

ビジネス系情報システムのための多層型オブジェクト構造 —情報の共有化と振るまいのカプセル化の両立を目指して—

岡部雅夫 小熊康弘 渡辺香里

東京電力株式会社 情報システム部

ビジネス系情報システムの分析・設計の分野においても、オブジェクト指向が注目を集めている。しかしながら、一方で情報の共有化の求められるビジネス系情報システムにおいては、動的な振る舞いをオブジェクトにカプセル化することが困難であり、あまり成功を取っていないように思われる。

本稿では、ビジネス系情報システムの分析・設計をターゲットに、この問題を解決するために、情報の共有化のための原子オブジェクトと動的な振る舞いをカプセル化するオブジェクトとしての役割場を階層的に導入することを提案する。

Layered object structure for business information systems - to make encapsulation of behavior compatible with information sharability

Masao Okabe Yasuhiro Oguma Kaori Watanabe

Tokyo Electric Power Co., Inc.
Information Systems Department

Object-oriented techniques have made great success in the various fields of information systems. In the area of analysis and design of business information systems, however, they have not so much. It is because, in the traditional object-oriented techniques, encapsulation of behavior is incompatible with information sharability, which is one of the major requirements of business information systems.

In this paper, to solve this problem, we propose the layered object structure composed of atomic objects and role fields. The former are objects that realize information sharability, and the latter are objects that encapsulate their behavior.

1. はじめに

最近、ビジネス系情報システムの分析・設計の分野においても、OMT [1]をはじめとして、多くのオブジェクト指向の方法論が注目を集めている。その理由は、大規模・複雑化するシステムの分析・設計において、以下のオブジェクト指向の特徴が有効と考えられているからに他ならない。

(1) クラス間の継承による再利用性の向上

(2) オブジェクトと振る舞いのカプセル化

による局所化・隠蔽化

しかし、実際の適用事例をみると、多くは、オブジェクト図の作成が中心であり、リレーショナルデータベースのテーブル設計等に活かされてはいるが、動的振る舞いに関しては、オブジェクト図にコメント的記される操作等を含め、後工程にほとんど活用されていない。

本稿では、ビジネス系情報システムの分析・設計をターゲットに、動的振る舞いのオブジェクトへのカプセル化を可能にするためのオブジェクト指向の拡張を提案する。

なお、本稿は、東京国際大学・佐藤英人教授、富士ゼロックス情報システム(株)・羽生田栄一氏^(注1)、皆川誠氏の指導の下に、東京電力で研究してきたビジネス系業務を対象としたオブジェクト指向モデリング手法であるMELON^(注2)の研究成果をもとにしている。ただし、説明の都合上、多くの簡略化を行い、また、一部、拡張を行っている。

また、この内容は、現在、日本規格協会で検討されているJDMF [2]の振る舞いモデル機能の拡張と関連しており、動的振る舞いを、オブジェクト間の関連として表現するか、または、オブジェクトの属性として表現するかという等価な表現方法の差を除けば、基本的には、両者は整合の取れたものである。

2. 問題認識

オブジェクト指向を適用する上で、「何がオブジェクトか」が、まず、問題になる。

制御系システムを考える場合には、あまり大きな問題ではないようであるが、抽象的な概念を扱

うことの多いビジネス系情報システムでは、大きな問題である。

現状、ほとんどの場合、E/Rモデルでの実体をオブジェクトとして捉えているように思われる。これが、オブジェクトに動的振る舞いをカプセル化させることの障害になっているというのが基本的な問題認識である。

E/Rモデルの意味としては、以下の2つが考えられる。

(1) ものごとの自然な表現

(2) one fact in one place による情報の共有化

E/Rモデルが提唱された時は、認識されている実体を表現するために、実体集合に対して必要な属性を定義するとされていた[3]が、リレーショナルデータベースのテーブル設計のための手法として用いられる中で、いつしか、正規化された関係を実体として扱うことにより、(2)の色彩を強めてきた。

ビジネス系情報システムにおいて、情報の共有化は重要な要件ではあるが、正規化された関係は、言うまでもなく、関係代数により組み合わせられてはじめて業務的な意味を持つものである。これをオブジェクトとみなして、動的振る舞いをカプセル化しようとすることには、無理がある。

3. 基本的な考え方

E/Rモデルの原点に戻り、例えば、図-1のような単純なE/Rダイアグラムを考えて見る。

ビジネス系情報システムにおいて管理しなければならない動的振る舞いとは、まさに、この学生がある科目を履修しているという関連に関する、履修登録から、出席状況等を含め、最終成績までの状況である。

また、このE/Rダイアグラムに参加している実体である学生や科目は、関心領域であるこの関



図-1 単純なE/Rダイアグラム

連への参加において必要となるあらゆる属性を含んだものであり、必ずしも正規化された関係ではない。

MELONにおいては、おおよそ、このE/Rダイアグラムに相当するものとして「役割場」を定義し、動的振る舞いをカプセル化する単位としてのオブジェクトとする。「役割場」にはイベントの送受信に基づく状態遷移および動作が記述される。また、このE/Rダイアグラムに参加している実体に相当するものとして「役割オブジェクト」を定義する。

さらに、システム全体の振る舞いを示すために、参加する「役割場」とそれら間のイベントの送受信を表現する「役割場協調モデル」を導入する。

一方で、動的振るまいの記述は一切行わない情報の共有化のための「原子オブジェクトモデル」を導入する。「役割場」及び「役割オブジェクト」の静的な構造を、この「原子オブジェクトモデル」をもとに定義することにより、情報の共有化を実現する。

4. 具体例

以下においては、MELONによる記述の概略を、具体例に基づき「原子オブジェクトモデル」から「役割場」「役割オブジェクト」「役割場協調モデル」までを組立ることにより説明する。なお、詳細な記述は、Appendixを参照されたい。

4. 1 ISO例題

題材として、ISOでの概念スキーマモデル機能標準化の検討で用いられている例題の一部を取り上げる。この例題は、「配送センターが、小売店からの注文に応じ配送すると共に、在庫管理を行い、工場への必要な発注も行う」というものである[4]。JDMFにより記述した「名前付きオブジェクト関連図」を図-2に示す[5]。

本稿では、そのうち、一部分を、さらに簡略化し、「配送センタが、小売店からの注文に対し、その場で在庫を確認し、すべての注文品目の在庫があれば注文を受ける。」という局面に絞って考える。

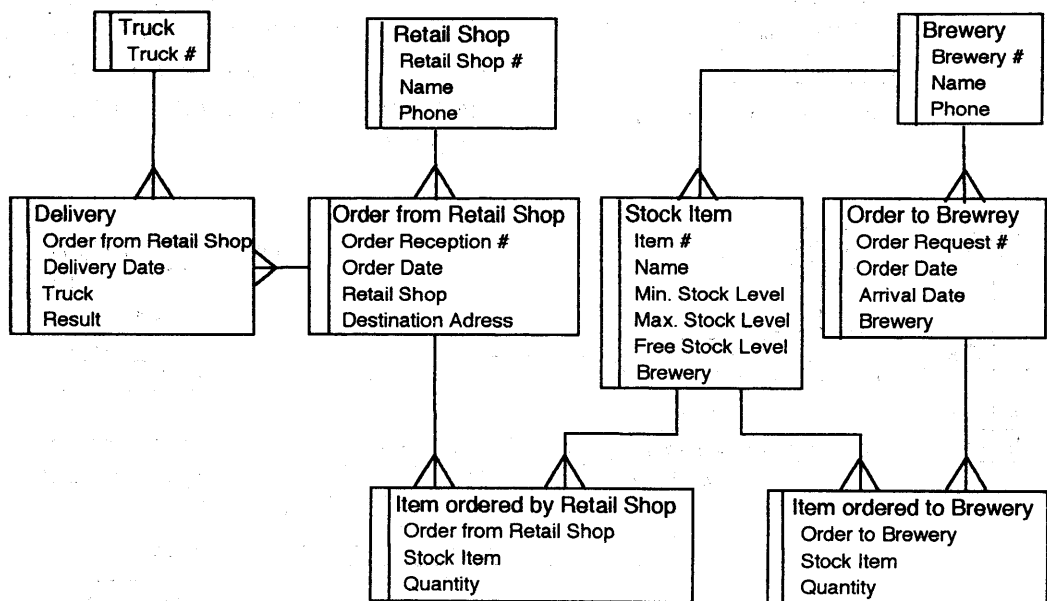


図-2 JDMFによる名前付きオブジェクト関連図

リレーショナルデータベースのテーブル設計を意識してE/Rダイアグラムを記述すれば、およそ、図-3のようになるであろう。OMTをはじめとするオブジェクト指向の方法論によった場合も、オブジェクト図の静的構造は、このE/Rダイアグラムと大同小異であろう。実際、図-3は、図-2のJDMFによる「名前付きオブジェクト図」とも、ほとんど符合している。

また、ここでの業務的な振る舞いは、「小売店から注文を受ける」「注文品目の在庫を確認する」の2つである。

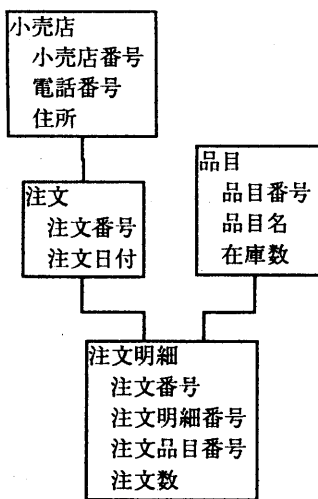


図-3 一般的なE/Rダイアグラム

4.2 原子オブジェクトモデル

一般的には、オブジェクトは属性を持つ。しかし、オブジェクトと属性という主従の関係は、本来、ある見方にたった場合にはじめて生じるものであり、見方を変えれば逆転することもありうる。原子オブジェクトモデルでは、特定の業務の見方に依存することなく、情報の共有化を図るために、オブジェクトと属性という主従の関係を排除して、全てを対等な原子オブジェクトの関連として扱う。具体的には、図-4に示す通りである。およそ、図-3のE/Rダイアグラムにおいて、全ての属性を、他のオブジェクトと対等なオブジェクトとして切り出したものになる。ただし、オブジェクトの符号化した表現である識別番号や名称は、ここでは捨象する。これらは、全てのオブジェクトに存在しうるもので、実装を意識した詳細な設計工程において付加すべきものである。

原子オブジェクトモデルは、ノーテーションの差を除けば、基本的には2項関連モデルであるNIAM[6]のサブセットである。ただし、NIAMでは、関連（センテンス）がオブジェクトとして他のオブジェクトと関連をもつ場合をオブジェクトファイブセンテンスとして扱っているが、原子オブジェクトモデルでは、これを「関連の対象化」と「関連のオブジェクト化」の2通りに区分する。

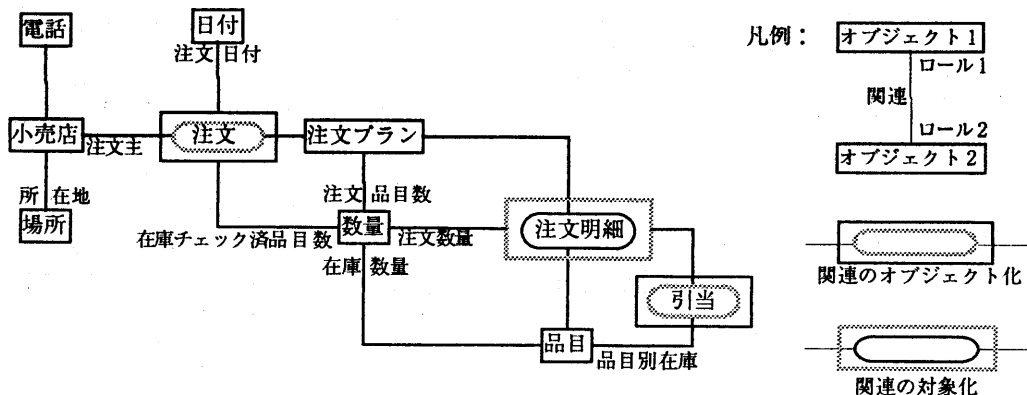


図-4 原子オブジェクトモデル

「関連の対象化」では、そのオブジェクトは関連を構成する原子オブジェクトにより決定される静的なものであり、NIAMにおけるオブジェクトタイプ定義センテンスと同一である。

これに対し、「関連のオブジェクト化」では、そのオブジェクトは関連を構成するオブジェクトが関連を生じさせる毎に生成される動的なものであり、関連を構成する原子オブジェクトが同一な異なるオブジェクトが存在しうる。例えば、図-4において、同じ「小売店」と同じ「注文プラン」により発生する「注文」であっても、関連のオブジェクト化された「注文」としては、その発生の都度、別のオブジェクト、すなわち、別の注文と認識される。

この「関連のオブジェクト化」が、以下に述べる役割場の定義において中心的な役割を担う。

4.3 役割場

役割場は、業務的な振る舞いを表すオブジェクトである。従って、先に述べた「小売店から注文を受ける」「注文品目に在庫を引き当てる」に対応して、それぞれ役割場「注文管理」^(注3)「引当処理」が図-5、図-6に示されるように定義される。ここで、この役割場の静的構造の定義は、原始オブジェクトクラスに特定の業務の見方を導入することにより行われる。

例えば、役割場「注文管理」の静的構造は、図-5に示すように、役割オブジェクト「注文主」

「注文プラン」とそれらの結びつける「注文」からなる。すなわち、原始オブジェクトモデルの関連のオブジェクト化「注文」、ならびに、その関連を構成する原始オブジェクト「小売店」「注文プラン」に着目している。役割オブジェクト「注文主」「注文プラン」は、原始オブジェクトモデルにおいて、それぞれ、原始オブジェクト「小売店」「注文プラン」に着目して、これと関連のある原子オブジェクトのロールの内、この役割場の振る舞いに関係する「所在地」「注文明細」等を属性として取り組んだものとして定義される。同様に、関連のオブジェクト化「注文」と関連のある原始オブジェクトのロール「注文日付」等も、役割場のそのものの属性として定義される。2項関連モデルにおいて特定のオブジェクトに着目し、これと関連のあるオブジェクトのロールを属性として取り込むという考えは、[7]によった。

さらに、この静的構造に加えて、イベントの送受信に伴う状態遷移や動作が定義される。イベント・状態遷移・動作の定義は、おおむねOMTに従った。状態遷移の概略は図-5^(注4)に示される通りである。

同様に、役割場「引当処理」は図-6のようになる。役割場「引当処理」は、役割場「注文管理」からのイベント「在庫チェック依頼」を受けて在庫チェックを行い、その結果をイベントとして役割場「注文管理」に返す。ここで、イベントは、そのイベントを発した役割場のインスタンス

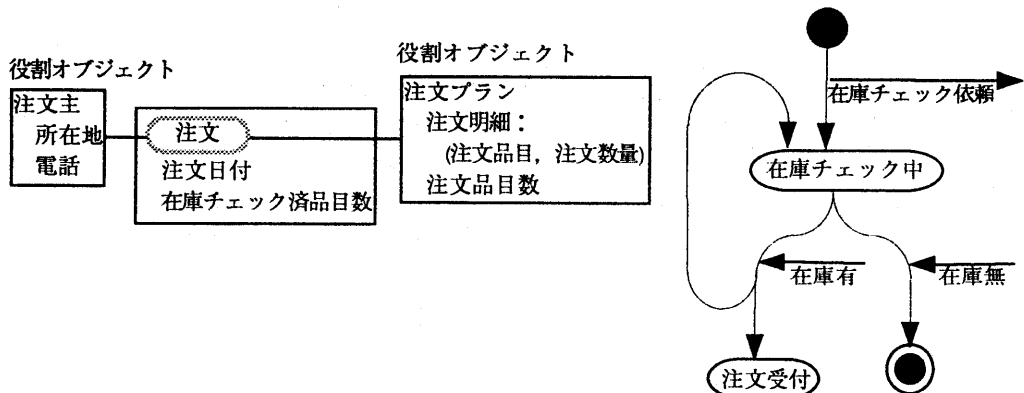


図-5 役割場「注文管理」

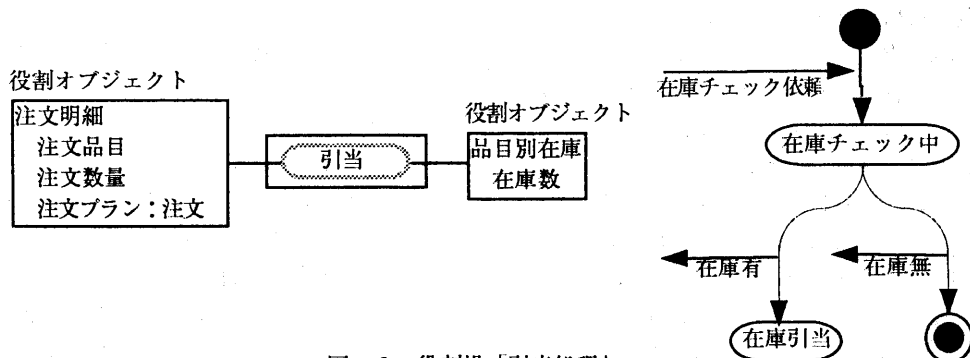


図-6 役割場「引当処理」

が暗黙の引数となる以外は、引数を持たない。すべての情報は、原始オブジェクトモデルを介して共有される。その際、暗黙の引数であるイベントを発信した役割場のインスタンスは、イベントを受信した役割場がそのイベントを受け取るインスタンスを特定するために用いられる。例えば、役割場「引当処理」に参加する役割オブジェクト

「注文明細」は、役割場「注文管理」に参加する役割オブジェクト「注文プラン」を、逆に、各注文明細側から見たものであり、原始オブジェクトモデルを介して情報を共有する。

4. 4 役割場協調モデル

図-7に表されるように、対象としているシステムに参加する役割場を定義する。役割場間のイベントの授受は、各役割場で記述されていて冗長であるが、わかりやすさのために記述した^(注3)。

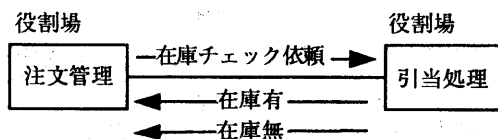


図-7 役割場協調モデル

5. 思考過程

4. においては、原子オブジェクトモデル、役割場、役割場協調モデルの順に説明したが、

これは、説明上の都合であって、実際の思考過程を意味しない。

MELONは現状の仕事のやり方にとらわれないトップダウン的な方法論を目指している。実際の思考過程としては、はじめに役割場協調モデルで概略の役割場を洗い出し、その役割場の洗練・詳細化を通じて、最終的に、その静的構造を原子オブジェクトモデルをもとに定義する。その際、既定義の原子オブジェクトの再利用はもちろん、役割オブジェクト、役割場に関しても、積極的に再利用を図っていく。

6. 継承について

役割オブジェクトのクラス階層に関しては、同じ原子オブジェクトに着目している役割場に関し、関連する原子オブジェクトの包含関係により、自然なスーパー・サブクラス関係が定義される。

役割場に関しても、同様に、同一の「関連のオブジェクト化」に着目している役割場に関し、参加している役割オブジェクト間のスーパー・サブクラス関係から、役割場間の自然なスーパー・サブクラス関係が定義される。自然なサブクラスの役割場には、そのスーパークラスの役割場の静的な構造だけでなく動的振る舞いも継承される。

7. 終わりに

本稿では、ビジネス系情報システムに求められる情報の共有化を損なうことなく、動的振る舞い

のオブジェクトへのカプセル化を可能にするために、原子オブジェクトモデルと役割場の階層的な導入を提案した。この構造の中に、オブジェクト指向のもう一つの特徴であるクラス間の継承をどう織り込んでいくかが今後の課題である。6. に述べた自然に導かれるクラス間の継承関係に加えて、役割場間の状態のクラス階層等を含めたより一般的なクラス間の継承を今後検討していく予定である。

Appendix

A. 1 原子オブジェクトモデル

atomic model

context 配送センタ;

object

@住所, @小売店, @電話, @注文プラン,
@品目, @数量, @日付, \$注文受付,
\$在庫引当, ¥注文明細;

relation

住所-小売店: (住所:@住所(1..1),
小売店:@小売店(0..n)),
電話-小売店: (電話:@電話(0..n),
小売店:@小売店(0..1)),
注文受付: (小売店:@小売店(0..n),
注文プラン(0..n)),
注文受付-日付: (注文受付:\$注文受付(0..n),
注文日付:@日付(1..1)),
注文明細: (注文プラン:@注文プラン(0..n),
注文品目:@品目(0..1)),
注文プラン-数量: (注文プラン:@注文プラン
(0..n), 注文品目数:@数量(1..1)),
注文明細-数量: (注文明細:@注文明細(0..n),
注文数量:@数量(1..1)),
在庫引当: (注文品目:@注文明細(0..n),
在庫品目:@(1..1));

end.

A. 2 役割オブジェクト

role object 注文主

focus 小売店;

attribute

電話: 電話;

住所: 住所;

end.

role object 注文プラン

focus 注文プラン;

attribute

注文品目: 注文品目,
注文品目数: 注文品目数
:= Cardinality(.注文品目),
注文数料: 注文明細.注文数量;

end.

role object 注文明細

focus 注文明細;

attribute

注文品目: 注文品目,
注文数量: 注文数量,
注文: 注文プラン.注文;

end.

role object 品目別在庫

focus 品目;

attribute

在庫数量: 在庫数量;

end.

A. 3 役割場

field 注文管理

focus 注文;

role

注文主: 小売店,
注文プラン: 注文プラン;

creation on 注文主 := an actor,
注文 := one given by
注文主;

attribute

注文日付: 注文日付
:= date(*在庫チェック完了),
チェック済み品目数: チェック済み品目数
:= チェック済み品目数 + 1
on event *在庫有
:= チェック済み品目数 + 1
on event *在庫無;

input event *在庫有, *在庫無;

output event *在庫チェック依頼;

state <●>, <在庫確認中>, <注文受付>,
<<確認状況チェック中>>, <◎>;

life cycle

<●> -> 在庫チェック -> <在庫確認中>,

```

<在庫確認中> - *在庫有/#チェック状況確認
    <<確認状況チェック中>>,
<<確認状況チェック中>>
    - *在庫チェック未了/-> <在庫確認中>,
<<確認状況チェック中>>
    - *在庫チェック完了/-> <注文受付>,
<在庫確認中> - *在庫無/-> <◎>;

action
    #チェック状況確認:
        send *在庫チェック完了
        iff 注文.注文品目数=チェック済み品目数
            send *在庫チェック未了 otherwise;
end.

field          引当処理
focus         引当;
role
    注文明細:    注文明細,
    品目別在庫:  品目別在庫;
creation on
    注文明細      := one such that .注文 =
                    sender(*在庫チェック依頼),
    品目別在庫    := 注文明細.注文品目;
input event     *在庫チェック依頼;
output event    *在庫有, *在庫無;
state <●>, <在庫確認中>, <◎>;
life cycle
    <●> - *在庫チェック依頼/#在庫チェック->
        <在庫確認中>,
    <在庫確認中> - *在庫有/-> <◎>,
    <在庫確認中> - *在庫無/-> <◎>;

action
    #在庫チェック:
        send *在庫有 and 品目.在庫数量 :=
            品目.在庫数量 - 注文明細.注文数量
        iff 品目.在庫数量 >=
            品目.在庫数量 - 注文明細.注文数量
            send *在庫無 otherwise;
end.

A. 4 役割場協調モデル
model 例題
    participant filed    注文管理, 引当処理;
end.

```

注

- (1) 当時。現在は、(株)オーズ総研。
- (2) Multi-layered Epstemic Logical Network
多層型認識論理オブジェクトネットワーク
- (3) 例題としては注文受付までを対象としているが、本来は、注文受付から配送完了までを扱う役割場である。
- (4) 正確には、在庫無の品目が発生した場合、すでに在庫引当を行っている品目に対し在庫引当の解消を依頼する2フェーズコミットと全く同じ処理が必要であるが、簡単のため省略した。
- (5) 本来のMELONにおいては、役割場における局所イベントと役割場協調モデルにおける大域イベントを区別しているため、両者のバインディングが必要となる。

参考文献

- [1] Rumbaugh J., et al.: Object-Oriented Modeling and Design, Prentice Hall (1991) (羽生田栄一監訳 オブジェクト指向方法論OTM、トッパン (1992))
- [2] 日本規格協会: データモデル機能 JDMF/MODEL-1992 (1993)
- [3] Chen P.P.: The Entity-Relationship Model - Toward a Unified View of Data, ACM Trns. on Database Syst., Vol.1 No.1, pp.9-36
- [4] ISO/IEC JTC1 SC21/WG3 CSMF ABQ-4R2: Test case to be modeled by a CSMF (1994)
- [5] ISO/IEC JTC1 SC21/WG3 ABQ-23: 'Wine Distribution Center' modeled by JDMF (1994)
- [6] Nijssen G.M.: Current issues in conceptual schema concepts, Proc IFIP TC2 Conf Nice, North-Holland Publ Co (1977) pp.31-65
- [7] Troyer O.G.: NORM: An object oriented conceptual model based on NIAM (presentation paper) (1991)