Multi-Modeling of an Industrial Application with DFD, E-R and Object-Oriented Methods

J. Palazzo M. de Oliveira, N. Edelweiss

Instituto de Informática

Universidade Federal do Rio Grande do Sul

Av. Bento Gonçalves, 9500 - Agronomia - Caixa Postal 15064

91501-970 - Porto Alegre - RS - Brazil

Fax +55(51)336 5576

E-mail: [palazzo, nina]@inf.ufrgs.br

Abstract

Modeling industrial systems requires different modeling paradigms to capture the static and dynamic systems' characteristics. The modeling process must be divided in two major phases: the high-level description and the detailed system description. In this paper two modeling methods are used for the first level: the dynamic behavior is described in terms of Data Flow Diagrams and the static data structures and resources relationships are described with Entity-Relationship models. The detailed level description is done by an object-oriented model. At this level the industrial processes' states and temporal constraints are defined. The final model may be translated to a extended relational database system allowing the modeling tools integration and the operational database construction.

Keywords

Computer-aided engineering, computer-aided design, conceptual modeling

1. INTRODUCTION

The modeling of industrial systems differs from information systems modeling in some important ways. Two of the most meaningful are the multidimensional representation of industrial systems and the necessity of a *rough modeling methodology*: the construction of a model of the first levels of a plant project, as seen from different viewpoints to promote the communication between the various parts during the early phases of a plant project. Such a model must satisfy the following requirements: (i) to establish a common language for the different plant views; (ii) to represent the industrial system at different levels of abstraction, (iii) to provide the way to link the different modeling tools and concepts and, (iv) to integrate the business and industrial plant's features [Volainen 92]. In the next level of modeling the first high level model shall be translated to a detailed formal description. This is obtained by detailing each partial view and integrating all the views in the same underlying model. Some of the methods for modeling industrial systems at the different levels are:

- **SADT** [Ross 77], a software engineering method used for the decomposition of complex software systems in functions, sub-functions and activities. It is used to describe or specify

systems. It does not handle flows but dependencies. It cannot be used to control systems operations.

- **Petri Nets** [Reisig 85], formal analytical tool for parallel and concurrent system analysis. Has qualitative and quantitative power. Provides an executable model.
- **ER - Entity-Relationship** [Chen 76] is focused on information modeling. Is widely used and easy to understand. Very good for conceptual data modeling. Supported by good graphical representation.
- **IDEF** [Bravoco 85], an attempt to extend SADT to CIM - Computer Integrated Manufacturing. Composed by three non-integrated modeling techniques: IDEF0 - function modeling based on SADT, IDEF1 - information modeling based on ER, and IDEF2 - dynamics modeling based on queues.
- **Object-Oriented Modeling Methods** [Coad 91, De Antonellis 90, Henderson-Sellers 90], representing the static and dynamic properties of real world entities using the object and message concepts.

This paper's main concern is to show the use of some modeling tools - DFD, E-R and an OO model - in an industrial environment. Initially all the *domains* are established and a high level DFD model is derived ensuring communications with and between system users and operators. The next step is the mapping of the functional requirements to the E-R model detailing the data structures. Then the OO model is constructed, using the temporal object-oriented data model TF-ORM (*Temporal Functionality in Objects with Roles Model*) [Edelweiss 93]. This model is formal and incorporates in a strict syntax the systems' behavior, temporal constraints and data structures.

The formal OO description is implemented by a database acting as the integrating substrate to all the modeling tools. The TF-ORM may be mapped to an OO database or to an extended relational database as POSTGRES [Oliveira 95]. The particular industrial semantic is associated with the application programming interface to allow convenient a data access to the modeling tools and production procedures. The final design database is used to generate the operational database necessary to support the production control and administrative applications integration, Figure 1.

Figure 1 Tools integration

In section 2 a case study is presented. The system's dynamic behavior is described in terms of DFDs in section 3. In section 4 the data structures are presented in the E-R model. In section 5 the formal OO representation is presented using the TF-ORM data model. Finally, in section 5.3 the main characteristics of the three models are revised and the future work is described.

2. A CASE STUDY

A simple case is described here to be used in the examples of this paper. The example is based on the one developed in [Lutherer 94] modeling an industrial enterprise. This example represent the automation of a concrete plant including the sales office and the industrial plant shown in

Figure 2. This example is being used as a common reference for modeling methods comparison purposes.

The plant is composed of: several bins to store the cement, sand, aggregates and the water; a mixer for the concrete preparation; two scales to weight the components; a metering valve to control the water discharge; two conveyor belts to carry the solid ingredients to the mixer.

Figure 2 The concrete business

The customer places an order for some concrete quantity and kind for one or more days. The *Sales Department* schedules the orders and eventually sends it to the plant. After the order is delivered at the *Construction* site the *Customer* is billed.

3. SPECIFICATION WITH DATA FLOW DIAGRAMS

At the requirement level the goal is to define the enterprise requirements in terms of enterprise functionality, behavior and structure by means of a functional decomposition process. The functional decomposition identifies the *Domains*. A domain is a subset of the enterprise defined by its *domain objectives* which are fulfilled by the *domain processes*. In the example two domains are identified: the "sales department" and the "concrete plant".

The first description of domains may be obtained by the use of a semi-structured definition containing the main domain's components: the description, the objectives, the constraints, one or more domain processes, the domain relationships with other domains (boundary), the object views and the associated events. This process - the requirements collection and organization - may be supported by *templates*, in this paper this phase is not detailed, only one template is presented. See Table 1.

For each *domain* process a new description must be built. The domain processes are composed by *business processes*. Business processes represent intermediate structures in the function decomposition hierarchies. At the lower level, the leaves of the hierarchy, appears the *enterprise activities*. Enterprise activities are identified as tasks which consume their *function input* and produce *function outputs* as objects views, and which will require allocation of *resources* and time for their execution [Vernadat 93]. Each level should be detailed by an appropriated description. The decomposition is concluded when all the enterprise activities are clearly defined.

Table 1: Concrete plant

DOMAIN	Production
Type:	Industrial
Identifier:	P
Name:	Producing concrete
Design authority:	Chief engineer
Description:	the concrete plant
Objectives:	to produce the concrete “just-in-time”
Constraints:	to use 80% of the total plant capacity
Processes:	production control, cement injection, aggregate injection, sand injection, water injection, mixing
Boundary:	concrete specification, concrete load
Object views:	production plan, loaded truck to construction
Events:	charge specification received, start cement injection, close cement injection, start water injection, close water injection, start sand injection, close sand injection, start aggregate injection, close aggregate injection, start mixing, discharge concrete

Figure 3 The concrete business, second level DFD

The requirements description process corresponds closely to the elaboration of Data Flow Diagrams - DFD. To each level in the enterprise modeling (domain process, business process, enterprise activities) corresponds a DFD diagram. In this phase, DFD construction, the *behavior analysis* is performed and the functional relationships among the processes and activities are detailed. In the present example at the second level DFD the business processes are identified, Figure 3 (the first level DFD is the context specification representing the system’s relationships with the external world).

Figure 4 The concrete plant, third level DFD

The industrial components are identified in this example at the third level DFD. We will concentrate our attention on the *concrete plant* (1.2), described in the following DFD, Figure 4.

The *concrete plant* is decomposed in the component elements representing the control structure of the plant. The kind of DFD here employed is an extension of the traditional representation where only data flows were allowed. In an industrial case the model should be extended to a DFD* to include not only the data but the control and material flow [Santos 86] represented here by a thick arrow. The data and control flows are equally represented by a narrow arrow as the control information is associated with a data value.

4. THE E-R MODEL

The E-R model is very well adapted to the data's structure representation. In an industrial system two well defined E-R structures are identified: the *administrative view* and the *industrial view*. In the first one each entity and relationship corresponds to a usually huge volume of occurrences, as in a database. In the industrial view the control structure (partially shown in Figure 5) produces a complex E-R diagram but the extension of the corresponding database contains very few occurrences, representing the production states. In the *industrial view* the entities represent the state variables and the relationships attributes are the operating state of the equipment.

The construction of the DFD and E-R representations is a joint procedure described in the chapter 9 of Batini's book on Conceptual Database Design [Batini 92]. This author stresses the complementary characteristics of *data analysis* and *functional analysis*. The methodology links together data and functional schemas, through external schemas, mutual refinements, and navigational schemas to produce a conceptual specification of data and functions.

5. THE TF-ORM SPECIFICATION

In this section the two previously obtained schemas are integrated. To represent the resulting schema the temporal object-oriented data model TF-ORM is used.

5.1 The TF-ORM Model

In the TF-ORM model a class is defined by a name and a set of roles. A role is defined by a name, a set of properties, a set of abstract states the role can assume while playing this role, a set of messages the role can receive or send, and a set of rules (state transition rules and integrity

rules). Logic formulas can be associated to the state transition rules, acting as conditions that constrain the state transitions. The TF-ORM model supports specialization and aggregation mechanisms, with the possibility of inheriting roles, or redefining them.

Figure 5 The E-R model

Three different types of classes can be defined: (i) resource classes, modeling information and resources; (ii) process classes, representing the processes that can be executed with this information and resources; and (iii) agent classes, representing the persons that execute the processes. Agent classes present also, besides the above mentioned messages, personal decisions, representing non-structured work in a formal environment.

All instances (class and role instances) have temporal information associated - the instance's creation time and destruction time, and the instances in which the instance's activity was suspended and resumed. This information is stored in special pre-defined properties and is used by the query language to retrieve information.

Each class presents a special role, the *base-role*, where the global properties inherited by all other roles and the initial characteristics of the other roles are described. Properties may be static (having the same value during all the instance lifetime) or dynamic (properties that may assume different values with time). Dynamic properties have two different time points associated with each value: the *transaction time*, corresponding to the moment when the information is introduced in the database, and the *valid time*, the time when that information starts to be valid in the real world. When defining a property, a domain is assigned to the properties' values. Possible domains are: an object, a set of objects or a list of objects. TF-ORM presents a set of pre-defined classes to be used as domains in properties definition, called *data types*. Beside usual numeric data types (*real*, *integer*), a number of temporal data types are defined (like *date*, *hour*, *instant*, *semester*). Time is modeled as varying in a discrete form.

5.2 Stepwise specification construction in TF-ORM

When using the TF-ORM data model, a specification can be constructed stepwise, presenting different refinement degrees. Using a top-down specification method, a class can be refined in several classes - each one of the roles present in the class becomes a class in the next refinement. The refinement technique is specially important when using process classes. The roles of a process class represent different tasks executed in the process. In a next refinement, each one of these tasks will become an independent process.

In the refinement process, messages received by the role transformed in a class will appear as incoming messages in at least one of the roles of the new classes; messages sent by the former role will be sent by at least one of the new classes roles. The refinement will complete the specified information, identifying additional features.

5.3 The TF-ORM Modeling

The TF-ORM data model can be used to integrate the two previously developed DFD and the E-R models. The resulting model will be a complete and formal representation of the enterprise. The advantage of developing these two models prior to the complete one is the partial modeling of the main characteristics separately simplifying the understanding of the whole model. The stepwise construction of the specification will be performed, following the construction of the DFD diagrams. At the same time, the E-R model is considered to complete information about data and states. As presented in section 3, the first level of the enterprise modeling corresponds to the *domain processes* representation. Each one of these domains is represented in the TF-ORM model as a process class. Each *business processes* that composes a domain process is represented as a role in the corresponding TF-ORM process class. Refinements are applied to these classes, transforming the roles in new process classes, until the roles of the new classes represent independent *enterprise activity*. These refinements represent the further levels represented in the DFD diagrams.

Control flow, defined in the DFD diagrams, is represented in TF-ORM through state transition rules. Data and material flows are represented by message parameters involved in the state transition rules. Information about the possible states of processes and activities are defined in the E-R model.

The mapping of the DFD diagrams to the TF-ORM model defines also resource classes, represented by repositories in these diagrams. The necessary properties to define data are recovered from the E-R model. In the case study presented in this paper we start with the domain process concrete production, represented by the DFD of Figure 4. This process is represented in TF-ORM as a process class; the two business processes (concrete plant and sales department) are represented as roles in this process class. Four resource classes are also represented (enterprise, customer, sale and construction) to model their interaction with the process class and to define the data structures. Properties of resource and process classes are identified in the E-R diagram. The message exchange between roles and classes is identified by the flows of the DFD specification, representing control, data and material flows, with parameters identified in the E-R model. As the specification is being constructed stepwise, through successive refinements, classes that are not yet defined are represented as EXTERNAL_WORLD.

```

Process class (
  CONCRET_PRODUCTION,
  < Base_Role,
  rules = {
    r1: msg(create_object) => msg(add_role(sales_department)),
                                msg(add_role(concret_plant)), state(active),
    r2: state(active), msg(suspend_object) => msg(suspend_role(sales_department)),
                                msg(suspend_role(concret_plant)), state(suspended),
    r3: state(suspended), msg(resume_object) => msg(resume_role(sales_department)),
                                msg(resume_role(concret_plant)), state(active) }
),
< Sales_Department,
states = { active, waiting, suspended },
messages = {
  order(Quantity: integer, Concret_Type: {strong, normal, week },
        Customer_incr: string, Delivery: date) from EXTERNAL_WORLD,
  reg_cust_order(Customer: string, Sale_inscr: integer) to CUSTOMER.Register,
  reg_order(Customer_number: string, Quantity: integer,
            Type: {strong, normal, weak}, Price: real, Del_date: date) to SALE.Register,
  production_order(Order_number: integer) to CONCRET_PRODUCTION.concret_plant,
  concret_received (Sale_nr: integer, Delivery_date: date)
    from CONSTRUCTION.Reception,

```

```

    reg_delivery(Sale_nr: integer, Delivery_date: date) to CUSTOMER.Register },
rules = {
    r1: msg(add_role) ⇒ state(active),
    r2: state(active), msg(suspend_role) ⇒ state(suspended),
    r3: state(suspended), msg(resume_role) ⇒ state(active),
    r4: state(active), msg(order(Quantity, Type, Customer, Delivery)) ⇒
        msg(create_object(sale)),
        msg(reg_order(Customer, Quantity, Type, Price, Del_date)),
        msg(reg_cust_order(Customer, Sale_inscr)),
        msg(production_order(Sale_inscr)), state(waiting),
    r5: state(waiting), msg(concret_received(Sale, Delivery_date)) ⇒
        msg(reg_delivery(Sale, Delivery_date)), state(active) }
>,
< Concret_Plant,
states = { active, suspended },
messages = {
    production_order(Sale_nr: integer) from CONCRET_PRODUCTION.sales_department,
    concret_delivery(Sale_nr: integer, Delivery_date: date)
        to CONSTRUCTION.Reception},
rules = {
    r1: msg(add_role) ⇒ state(active),
    r2: msg(suspend_role) ⇒ state(suspended),
    r3: msg(resume_role) ⇒ state(active),
    r4: state(active), msg(production_order(Quantity, Type, Customer, Sale, Delivery)) ⇒
        msg(concret_delivery(Sale, Delivery)), state(active) }
> )
Resource class (
CUSTOMER,
< Base_Role,
    static properties = {
        (Customer_number, string) },
    dynamic properties = {
        (name, string),
        (address, string),
        (phone, string),
        (sales, set of SALE) },
    rules = {
        r1: msg(create_object) ⇒ msg(add_role(Register)), state(active),
        r2: state(active), msg(suspend_object) ⇒ msg(suspend_role(register)), state(suspended),
        r3: state(suspended), msg(resume_object) ⇒ msg(resume_role(register)), state(active) }
>,
< Register,
states = { active, suspended },
messages = {
    reg_customer(Customer: string, Name: string, Addr: string, Phone: string)
        from EXTERNAL_WORLD,
    change_name(Customer: string, NewName: string) from EXTERNAL_WORLD,
    change_addr(Customer: string, NewAddr: string) from EXTERNAL_WORLD,
    change_phone(Customer: string, NewPhone: string) from EXTERNAL_WORLD,
    reg_cust_order(Customer: string, Sale_inscr: integer)
        from CONCRET_PRODUCTION.Sales_Department },
rules = {
    r1: msg(add_role) ⇒ state(active),
    r2: state(active), msg(suspend_role) ⇒ state(suspended),
    r3: state(suspended), msg(resume_role) ⇒ state(active),
    r4: state(active), msg(reg_customer (C, N,A,P)) ⇒ state(active);
        value(Customer_number) = C,
    r5: state(active), msg(change_name(C,N)) ⇒ state(active);

```



```

        value(Customer_number) = C,
r6: state(active), msg(change_addr(C,A)) ⇒ state(active);
        value(Customer_number) = C,
r7: state(active), msg(change_phone(C,P)) ⇒ state(active);
        value(Customer_number) = C,
r8: state(active), msg(reg_cust_order(C, S)) ⇒ state(active);
        value(Customer_number) = C
>)

Resource class (
    SALE,
    .....
>)

Resource class (
    CONSTRUCTION,
    .....
>)

```

6. CONCLUSION

Our objective is to integrate the different modeling methodologies in a complete framework. This multi-modeling methodology is of great interest in industrial systems due the multi-disciplinary nature of this environment. Each modeling tool is best adapted to the representation of some aspect of the complex environment. Table 2 compares the expressiveness of the models used in this paper. The DFD model has the more abstract representation of process and flows defined only by names. No information about synchronization or the control role they play are present. Data is represented as repositories or as data flows. In the E-R model the data structures are well defined. In the industrial view the processes and control roles are represented by relationships among physical components. The same relationships partially represent the physical and control flows. All these features are present in the TF-ORM model.

Table 2: Modeling concepts

modeling tools concepts	DFD*	E-R	TF-ORM
Process	↗	↗	↗
Dataflow	↗	↗	↗
Physical flow	↗	±	↗
Control role		↗	↗
Control flow	±	±	↗
Data structure	±	↗	↗
State		±	↗

The ongoing research will produce a complete methodology to derive an OO enterprise schema at which the TF-ORM schema is the integration model. Some case studies are been conducted by graduate students and some dissertations are formalizing the multi-modeling of industrial systems. Our claim is that this approach facilitates the complete modeling by the use of complimentary modeling tools, integrated through a powerful O-O temporal model, to represent partial system's characteristics. At the project completion the modeling methodologies will be integrated through a database, Figure 1. The mapping of the TF-ORM schema onto an O-O or onto a relational DBMS is described in [Oliveira 95]. At this final implementation stage any modification at each of the models will reflect in the other models. This characteristic allows the

incremental model development facilitating the progressive understanding of the complex system.

7. REFERENCES

- [Batini 92] BATINI, C.; CERI, S.; NAVATHE, S.B. *Conceptual Database Design*, The Benjamin/Cummings Publishing Company, 1992, 470 p.
- [Bravoco 85] BRAVOCO, R.R.; YADAV, S.B. A Methodology to model the dynamic structure of an organization. *Information Systems*, v.10, n.3, p.299-317, 1985.
- [Chen 76] CHEN, P.P. The Entity-Relationship Model: towards a unified view of data. *ACM Transactions on Database Systems*, v.1, n.1, p.9-37, March 1976.
- [Coad 91] COAD, P.; JOURDON, E. *Object-Oriented Analysis*. Englewood Cliffs, Prentice Hall, 1991. 233p.
- [De Antonellis 90] De ANTONELLIS, V.; ZONTA, B. A Disciplined approach to office analysis. *IEEE Transactions on Software Engineering*, v.16, n.8, p.822-28, Aug. 1990.
- [Edelweiss 93] EDELWEISS, N.; OLIVEIRA, J.P.M. de; PERNICI, B. An Object-Oriented Temporal Model. *Proceedings of the 5th International Conference on Advanced Information Systems Engineering - CAISE '93*, June 8-11, 1993, Paris, France, p.397-415. (Lecture Notes in Computer Science, 685).
- [Henderson-Sellers 90] HENDERSON-SELLERS, B. Three methodological frameworks for object-oriented systems development. *Proceedings of the 3rd Int. Conference Technology of Object-oriented Languages and Systems*, Sydney - TOOLS, 1990. p.118-31.
- [Lutherer 94] LUTHERER, E. et al. Modeling with CIMOSA: a case study, *Proceedings of the IFIP WG 5.7 Working Conference on Evaluation of Production Management Methods*, C. Walter, F.J. Kliemann e J. Palazzo M. de Oliveira eds., Gramado, March 21-24, 1994, p. 233-241. Elsevier Science Publishers.
- [Oliveira 95] OLIVEIRA, J. PALAZZO M. de et al. On the Implementation of an Object-Oriented Temporal Model using Object-Oriented and Relational DBMSs, *Proceedings of the DEXA '95 Conference*, London, Set 4-8, 1995.
- [Reisig 85] REISIG, W. *Petri Nets : an Introduction*. Springer-Verlag, Berlin, 1985. 161p.
- [Ross 77] ROSS, E. T. Structured Analysis for Requirements Definition. *IEEE Transactions on Software Engineering*, 3, 6-15.
- [Santos 86] SANTOS, S.A.; OLIVEIRA, J. PALAZZO M. de; TAZZA, M. Projeto de Sistemas Distribuidos: uma solução integrada (in portuguese). *Proceedings of the XIX Congresso Nacional de Informática*, (Brazilian National Dataprocessing Conference) p. 245-254, 18-24 Aug., Rio de Janeiro.
- [Vernadat 93] VERNADAT, F. Tutorial: "Enterprise Modeling and CIMOSA" in *IEPM '93 International Conference on Industrial Engineering and Production Management*, Mons, Belgium, Jun. 2-4, 1993.
- [Volainen 92] VOLAINEN, T. S. ; SALMINEN, V. Rough Modeling Methodology for Plant Projects; *Proceedings of the IFAC Workshop on CIM in Process and Manufacturing Industries*, Nov. 23-25, 1992, Espoo, Finland. Elsevier Science Publishers.

8. BIOGRAPHIES

José Palazzo M. de Oliveira received the B.S. degree in Electric Engineering in 1969, the M.Sc. degree in Computer Science in 1976 both from Federal University of RS - UFRGS, Brazil and the Dr. Ing. from INPG, Grenoble, France in 1984. He is researcher and Ph.D. advisor at the Graduate Program in Computer Science and at the Graduate Program in Business at UFRGS. His current research interest is Conceptual Information Systems Modeling, Distributed Systems and Temporal Databases.

Nina Edelweiss received the B.S. degree in Electric Engineering in 1969, the M.Sc. degree in Computer Science in 1975 and the Ph.D. in 1995 from Federal University of RS - UFRGS, Brazil. She is researcher at the Graduate Program in Computer Science, UFRGS. Her current research interest is Conceptual Information Systems Modeling and Temporal Databases.