PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 18/07

Applying Analogy for the Generation of Entity-Relationship Schemas

Antonio L. Furtado

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 18/07 ISSN 0103-9741
Editor: Prof. Carlos José Pereira de Lucena July 2007

Applying Analogy for the Generation of Entity-Relationship Schemas*

Antonio L. Furtado

furtado@inf.puc-rio.br

Abstract: To support the generation of database schemas of information systems, starting from analogous predefined schemas, a five-step process is described. It involves generic and blended spaces, whose utilization is essential to achieve the passage from the source space into the target space in such a way that differences and conflicts can be detected. and, whenever possible, conciliated. The convenience to work with multiple source schemas to cover distinct aspects of a target schema, as well the possibility of creating schemas at the generic and blended spaces, are briefly considered.

Keywords: .Schema Generation, Analogy, Blending, Lattices, Entity-Relationship Model, Logic Programming.

Resumo: Para apoiar a geração de esquemas de bancos de dados de sistemas de informação, partindo de esquemas análogos predefinidos, é descrito um processo em cinco etapas. Envolve espaços genéricos e espaços aglutinados, cuja utilização é essencial para efetuar a passagem do espaço fonte ao espaço alvo de tal modo que as diferenças e conflitos possam ser detetados e, sempre que possível, conciliados. A conveniência de trabalhar com múltiplos esquemas fonte para cobrir aspectos distintos de um esquema alvo, bem como a possibilidade de criar esquemas nos espaços genéricos e aglutinados, são brevemente consideradas.

Palavras-chave: Geração de Esquemas, Analogia, Aglutinação, Reticulados, Modelo Entidades-Relacionamentos, Programação em Lógica.

* This work has been partly sponsored by the Ministério de Ciências e Tecnologia da Presidência da República Federativa do Brasil.

In charge of publications

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

1. Introduction

Designers of information systems soon learn that reusing their previous experience, and also that of other designers, is a rewarding strategy.

In particular, we have been working [BBCF, BBFC] on methods and tools for, starting from some predefined database schema regarded as a *source schema*, abstract a *pattern* that captures its structure, which is then repeatedly used to generate one or more *target schemas*. What makes this strategy viable is the intuitive perception of an *analogy* between source and target, expressed by saying that the latter *is like* the former.

Additionally, the source schema should be a typical example among those that are analogously structured, and the terminology of its underlying domain should be familiar even to the less experienced designers. If these requirements are satisfied, it will be possible to instantiate the positions occupied by variables in the pattern, by prompting the designer to indicate which names in the target schema being generated correspond to each name in the example source schema.

In the present paper, adopting an approach applicable in widely different areas [FT1], we extend our method so as to take four spaces into consideration. The diagram in figure 1 represents these four spaces, and shows how they are articulated in view of a process whereby, starting from the source, the target is gradually constructed.

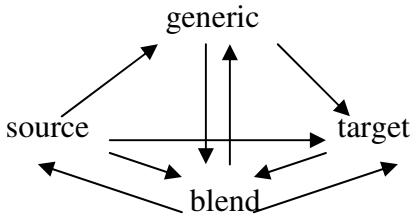

fig. 1: the four-space approach

Informally, the generic space originates from the source by importing, in a generalized format, the elements for which corresponding elements in the target will eventually be characterized. In practice, both the source and the target will contain other non-corresponding elements, since analogy is rarely bijective. Viewing the diagram as a lattice [MB], the generic constitutes the *meet* of the source and the target spaces. Whereas it denotes the elements in correspondence in these two spaces, the blended space, as the *join* of source and target, inherits all their elements, corresponding or not. Again informally, it is the space wherein whatever is incomparable or conflicting when putting together source and target can be detected, often calling for some creative form of adaptation to be remedied or conciliated [Tu, FT2]. In [Go] blending is formalized in category theory.

Thi text is organized as follows. Section 2 details how our five-step process can apply the four-space approach to the interactive generation of target database schemas of an information system, starting from an example previously specified source schema, which is a typical illustration of the *weak entity* concept. The Entity-Relationship (ER) model [BCN] is used in the presentation. Sections 3 and 4 briefly discuss, respectively, the advantages of bringing in a multiplicity of source schemas for designing distinct *aspects* of a target schema, and the possibility of also creating schemas directly from elements at the generic and/or blended spaces. Section 5 contains the conclusions.

2. The four-space schema-generation process

The process will be illustrated through a simple example. We start whith a schema, or, more precisely, a schema *fragment*, specifying `employees` and their `dependents`, which is probably the most frequently mentioned illustration of the weak entity concept in ER modelling. As a fragment, it only needs the elements relevant to characterize weak entities.

The clauses below introduce two entity classes, `employee` and `dependent`. The identifying attribute of employee is `empno`, whereas `dependent`, being a weak entity, relies on the identifying relationship `isdepof`, combined with the discriminating attribute `depno`. The identifying relationship is 1 to n, being total with respect to `dependent` and partial with respect to `employee`; these properties are indicated by associating pairs of minimum and maximum values for the participation of instances of each entity in relationship instances: at least 0 and at most n `dependents` can be related with exactly one `employee`. The relationship has attribute `family_tie`, with values such as `wife`, `husband`, `son`, `daughter`. Note that the fragment does not include, as unessential to the characterization of weak entities, certain basic properties of `employee`, such as those referring to the employment aspect itself.

```
Schema: Emp_Dep
Clauses --
  entity(employee, empno)
  attribute(employee, empno)
  entity(dependent, [empno/depno-isdepof-empno, depno])
  attribute(dependent, depno)
  relationship(isdepof, dependent/0/n, employee/1/1)
  attribute(isdepof, family_tie)
```

The schema will be used at the source space, wherefrom target schemas based on the weak entity concept can be derived. Starting from this source schema, the process goes through five consecutive steps, to be described in the sequel.

Step 1 - generating the pattern

From the source schema Emp_Dep, the Weak Entity pattern is obtained (fig. 2.1) by consistently subbstituting variables for the names of entities, relationships and attributes.

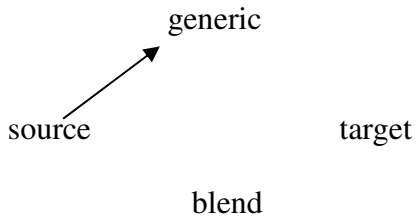

fig. 2.1: generating the pattern

Besides clauses built from those of the source schema, the pattern contains *mappings*, associating the variables introduced with the corresponding source schema names. Consistent substitution implies that, to give one example, variable `A` refers to entity `employee` wherever it occurs in the clauses of the pattern.

```
Pattern: Weak Entity
Example schema: Emp_Dep
Clauses --
  entity(A, B)
  attribute(A, B)
  entity(C, [B/D-E-B, D])
  attribute(C, D)
  relationship(E, C/0/n, A/1/1)
  attribute(E, F)
Mappings --
  A:employee
  B:empno
  C:dependent
  D:depno
  E:isdepof
  F:family_tie
```

Step 2 - generating the target schema

Suppose the designer wants to specify a Bk_Ed schema, and realizes that this too involves the weak entity concept: the `editions` of a `book` are comparable to the `dependents` of an `employee`, in that to identify an instance of `edition`, the indication of the book in question is needed, besides the `year` of publication as discriminating attribute. The generation (fig. 2.2) is basically done by specializing the clauses of the pattern (belonging to the generic space), but the diagram also refers to the originating source space, to stress that from it were extracted the names figuring in the pattern mappings.

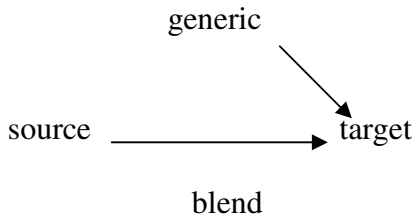

fig. 2.2: generating the target schema

Specializing the clauses of the pattern is done by substituting an appropriate name belonging to the underlying domain of Bk_Ed for each pattern variable. Relying on the assumption of an intuitive understanding of the analogy between the two domains, the designer is prompted to supply the target schema names through queries of the form:

```
- What corresponds to <name in the source schema>?
```

In our example, this would instantiate the pattern mappings as follows:

```
employee → book
empno → isbn
dependent → edition
depno → year
isdepof → isedof
```

noting that the designer may in general, with limitations, deny one or more correspondences by replying nil. So it may happen, at this stage, that nothing corresponding to the attribute `family_tie` comes to mind:

```
family_tie → nil
```

This is indeed the only element in this case that can be absent. Having indicated `book` as corresponding to entity `employee`, the indication of what corresponds to `empno` is mandatory, since no entity can lack an identifier. Likewise, if nothing corresponds to `dependent`, the indication of `isedof` as corresponding to `isdepof` would be an error, because a binary relationship requires the presence of two participating entities. The absence of `isedof`, on the other hand, though not erroneous, would defeat the purpose of the entire process – the weak entity concept makes no sense without an identifying relationship.

After inspecting the resulting target schema, the designer's knowledge of the target domain must be used to check its clauses, with a special attention to:

a. possible additions to the target schema, without correspondence in the source schema.
b. all sorts of modifications to be done in the generated clauses.

Suppose that the addition and the modification below were judged necessary:

addition: `attribute(book,subject)`
modification: `isedof - min-1:1`

with which the Bk_Ed target schema becomes:

```
Schema: Bk_Ed
Clauses --
  entity(book, isbn)
  attribute(book, isbn)
  attribute(book, subject)
  entity(edition, [isbn/year-isedof-isbn, year])
  attribute(edition, year)
  relationship(isedof, edition/1/n, book/1/1)
```

Step 3 - blending the source and target schemas

The blended space is pictured as a confluence of the source and the target spaces, taking into consideration the correspondences registered in the generic space (fig. 2.3).

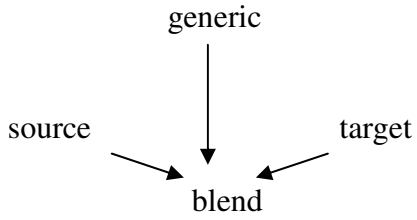

fig. 2.3: blending the source and target schemas

In the database schema-generation process, its elements will be obtained by joining each entity and relationship of the source schema with its counterpart in the target schema. To begin with, all information about each entity and relationship, contained in the various clauses of the two schemas, is collected in separate *frames*, structured as lists of property:value pairs.

Each property of an entity E is represented either by an attribute name, or by a relationship name tagged with 1 or 2 to indicate, respectively, whether E is the first or the second participant in the relationship. Since in the present example no restrictions are being imposed on the values, all value positions are filled with an underscore, a usual convention for an anonymous variable.

The properties of a relationship R are similarly represented. They include the identifying attributes of the two participating entities, the minimum and maximum occurrences for the first and for the second participant, and other relationship attributes if any. The frames extracted from the Emp_Dep schema are:

```
frame of employee =   [empno:_, isdepof/2:_]
frame of dependent =  [depno:_, isdepof/1:_]
frame of isdepof =    [depno:_, empno:_, min-1:0, max-1:n, min-2:1,
                       max-2:1, family_tie:_]
```

and those taken from the Bk_Ed schema are:

```
frame of book =       [isbn:_, subject:_, isedof/2:_]
frame of edition =    [year:_, isedof/1:_]
frame of isedof =     [year:_, isbn:_, min-1:1, max-1:n, min-2:1,
                       max-2:1]
```

We shall introduce here a *join* operation on frames, specifying that, when applied to entity or relationship frames F_1 and F_2, a frame J results, whose property-value pairs comprise:

a. pairs p_1:v_1 from F_1, for each property p_1 not corresponding to any property in F_2;
b. pairs p_2:v_2 from F_2, for each property p_2 not corresponding to any property in F_1;
c. pairs p_1-p_2:$v_{1\text{-}2}$, for each two corresponding properties p_1 and p_2 in F_1 and F_2, respectively.

Value $v_{1\text{-}2}$ in item c is obtained by, in turn, joining the two values v_1 and v_2, according to the following criterion: if the values are identical constants, or at least one of them is a

variable, $v_{1\text{-}2}$ is the result of their *unification* [Kn]; otherwise the result is a term formed by the two values prefixed by an asterisk to indicate that they are in *conflict*.

The frames characterizing the blended space, obtained by joining the frames taken from the source and the target schemas, are shown below. Non-corresponding properties and conflicting values are stressed (in italic, boldface):

$F_{employee} \vee F_{book}$ = [empno-isbn:_, isdepof/2-isedof/2:_, ***subject***:_]
$F_{dependent} \vee F_{edition}$ = [depno-year:_, isdepof/1-isedof/1:_]
$F_{isdepof} \vee F_{isedof}$ = [depno-year:_, empno-isbn:_, min-1:****(0,1)***, max-1:n, min-2:1, max-2:1, ***family_tie***:_]

A disclaimer is in order here. We have considered only one simple type of conflict. If the designer is allowed to perform arbitrary modifications to the target schema initially obtained by instantiating the pattern variables (cf. step 2), other types of conflict may occur, calling for the specification of appropriate criteria to handle them. As noted in [FT2], blending is, in general, a most complex task, requiring a great deal of creativity from the part of the designer, who may have to devise ad-hoc ways to achieve consistency.

Step 4 - revising the target (and source) schemas

The resulting blended space can be *reinjected* into the derived target space, and even into the originating source space, if the designer admits the possibility of also reconsidering it (fig. 2.4).

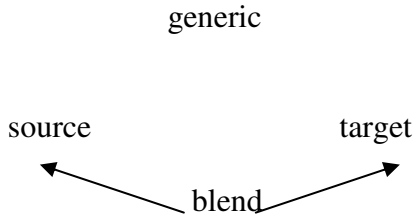

fig. 2.4: revising the target (and source) schemas

In our example, a convenient way to call the designer's attention to what was not used from the source schema is to display together, in frame format, the entire list of current properties of each entity and relationship in the target schema, expanded as the result of blending. Such frames are directly obtained from the blend frames by reducing the corresponding properties back to their names in the target space, and, naturally, keeping the names of the source space properties until now disregarded:

frame of $book_{employee}$ = [isbn:_, isedof/2:_, ***subject***:_]
frame of $edition_{dependent}$ = [year:_, isedof/1:_]
frame of $isedof_{isdepof}$ = [year:_, isbn:_, min-1:1, max-1:n, min-2:1, max-2:1, ***family_tie***:_]

Surely, the designer may or may not judge appropriate to reconsider what was initially left out, in this case the relationship attribute `family_tie`. Would there be different "ties"

between edition and book? Ironically, the remark that "so-and-so is a revised edition of his father" is not uncommon, a playful but expressive metaphoric connection between the domain of human beings, underlying employee, and books, which may bring to mind that an edition may be classified as revised, or corrected, or expanded, or abridged, possible values for a new edtype attribute for the isedof relationsip.

The reconsideration of a source schema such as Emp_Dep for expansion is more rarely desirable, especially if one wishes to keep it as a fragment containing only the features necessary to characterize weak entities. But in case one wants to examine the possibility, the blend frames can be alternatively renamed as follows:

```
frame of employee_book =      [empno:_, isdepof/2, subject:_]
frame of dependent_edition =  [depno:_, isdepof/1:_]
frame of isdepof_isedof =     [depno:_, empno:_, min-1:0, max-1:n, min-2:1,
                               max-2:1, family_tie:_]
```

What can be the "subject" of an employee? The subject of a book can be some fictional genre, but can also be a professional field, such as engineering, or accounting, which may suggest a new attribute profession for the employee entity, with possible values including engineer and accountant, among others.

More likely to happen is a further reduction of Emp_Dep to suppress the family_tie attribute. This would become advisable if the attribute is systematically disregarded, even at this revision step, in a long series of target schemas generations. Reconsidering a source schema, and consequently the pattern abstracted from it (as covered in step 5) is a case of *double-loop learning* [AS]: the continued use of a model providing clues for its correction and refinement.

Step 5 - revising the pattern

Since the generic space is often intended as a help to generate not just one but a plurality of target spaces, conflicts located at the blended space, as well as changes made at the source space from suggestions motivated by observing the blend, may entail its reconsideration (fig. 2.5).

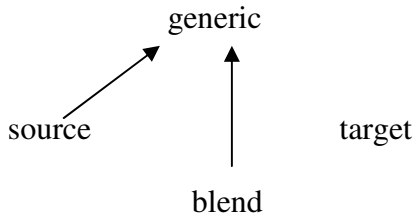

fig. 5: revising the pattern

In our example, the blend mirrors the fact that an identifying relationship must be total with respect to the weak entity, but no such requirement is imposed with respect to the entity on which it relies for identification. So the conflict registerd in property:value pair min-1:*(0,1) of the frame resulting from the join $F_{isdepof} \vee F_{isedof}$ should motivate the

insertion of a *hot spot* [Pr], i.e. a place where the specification becomes flexible, in the Weak Entity pattern.

The adpoted notation, using a question mark as prefix, will signal that the designer should be queried about the `min-1` property of the relationship denoted by variable `E`, and that the value supplied must be chosen as 0 or 1.

Moreover, if at step 4 a new attribute such as `profession` is added to the source target, or if the `family_tie` relationship attribute is removed from it, the pattern must be modified accordingly, so that it will continue to reflect the Emp_Dep schema.

If all these modifications occur, the pattern would become, after the deletion of the lines

```
  attribute(E, F)

  F:family_tie
```

and the addition or modification of three lines (in boldface):

```
Pattern: Weak Entity
Example schema: Emp_Dep
Clauses --
  entity(A, B)
  attribute(A, B)
  attribute(A, G)
  entity(C, [B/D-E-B, D])
  attribute(C, D)
  relationship(E, C/?(0,1)/n, A/1/1)
Mappings --
  A:employee
  B:empno
  G:profession
  C:dependent
  D:depno
  E:isdepof
```

3. Covering different aspects through multiple source schemas

Patterns to model the same concept can be obtained from different source schemas. We chose the Emp_Dep example to construct the Weak Entity pattern, but other examples could be selected, from which a family of versions of the pattern would be obtained and made available to designers. Originating from source schemas featuring different sets of names, the mapping section of each version would differ from that of the others. More importantly, not all clauses might be identical, reflecting permissible structural variations, according to which the versions could be classified. A designer would then have a chance to utilize the version appearing more congenial to the case on hand.

Repeating the generation process with a second version is another advantage, allowing one way to check the result. Assume, for instance, that a version of Weak Entity is available, wherein the identifying relationship is total with respect to both participating entities. If the designer of Bk_Ed had not noted at step 2 the need to correct the specification of `isedof`, blending it with the schema generated from the second version of the pattern would reveal the conflict.

But the application of more than one source must also be considered along a separate line of reasoning. Early studies on analogy and metaphor [LJ] already argued in favour of the use of multiple sources to provide a fuller characterization of a target possessing many properties, which might however be grouped into a manageable number of clusters. In [Mo], a set of eight metaphors served to explore the concept of *organization* from the viewpoints of different competing theories.

We worked with Emp_Dep as source schema to characterize a structural feature of the Bk_Ed schema, namely the reliance on an identifying relationship to designate instances of a weak entity. Many other sources can be brought in to suggest other types of properties. Clearly books can be seen as products, merchandises, objects of intellectual property, library items, etc.

Besides attributes and relationships, *operations* can be defined for books. As a library item, for instance, a book can be lent to a reader, if lost or damaged it can be replaced, etc. In [FCBB] we included, both in schemas and in patterns, clauses defining operations in terms of their *pre-* and *post-conditions* [FN]. *Integrity constraints* expressed e.g. in first-order logic notation could also be added.

Notice that such extensions, obviously of practical interest, especially in the context of a combined use of a multiplicity of source schemas, would lead to more difficult consistency verification, and, in particular, would necessitate a far more involved treatment of blends than we presented here.

On the other hand, the name of the source schema used to derive a certain set of properties of a concept serves to designate a distinct *aspect* of the concept. And, as stressed in [HT], when performing a problem-solving algorithm of exponential or high polynomial complexity to instances of an entity, for example, one can establish that only the properties derived from the one (or the few) designated source(s) will be considered, thereby reducing the computational effort.

4. Categorizations from the generic and the blended spaces

Whereas the patterns at the generic space are preserved to help in the future creation of any number of target schemas, the frames composed at the blended space are only used in connection with a specific source-target pair, and can in principle be discarded after the generation process terminates.

And yet both spaces, whose role is no more than auxiliary in the derivation of targets from sources, can give rise to new full-fledged conceptual spaces, through a process sometimes called *categorization* [FT1]. This is more easily accomplished when generic and blend represent the confluence of spaces associated with the same underlying domain.

Entities `employee` and `student` provide an example of this situation, since both have human beings as underlying domain. Their corresponding properties can conveniently be named identically, so that they can more appropriately be called *common* properties, to be *factored out* to characterize a `person` entity – in a sense, a materialization of the generic space. Both the common and the exclusive properties of `employee` and `student` are, in turn, *inherited* by the `trainee` entity, which materializes the blended space. In [BBFC] we represented these four entity classes as nodes of the lattice induced by *is-a* links, and

showed that, their properties being so specified, the meet and the join of the frames of `employee` and `student`, yield, respectively, the frames of `person` and `trainee`.

When different underlying domains are involved, categorization can still be envisaged. The resulting blend is then populated with hybrid entities, which may either appear realistic or fantastic, depending on the context. Conflating persons, objects or events is a powerful literary practice, and, surprisingly, offers sometimes intuitive clues to solve problems, as in the buddhist monk riddle expounded in [Tu]. A blend conflating persons and books, for instance, might make sense in a cartoon universe, as a Digital Storytelling application aiming to teach children how to use the facilities of a library. Apart from Information Systems, on which the present paper concentrates, and Digital Storytelling, other Computer Science areas such as Software Engineering have drawn significantly from the notions of analogy [BS] and blending [IB].

5. Concluding remarks

Although simple, the weak entity example helped us to gain a better understanding of design by analogy. Having developed an interactive logic programming tool, we were able to run experiments with the current version of the five-step process.

Much work remains to be done, especially to extend the process as described in section 2, in order to cope with an ampler variety of conflicts, and to develop semi-automatic algorithms and/or heuristics to recommend adequate strategies for handling the different situations that may arise in practice.

The topics sketched in sections 3 and 4 should also be included as objectives for future research, aiming at their integration in a more comprehensive treatment of the schema generation problem.

Referencces

[AS] Argyris, C. and Schon, D. A. *Organizational Learning II: Theory, Method, and Practice*, FT Press, 1995.

[BBCF] Breitman, K. K., Barbosa, S. D. J., Casanova, M. A. and Furtado, A. L. “Conceptual modeling by analogy and metaphor”. *Proceedings of CIKM*, 2007.

[BBFC] Barbosa, S. D. J., Breitman, K. K., Furtado, A. L. , Casanova, M. A.. "Similarity and analogy over application domains". *Proceedings of SBBD*, 2007.

[BS] Barbosa, S. D. J. and de Souza, C. S. “Extending software through metaphors and metonymies”. In *Knowledge-Based Systems*, 14, 2001.

[BCN] Batini, C., Ceri, S. and Navathe, S. *Conceptual Design – an Entity-Relationship Approach*. Benjamin Cummings, 1992.

[FCBB] Furtado, A.L., Casanova, M.A., Barbosa, S.D.J., Breitman, K.K. "Plot mining as an aid to characterization and planning". Technical Report MCC07, PUC-Rio, 2007.

[FN] Fikes, R. E. and Nilsson, N. J. "STRIPS: A new approach to the application of theorem proving to problem solving". *Artificial Intelligence* , 2(3-4), 1971.

[FT1] Fauconier, G. and Turner, M. Conceptual projection and middle spaces. Technical Report 9401, University of California, San Diego, 1994.

[FT2] Fauconier, G. and Turner, M. *The Way we Think*. Basic Books, 2002.

[Go] Goguen, J. "An Introduction to Algebraic Semiotics, with Application to User Interface Design". In *Computation and Metaphor, Analogy and Agents*. Nehaniv, C. (ed.). Springer-Verlag, 1999.

[HT] Holyoak, K. & Thagard, P. *Mental Leaps*. The MIT Press, 1996.

[IB] Imaz, M. and Benyon, D. *Designing with Blends*. The MIT Press, 2007.
[Kn] Knight, K. Unification: "A Multidisciplinary Survey". *ACM Computing Surveys*, Vol. 21, No. 1, March, 1989.
[LJ] Lakoff, G. & Johnson, M. *Metaphors We Live By*. University of Chicago Press, 1980.
[MB] MacLane, S. and Birkhoff, G. *Algebra*. MacMillan, 1967.
[Mo] Morgan, G. *Images of organization - Executive edition*. Sage Publications, 1998.
[Pr] Pree, W. *Design Patterns for Object-Oriented Software Development*. Addison-Wesley, 1995.
[Tu] Turner, M. *The Literary Mind*. Oxford University Press, 1996.