Research Paper ■

Development of a Hospital Information System Using the TAD Method

TALIB DAMIJ, PHD

Abstract **Objective:** To examine the capability of a new object-oriented method called Tabular Application Development (TAD) in developing a hospital information system for a gastroenterology clinic.

Design: TAD has five phases. The first phase identifies the problem to be solved. The second phase defines the business processes and activities involved. The third phase develops the object model. The fourth phase designs the application model. The final phase deals with implementation.

Results: Eight requirements for the system were identified by hospital management; 17 specific tasks were grouped into three activity categories. The process model, the object model, and the application model of the system are described.

Conclusion: The TAD method is capable of developing such an information system without any problem. Furthermore, the method minimizes the time needed to do this in such a way that the process is completely visible to the analyst.

■ **JAMIA.** 1998;5:184–193.

A hospital information system was created using a new object-oriented method called Tabular Application Development (TAD). With this method, the process of application development is easily and completely visible to the analyst. TAD has five phases and introduces a new approach to developing information systems that involves creating several tables. The first phase identifies the problem to be solved using the entity table. The second phase defines the business processes and activities of the organization using the activity and task tables. The third phase develops the object model of the system. The fourth phase designs the application model using the information collected in the entity and the activity tables. The final phase deals with the implementation of the system.

Affiliation of the author: University of Ljubljana, Ljubljana, Slovenia.

Correspondence and reprint requests to: Talib Damij, PhD, Faculty of Economics, University of Ljubljana, Kardeljeva ploscad 17, 1000 Ljubljana, Slovenia. e-mail: ⟨talib.damij@uni-lj.si⟩.

Received for publication 3/21/97; accepted for publication 11/14/97.

Background

The Gastroenterological Internal Clinic is a small and very successful hospital specializing in internal diseases. In addition to Management and a Reception Office, the Clinic has four Care Departments, one Intensive Care Department, and three Diagnostic Departments.

Before the implementation of this system, this hospital had no information system. Almost all activities were done manually. For this reason, the management could not obtain needed information in a timely way. They decided to develop an information system, which provided a good opportunity to determine whether the TAD method was capable of solving such a problem.

Method

First Phase

In the first phase of TAD, the problem to be solved is identified and reduced to understandable terms. The best way to begin this is to interview all users. We first interview the management and then continue with the other users. Usually we create a plan for the interviews in accordance with the hierarchical scheme of the organization. We start the process with the top management, and work down to the lower levels.

The purpose of these interviews is to identify:

- The organization's structure;
- Outputs and analyses that are vital to the management's decision making; and
- Management's decision-support problems.

We use the term entity instead of user. An entity is any source of information that is part of the system or is connected with the system by some interaction. An internal entity is inside the system and takes part in the system's operation. An external entity is not a part of the system, but it has one or more interactions with the system. An entity may be a user or any other source that sends input to the system, participates in some task in the system, or receives output from the system.

Interviews should be organized with the internal users only. Internal users inform us about the behaviors of the external users and other entities.

Identication of the entities and their requirements is achieved by developing a table called the entity table. This table is developed and completed during the interviews with the management at different levels. The entity table is structured as follows: The columns of the table represent the entities. The rows of the table represent the analyses required by the entities.

An asterisk in any square (i, j) in the entity table means that the entity defined in column j requires the analysis defined in row i, where i ranges from 1 to the number of rows and j ranges from 1 to the number of entities.

Second Phase

This phase deals with identifying the tasks, activities, and processes of the system, and has four steps. The first step creates the activity and task tables. The second defines the activities and the processes of the system. The third step tries to optimize the activity table. The fourth step transforms the activity table into a process model.

Task Identification

The aim of the interviews with the management at different levels organized in the previous phase is to complete the entity table and to identify the users of the system.

The task identification step deals with identifying and analyzing the activities of the organization by organizing further interviews with each user of the system.

The purpose of these interviews is to identify every task performed by any user in the framework of the identified entities. These interviews are also useful for completing the entity table with users' requirements and decision-support problems.

The best way to identify the tasks is by developing another table, called the activity table. As we develop the activity table, we simultaneously develop another table, called the task table, which is discussed later.

The activity table is organized as follows: The entities are represented by the columns and the tasks are listed in the rows of the table. Every task occupies one row. A non-empty square (i, j) shows a certain job performed by an entity defined in column j inside the task defined in row i. A job is work performed by a determined entity in the framework of a certain task. A task is a collection of jobs performed by a number of entities.

Developing the activity table is a result of interviews organized with the internal entities defined in the columns. In the rows of the activity table we first register each task performed by any user and then link this task with the entities in the columns, which cooperate in doing the task. In the other words, in the rows of the table we list all tasks one by one in the order in which they occur in the real world. Each of them must be connected with those entities in the columns that perform jobs in the framework of this task.

For every task defined in row i, where i ranges from 1 to the number of tasks, we list the entities in the columns and try to link the current task with each of these entities. If any connection exists between task (i) and entity (j), where j ranges from 1 to the number of entities, then a letter S or T is written in square (i, j).

Letter S in square (i, j) means that entity (j) is a source entity for task (i). This entity (j) performs a determined job in the framework of task (i) (creates, completes, sends, etc.) defined in square (i, j). Letter T in square (i, j) means that entity (j) is a target entity for task (i). This entity (j) accepts or registers an output from other entities.

Any task may have one or more source and target entities. For this reason, the letters S and T, used to indicate a certain task, are also indexed by the sequence number of the job as it occurs in the framework of the treated task.

In addition, we consider only the internal entities, and use the letters P and U to connect the tasks in which a certain internal entity is included. This is to connect the jobs of any internal entity defined in a determined column. Letter P in square (i, j) means that task (i) is a predecessor to some task (tasks) indicated by U in column j. Letter U in square (i, j) means that task (i) is a successor to another task (tasks) indicated by P in column j. Because any task may have one or more predecessors and also one or more successors, the letters P and U are indexed by the index of the predecessor task (see Table 2).

We have described how to define the tasks of the system. Each task consists of one or more jobs and occupies one row of the activity table. Jobs of any task are indicated by non-empty squares in the task's row of the activity table.

We often find ourselves in situations where we need more information about the tasks. We especially need detailed information about jobs to identify all the circumstances in which they are accomplished. The best way to identify jobs, their characteristics, and the circumstances linked with them is by developing a table called the task table. As mentioned above, the task table is developed at the same time as the activity table.

The task table is organized as follows: The jobs are represented by the rows of the table and the characteristics of the jobs are defined in the columns. Thus, each job, defined by a non-empty square in the activity table, occupies one row in the task table. Each job is represented by its code Jij, where the letter J means job, and i and j are indices of row i and column j of the activity table where the job is defined.

In the columns of the task table, we define job characteristics in terms of Description, Time, Condition, and Document. Description is used to write a short description of the particular job. Time is used to denote that entity (j) in the activity table needs a determined time to perform job Jij. Time may become very useful in case we decide to optimize the activity table, as is discussed below. Time is not defined for those jobs that do not take a lot of time. Condition is used to define that performing job Jij requires that one or more preconditions or conditions be fulfilled. Document is used to indicate which documents (inputs and outputs) are connected with job Jij. We may define other job characteristics if necessary.

In addition to the jobs, the task table represents the tasks defined in the activity table. The purpose of this is to show in detail the linkages between the tasks and their jobs. As mentioned above, each job occupies one row in the task table. For this reason, each task occupies one or more rows in the task table (see Table 3).

Developing the activity and task tables is an iterative process. Some of the interviews have to be repeated to arrive at a precise understanding of the user's work. If anything is not understandable, then we have to organize a new interview with the responsible user until everything is clear.

Activity and Process Definition

After finishing the process of interviews with the users, we complete the entity, activity, and task tables. This is achieved by organizing a joint meeting with the main or important users from different entities. In this meeting we represent the entity, activity, and task tables to them. The aim of this presentation is to enable us to correct any mistakes in the tables and to persuade the users to complete the entity, activity, and task tables.

In the next step, the analyst has to rethink and analyze the activity table to define the activities of the system by grouping the tasks into suitable collections. Each of these collections represents a determined activity. An activity is a collection of one or more tasks, which are followed in a determined order. Each activity occupies one or more rows in the activity table. If necessary we may go on grouping the activities into suitable collections; each of these collections is a business process. Each process consists of one or more successive activities and occupies one or more rows of the activity table.

A business process is a collection of activities that takes one or more kinds of inputs and creates an output that is of value to the customer.[1] Creating the activity table leads to the discovery of all the tasks and all the jobs inside each task, all activities defined by grouping the tasks in convenient collections, and all business processes identified by grouping the activities in appropriate collections. Thus, we can say that creating the activity table leads to the discovery of the entire system and its subsystems.

Optimization of the Activity Table

This step may be called reengineering the business processes of the organization. To do this we need the agreement of the top management. In this step we try to analyze and rethink every activity to optimize the time and the resources needed to perform any task and to make every job more effective. This approach enables us to develop an efficient information system, which helps the organization to be more successful. This goal is achieved by removing redundant tasks or jobs, by moving jobs from one entity to another, by shortening the times needed to perform some jobs, or by using other optimization strategies.

Optimization of the activity table is performed using the information collected in the activity and task tables, because these tables enable us to analyze the en tities, tasks, and jobs from various points of view. From these tables we can easily find redundant tasks or jobs and obtain other useful information. The task table enables us to analyze the time needed to perform any job. We can quickly find the time-consuming jobs and warn the management about them. In addition to this, without much additional effort we can use the information collected in the activity table to create event trace diagrams or other diagrams that may increase our understanding of the system.

Optimization of the activity table is an iterative process. Each further iteration may help in creating a better solution. We repeat the process until an optimal solution is found. If a better solution is found, we must present it to the management before using the new activity table.

This very important step enables TAD to be a useful and effective method.

Process Model

This step deals with creating the process model. It may be skipped because the activity table sometimes seems to be more visible than the process model. The activity table and the process model are essential for understanding the behavior and the operations of the system.

To develop the process model we transform the activity table into the model. The whole model is developed in accordance with the activity table. To do this, we list all tasks in the rows of the activity table. For every task (i), where i ranges from 1 to the number of tasks, we list every entity (j), where j ranges from 1 to the number of entities. We transform each non-empty square (i, j) into an elementary process (circle) if entity (j) is an internal entity. This process represents the job performed by entity (j) in the framework of task (i). For this reason, we write in it the indices i and j. Otherwise we transform each non-empty square (i, j) into a source (rectangle) and write the name of entity (j) in it.

After transforming the whole table, we connect the sources and processes drawn horizontally and vertically. Horizontally we link each process or source indicated by S_q in square (i, j) by arrows with those processes and sources indicated by T_q in row i, where q is the sequence number of the jobs in row i. Vertically we link every process indicated by P_i in square (i, j) by arrows with those processes indicated by U_i in column j. Using the described procedure, we transform the whole activity table into a single data flow diagram (DFD). To indicate each activity, we draw a border around the processes of every activity giving the activity-level DFD. To denote every business process, we also draw a border around the activities of each process, giving the process-level DFD. All processes together give a system-level DFD.

Otherwise, we may draw each activity-level DFD separately. This DFD contains all the elementary processes of each activity. In other words, the complete system-level DFD represents the whole activity table. Every process-level subsystem represents a determined business process in the table if such a process is defined. Each activity-level subsystem represents a certain activity in the table.

Third Phase

The third phase has two steps and deals with developing the object model of the system. Before describing these steps, let us give some important definitions:

- An *object* is anything, real or abstract, about which we store data and those operations that manipulate the data.[2]
- An *object* is simply something that makes sense in an application context.[3]
- An *object class* describes a group of objects with similar properties (attributes), common behaviors (operations), common relationships to other objects, and common semantics.[3]

Corresponding to these definitions and to the TAD method, we develop the object model using the results of the first and second phases. In the first step of this phase we identify the object classes, their attributes, and their associations. Identification of object classes is actually a continuous process, which is started in the first phase, continued in the second phase, and completed in the third phase. Information about object classes is obtained from interviews with the users and by analyzing the entity, activity, and task tables.

During the interviews organized in the first and second phases with the management and other users, we try to register everything that could be relevant to the application system mentioned by the interviewers. For example, during interviews with management we stated that "we would like to have an analysis about the patients, the examinations ordered, and the doctors who ordered them." The result of this work is a list of statements, which is an important source of information about the object of the system.

The first step of this phase deals with developing the object model by analyzing the results of the previous phases. We start by analyzing the list of the statements elicited in the previous phases. For this purpose, we

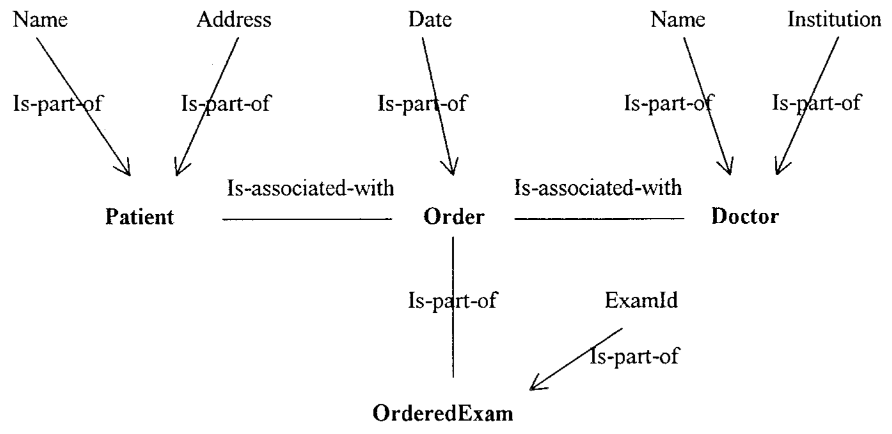

Figure 1 Identification of four classes of objects.

carefully consider each statement and try to find out which object classes, attributes, and associations are connected with it. The result of this analysis is a group of object classes, their attributes, and their associations.

This work is continued by analyzing the entity table, and particularly the outputs and reports defined in the rows of the table. If any new object class, attribute, or association is identified, the existing group of classes is extended by it. Furthermore, we continue the process by analyzing the activity and task tables, particularly the documents defined in the Document column of the task table. Any newly discovered object class, attribute, or association is added to the existing ones. In addition, the analyst may add any other object class if it is useful to the application.

Such an analysis is carried out using three types of relationships that can exist between classes and attributes. These relationships are Is-part-of, Is-associated-with, and Isa structures. A class of objects can be constructed, described, and qualified by the objects' attributes and by other classes (the Is-part-of structure). A class, then, is constructed from components consisting of attributes and other classes. This is referred to as the class aggregation abstraction.[4] For example, Name Is-part-of Patient and Address Is-part-of Patient are used to describe the aggregation of attributes. Furthermore, OrderedExam Is-part-of Order is an example used to describe the aggregation of classes.

A class can interact with other classes, and a class can affect the attribute values of other classes through associations or collaborations (the Is-associated-with structure). This is referred to as the class association abstraction.[4] For example, Order Is-associated-with Doctor is used to describe the relationship between the classes Order and Doctor. Sometimes it is useful to model an association as a class.[3] Thus classes are associated directly or via an association class. Figure 1 shows that such an analysis leads to the identification of four classes, Patient, Order, OrderedExam, and Doctor, and some of their attributes and associations.

From the above discussion we can conclude that in the first step we try to develop the initial object model by implementing the Is-part-of and Is-associated-with structures. In the second step we try to organize the classes of the object model obtained in the previous step using inheritance (the Isa structure) to arrive at common structures. Inheritance is a critical feature as

Table 1 ■

Entity Table of the Clinic

	Entity			
Requirements	Management	Reception–Office	Care Department	Endoscopy
Analysis				
1. Information about hospital occupation	*		*	
2. Information about patients and diseases	*		*	
3. Information about successful treatments	*		*	
Reports				
4. Information about problematic patients	*		*	
5. Information about doctor's occupation	*		*	*
6. Information about medications used	*		*	
7. Information about hospital plan realization	*			
Decision support				
8. Developing work plan for medical staff	*		*	*

Table 2 ■

Activity Table of the Clinic

Process	Activity	Task	1. Manage-ment	2. Recep-tion-Office	3. Care Depart-ment	4. Endos-copy	5. Labora-tory	6. Phar-macy	7. External Special-ist	8. Insur-ance Com-pany	9. Personal Doctor	10. Patient
	Register	1. Register pa-tient's data		T1, S2 P1	T2 P1							S1
		2. Identify payer		T1 U1, P2								S1
		3. Accept doctor's order		T1, S2 U2	T2 P3						S1	
		4. Open patient's card			S1 U1, U3, P4							
		5. Register anam-nesis			S1 U4, P5							
		6. Order for inter-nal diagnosis			S1 U5, P6	T1 P6						
Patient	Treatment	7. Accept internal findings			T1 U6, P7	S1 U6						
		8. Order medicine			S1 U7, P8			T1 P8				
		9. Accept medicine			T1 U8			S1 U8				
		10. Order for labo-ratory			S1 U6, P10		T1 P10					
		11. Accept labora-Ktory findings			T1 U10, P11		S1 U10					
		12. Order for exter-nal specialist			S1 U6, U11, P12				T1 P12			
		13. Accept special-ist's findings			T1 U12, P13				S1 U12			
		14. Prescribe ther-apy			S1 U6, U11, U13, P14							
	Release	15. Inform about state of health		T1, S2 P15	S1 U14						T2	
		16. Create invoice		S1 U15, P16						T1 P16		
		17. Accept pay-ment		T1 U16						S1 U16		

NOTES: P, predecessor to a task indicated by U. S, source entity for a task. T, target entity for a task. U, successor to task indicated by P.

well as a primary goal of object-oriented development.[4] A class can be related to other classes through subclasses (the Isa structure). Thus, a subclass can inherit attributes from superclass, and it can have its own unique attributes. This is referred to as the class generalization and specialization abstraction.[4] Inheritance can be added in two directions: by generalization of common aspects of existing classes into a superclass (bottom-up) or by refining existing classes into specialized subclasses (top-down).[3] For example, Order is a superclass of classes AcceptOrder, ExterOrder, InterOrder, MedicOrder, and LaborOrder. This is an example of generalization of common aspects into superclass (see Figure 3).

Fourth Phase

This phase deals with designing the system, i.e., developing the application model. This model consists of several parts. Development of the application model is derived from the information collected in the entity and activity tables. This goal is achieved in two steps.

The first step develops the first two parts of the application model in accordance with the information in the entity table. The first part represents the reports

Table 3 ■

Task Table of the Clinic

Task	Job Code	Description	Time	Condition	Document
1. Register patient's data	J12	Reception office registers patient's data			Patient's medical card
	J13	Care department accepts patient's data from reception office			Patient's medical card
2. Identify payer	J22	Reception office identifies the treatment payer		If the payer is unknown do not admit the patient	Patient's medical card
3. Accept doctor's order	J32	Reception office accepts doctor's reception order from patient		If the patient does not have a reception order from the doctor do not admit the patient	Reception order
	J33	Care department gets doctor's reception order from reception office			Reception order

defined in this table. The second part considers the decision-support problems if there is such a requirement in the entity table. To develop these two parts we simply list the rows of the entity table and create the parts corresponding to the content of the rows. To do this we copy the required reports to the first part and the required decision-support problem to the second part. The created parts are especially important for management, because they deal with various analyses that are essential to management's decision making. These analyses provide the managers at different levels with all the necessary information about their plans and objectives and give them a clear picture of any particular problem in the organization. The aim of these analyses is to help management to optimize the operation of the organization.

The second step develops the other parts of the application model in accordance with the information stored in the activity table and considering the internal entities only. Each of these parts presents a particular business process if such a process is defined in the activity table; otherwise, it represents a determined activity. Every activity includes a number of tasks, and each task has one or more jobs. To achieve this, we list the activity table and create a part of the application model for the individual business processes that are defined in the activity table; in the absence of individual processes, we create a part for each activity. For every process, we copy its activities from the table to the application model, and for each activity, we copy all its tasks and jobs from the activity table. Thus, we can conclude that the application model gives a clear picture of the requirements and decision-support problems defined in the entity table and, regarding the business processes, the activities and tasks defined in the activity table. In other words, it represents the contents of the entity and activity tables.

Fifth Phase

The last phase of TAD deals with the implementation of the system developed in the previous phases. To complete this phase, we analyze the object model and the application model. The first step deals with implementing the object model and writing algorithms for accomplishing the operations of every class defined in the object model (see Figure 3). The second step defines the algorithms that implement the content of the application model (see Figure 4). We write an algorithm for every process, activity, task, or job defined in the application model.

Results

First Phase

In accordance with the first phase of the TAD method, we organized two meetings with the management of the clinic. In addition, we interviewed the management of each department. The results of these interviews are shown in Table 1. This table represents the entity table of the hospital. It shows the organization's entitites, vital outputs and analyses, and decision-support problems. Thus, the clinic has four internal entities: Management, Reception Office, Care Department, and Endoscopy. The entity Care Department represents four care departments and in intensive care department, because these departments have almost the same activities. The rows of Table 1 show eight analyses, the number required by the entities in the columns.

Second Phase

Task Identification

To develop the activity table, develop the task table, and complete the entity table, we organized interviews at the department (entity) level. We first interviewed the management of each department and then continued with all users in the framework of the department. The purpose of these interviews was to identify the tasks and jobs performed by the department. The results of these interviews are shown in Tables 2 and 3, which represent the activity and task tables of the clinic. (These tables are only part of the real activity and task tables of the clinic. The real tables are too large to be presented here.)

Table 2 has 17 tasks and ten entities. The first four entities are internal and the last six are external. The first task, "Register patient's data," means the reception office gets personal data from the patient and sends it to the care department. Thus, we write S1 in square (1, 10), T1 in square (1, 2), S2 in square (1, 2) and T2 in square (1, 3). The second task "Identify payer" means that the reception office gets information from the patient about who is paying for the treatment. We write S1 in square (2, 10) and T1 in square (2, 2). All other tasks are defined using the same rule.

Furthermore, the first and third tasks are a predecessor to the fourth task performed by the entity Care Department. Thus, we write P1 in square (1, 3), P3 in square (3, 3), and U1, U3 in square (4, 3). The fourth task is a predecessor to the fifth task. We write P4 in square (4, 3) and U4 in square (5, 3). All tasks are linked in the same way where such connections exist between them.

Table 3 shows only the tasks of the activity "Register." This activity has three tasks. Let us explain the second task "Identify Payer," which contains only one job. The code of this job is J22, which means a job of task 2 performed by entity 2. In the column Description we briefly describe the job. Column Time is empty, because performing this job does not require a lot of time. In the column Condition we define a condition that must be satisfied before the patient is accepted. Finally, in the column Document we link the job with the patient's medical card. Other tasks and jobs are defined using the same concept.

Activity and Process Definition

Corresponding to this step, we grouped the tasks defined in Table 2 into three activities: Register, Treatment, and Release. The first activity has three tasks, the second has nine tasks, and the last has three tasks.

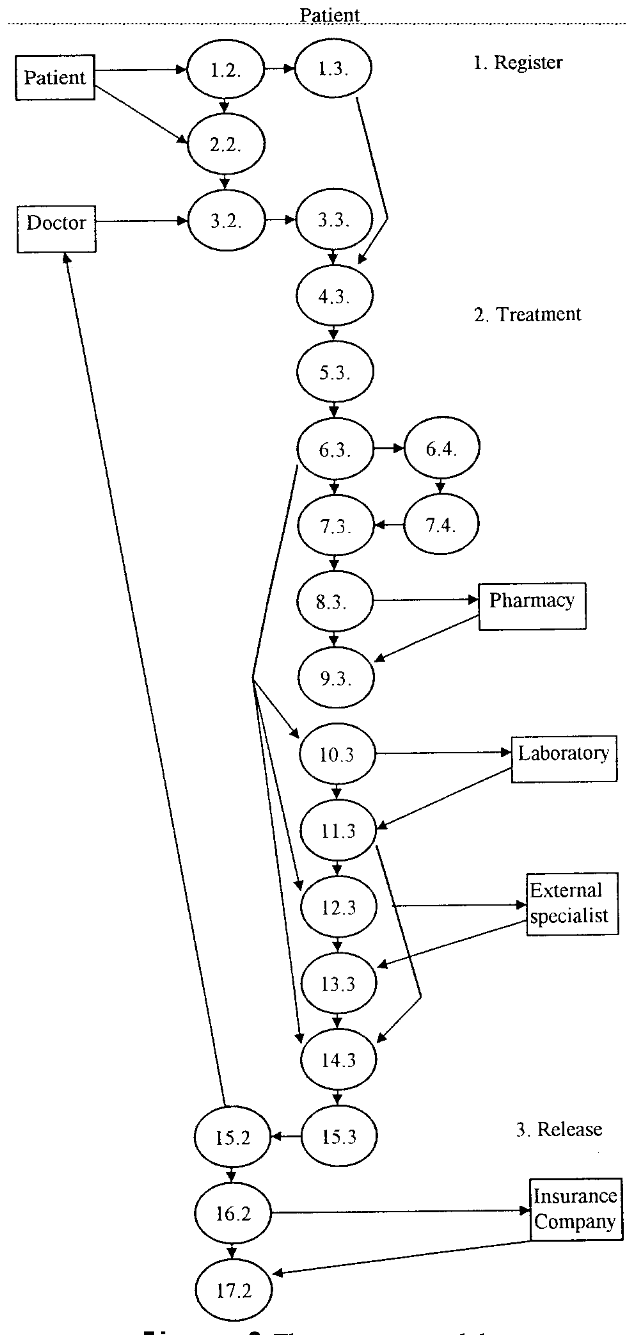

Figure 2 The process model.

Optimization of the Activity Table

The third activity in Table 2 needs to be accelerated. This activity deals with releasing the patient home. In reality the patient has to wait some time to obtain all the needed documents. We hope that implementing this system will solve this problem.

Developing the Process Model

Figure 2 shows the process model of the Clinic.

Third Phase

In accordance with the first step of the third phase we analyzed the list of statements registered during the interviews, the entity table and its outputs, the activity and task table, and the documents collected in the Document column of the task table. The result of this analysis is the initial object model, which contains set of classes, their attributes, and their associations. These classes are: Patient, Card, Department, Diagnosis, Hospital, Doctor, Medicine, Examination, AcceptOrder, ExterOrder, InterOrder, MedicOrder, LaborOrder Treatment, PatientState, PatientDiag, Findings, Order-Medic, Labor-Exam, Institution, and Specialization. Corresponding to the second step, we analyze the classes obtained, trying to find the inheritance between them. The result of this analysis is to create a superclass Order, which is a superclass of the classes AcceptOrder, ExterOrder, InterOrder, MedicOrder, and LaborOrder. Because of the complexity of the real object model of the clinic and space limitations, Figure 3 represents only part of the model.

Fourth Phase

Figure 4 shows the application model of the clinic. The model contains three parts. In the first step of this phase we create the first two parts corresponding to the content of the entity table. These are Reports and Decision Support. In the second step we develop the

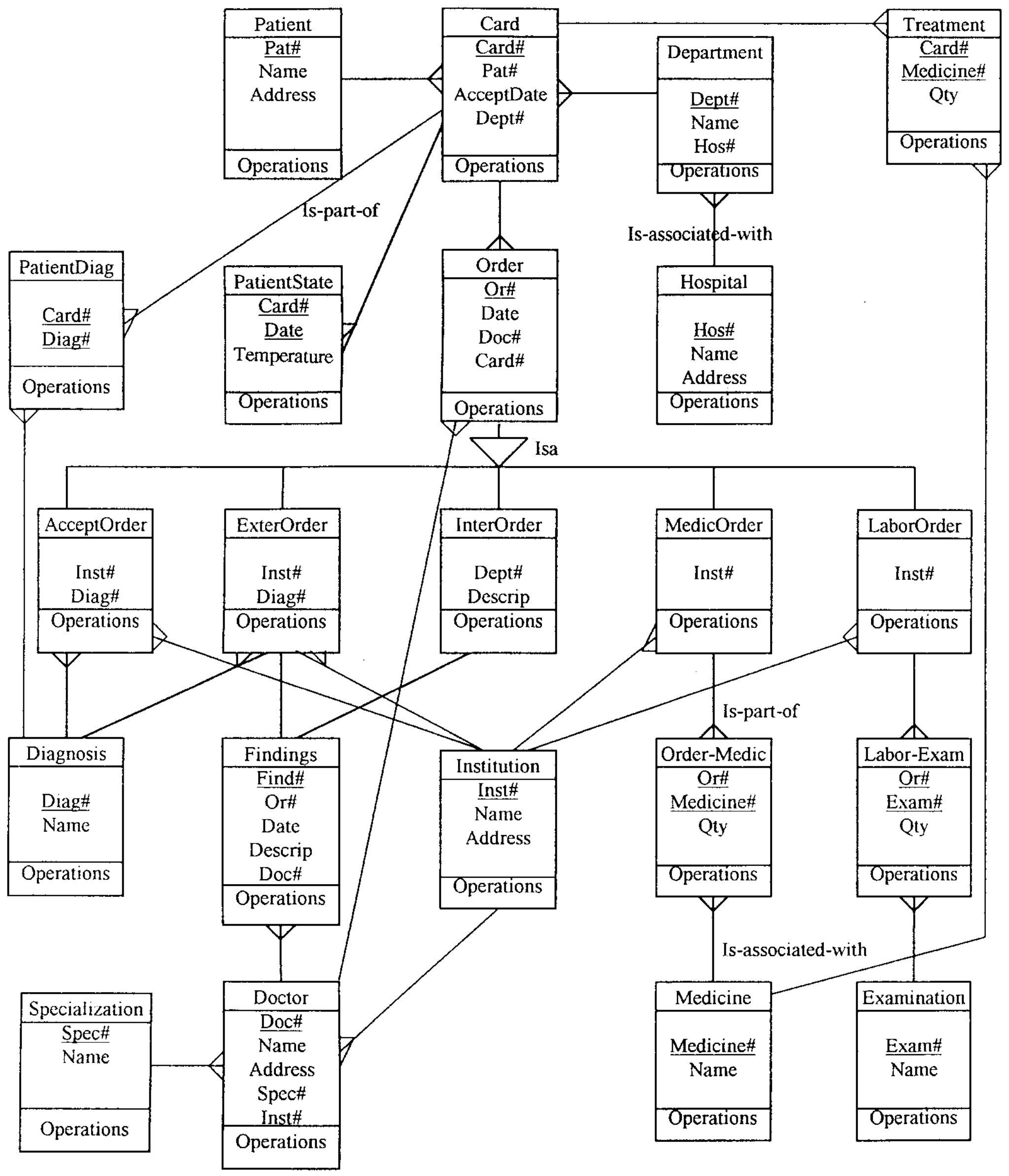

Figure 3 The object model.

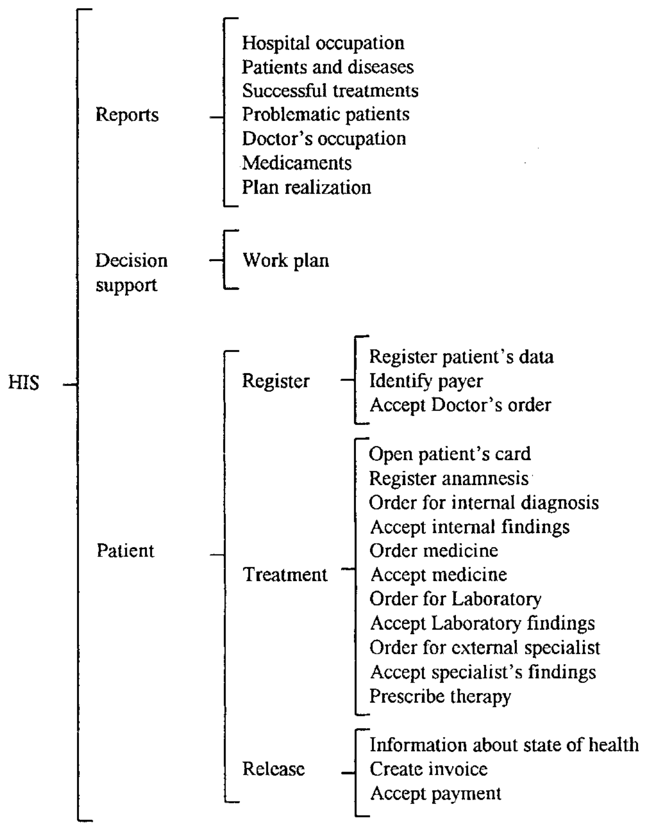

Figure 4 The application model.

last part of the model to represent business-process Patient, which consists of three activities in accordance with the content of the activity table, Register, Treatment, and Release.

Fifth Phase

The results of the final phase are a set of methods that implement the operations of the classes of the object model and a set of algorithms that implement the application model. (Because of space limitations we cannot present these algorithms here.)

Conclusion

The aim of this work was to develop a hospital information system using a new object-oriented method called TAD. We found that the TAD method is capable of developing such a system without any problem. In addition, we found that in spite of the large number of tasks, the process of systems development remained completely visible to the analyst at all times, independent of the analyst and his or her experience. Furthermore, we discovered that the TAD method enables us to minimize the time needed to create such a system. This method has the following characteristics, which are important in making it acceptable and useful in practice:

- Creates the analyses and outputs needed by management at different levels.
- Includes decision-support problems.
- Shows the functioning of the organization in a visible and understandable way using the activity table.
- Includes the reengineering of business processes by optimization of the activity table.
- Defines an algorithm to develop the process model independent of the analyst.
- Prescribes an algorithm to develop the application model.

References ■

1. Hammer M, Champy J. Reengineering the Corporation, A Manifesto for Business Revolution. New York: Harper Business, 1993.
2. Martin J. Principles of Object-Oriented Analysis and Design. Englewood Cliffs, NJ: Prentice-Hall, 1993.
3. Rumbaugh J, et al. Object-Oriented Modeling and Design. Englewood Cliffs, NJ: Prentice-Hall, 1991.
4. Sanders GL. Data Modeling. Danvers, MA: Boyd and Fraser, 1995.