.deleteDisplay()
TrayDock.adjustQty(qty)
TrayDock.checkItemAndDock(minQtyFPW;stat)
  IF reached min. qty. in TrayDock
    TrayDock.reachedMinQtyl(;TrayDock)

TrayDock.calculateDynamics(qtyl;dynamics)
Item.adjustQtyInFPW(qtyl)
Item.checkQtyInFPW(qty, minQtyInFPW; enoughItems)
IF reached min. qty. in FPW
  lem.signalMinQtyInFPW(;status)
  FPW.signalMinQtyFPW(;Item)




TrayDock.reachedMinQtyl(;trayDock)


IF processed last order or process stoped
  AcceptedOrders.closeOrder()
  AcceptedOrders.labelOrder()
  BarCodeWriter.determineOrderCode(Customer;code)
  BarCodeWriter.labelOrder()
```

Figure 7. An object scenario for a part of a real project (*case study*)

5. CONCLUSION

Object-oriented and structured methodologies of information systems modelling and application software designing are not mutually exclusive, but they are complementary. It is especially evident in designing complex business-management systems, in which business and technical processes must be integrated. Choosing which method to use in the process and data modelling isn't solely up to the computer expert (as they tend to use certain CASE-tools or programming languages) but it depends on the *nature of the problem* at hand.

MRP systems have been evolved over a long period of time and they have not been based on a certain methodology. This inconsistency in approach is the main reason why there is no relevant literature to cover this area. Existing publications are mainly concerned with the functions of a business manufacturing system that is the subject of an analysis and not with the methodologies used therein. This article presents a different approach to this problem, with the emphasis on the methodology. It shows a methodology with which complex business manufacturing systems can be analysed in order to obtain the appropriate information system.

Together with the necessary theoretical explanation, we have shown the possibilities of integrating structured and object-oriented methods and techniques for a case study and recommended a HIPO-diagram as the modelling technique for that phase of the information system development in which this integration actually takes place. For those programs designed by object techniques we recommended *Coad* notation. Its usage is exemplified in the case study which relies on the work results from a real project.

REFERENCES

[1] R. Barker. *CASE*Method-Entity Relationship Modelling*. Addison-Wesley, Wokingham, England, 1990.

[2] M. Blaha, W. Premerlani. *Object-Oriented Modelling and Design for Database Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[3] G. Booch. *Object-Oriented Analysis and Design*. The Bejamin/Cummings Publishing Comp, Redwood City, California, 1994.

[4] J. Bourne. *Object-Oriented Engineering*. Irwin Inc., Homewood, Illinois, 1992.

[5] J. Brumec. Optimizacija strukture informacijskih sustava. *Zbornik radova,* broj 18, Fakultet organizacije i informatike, Varaždin, 1994.

[6] J. Brumec. Epistemologija CASE-alata. *CASE6-šesto savjetovanje o CASE-alatima*, Opatija, 1994.

[7] J. Brumec. Objektni pristup razvoju IS-a i CASE-alati. *CASE8-osmo savjetovanje o CASE-alatima*, Opatija, 1996.

[8] K. Chung. A model for the planning of business process reengineering. *Journal of Computer Information Systems*, Fall 1998.

[9] P. Coad, E. Yourdon. *Object-Oriented Analysis*. 2nd ed. Yourdon Press, Prentice Hall, New Jersey, 1991.

[10] P. Coad. *Object Models-Strategies, Patterns & Applications*. 2nd ed. Yourdon Press, Prentice Hall, Upper Saddle River, New Jersey, 1997.

[11] C. J. Date. *An Introduction to Database Systems*. 6th ed. Addison-Wesley, Reading, MA, 1995.

[12] I. Graham. *Object-Oriented Methods*. Addison-Wesley Publishing Company, London, 1994.

[13] J. Hoffer, J. George, J. Valacich. *Modern Systems Analysis and Design*. The Benjamin/Cummings Publishing Company, Menlo Park, California, 1996.

[14] K. Klasić. Modeli optimizacije strukture informacijskog sustava. Doktorska disertacija, Sveučilište u Zagrebu, Fakultet organizacije i informatike, Varaždin, 1998.

[15] K. Kurbel. *Produktionsplanung und Steuerung*. R. Oldenburg Verlag, Wien, 1995.

[16] J. Martin, J. *Information Engineering, I, II, III*. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[17] J. Martin, J. Odell. *Object-Oriented Analysis and Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[18] S. Montgomery. *Object-Oriented Information Engineering*. Academic Press, London, 1994.

[19] M. Robson. *Strategic Management and Information Systems*. Pitman Publishing, London (1997).

[20] J. Rumbaugh et al. *Object-Oriented Modelling and Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[21] P. Sallis et al. *Software Engineering*. Addison-Wesley Publishing Company, Sydney, 1995.

[22] T. J. Strader, F. R. Lin, M. J. Shaw. The impact of information sharing on order fulfillment in divergent differentiation supply chains. *Journal of Global Information Management*, Vol 7, Jan-Mar, 1999.

[23] I. Sommerville. *Software Engineering*. 5th ed. Addison-Wesley Publishing Company, Reading MA, 1995.

[24] J. Ward, P. Griffiths. *Strategic Planning for Information Systems*. John Wiley & Sons Ltd., Chicester, England, 1996.

[25] E. Yourdon. *Modern Structured Analysis*. Yourdon Press/Prentice Hall, Englewood Cliffs, New Jersey, 1989.

Received: 17 June 1998
Accepted: 5 October 1998

Brumec Josip
Vesna Dušak
Neven Vrček

STRUKTURNE I OBJEKTNO ORIJENTIRANE METODE U SLOŽENOM IS PROJEKTU

Sažetak

Objektno orijentirane i strukturne metode nisu međusobno isključive, iako svaka od njih ima prednosti pri korištenju u izgradnji specifičnih tipova informacijskih sustava. Zbog toga ovaj rad obrađuje određena svojstva poslovnih sustava i njima pripadajućih informacijskih sustava koja utječu na odabir metoda razvoja informacijskih sustava. Može se zaključiti da se za razvoj složenih informacijskih sustava, posebice onih koji povezuju poslovne i tehnološke procese, najbolji rezultati postižu kombinacijom strukturnih i objektno orijentiranih metoda. U radu je prikazan način na koji se strukturne i objektno orijentirane metode povezuju u konzistentnu metodologiju za razvoj informacijskih sustava te je prikazano njezino korištenje na stvarnom primjeru.

Ključne riječi: razvoj informacijskih sustava, objektno orijentirane metode, strukturne metode, BMS-metodologija.