# AN APPROACH FOR CLASS MODEL DEVELOPMENT

Nadja Damij and Talib Damij
*Faculty of information studies, Novi trg 5, SI-8000 Novo mesto, Slovenia*
*Faculty of economics, University of Ljubljana, Kardeljeva ploščad 17, SI-1000 Ljubljana, Slovenia*

Keywords: Database design, Normalization, Class model.

Abstract: This paper aims to introduce an approach that can be used by both students and practitioners to develop a class model by analysing users' documents. The approach is based on the concept of functional dependence and enables the use of the normalization technique in the field of object-oriented modelling. We believe that the normalization technique is applicable, useful and even essential in this field. The approach consists of six steps that lead the analyst through identifying identity attributes, determining functional dependences, defining associations between identity attributes, integrating the analyses, developing an initial class model, and completing the class model by using inheritance. Two documents of a hospitalization process are used as an example to implement the steps of this approach.

## 1 INTRODUCTION

The purpose of this paper is to present an approach for class model development based on the concept of functional dependence.
The reason for developing such an approach is that the well-known object-oriented techniques and methods do not offer a specific approach to developing the class model. For this reason, the development process remains dependent on the analyst and his/her experience.
In UML (Unified Modelling Language), for example, the class diagram is developed on the basis of the objects listed on the interaction diagrams (sequence or cooperation), which are created using the information contained in the use cases descriptions, particularly in the framework of their event flows.
On the other hand, it is known in relational modelling that the normalization technique is essential for creating a database design.
Normalization is a simple technique, which leads the database administrator to create a data model by identifying different kinds of dependencies existing between the attributes of a certain relation. Such a data model is free of anomalies which could cause problems in inserting, updating, and deleting records in the database.
This paper aims to show that the normalization technique is very useful in the field of object-oriented modelling and could be employed to develop a class model.

The paper is structured into five sections, in addition to the Introduction. In Section 2, the concept of functional dependence is discussed. In Section 3, the normalization technique is presented. In Section 4, the approach of class model development is introduced. In Section 5, some useful remarks and conclusions are presented. Throughout the paper two simplified documents of a hospitalization process are used as an example to implement the approach's steps.

## 2 FUNCTIONAL DEPENDENCE

In the process of information systems development we usually collect many users' documents. These documents contain a lot of information about the objects of the system discussed, their attributes and relationships. For this reason, such documents should be analysed carefully to develop the class model of the system. To achieve this, we use the concept of functional dependence.
Functional dependence is an excellent way to analyse in detail the relationships existing between the attributes of a certain user's document.
The following definition clarifies the use of functional dependence to identify relations existing between attributes of a determined relation: "An



Damij N. and Damij T. (2010).
AN APPROACH FOR CLASS MODEL DEVELOPMENT.
In *Proceedings of the Third International Conference on Health Informatics*, pages 414-421
DOI: 10.5220/0002717404140421

attribute B of a relation is functionally dependent on the attribute A of it if at every instant of time each A-value is associated with no more than one B-value" (Vetter, 1981).
From the above definition, we may conclude that attribute A is an identity attribute and B is a non-identity attribute.
This definition may be extended to consider the functional dependence existing between attributes of a certain user's document as follow: "An attribute B of a certain document is functionally dependent on attribute A (attributes $A_1,\ldots,A_n$) if every value of attribute A (attributes $A_1,\ldots,A_n$) is associated with one and only one value of attribute B" [4].
The notation used to indicate that attribute B is functionally dependent on attribute A is:

$$A \longrightarrow B$$

For example, PatientName is functionally dependent on Patient#, because each value of attribute Patient# determines or identifies one value of attribute PatientName. Furthermore, attribute Dose is functionally dependent on both attributes Order# and Medication#, because a value of the attribute Dose is not determined only by an Order# value, neither by a Medication# value. but a value of attribute Dose is determined by values of both attributes Order# and Medication# as a pair. These functional dependencies could be written:

$$\text{Patient\#} \longrightarrow \text{PatientName}$$
$$\text{Order\#,Medication\#} \longrightarrow \text{Dose}$$

# 3 NORMALIZATION

Edgar F. Codd, the inventor of normalization, had this to say about it: "We all have trouble organizing even our personal information. Businesses have those problems in spades. It seemed to me essential that some discipline be introduced into database design. I called it normalization because then-President Nixon was talking a lot about normalizing relations with China. I figured that if he could normalize relations, so could I" (Rapaport, 1993).
Normalization is used to identify, analyse and organize information contained in users' documents. It is a bottom-up technique that starts with analysing attributes and linking them with their entities (relations). Meanwhile, ER diagrams represent a top-down approach that first identifies entities and then defines their attributes.
Relational normalization requires that in order to avoid anomalies related to inserting, updating and deleting database records, the relations within a database design must be at least in the third normal form. In (Vetter, 1981) we find the following definitions of the normal forms:
A relation is in the first normal form if presented in table form contains at each row-column-intersection precisely one value, never a set of values.
A relation is in the second normal form if it is in the first normal form and every non-identity attribute is functionally dependent on complete identity attribute.
A relation is in the third normal form if it in the second normal form and every non-identity attribute is not transitively dependent on any other identity attribute.
From these definitions, we may conclude:

- There is no repeating groups, which means that each row-column-intersection contains only one value, (first normal form);
- There is no functional dependency on a part of an identity attribute (second normal form); and
- There is no transitive dependency on any other identity attribute (third normal form).

We think that the normalization technique is necessary, applicable and useful in the field of object-oriented modelling. Sanders mentions the term object normalization to indicate the use of the normalization concept in the field of object- oriented modelling (Sanders, 1995).
Each object in a class has a unique identity attribute, which represents it and distinguishes it from other objects of the class. Therefore, we think that using the functional dependence concept is essential in analysing user documents in order to develop a class model.

# 4 NORMALIZATION

The purpose of this work is to introduce an approach which leads the analyst through simple steps toward successful creation of a class model.
The fundamental idea of this approach is that using the concept of functional dependence to analyse the relationships which exist between the attributes of users' documents can lead us to successfully develop a class model of the system discussed.
This approach consists of six steps as shown in Figure 1.

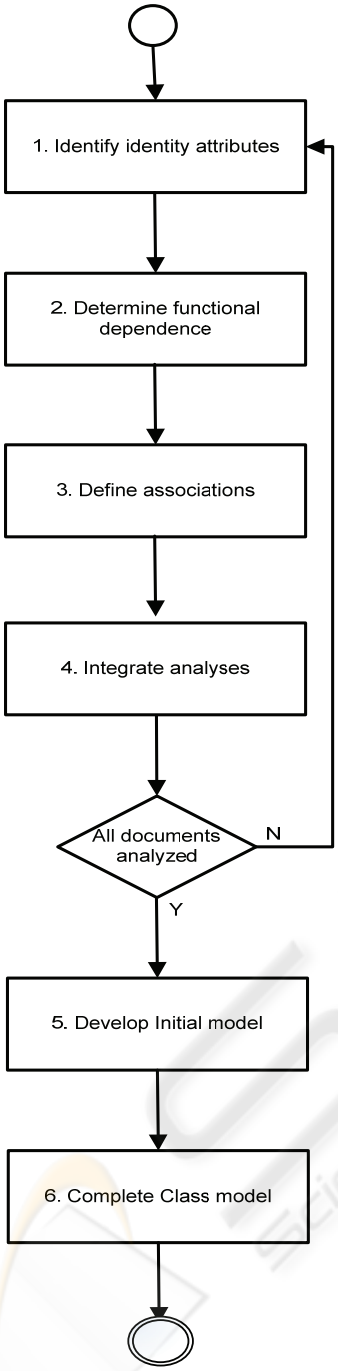


Figure 1: The Approach.

As already mentioned, two simplified documents of the patient hospitalization process are used to demonstrate application of the approach. These documents are a temperature form (TEMSHEET) and a doctor form (DOCTOR), as shown in Figure 2.

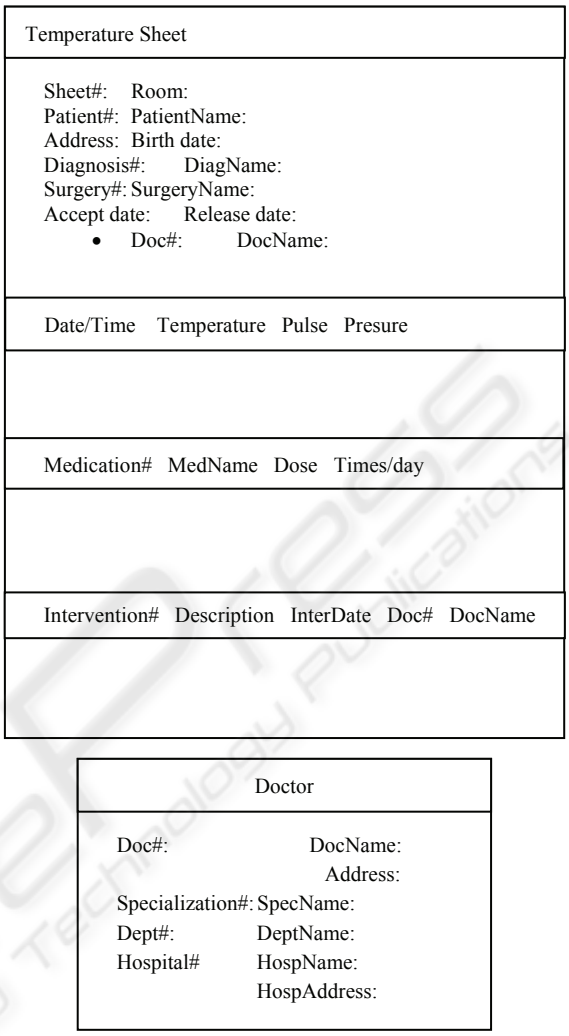


Figure 2: Documents TEMPSHEET and DOCTOR.

## 4.1 Identify Identity Atributes

An object is anything, real or abstract, about which we store data and those operations that manipulate the data (Martin et al., 1992).
An object class describes a group of objects with similar properties (attributes), common behaviour (operations), common relationships to other objects, and common semantics (Rumbaugh et al., 1999).
Each object has an identity which represents it. Object identity or identity attribute is the property of an object that uniquely distinguishes it from other objects.
An identity attribute is a minimal set of attributes that uniquely identifies and represents an object instance (Rumbaugh et al., 1999).
The first step deals with identifying an object is identity attributes contained in users' documents.

This is achieved by analysing the collected documents one by one. Such an analysis enables us to identify a number of identity attributes which belong to a set of object classes.
In our experience, it is better to start with a complex document, which later plays an important role in integrating other analyses (see the fourth step).
*Hospitalization:* Let us start with the document TEMPSHEET. Carrying out the first step on this document leads us to identify the following identity attributes: TempSheet#, Patient#, Diagnosis#, Doc#, Surgery#, Medication#, and Intervention#, which belong to the following classes: TempSheet, Patient, Diagnosis, Doctor, Surgery, Medication, and Intervention.

## 4.2 Identify Functional Dependence

The second step identifies the functional dependencies existing between identity attributes and non-identity attributes in the framework of each of the collected documents.
- for each identity attributes (say A, B) identified in the framework of a certain document (say DOCUMENT1)
  - for each of the document's non-identity attributes (say X, Y, Z)
    - if each value of the chosen identity attribute is associated with one and only one value of the non-identity attributes then the non-identity attribute is functionally dependent on the chosen identity attribute.

After completing the analysis of DOCUMENT1, the process goes on by analysing another document until all the documents are analysed.
*Hospitalization:* To implement this, we start by analysing the relationships existing between the identity and non-identity attributes of the document TEMPSHEET.
Figure 3 shows that the attributes RoomNo, AcceptDate and ReleaseDate are functionally dependent on attribute TempSheet#, because each TempSheet# value determines one value of RoomNo, AcceptDate and ReleaseDate.
Attributes PatientName, Address and BirthDate are functionally dependent on attribute Patient#, because every Patient# value identifies one value of each of the attributes Patient, Address and Birthdate.
Attributes DiagName, SurgeryName and DocName, MedName and Description are functionally dependent on identity attributes Diagnosis#, Surgery#, Doc#, Medication# and Intervention# sequentially.

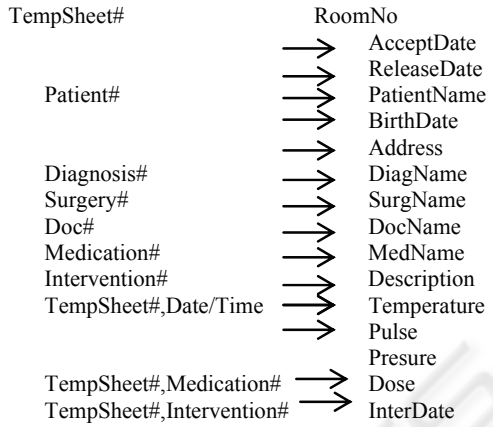


Figure 3: Functional dependencies of Document TEMPSHEET.

In addition to this, we find that attributes Doc# and DocName occur again in the lower part of the document and their values are repeated in the rows of this part. The functional dependence between these two attributes has already been defined and there is no need to define it again. The presence of the identity attribute Doc# in these rows means that an association has to be defined to link these rows with Doc#, as will be shown in step 3.
Furthermore, attributes Temperature, Pulse and Presure are functionally dependent only on a combination of the attributes TempSheet# and Date/Time, because every value of the combination of these attributes determines only one value of each of the attributes Temperature, Pulse and Pressure.
For the same reason, attributes Dose and Times/Day are functionally dependent on a combination of the attributes TempSheet# and Medication#. Finally, we find that attribute InterDate is functionally dependent on a combination of attributes TempSheet# and Intervention#.

## 4.3 Define Associations

The third step of the approach deals with defining the associations existing between the identified identity attributes of the document analysed.
In general we may define an association as follows: An association is a relationship that exists between objects of two classes. An object is represented by its identity attribute. Therefore, we may say that an association is a relationship between identity attributes.
An association could be one-to-one, one-to-many or many-to-many. A one-to-one association exists between identity attributes of objects of two classes, where every value of one identity attribute is related to only one value of the other identity attribute. A

one-to-many association exists between identity attributes of objects of two classes, where every value of one identity attribute is related to none, one or more values of the other identity attribute. A many-to-many association exists between identity attributes of objects of two classes, where every value of one identity attribute is related to none, one or more values of the other identity attribute.

To introduce different type of association, we use general notations.

*Hospitalization:* To implement the third step concerning the document TEMPSHEET, we continue with the analysis given in Figure 3 to define the associations existing between the identity attributes.

Figure 4 shows that the identity attributes Patient# and TempSheet# are associated by a one-to-many association, because each TempSheet# value determines only one value of attribute Patient# and every Patient# value could be related to one or more TempSheet# values.

Furthermore, we also find that attribute TempSheet# is associated by a many-to-one association to the attributes Diagnosis#, Surgery# and Doc#, because a certain diagnosis may be written on different temperature sheets, whereas only one diagnosis is defined on a temperature sheet. A similar explanation is valid concerning the other two associations.

We also find that TempSheet# is connected to attributes TempSheet#,Intervention#, TempSheet#,Medication# and TempSheet#,Date/Time by one-to-many associations. TempSheet#,Intervention# and TempSheet#,Medication# are linked to Intervention# and Medication# by many-to-one associations sequentially. Finally, we define a one-to-many association between Doc# and TempSheet#, Intervention# to link doctors with intervention rows.

The result of analsying the first document using the first three steps is an analysis called the master analysis (Figure 4). After that we continue analysing other documents. For each following document, we first repeat the implementation of the first three steps and secondly continue with the fourth step to integrate the document analysed into the master analysis.

Analysing the document DOCTOR identifies in the first step the following identity attributes:

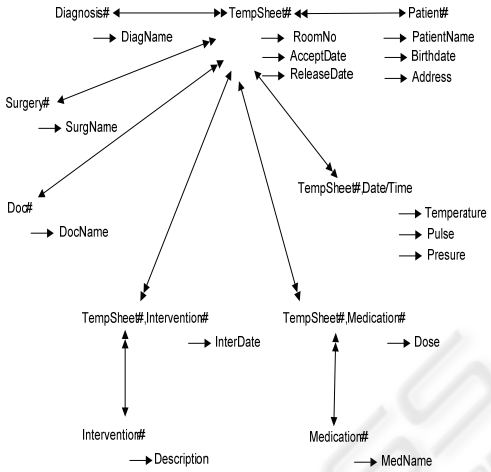


Figure 4: Associations between identity attributes of document TEMPSHEET.

Doc#, Specialization#, Hospital#, and Dept#, which belong to the classes Doctor, Specialization, Hospital, and Department.

The result of using the second step is shown in Figure 5, which represents the functional dependencies existing between identity and non-identity attributes of this document.

| | |
|---|---|
| Doc# | → DocName |
| | → Address |
| Specialization# | → SpecName |
| Hospital# | → HospName |
| | → HospAddress |
| Dept# | → DeptName |

Figure 5: Funcional dependencies of Document DOCTOR.

The third step is implemented by the analysis shown in Figure 6, which defines associations existing between the identity attributes of the mentioned document.

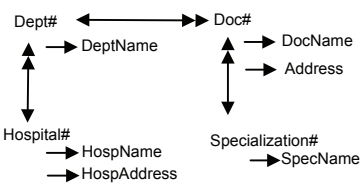


Figure 6: Associations between identity attributes of document DOCTOR.

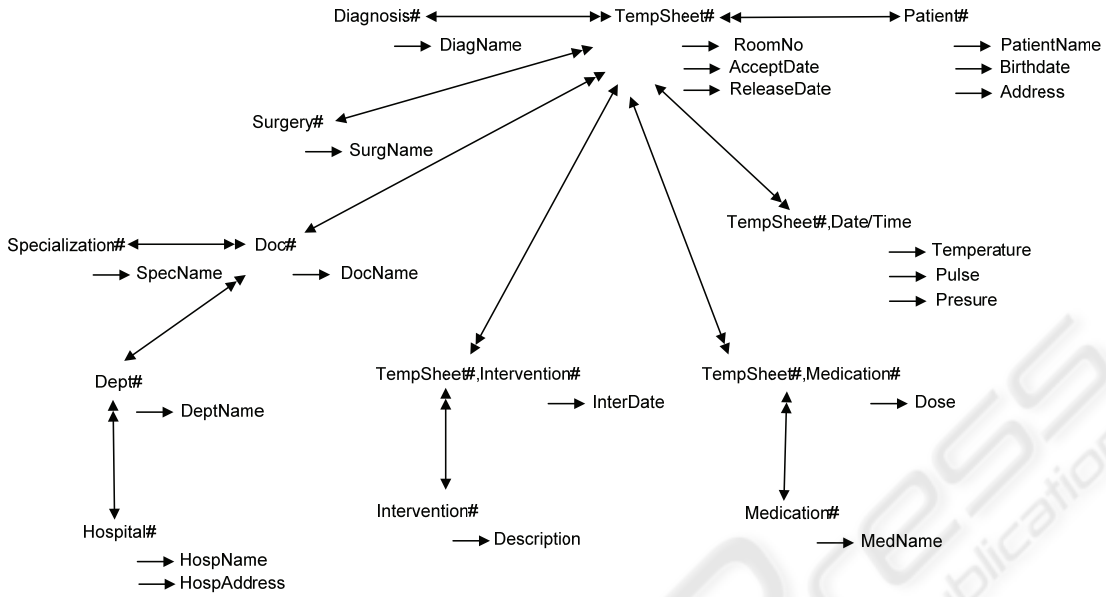


Figure 7: Integrated analyses.

## 4.4 Integrate Analyses

The fourth step starts by integrating analyses from the second analysis into the master analysis. Each of these analyses is a result of implementing the first three steps on each document from the second to the last document.

This step consists of integrating these analyses and grouping them into one analysis, which represents all the objects of the system, their attributes and associations.

This step has the following sub-steps:

Integrating the analysis (from the second to the last one) into the framework of the master analysis is done using the following rules:

a) for each identity attribute existing in both analyses (the master and the integrating analyses)
   - connect every non-identity attribute that is defined only in the integrating analysis to the master identity attribute
b) for each identity attribute that exists only on the integrating analysis
   - define this identity attribute and all non-identity attributes connected with it on the master analysis
c) define associations that connect the newly defined identity attribute to other identity attributes defined in the master analysis.

*Hospitalization:* Figure 7 shows the integration of the analysis obtained from the document DOCTOR (Figure 6) into the master analysis.

By using rule b, the master analysis was extended by new three identity attributes, Specialization#, Dept# and Hospital#, and their non-identity attributes.

To implement rule c, we define two many-to-one associations which were defined to connect Doc# to Specialization# and Doc# to Dept#. Furthermore, a many-to-one association was defined between Dept# and Hopspital#.

## 4.5 Develop Initial Model

The fifth step transforms the analysis obtained from the fourth step, as a result of the process of integration all analyses into the framework of the master analysis, into the class model. This is done using the following rules:

- for each identity attribute presented on the master analysis
  - define a class by drawing a rectangle and writing the class name in it
  - write the new class's identity attribute(s) and its non-identity attributes corresponding to the master analysis
  - connect the newly defined class with other classes in accordance with the associations presented on the master analysis, which connect its identity attribute(s) to other identity attributes
  - identify operations of the newly defined class by studying the behaviour of its objects corresponding to the descriptions of

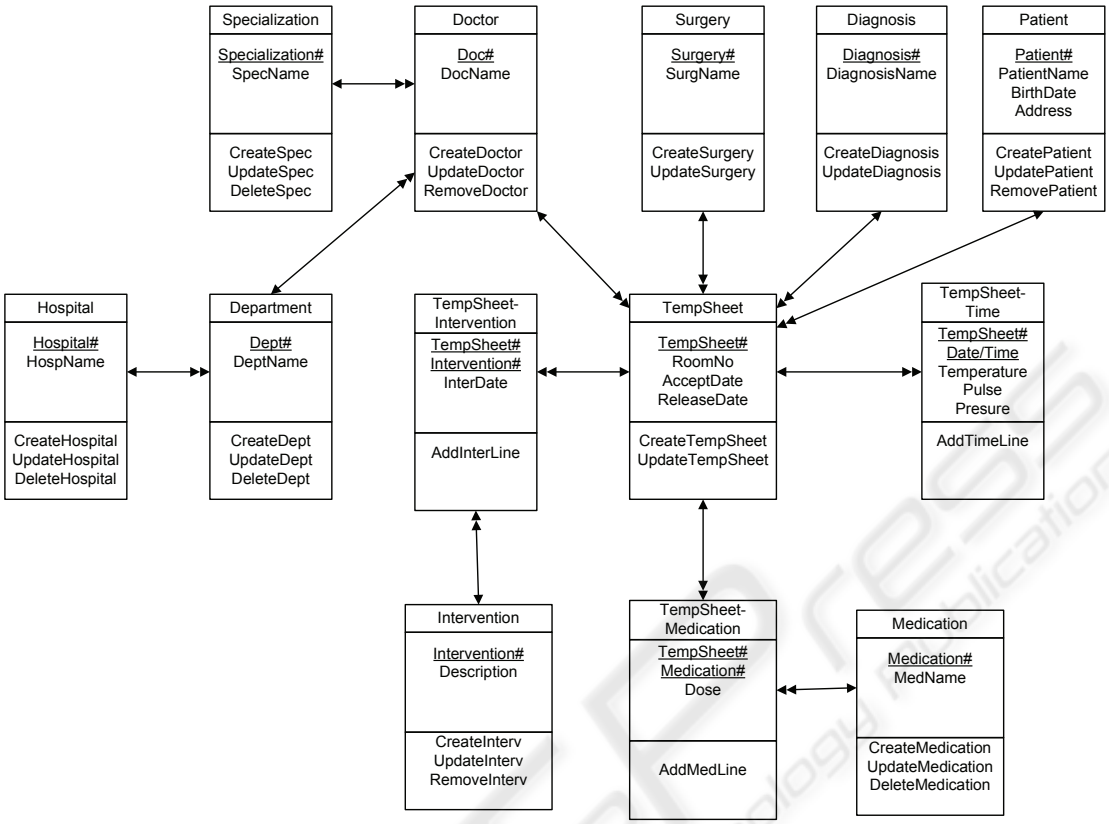


Figure 8: Class model.

the documents and add them to the class definition.

*Hospitalization:* Figure 8 shows the result of transformation of the master analysis into the class model, which is called the initial model of the system

### 4.6 Complete Class Model

In the last step, we complete the initial model and develop the final class model.

Transforming the initial into the final class model is done by:

- studying the possibility of defining the inheritance between classes,
- adding new classes from the analyst's knowledge, if needed.

*Hospitalization:* Analysing the class model shown in Figure 8, we find that inheritance does not exist between the classes of the model.

## 5 CONCLUSIONS

The aim of this paper was to introduce an approach which is capable of leading the analyst to overcome the complex problem of class model development.

The reason for developing such an approach to solve this important problem is that often we find that analysts create the class model on the basis of their experience alone, which may lead them to overlook some important facts.

We add the following useful remarks and conclusions. Firstly, users' documents are very rich in information about objects, their attributes, associations, and operations. This fact is very important and could be used with great success to develop an initial class model that represents the real world of the system discussed.

Secondly, the approach introduced is effective, easy to use by students and practitioners, and capable of producing a class model that satisfies the terms of the third normal form.

Thirdly, the concept of functional dependence is essential for analysing the associations existing between the documents' attributes. Implementing the functional dependence enables the modeller to remove repeating groups, and partial and transitive dependences.

Fourthly, the normalization technique is very useful

and easily applicable in the field of object-oriented modelling. Implementing normalization in this field could help a great deal in establishing order in the process of class model development.

## REFERENCES

M. H. Rapaport, A Fireside Chat, DBMS, Vol. 6, No. 13, pp. 54-60, 1993.

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

M. Vetter, R.N. Maddison, Database Design Methodology, Prentice Hall, New Jersey, 1981.

T. Damij, Tabular Application Development for Information Systems. Springer-Verlag New York, Inc, 2000.

J. Martin, J.J. Odell, Object-Oriented Analysis and Design, Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1992.

J. Rumbaugh, I. Jacobson, G. Booch: The Unified Software Development Process, Addison-Wisley, 1999.