8-15-1997

TAD: An Object-Oriented Method

Talib Damij
University of Ljubljana, talib.damij@uni-lj.si

TAD: An Object-Oriented Method

Talib Damij
University of Ljubljana, Faculty of Economics
Kardeljeva pl. 17, 1000 Ljubljana, Slovenia
talib.damij@uni-lj.si

Abstract

Tabular Application Development (TAD) is a new method which develops the application by creating several tables. TAD has five phases. First phase identifies the entities of the system. The second phase defines the activities of the organization. The third phase identifies the objects of the system. The fourth phase designs the application model. The final phase deals with the implementation. TAD was used successfully to develop a hospital IS.

Keywords: Information systems design methodologies, Object-oriented methods

First Phase

The first phase tries to understand the problem to be solved by organizing interviews with all users. First we interview the management and then continue with other users. The purpose of these interviews is to identify:

- The organization's structure.
- Analyses which are vital to the management.
- Management's decision support problems.

An entity is any source of information which is part of the system or is connected with the system by some interaction.

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

An asterisk indicated in any square(i,j) in the entity table means that the entity defined in column j requires the analysis defined in row i.

Second Phase
The Activities

This step defines the activities of the organization by interviewing each user. The purpose of these interviews is to identify every task performed by any user. A task is a collection of jobs performed by one or more entities.

The best way to identify the tasks is by developing a table called the activity table. The activity table is organized as follow: The entities are represented by the columns and the tasks are listed in the rows of the table.

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.

Developing the activity table is a result of interviews organized with the internal entities. In the rows we first register each task and then link it with the entities in the columns which cooperate in doing it.

For every task defined in row i we list the entities in the columns and try to link this task with each entity. If any connection exists between task(i) and entity(j) then a letter S or T is written in square(i,j).

Letter S means that entity(j) is a source entity for task(i). Letter T means that entity(j) is a target entity for task(i).

Any task may have one or more source and target entities. For this reason letters S and T, used to indicate a certain task, are also indexed by the index of the source entity.

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

Letters S and T are used to define the source and target entity for every task. Letters P and U help to determine a precise place for each task in the activity table.

In addition to that we continue with describing the jobs defined in the activity table by identifying all their characteristics.

A job is work performed by a determined entity in the framework of a certain task.

The best way to identify the characteristics of each job is by developing a table called the job table. This table is organized as follows: The jobs are represented by the rows of the table and the characteristics are defined in the columns.

Each job is represented by its code, which is defined as Jij, where letter J means job, and i and j are indices of row i and column j.

A certain job will be defined in the job table if this job is linked with something which have to be registered.

We found characteristics such as: time, input/output, and condition are very important.

Time is used to denote that entity(j) needs a determine time to perform job Jij.

Input/output is used if job Jij is linked with one or more inputs or/and outputs.

Condition is used if performing job Jij requires testing one or more conditions.

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. So, each activity occupies one or more rows in the activity table.

If necessary we may go on with grouping the activities into suitable collections; each of these collections is a business process.

Activity table optimization

This step deals with optimization of the activity table which may be called reengineering the business processes of the organization. In this step we try to analyze every activity to optimize the time and the

resources needed to perform any task. This fact enables us to develop an efficient information system. This goal will be achieved by removing redundant tasks or jobs, by moving jobs from one entity to another, by shortening the time needed to perform some job or by using other optimization criteria.

Process model

This step transforms the activity table into the process model. To do this we list all tasks in the rows of the activity table. For every task(i) we list every entity(j) in the columns. We transform each non-empty square(i,j) into an elementary process (circle) if entity(j) is an internal entity. Otherwise we transform each non-empty square(i,j) into a source (rectangle).

After transforming the whole table, we connect the sources and processes horizontally and vertically. Horizontally we link each process or source indicated by S_j in square(i,j) by arrows with those processes and sources indicated by T_j 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.

This procedure transforms the activity table into a DFD. We used line border to indicate activity-level, process-level, and a system-level DFD.

Third Phase
The Objects

An object is any thing, real or abstract, about which we store data and those operations that manipulate the data [Martin, 1993]. So, an object is any entity, activity or task about which we store data and operations.

Identifying the objects is derived from the activity table. This process has two steps.

The first step identifies the first group of objects. This group is obtained by analyzing the entities in the columns of the table. If we collect information about any entity then we transform this entity into an object.

The second step identifies the second group of objects. This group of objects is gained by transforming any input or output created or used by any task into an object. We also transform into an object any task or activity if the organization stores information about it.

The Classes

This step identifies the object classes by defining the attributes and the operations of the objects already found. The best way to identify the attributes of classes is by using the normalization technique to normalize the inputs and outputs gained in the previous step.

The result of the normalization is a set of normalized relations. This set of relations consists of three subsets. The first subset of relations defines the attributes of the classes of the objects already identified. The second subset finds the attributes of classes of new objects. The third subset of relations shows the attributes of association classes which represent the associations between the identified object classes. An association class contains the identifiers of the associated classes.

The Linkage table

In this step we first organize the classes using inheritance to share a common structure. After identifying the inheritance between classes we define the connections between the classes.

The best way to do this is by developing a table called the linkage table. This table is structured as follows: all classes are represented in the rows and also in the columns in the same order.

We develop the linkage table by listing the classes defined in the rows. For every class in row i we list all classes defined in the columns. If class(i) contains a key attribute of class(j) in the columns then a letter (K, F or P) will be written in square (i,j).

Letter K (key) indicates that class(i) is a superclass of class(j) and means that both classes have the same key attribute. Letter F means that the data structure of class(i) contains a key attribute of class(j) as a foreign key. Letter P means that the data structure of class(i) contains a key attribute of class(j) as a part key.

For these reasons we can say that letter K indicates an Isa structure, letter F indicates an Is-associated-with structure and letter P indicates an Is-part-of or Isassociated-with structure.

The object model

To develop the object model we use the information collected in the linkage table for transforming this table into the object model.

To do this we list all classes defined in the rows of the linkage table. For every class(i) draw class(i) and all those classes in row i which are connected with this class.

Fourth phase

This phase has two steps. The first step develops the first part of the application model according to the information collected in the entity table. To do this we list the rows of the entity table and create this part in accordance to the content of the rows. In addition to the name of each output or analysis we write the index of each entity in the columns of the entity table which required this output or analysis.

The second step creates the second part of the application model using the information collected in the activity table about the internal entities. This part of the model consists of one or more submodels. Each submodel represents a certain business process or activity. Every activity has a number of tasks, and each task is performed by one or more jobs. In addition to the name of each task or job we write the index of each entity in the columns of the activity table which performed it.

Fifth Phase

The last phase of TAD deals with the implementation of the system developed in the previous phases.

The first step of the implementation deals with writing algorithms for realizing the operations of every class defined in the object model.

The second step defines the algorithms which represent the content of the application model.

These algorithms must implement the users data access corresponding to the information represented in the application model and represented by the index value of each entity which has the right to use an algorithm.

References

Booh, G. Object-Oriented Analysis and Design with Applications. CA: Benjamin-Cummings Publishing Co, 1994.

Damij, T. "Tabular Based Approach for Systems Development", Proceedings of International Conference on Organization and Information Systems, Bled, Slovenia, 1995.

Damij, T. "Tabular Application Development", Proceedings of Information Systems Conference of New Zealand, Palmerston North, New Zealand, 1996.

Jacobson, I. (1995) Object-Oriented Software Engineering, A use Case Driven Approach. Addison-Wesley Publishing Company, Wokingham, England, 1995.

Martin, J. Principles of Object-Oriented Analysis and Design, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.

Rumbaugh, J. et al. Object-Oriented Modeling and Design, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991.

Sanders, G. L. Data Modeling, Boyd & Frase Publishing Company, Danvers,1995.