Generic Data Model Patterns using Fully Communication Oriented Information Modelling (FCO-IM)

Fazat Nur Azizah[1], Guido P. Bakema[2], Benhard Sitohang[1], Oerip S. Santoso[1]

[1]*School of Electrical Engineering and Informatics, Bandung Institute of Technology*
Jln. Ganesha no. 10, Bandung, Indonesia
fazat@stei.itb.ac.id
benhard@stei.itb.ac.id
oerip@stei.itb.ac.id

[2]*Faculty of Engineering, Institute of Information Technology, Media and Communication,*
HAN University of Applied Sciences
Ruitenberglaan 26, 6802 CE, Arnhem, The Netherlands
[2]guido.bakema@han.nl

Abstract: In the specific area of data Modelling, data model patterns have been introduced and used to help creating high-quality conceptual data models. Generic data model patterns, in particular, are expected to be more useful in helping to solve more data Modelling problems in comparison to domain-specific patterns. A collection of generic patterns using Fully Communication Oriented Information Modelling (FCO-IM) as the Modelling approach is described in this paper. They can generally be divided into two categories: (1) Patterns that are based on the identification of an object, consist of 6 patterns; (2) Patterns that are based on the relation between two objects, consist of 7 patterns. The generic patterns are needed to be documented in full details and then to be shared in wider audience to have comments and improvements.

Keywords: pattern, data model pattern, FCO-IM

1. Introduction

Patterns have been used in many areas of software engineering to help design processes in creating high-quality solutions in shorter amount of time. Design patterns (see [14]), for instance, are used to help designers creating object-oriented programs. In the specific area of data Modelling, data model patterns (see [15], [16]) have been introduced and used to help creating high-quality conceptual data models for an enterprise.

The use of patterns to guide design processes is originally an idea of Christopher Alexander, a physical architect known for his writings on patterns in urban planning and building architecture, who defines a pattern as "a three-part rule which expresses a relation between a certain *context*, a *problem*, and a *solution*" [1]. Based on this definition, a pattern can be defined as an instruction or a description of a solution to a recurring problem within its goals and constraints which takes place in a certain context [2], [6]. In the case of data model patterns, the problems are specific problems in data Modelling with data models as the solutions.

Most of current works on data model patterns are domain-specific patterns i.e. patterns that are usable for particular area of application, mainly enterprise domain (see [4], [12], [13], [15], [16], [19]). These works use Entity-Relationship Modelling (ERM) [8], [18] or Object-Oriented Modelling (OOM) [10] as the Modelling approach.

Since the patterns are domain-bounded, their applications are limited to the specified area. Even in the respective domain, troubles may be encountered when only small portions of the patterns can be used or too many changes should be carried out. A lot of data Modelling cases do not fall neatly to any parts of the patterns. It would be more useful if data model patterns are defined in more generic ways, so that they can be used to solve more problems. This leads to several works on generic or domain-independent patterns (see [11], [17]). Nevertheless, these works still use terms and constructs that are closely related to the domain-specific patterns.

The aim of our research is to investigate the concepts of data model patterns using Fully Communication Oriented Information Modelling (FCO-IM) as the conceptual data Modelling method. FCO-IM is a fact-oriented data Modelling (FOM) method which views a universe of discourse (UoD) as a collection of facts [5], [7]. The use of fact-oriented way of thinking, especially in FCO-IM, is expected to bring new insights in the discussions of data model patterns [6].

The goal of this paper is to describe a collection of generic data model patterns with FCO-IM as the Modelling approach. The problems defined for the patterns come from common structures of data that may be encountered by data modelers during Modelling processes. This paper presents how these problems are handled in FCO-IM. For specific terms used in FCO-IM please refer to [5] or [7].

2. The data modelling problems

From time to time, data modelers encounter structures of data that come repeatedly from one case of data Modelling to another. No matter what the semantics that are defined above the structures, they are always solved in particular fashions. Experienced data modelers usually can recognize the structures immediately when they see them and they have developed a tacit knowledge in their heads on how to handle the structures in data models.

Accordingly, in our view, generic data model patterns should be derived from the definition of such structures. It should not use specific terms/vocabularies that may lead to specific interpretations of the structures. Using FCO-IM as the Modelling approach, we are going to describe the problems using FCO-IM terms and constructs, including Object Type (OT), Fact Type (FT), Label Type (LT), population, and Uniqueness Constraint (UC) [5], [7].

The first group of problems that we identified are problems related to the structures of a single object. They are specifically related to how an object is identified. Identification of an object is important in data Modelling because every object must be recognized uniquely in order to maintain the single definition of data. The problems are:

- *Single Identification*: How to model an object that is identified uniquely by only one particular way of identification.
- *Generalized Identification:* How to model two or more objects that are required to be recorded for the same type of facts, and thus, those objects should be treated as a single object.
- *Synonymy:* How to model an object that is identified using two or more different ways of identification.
- *Homonymy:* How to model one name (or other way of identification) that can be used to define two or more objects.
- *Recursive Identification*: How to model an object which is identified partly by itself.
- *Set Identification*: How to model the identification of a set of objects of the same type.

The second group of problems that we are identified are defined as the relations of two objects in a UoD. The problems are:

- *Attribute*: How to model the attribute (property) of an object. The attribute is most probably an object of different type.
- *Assembly-Part*: How to model an object that becomes a part of other object, or the other way around, how to define an object that assembles other object. Both objects are of different types.

- *Is-a (Specialization)*: Some objects have different characteristics in comparison to other objects of their type, implying a kind of specialized type of the object.
- *Matrix*: How to model an object which is mapped (with specific kinds of relations) to several other objects of different type and vice versa.
- *Graph*: How to model the relation between two objects of the same type. An object can be related to one or more objects.
- *Successor-Predecessor (Sequence)*: How to model an object (predecessor) that comes after another object of the same type (successor) in a sequential manner. Specific characteristic: an object may only be followed by one other object. Sequence can be considered as a special type of graph problem.
- *Parent-Child (Tree)*: How to model an object that has parent-child relationship with other object(s) of the same type, possibly not only one level of parent-child relationship, but also several levels of parent-child hierarchy, forming a tree structure. Specific characteristic: a child may only have at most one parent. Parent-Child can be considered as a special type of graph problem.

3. The Generic patterns

Each of the problems described in chapter IV is the basis of the generic data model patterns. According to Alexander's concept, a full description of a pattern should contain several elements, such as name, problem, context, solution, etc. [2]. The following descriptions of patterns are emphasized on problems, solutions, and some examples, but they are expressed in an open way to emphasize the discussion of the problems and their solutions and to avoid too many details.

A. Patterns Based on the Identification of an Object

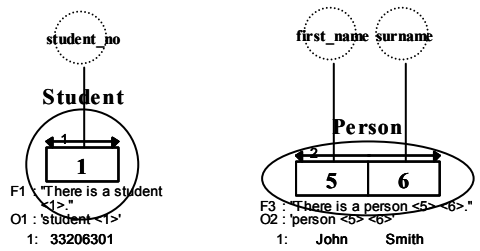
The patterns discussed in this section are based on the problems of identification of an object.

1) Single Identification Pattern

Most objects are identified uniquely using only one particular way of identification. The identification can be either single value or composite values (the unique combinations of several values).

The main characteristic of the solution (and its variants) in FCO-IM is the presence of a UC that covers all roles that constitute an OT altogether and strictly applies n rule (see [7] for explanation on n rule).

Typical examples include: a student is identified by a student number; an employee of a company is identified by employee number, a person is identified by the combination of first name and surname, etc. Fig. 1 presents examples of applications of the patterns for object types identified by a single value and by composite values.



(a) (b)

Fig. 1. Examples of applications of Single Identification Pattern's solution for object type identified by a single value (a) and by composite values (b)

2) Generalized Identification Pattern

Sometimes two or more different objects are recorded for the same kind of facts and thus, it is needed to define a new object that unites the previous objects. The solution of such problem is by using the so called generalization construct. Generalization is a way to unite two or more different OT's into a new OT that contains all those OT's [7]. The main characteristics of the new OT are the presence of roles with each role played by the different OT's and several UC's that cover only parts (not all) of the roles of the new OT.

As an example, in a company, there is an OT called 'Project' (identified by a project number denoted by LT 'project_no') and there is another OT called 'Assignment' (identified by the name of the departement that carries it out, denoted by OT Department, and a sequence number, denoted by LT 'sequence_no'). There is an FT 'Budget' that must be recorded for both Project and Assignment. Thus, a new OT called Task is defined, which is the generalization of 'Project' and 'Assignment'. Fig. 2 shows how this example is modeled in FCO-IM.

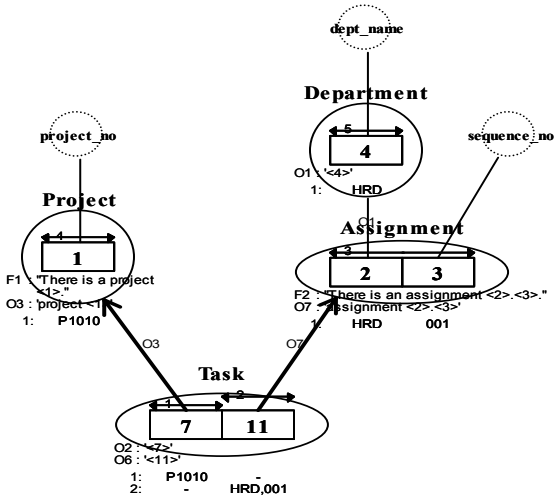


Fig. 2. An example of application of Generalized Identification Pattern's solution for Project-Assignment case

3) Synonymy Pattern

Synonymy is the phenomenon that an object is identified in more than one way [7]. The starting point of this problem is different with the generalized identification problem, since this problem is about a single OT with more than one way of identifications, while the generalized problem is about two or more OT's to be made a single OT. Nevertheless, the solution of synonymy problem in FCO-IM is also by using generalization construct. Thus, the main characteristics of the solution are basically the same as the solution for generalized identification problem.

Typical examples of synonymy problem include: an employee in a company is identified using either social security number or an employee number; a student in a university is identified using either a registration number or a student number or even with an employee number (if the student is also a teacher/an employee in the same university). Fig. 3 depicts an example of the application of Synonymy Pattern.

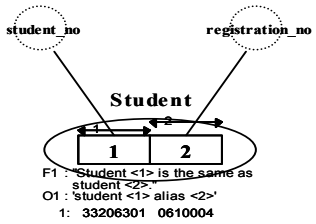


Fig. 3. An example of application of Synonymy Pattern's solution for Student case

4) Homonymy Pattern

Homonymy is the phenomenon that one name indicates two or more different objects [7]. This problem differs from the two previous problems since it concerns a particular name (or other ways of identification) that may be used as the identification for more than one OT's. The solution in FCO-IM, however, is also using generalization construct.

A typical example is the Modelling of bus routes and flights in a travelling company [7]. For example: there is an OT 'Bus Route' with one of the population 'AX11' which is also a population of OT 'Flight'. An OT called Itinerary' is defined which is the generalization of OT 'Bus Route' and 'Flight'. Fig. 4 depicts this example.

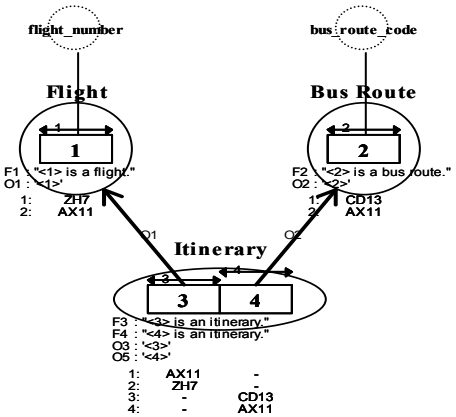


Fig. 4. An example of application of Homonymy Pattern's solution in Bus Route and Flight case

5) Recursive Identification Pattern

Some OT's are identified partly by itself. This is called a recursive OT. An object of this type can be identified uniquely based on their positions within a structure that consists of objects of the same OT.

The solution of this problem in FCO-IM is by using a special construct of generalization that is typical for recursive problems. The main characteristic of a recursive OT is the presence of at least one role that is played by the OT itself.

Typical examples of application of this pattern include: the numbering of chapter and section in a book [7], the identification of files/folders in a computer file system, family tree

(where names of family members are not unique and therefore, a particular person can only be identified uniquely by its path to the root of the tree), etc.

Fig. 5 presents an example of hierarchy of files, folder, and drives of a typical computer file system. As it can be seen in the example, there are two files with the name File-1.txt. They are different objects and are recognized by their position in the file system hierarchy. Fig. 6 shows how it is represented with recursive identification pattern. As it can be seen in Fig. 6, role 1 of OT 'File/Folder' is played by OT 'File/Folder' itself.

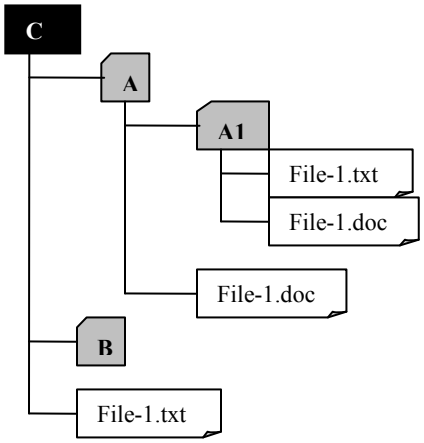


Fig. 5. An example of typical hierarchy in a computer file system

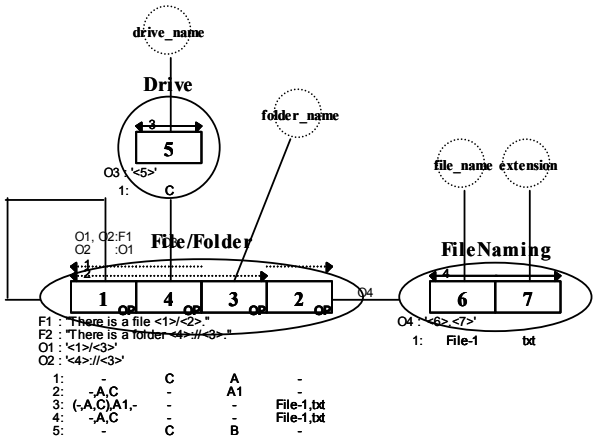


Fig. 6. An example of application of Recursive Identification Pattern in Files and Folder case

6) Set Identification Pattern

In some cases, several objects (of the same type) are grouped together in a set. The set is identified by the objects belong to it. As an example, as discussed in [8], a team of game players consists of an arbitrary number of players. So, there is a team called the team of Steve and Colin, and there is another team called the team of Macy, Marty, and John. Such identification of a team requires a specific way of Modelling which introduce a specific form of OT in FCO-IM called the set type. Fig. 7 provides an example of the use of a set type in an FCO-IM diagram. In this example, the OT TEAM is a set type. Box (role) 2 indicates that it is

played by OT PERSON. The UC 2 on top role 2 indicates that within a set, a PERSON can appear only once. The UC 3 (which covers the braces '{}') indicates that each set uniquely defines a TEAM.

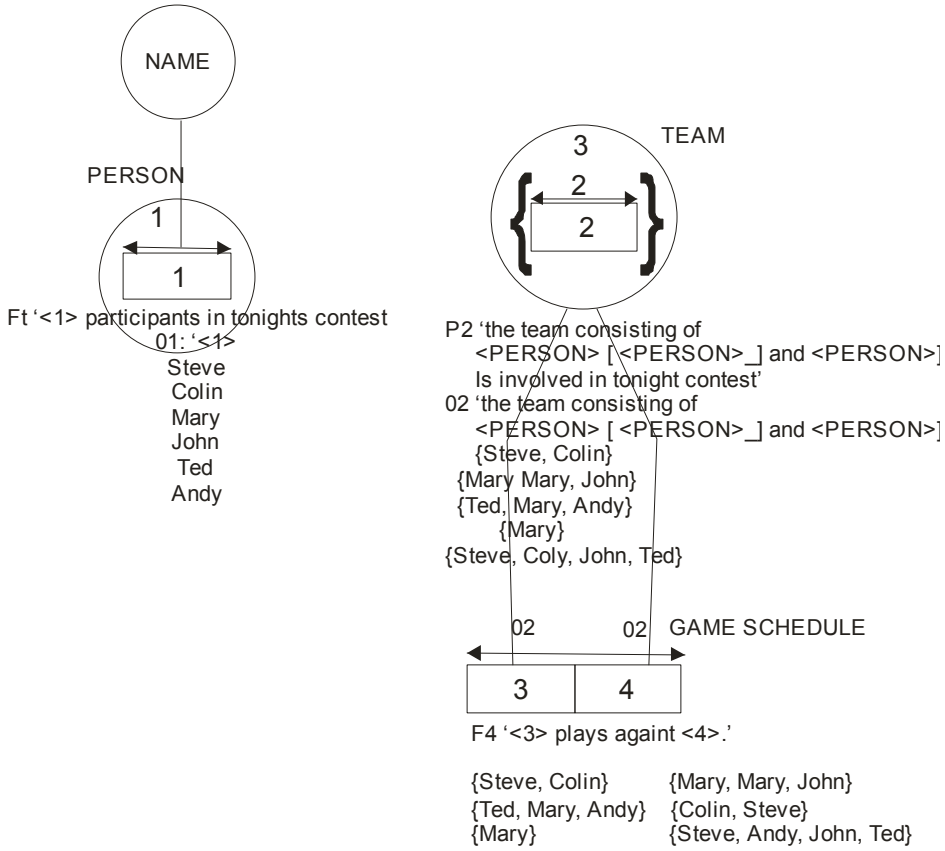


Fig. 7. An example of set type (TEAM) [8]

B. Patterns Based on the Relation between Two Objects

The patterns discussed in this section are based on the problems of the relations between two or more objects, either objects of the same type or of different types. Each of the objects can be identified uniquely using one of the patterns discussed in the previous section.

Although the relations defined are only on two objects, the relations can always be expanded into three or more objects. Nominalization concept (see [5], [7]) allows such expansion. Unfortunately, this will not be discussed in detail in this paper.

1) Attribute Pattern

Almost every object in the world has attributes/properties attached to it. For example: a person has name, address, and birthday as the properties, etc. Each of the attribute is actually also an object (can be of different or same OT) or only a certain value that describes a certain attribute.

The typical solution of an FCO-IM model for this problem is an FT with two roles; one of them is played by an OT which represents the object that possesses attribute and the other role is played by either an OT or an LT which represents an object or certain values that become the attribute of the former OT. To ensure that the attribute is dignified to the first OT, a UC is placed on top of the role that is played by the first OT.

Fig. 8 depicts an example of an FT ‘Gender of Person’ that defines the attribute gender of OT ‘Person’. The attribute itself is played by OT ‘Gender’. Note that the application of a TC at OT ‘Person’ side of this FT is optional.

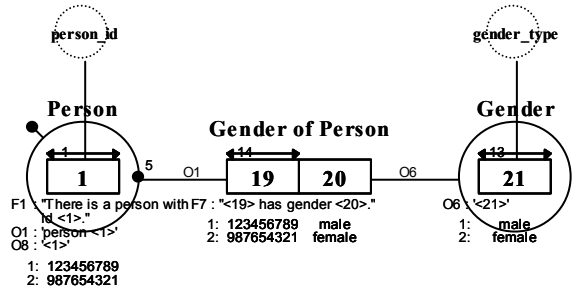


Fig. 8. An example of application of Property Pattern

2) Assembly-Part Pattern

In a lot of cases, a thing can be part of other things, or the other way around, a thing can have parts of other things. For example: a car product may be assembled of certain spareparts, a food product may be made of several ingredients, a building consists of several rooms, etc. Such relations are typical assembly-part problem. The special characteristic of this problem is that each of parts can become a part of only one assembly at a time.

A typical solution of this problem in FCO-IM is an FT with two roles; each played an OT (the OT’s can be the same or different OT’s). One OT plays the ‘assembly’; the other plays the ‘part’. A UC must be placed on top of the role that is played by OT that plays the ‘part’. This ensures that each ‘part’ will only be used in one ‘assembly’.

A typical example of this problem is the relation of product and spare part in a manufacturing company. This example is depicted in Fig. 9. OT ‘Product’ is assembled from OT ‘Sparepart’.

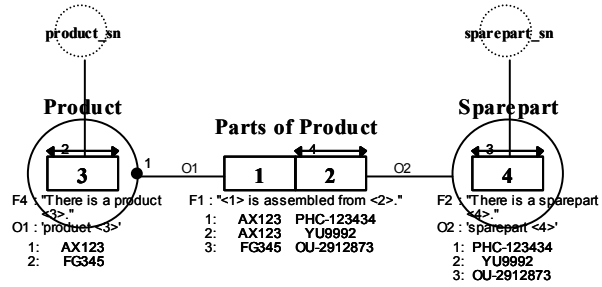


Fig. 9. An example of application of Assembly-Part Pattern (Product-Sparepart case)

3) Is-A (Specialization) Pattern

In a lot of cases, a role of an FT can be populated only by a special well-defined subset of the population of an OT that plays it. This requires a new OT which holds the population of the subset. Is-a (specialization) problem shows up in a lot of data Modelling cases. A lot of taxonomy or classification problems can be modeled using this pattern.

In FCO-IM, this kind of relationship between the two OT's is modeled using specialization construct [7]. The OT that holds all possible population in question is called the supertype; whereas the OT that holds only the subset of the population is called the subtype.

An example of specialization problem is depicted in Fig. 10 in which the OT 'Person' is the supertype and the OT 'Woman' is the subtype. The subtype 'Woman' appears because of the presence of fact(s) that records something only for 'Woman', for instance 'Number of Childbirths'.

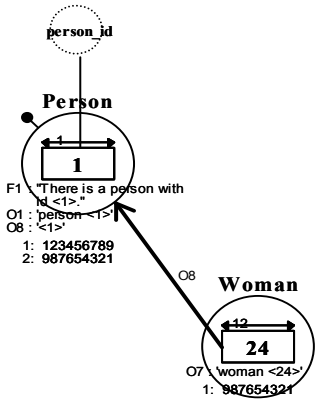


Fig. 10. An example of application of Is-A Pattern on the relation between Person (supertype) and Woman (subtype)

4) Matrix Pattern

Things that are logically modeled as matrices can be modeled using the Matrix Pattern. Suppose there is a group of objects A and B, the objects of group A can be mapped (in a particular type of relation) to several objects of group B and vice versa.

The solution of such problem in FCO-IM is by defining an FT with two roles; each role is played by an OT (can be the same or different OT). The mapping relationship is defined by putting a UC that covers both roles. This ensures that the combinations of values from both OT's are unique.

A typical example of such relation is the relation between a Person and his/her Role in a company. A Person in a company can have several Roles, such as a customer and as an employee of the company. The Role customer involves several Persons, and so does the Role employee. Fig. 11 shows the IGD of the example of Person and Role which is defined by the FT 'Role of Person'. Note that the TC's at both sides of the roles of FT 'Role of Person' are optional.

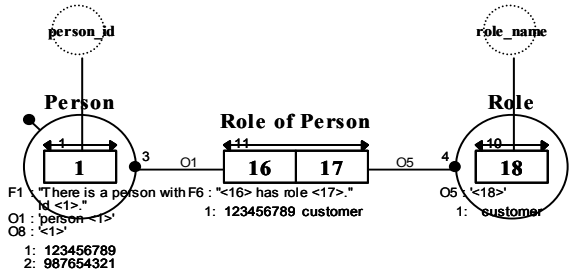


Fig. 11. An example of application of Matrix Pattern

5) Graph Pattern

In graph problem, an object is related to one or more objects in a certain type of relation. The relation is defined per pair of objects. An object can have one or more pairs. The pair can also be itself.

The solution of such problem in FCO-IM is by defining an FT that holds two roles; each role is played by the same OT. A UC is placed on both roles to uniquely define the pair.

An example of the problem is the connection between cities in a map, as can be observed in Fig. 12. Each city is modelled as OT City. UC 2 ensures that each pair occurs only once.

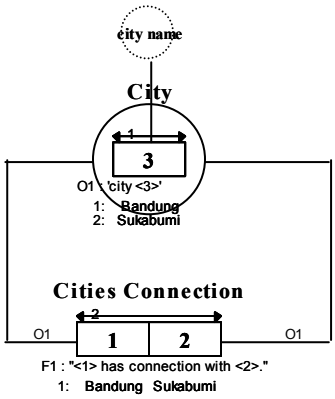


Fig. 12. An example of application Graph Pattern

6) Successor-Predecessor (Sequence) Pattern

In a sequence problem, one object is related to another object (of the same type of objects) in a sequential manner. This requires one object (the successor) is 'followed' by another object (the predecessor). A successor may only be followed by exactly one predecessor and one predecessor may follow at most only one successor.

This pattern is considered as a special type of Graph pattern. The problem defined in successor-predecessor problem is more restricted than the graph pattern in a way that an object can only be related to at most one other object and there is a sequential relation that must be kept among the objects.

A solution for this problem in FCO-IM is by defining an FT which holds two roles. Each role is played by the same OT. One of the roles is the successor role; the other is the predecessor role. To make sure that each successor is followed at most by one predecessor, a UC is placed on top of the successor role and to make sure that each predecessor only follows at most one successor, a UC is placed on top of the predecessor role.

An example of the application of this pattern is an assembling process of a product out of spare parts, as depicted in Fig. 13. Suppose there is a machine that is used to assemble a certain product. OT 'Assembly Process' is identified by OT 'Process' which is identified by a process number and an OT 'Sparepart' that defines which spare part is currently involved. The FT 'Sequence of Process' defines the sequence that must be carried out during the assembly process. Role 8 is played by the successor, while role 9 is played by the predecessor.

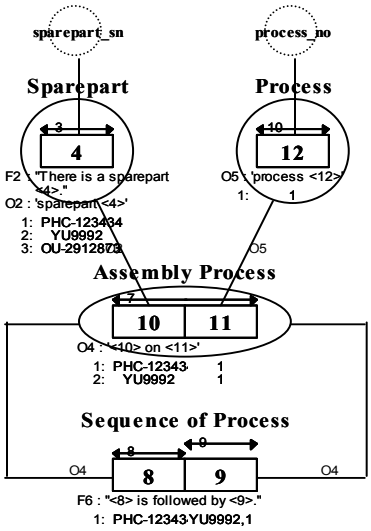


Fig. 13. An example of application Sequence Pattern

7) Parent-Child (Tree) Pattern

Objects of the same type can have hierarchy in which one object has higher hierarchy in comparison to other objects. Objects with higher hierarchy are called the parents; whereas objects with lower hierarchy are called the children. Every child has exactly one parent, while a parent may have 0 to n children.

Parent-child pattern is considered as a special type of Graph Pattern. It restricts the graph pattern by defining that each object must at most have only one relation called parent with other object, but allow the object to have more than one relation called child(ren) with other objects.

The solution of such problem in FCO-IM is by defining an FT with two roles to define the parent-child relation. Both roles are played by the same OT. One role is played by the 'parent'; while the other is played by the 'child'. Because every child has only one parent, a UC is placed on top of the role that plays the 'child'.

As an example, there is a supervision hierarchy of employees as depicted in Fig. 14. The resulting data model should consider this hierarchy as well. Suppose that the names of the employees are unique and each employee may only show up once in the hierarchy, the related FCO-IM IGD is as shown in Fig. 15.

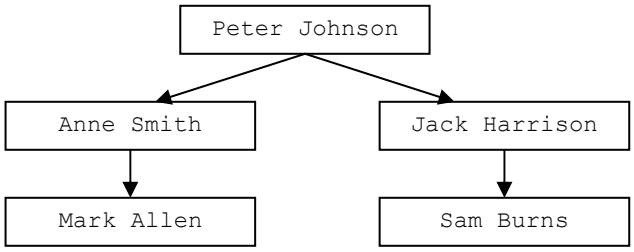


Fig. 14. An example of hierarchy of supervision of employees

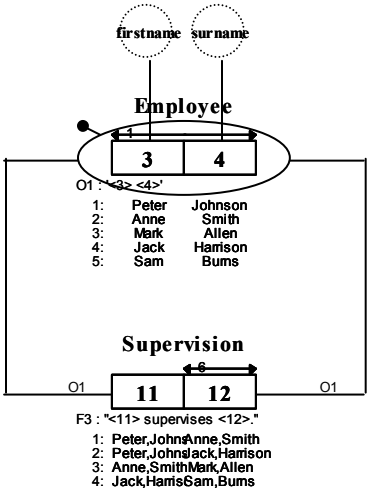


Fig. 15. Supervision Hierarchy

4. Remarks on The Generic Patterns

The problems defined for the generic patterns discussed in the previous chapter are found among the structures that can be found in a lot of data Modelling cases. Thus the solutions (in FCO-IM) of the patterns can be reapplied when a data modeler encounters and recognizes the problems in his/her work. The solutions prove to be not complicated. Even a novice data modeler should be able to understand them well. Of course, it requires the data modelers to understand the basic concepts of FCO-IM since some the solutions use typical concepts of FCO-IM.

The generic problems presented in this paper encompass small scope of problems, but they can cover the solutions for a wide variety of cases. The solutions provide a standard way of solving the problems. The knowledge embedded within the patterns can be used to make data Modelling faster and easier because data modelers do not have to take too much time to decide which one is the best way to model something.

The generic patterns can be viewed as the base patterns of bigger patterns which we will explore further in a form of pattern language [1]. Bigger data Modelling patterns are defined based on bigger scope of problems that consist of smaller problems. So, bigger patterns are basically the reapplications of smaller/generic patterns. Thus, the codifications of bigger patterns should always take the generic patterns into account. This is our view of generativity [1].

Such generativity concept can be perceived early on the relation of the two categories of patterns described in the previous chapter. Each object defined in the patterns based on the relation between two objects are identified uniquely using one of the patterns based on identification of an object.

5. Conclusions and Future Works

Typical problems found in data Modelling activities can be found from the structures of the facts found in a UoD and the problems can be solved not in complicated ways using FCO-IM. What a data modeler must do is finding in which type of problem does a specific case falls into and thus, the solution can be applied. In this paper, the generic patterns are divided into two categories, which are based on the identification of an object (6 patterns) and based on the relation between two objects (7 patterns). The latter group of patterns “use” the patterns of the first group in defining the identification of the objects involved.

Some works are still needed to be carried out regarding to the generic patterns. First, the patterns are needed to be documented in details, providing the elements of each pattern in order to create a full documentation of the patterns. Next, the patterns are required to be shared with other parties in order to receive comments and improvements. Furthermore, there may be other generic data Modelling problems that are not covered by the list provided in this paper; thus the list can be expanded in the future.

Bigger scope of patterns is also in the list of next works. Most current works on data model patterns are domain-specific patterns, i.e. patterns that can be applied directly in a very specific realm. This type or other type of patterns can be developed for FCO-IM data model patterns in the form of a pattern language.

References

[1] Alexander, C, "The Timeless Way of Building", *Oxford University Press*, USA, 1979.

[2] A Pattern Definition, http://hillside.net/patternsdefinition.html, accessed on 19/04/2006.

[3] Appleton, B, "Pattern and Software: Essential Concepts and Terminology", http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html, accessed on 19/04/2006.

[4] Arlow, J., Neustadt, I, Enterprise Patterns and MDA, Building Better Software with Archetype Patterns and UML, Addison Wesley, 2004.

[5] Articles on FCO-IM, http://www.casetalk.com/php/index.php? Articles, accessed on 18/3/2006.

[6] Azizah, F.N.; Bakema G, "Data Modelling Patterns using Fully Communication Oriented Information Modelling (FCO-IM)", *ORM Workshop 2006* (*part of OnTheMove Federated Conferences and Workshops 2006*), working papers, Montpellier, France, 2006.

[7] Bakema, G.; Zwart, J. P.; Lek, H. van der: "Fully Communication Oriented Information Modelling (FCO-IM)", 2002. The book can be downloaded for free in http://www.casetalk.com/php/index.php? FCO-IM%20English%20Book.

[8] Bakema, G.P., Zwart, J. P. C., Lek, H. van der: "Fully Communication Oriented Information Modelling", http://www.infagon.com/Content/_FCO_Article.html, accessed on 11/6/2009.

[9] Barker, R.: Case*Method: "Entity Relationship Modelling", *Addison-Wesley Professional*, 1990.

[10] Blaha, M., Premerlani, W, "Object-Oriented Modelling and Design for Database Application", *Prentice Hall*, 1998.

[11] Coad, P., North D., Mayfield, M, "Object Models: Strategies, Patterns, and Applications", *Prentice Hall*, 1997.

[12] Fowler, M, "Analysis Patterns Reusable Object Models", *Addison Wesley*, 1996.

[13] Fowler, M, "Patterns in Enterprise Software",

[14] http://www.martinfowler.com/articles/enterprisePatterns.html, accessed on tanggal 19/04/2006.

[15] Gamma, E., Helm, R., Johnson, R, Vlissides, J, "Design Patterns: Elements of Reusable Object-Oriented Software", *Addison-Wesley Professional*, 1st edition, 1995.

[16] Hay, D.C.: Data Model Patterns, "A Convention of Thought", *Dorset House Publishing*, New York, 1996.

[17] Hay, D.C.: Data Model Patterns, "A Metadata Map", *Morgan Kaufmann Publishers*, San Fransisco, 2006.

[18] Nicola, J., Mayfield, M., Abney, M, "Streamlined Object Modelling", *Patterns, Rules, and Implementation*, Prentice Hall, 2001.

[19] Silberschatz A., Korth H.F., Sudarshan S., "Database System Concepts", *Fourth Edition*, McGraw Hill, 2002.

[20] Silverston, L., “The Data Model Resource Book”, *Revised Edition*, Volume 1 and 2, John Wiley & Sons Inc., 2001.
[21] Simsion, G.; Witt, G, “Data Modelling Essentials”, *Third Edition, Morgan Kaufmann Publishers*, 2005.