UML Profile for Enterprise Distributed Object Computing

Initial Submission

CBOP

(Consortium for Business Object Promotion)

Version 1.0

October 25, 1999
OMG Document ad/99-11-05

Copyright 1999 by CBOP

The submitting companies listed above have all contributed to this joint submission. These companies recognize that this draft joint submission is the joint intellectual property of all the submitters, and may be used by any of

them in the future, regardless of whether they ultimately participate in a revised and or final joint submission.

In future, this documentation will be continuously maintained and enhanced through the activity for standardizing of CBOP members.

The companies listed above hereby grant a royalty-free license to the Object Management Group, Inc. (OMG) for worldwide distribution of this document or any derivative works thereof, so long as the OMG reproduces the copyright notices and the below paragraphs on all distributed copies.

The material in this document is submitted to the OMG for evaluation. Submission of this document does not represent a commitment to implement any portion of this specification in the products of the submitters.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The companies listed above shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material. The information contained in this document is subject to change without notice.

This document contains information which is protected by copyright. All Rights Reserved.

Except as otherwise provided herein, no part of this work may be reproduced or used in any form or by any means- graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems- without the permission of one of the copyright owners. All copies of this document must include the copyright and other information contained on this page.

The copyright owners grant member companies of the OMG permission to make a limited number of copies of this document (up to fifty copies) for their internal use as part of the OMG evaluation process.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c)(l)(ii) of the Right in Technical Data and Computer Software Clause at DFARS 252.227.7013.

CORBA and Object Request Broker are trademarks of Object Management Group.

OMG is a trademark of Object Management Group.

Table of Contents

1 PREFACE 2

1.1 SUBMITTING ORGANIZATION 2
1.2 STATUS OF THIS DOCUMENT 2
1.3 GUIDE TO THE SUBMISSION 2
1.4 CONVENTIONS 3
1.5 SUBMISSION CONTACT POINTS 3
1.6 ACKNOWLEDGEMENTS 4

2 PROOF OF CONCEPT 5

3 RESPONSE TO RFP REQUIREMENTS 6

3.1 MANDATORY REQUIREMENTS 6
3.1.1 Component Modeling 6
3.1.2 Modeling of Business Process, Entity, Rule, and Event Objects 6
3.1.3 Specification of Business Process Objects 6
3.1.4 Specification of Relationships 6
3.1.5 Meta-Object Facility Alignment 6
3.1.6 Proof of Concept of Profile 6
3.1.7 Proof of Concept of Mappability 7
3.2 OPTIONAL REQUIREMENTS 7
3.3 ISSUES TO BE DISCUSSED 7

4 OVERALL DESIGN RATIONALE 8

4.1 GOAL 8
4.2 SCOPE 8
4.3 RELATIONSHIP TO BFOP 9

5 FORMAL DESCRIPTION OF THE PROFILE 10

5.1 GENERAL DEFINITION OF A UML PROFILE 10
5.2 THE IDENTIFIED SUBSET OF THE UML METAMODEL 12
5.3 CONTENTS OF THE PROFILE 12
5.4 BUSINESS FUNCTION OBJECT PATTERNS 12
5.4.1 Background Facts 12
5.4.1.1 Meta concept and pattern 12
5.4.1.2 BFOP pattern principle 14
5.4.1.3 Concept and Mechanism of Patterns 15
5.4.2 UML Profile for BFOP 18
5.4.2.1 BFOP Constraints to collaboration 18
5.4.2.2 BFOP Collaboration and BFOP Package 21
5.4.2.3 BFOP Meta Model 21
5.4.2.4 The motivation behind the constraints 22

6 REPRESENTATION OF BUSINESS FUNCTION OBJECT PATTERNS 23

6.1 INTRODUCTION 23
6.1.1 Basic Patterns 23
6.1.2 Unit Patterns 24
6.1.3 Optional Patterns 25
6.2 PATTERN DEFINITION 26
6.2.1 Basic Patterns 26
6.2.1.1 Master & Detail 26
6.2.1.2 Association 27
6.2.1.3 Association-3 28
6.2.1.4 Dynamic Hierarchical Structure 29

6.2.1.5 Directed Graph 30
6.2.1.6 DAG (Directed Acyclic Graph) 31
6.2.1.7 Header & Detail 32
6.2.1.8 Stock & Flow 33
6.2.1.9 Period 34
6.2.1.10 Transaction Record 35
6.2.2 Unit Patterns 36
6.2.2.1 Correspondent-Transaction 36
6.2.2.2 Product-Stock-Place 37
6.2.2.3 Employee Assignment 38
6.2.2.4 Organization Structure 39
6.2.2.5 Transaction-Transaction Line-Product 40
6.2.2.6 Sales Order-Sales Order Line-Product 41
6.2.2.7 Purchase Order-Purchase Order Line-Product 42
6.2.2.8 Closing 42
6.2.2.9 Paying or Receiving 44
6.2.2.10 Settlement 45
6.2.2.11 Settlement for Receiving 46
6.2.2.12 Settlement for Paying 47
6.2.2.13 Sales Order & Purchase Order (SS) 48
6.2.2.14 Transport (SS) 49
6.2.2.15 Direct Transport (SS) 50
6.2.2.16 Product Information (SS) 51
6.2.2.17 Daily Inventory (SS) 52
6.2.2.18 Settlement for Receiving & Paying (SS) 53
6.2.2.19 Organization (SS) 54
6.2.2.20 Product Configuration (SS) 55
6.2.3 Optional Patterns 56
6.2.3.1 Unit Price by Correspondent 56
6.2.3.2 Unit Price by Period 57
6.2.3.3 Unit Price by Correspondent and Period 58
6.2.3.4 Employee Role 59
6.2.3.5 Employee Role Definition (SS) 60
6.3 OVERALL PACKAGE STRUCTURE 61
6.4 SAMPLE OBJECT MODEL FOR BFOP 63
REFERENCES 64

1 Preface

1.1 Submitting Organization

This following organization is pleased to submit this proposal in response to the OMG UML Profile for Enterprise Distributed Object Computing RFP (OMG document ad/99-03-10):

- Consortium for Business Object Promotion(CBOP)

1.2 Status of this document

This document was created based on CBOP working draft specifications developed by CBOP members IT Innovation Incorporated and Synergy Research Corporation.

The original document was developed as a part of CBOP's business objects standardization activities for the purpose of improving efficiency in application system development and encouraging circulation of software's parts for reuse.

Also, this documents was created based on the work of many people listed in References, and we would like to acknowledge all of them. In particular, many of the proposed patterns were built with the help of excellent books which were written by Martin Fowler [Fowler 97], Peter Coad [Coad 91], Erich Gamma [Gamma 95], Desmond F. D'Souza [D'Souza 99], and so on.

1.3 Guide to the Submission

Chapter 1 provides contact information and a guide to this submission.
Chapter 2 is the proof of concept statement.
Chapter 3 explains how this submission satisfies the mandatory requirements of the RFP, and how it addresses some optional requirements.
Chapter 4 presents a formal model of the proposed UML Profile for Enterprise Distributed Object Computing (EDOC).
Chapter 5 presents a formal model of the proposed UML Profile for EDOC.
Chapter 6 provides a detailed explanation of how each pattern is represented in the proposed profile.
Chapter 7 provides examples of the proposed profile usage(<TBD>).
Chapter 8 specifies the conformance criteria for the proposed profile(<TBD>).

1.4 Conventions

<TBD>

1.5 Submission Contact Points

(General discussions and comments can be sent to boi-wg@omg.org)

Hajime Horiuchi
Tokyo International University
1-13-1 Matoba-kita, Kawagoe-shi
Saitama 350-1102
Japan
Phone: +81-492-32-1111
Email:hori@tiu.ac.jp

Yoshihide Nagase
Technologic Arts Inc.
Shinko Bld. 5F
3-2-15 Hongo, Bunkyo-ku
Tokyo 113-0033
Japan
Phone: +81-3-5803-2788
Email:yoshi@tech-arts.co.jp

Masaharu Obayashi
Kanrikogaku Ltd.
Meguro Suda Bldg.
3-9-1 Meguro, Meguro-ku
Tokyo 153-0063
Japan
Phone: +81-3-3716-6300
Email: obayashi@kthree.co.jp

Tomoo Yoda
Synergy Research Corp.
3-16-306, Roppongi 7-chome,Minato-ku,
Tokyo 106-0032
Japan
Phone: +81-35414-0885
Email: yoda@synergy-res.co.jp

Kyo Suzuki
Sterling Software
1-4-1 Sumitomo Fudosan Bldg.
Koishikawa, Bunkyo-ku
Tokyo 112-0002
Japan
Phone: +81-3-5804-1476
Email: kyo.suzuki@sterling.com

1.6 Acknowledgements

This work and CBOP's activities are supported in part by Information-technology Promotion Agency, Japan (IPA). The submitter also appreciate discussion and cooperation with the members of the Japan National Body's Subcommittee of ISO/IEC JTC1/SC32/WG2 "Data Management and Interchange – Metadata", and others.

2 Proof of Concept

This submission is based upon the CBOP's Business Function Object Patterns• (BFOP). CBOP's Component Committee task forces, such as ATOLAS Task Force• (Sterling Software's COOL:Plex based), EJB Task Force (Enterprise Java Bean based) and Web Component Task• Force (DCOM based) are implementing components based upon BFOP.

3 Response to RFP Requirements

The following lists the requirements from the OMG UML Profile for Enterprise Distributed Object Computing RFP (ad/99-03-10) and describes how this submission addresses them.

3.1 Mandatory Requirements

3.1.1 Component Modeling

Proposals shall provide mechanisms allowing the specification of information necessary in a design using UML components for the implementation of a system in an enterprise distributed computing environment using emerging enterprise-class component models.

<TBD>

3.1.2 Modeling of Business Process, Entity, Rule, and Event Objects

Proposals shall provide mechanisms for the specification, at the design level, of the distinction between business process objects, business entity objects, business rule objects, and business event objects.

<TBD>

3.1.3 Specification of Business Process Objects

Proposals shall provide mechanisms for precisely defining the behavior of business process objects, including

- *Specification of business rules and their behavioral semantics*
- *Manipulation of business process objects at runtime*

<TBD>

3.1.4 Specification of Relationships

Proposals shall provide mechanisms for the specification of additional, specialized relationship semantics beyond the base UML metamodel, including the following;

- *Additional properties of relationships to specify constrains or operational semantics*
- *Classifications of relationships by their properties*
- *Derivation of per and post conditions for create/read/update/delete ("CRUD") operations applied to participants in the relationships, based on the above properties and classifications*

<TBD>

3.1.5 Meta-Object Facility Alignment

The profile shall conform to the OMG Meta-Object Facility (see section 6.3 for explanation of such conformance).

<TBD>

3.1.6 Proof of Concept of Profile

Submissions shall provide sample models expressed in terms of the profile.

<TBD

3.1.7 Proof of Concept of Mappability

Submissions shall also include a non-normative mapping of the profile to CORBA to prove that such a mapping is possible.

<TBD>

3.2 Optional requirements

3.3 Issues to be discussed

- Subset Integrity

Submissions should discuss the extent to which the subset defined by the profile is self-contained. Any dependencies of the profile subset metamodel on portions of the UML metamodel not in the subset should be noted and justified and the approach to dealing with them should be discussed.

<TBD>

- Simplification of and Aid to the Development Process

Submissions should discuss if and how the submitted profile simplifies or aids the development lifecycle.

<TBD>

- Tool support

Submissions shall discuss how the submitted profile enables tool support for object-oriented enterprise computing system development and how such tools may be judged more or less compatible with the profile.

<TBD>

- Alignment with Action Semantics for UML

A currently outstanding RFP(ad/98-011-01) addresses the need for more precision in the UML metamodel for actions, particularly within the context of state machines. The responses to the Action Semantics RFP will result in revisions to the current UML metamodel that should enable more precise and complete definition of the steps and rules of business processes. Proposals shall discuss how the support for specification of business process objects in the profile aligns with submissions to the Action Semantics RFP.

<TBD>

4 Overall Design Rationale

4.1 Goal

The proposed UML Profile for EDOC was designed to provide standard means for expressing object models such as Business Function Object Patterns (BFOP) using UML notation with pattern applying mechanisms which are required to describe models.

Successful implementation of an enterprise computing system requires the system operation to be directly related to the business processes it supports. In order to build up a good object-oriented model for enterprise computing reusable standard models are required.

The standard models have business entities, processes and rules. They also include a set of common and reusable patterns of relationship properties which occur in business modeling. CBOP's BFOP is being developed to achieve this objective.

4.2 Scope

BFOP is a set of object patterns laid out in a hierarchical multi-layer structure. The layers are called Basic, Unit, Basic Model, Product (application systems) and Option layers. Figure 4.2-1 illustrates how "BFOP: Organization(SS)" is composed from "BFOP: Employee Assignment" and "BFOP: Organization Structure". The UML parametric collaboration mechanism is used to materialize the pattern integration. The right down arrow shows unfolded "BFOP: Organization (SS)".

One of the major benefits for using this multi-layered structure is that it enables reuse (inheritance) of the constraints which have been defined and encapsulated in patterns in lower layers. It provides a normalized way to define constraints and is effective in keeping the consistency within the object model.

Figure 4.2-1 Overview of Architecture

4.3 Relationship to BFOP

This submission is based on BFOP which is a part of CBOP technologies.

5 Formal Description of the Profile

5.1 General Definition of a UML Profile

There currently is no normative definition of a UML profile. However, the Business Object Initiative RFPs provide the following working definition of a UML profile.
A UML profile is a specification that does one or more of the following:

- Identifies a subset of the UML metamodel (which may be the entire UML metamodel).
- Specifies “well-formedness rules” beyond those specified by the identified subset of the UML metamodel. “Well-formedness rule” is a term used in the normative UML metamodel specification (ad/97-08-04) to describe a set of constraints written in UML’s Object Constraint Language (OCL) that contributes to the definition of a metamodel element.
- Specifies “standard elements” beyond those specified by the identified subset of the UML metamodel. “Standard element” is a term used in the UML metamodel specification to describe a standard instance of a UML stereotype, tagged value or constraint.
- Specifies semantics, expressed in natural language beyond those specified by the identified subset of the UML metamodel.
- Specifies common model elements (i.e. instances of UML constructs), expressed in terms of the profile.

If it can be justified as compellingly necessary, a profile can also specify new metatypes, i.e. new instances of MOF constructs such as MOF Classes, MOF Associations, etc. (even though this would strictly go beyond a true “profile” of UML).
Figure 5.1-1 uses UML notation to express the definition of a UML profile.

The Business Object Initiative (BOI) roadmap includes four RFPs, Three of which have been formally issued. Figure 5.1-2 illustrates how the three issued RFPs and the draft RFP 4 relate to the concept of a UML profile.

Figure5.1-2 The Business Object Initiative RFPs

5.2 The Identified Subset of the UML Metamodel

The "identified subset" of the UML 1.3 metamodel includes the following MOF Packages in their entirety.

- Core
- Extension Mechanisms
- Common Behavior

Some of the Data Types Package elements are also includes. The following elements of the Data Types Package are excluded:

- Integer
- UnlimitedInteger
- String
- Time
- Boolean
- PseudostateKind
- LocationReference

All other elements of the Data Types Package are included in the identified subset.

5.3 Contents of the Profile

The proposed UML Profile for EDOC is composed of:

- Pattern notation using UML
- Patterns defined in Basic, Unit, Basic Model, Option and Products layers

The profile does not define any new metamodel elements. Pattern notation is expressed by using currently available metamodel elements. Patterns are described using the Pattern notation.

5.4 Business Function Object Patterns

This section presents Business Function Object Patterns (BFOP) defined in the UML Profile for EDOC. UML currently does not provide guidelines for how to express object patterns describing business functions. However, such patterns as defined in the BFOP are useful for developing enterprise systems based on distributed object components.

5.4.1 Background Facts

5.4.1.1 Meta concept and pattern

One of the difficulties in understanding meta data is caused by the arbitrarily use of "meta" concept.

So many dimensions can be used to induce a meta hierarchy. Someone use meta concept to distinguish an object from another object, which describes an object. The definition of pattern concept is also ambiguous. One would ask if it would be adequate to recognize a pattern as meta data because it is used to produce an object model, just

like meta data typically do? Clearer definition is required for " meta concept". Following are tentative classifications of meta hierarchy concept discussed in the CBOP.

Figure 5.4-1 Classifications of meta concept

1. Type and instance

This is a typical use of meta hierarchy concept. Meta data describes characteristics of a set in terms of properties. The set can have no, one or more elements in it. Type definition is typical meta data. Type definition should be used to restrict members in the set. All properties of a type should be inherited.

2. Description of elements

Description of elements exists in the meta level. However, the description are only used by human being to generate an instance of the set. It does not restrict characteristics of existing instances dynamically.

3. Description of format of and relationship of sets.

Meta data in this hierarchy is similar to "type and instance". However, nothing is inherited from meta level. Sometime, this type of meta data set is called "repository".

4. Relationship of lower level elements

Meta data in this hierarchy is dedicated to describe the relationship between elements in a particular set. Sometime, this type of metadata set is called "registry" or "directory". Also, nothing is inherited from meta level.

To cope with the consolidated UML extensions, relationship among UML, MOF, Profiles and object patterns need to be more clearly defined. This proposal intends to make a set of object patterns, a profile of UML as a part of it.

Figure 5.4-2 illustrates the relationship among UML, MOF, Model, and Pattern (profile). Using the classifications above, the relationship between MOF and UML is (3) and (1). Also, the relationship between UML and Model is (2). However, relationship between Pattern and Model should be (1).

This implies that a model should be developed according to UML rules by inheriting contents from patterns. Consequently, UML must provide not only notation and modeling constraints but also those inheritable contents in order to keep consistency and integrity among models developed by multiple people.

Figure 5.4-2 The relationship among UML, MOF, Model, and Pattern (profile)

5.4.1.2 BFOP pattern principle

In the world of object modeling, many patterns have been proposed, such as "Design Pattern" proposed by E.Gamma et.al [Gamma 95], or "Analysis Patterns" proposed by M. Fowler [Fowler 97], or "Catalysis Approach" proposed by D. D'Souza [D'Souza 99].

Pattern is something used to represent modeling know-hows or techniques to help developers in keeping efficiency and consistency among products.

In its object pattern standardization effort, the CBOP focuses rather on improving sharability and reusability of object models than on assisting modeling efforts by showing good modeling techniques to them.

To improve sharability and reusability of object models, patterns must support the following features:

1. Model must be consisted using the predefined normative modeling constructs, not only modeling manners and notations.
2. Predefined modeling constructs should include the common atomic objects, such as, Date, Currency, Country-code, which are needless to be discussed when they are used.
3. Common aggregated objects, such as Customer, Company, or Order, which represent business entities, also should be predefined as normative modeling constructs. They should be defined using the normative atomic objects,
4. Business concept, such as, Trade, Invoice, or Settlement, which are typically represented as relationship among objects, should be defined as aggregations of the common elementary aggregated objects or simple objects. They also have to be predefined as normative modeling constructs.
5. Those aggregations could be defined as object patterns, which can be predefined using the more basic and elementary patterns as base.
6. Patterns can represent business concept with the aggregation of more elementary patterns. Therefore, aggregation or composition mechanism of pattern must be provided.
7. Business rules which govern business concept can be represented with a pattern as constraints encapsulated in it. Therefore, the mechanism for constraint inheritance among patterns must also be provided.

5.4.1.3 Concept and Mechanism of Patterns

In this section, the concept and mechanism of patterns are discussed from the viewpoint of pattern notation, relationships among patterns and pattern types and their instances.

5.4.1.3.1 Notation for Pattern

We considered that there are three basic forms on expressing patterns. First, simple pattern which is a pattern consisting of minimal elements needed to form a pattern. Second, inherited pattern which is a pattern defined by inheriting from another pattern. And the third one is composite pattern which is a pattern defined as a result of combining more than two patterns. The composite pattern concept is an extension of the inherited pattern. Using the above three basic pattern forms as the base, we propose the following notations for expressing patterns and their mata model prototype .

• Simple Pattern

A simple pattern consists of minimal elements and does not involve another pattern. In BFOP, type (i.e. an abstract class) and relationship among types are significant elements for specifying the static structure of a simple pattern. In addition to the static structure, operations are defined to characterize the pattern's behaviors. Constraints for the operation can be specified as the pre/post conditions described in OCL. Figure 5.4-3 illustrates the notation for simple pattern and its meta model.

Figure 5.4-3 The notation of Simple Pattern and its

• Inheritance pattern

The pattern inheritance mechanism is provided to describe a pattern which is defined in conjunction with another already existing pattern. The names of types and attributes in the inherited pattern can be renamed as appropriate for the inheriting pattern. This provides the way to build various patterns for specific usages.

For instance, pattern <header>-<detail> can be used to generate many patterns which share the common characteristics of the header-detail. Typically, patterns inherited from the <header>-<detail> need stricter constraints than the original pattern. If the pattern <A'>-<B'> is created from the pattern <A>-<B>, the types A and B are replaced with A' and B' respectively.

Figure 5.4-4 shows the notation and mechanism of the inherited pattern and its meta model. The diagram of meta model indicates that a pattern has relationship with another pattern.

Figure 5.4-4 The notation of Pattern Inheritance and its metamodel

• Composite pattern

Third form of pattern, composite pattern provides a way to build more complex patterns. When combining two patterns to describe a composite pattern, a new type (i.e. logical class) which share the common characteristics of the original patterns. The new combining type is expressed using the parameterized collaboration in UML 1.3. The pattern composition is useful for building hierarchical structure of patterns. Figure 5.4-5 is a brief diagram illustrating the notation of composite pattern and its meta model.

5.4.1.3.2 Relationship among Pattern Mechanisms

Figure 5.4-5 The notation of Pattern Composition and its Metamodel

The relationship among pattern mechanisms described above can be explained using package diagram in UML notation as the following figure 5.4-6. Also, see the section 5.4.2.1.

Figure 5.4-6 The relationship for pattern mechanisms

5.4.1.3.3 Applying Patterns

As previously discussed in section 5.4.1.1, when one works at the meta level, he/she must always thinks from the viewpoint of types and their instantiations. To apply this discussion to patterns, patterns are what object models are instantiated from. On the other hand, to extract types as meta from object models is to produce patterns. Having many patterns as common "types" of object models means that you are equipped with various templates which are reusable.

When patterns are organized in hierarchy using pattern composition mechanism, instantiations are realized through a series of resolutions of composite patterns by performing "unfold". If composite patterns are granular enough to include components, instantiations can be performed not only by unfolding patterns into patterns at meta level but also by unfolding patterns into instantiated object models. This means that this pattern concept can be effectively utilized for component based development which is a requirement for the EDOC.

Figure 5.4-7 Instantiation of Patterns

5.4.2 UML Profile for BFOP

5.4.2.1 BFOP Constraints to collaboration

We have introduced some constraints to the standard UML collaboration in order to simplify the structure and the pattern interactions within BFOP. This simplification is achieved by constraining the abilities of the BFOP classes. To model those constraints virtually at meta level, we have introduced two stereotypes as extensions of the standard UML class which are BFOP Owner Class and BFOP Member Class defined with the following constraints.

1. BFOP Owner Class

- ✓ has no association with BFOP member classes
- ✓ has only one instance in a collaboration
- ✓ the instance does not know any member instance in a structural way
- ✓ the instance finds its member instances only through parameter visibility when owner operation is invoked

2. BFOP Member Class

- ✓ may have associations with other BFOP Member Classes
- ✓ uses its associations only to know other member instances
- ✓ can not invoke other member's operations

Figure 5.4.2-1 Collaboration has two aspects:
a structural part and a dynamic part

Figure 5.4.2-2 Collaboration also has two aspects: a structural part and a dynamic part. However, BFOP owner class has no association with BFOP member classes and BFOP member class instance does not interact with other member instances.

Figure 5.4.2-3 BFOP Package always contains only one BFOP collaboration.

5.4.2.2 BFOP Collaboration and BFOP Package

BFOP Collaboration is simply a UML collaboration realized only by BFOP Owner Class and BFOP Member Class.

BFOP Package is a UML Package with only one BFOP Collaboration inside of it.

Figure 5.4.2-4 UML Extension for BFOP

5.4.2.3 BFOP Meta Model

It is not necessary to define BFOP meta model since its all model elements are already defined in UML standard meta model and BFOP does not define anything which violates the UML standard.

However, we have created a virtual meta model highlighting BFOP constraints represented in the simplified standard UML meta model in order to help people understand the concept in BFOP. Fig. 5.4.2-5 and constraints table Table 5.4.2-1 are provided for this purpose.

Figure 5.4.2-5 Simplified Standard UML Meta-model and BFOP extension

Table 5.4.2-1

Classifier	Type			
	Operation Parameter	Attribute	AssociationEnd	AssociationEnd Cardinality
BFOPOwner	BFOPMember DataType	DataType	-	0
BFOPMember	BFOPMember DataType	BFOPMember DataType	BFOPMember	0..*

5.4.2.4 The motivation behind the constraints

1. Comprehensive organization

While BFOP member classes typically have persistence, BFOP owner classes do not, so a designer who refers to BFOP packages for reusing them can easily recognize that the owner could be a class working as a coordinator for the member classes that implements the business logic unique to the data stored in database.

2. Easy reuse

In BFOP, classes are connected to each other to meet high cohesion and low coupling principle that promote efficient reuse

6 Representation of Business Function Object Patterns

6.1 Introduction

6.1.1 Basic Patterns

No.	Pattern Name	Description
1010	Master & Detail	A Detail refers to zero or one Master. A Master should be referred by many Details. For example customer-Order, Product type-Product.
1020	Association	An Association Class exists between two classes. Stock-Warehouse and product.
1030	Association-3	An Association exists for three classes. Price-Product, Supplier, Period.
1040	Dynamic Hierarchical Structure	It's same as tree structure. Organization of company.
1050	Directed Graph	It's same as network structure. Rout of delivery.
1060	DAG (Directed Acyclic Graph)	This is extended from DAG. This is not allowed loop. Built of Material.
1070	Header & Detail	Parent and children. Order-Order Line. Bank-Branch.
1080	Stock & Flow	This is restricted DAG. This does not have multiple levels. Transportation-Location. Currency-Exchange Rate.
1090	Period	Period has From Date and To Date. Otherwise From (To) Date and Period notify TO (From) Date.

1110	Transaction Record	Many Transactions are summarized by one Time Bucket. Sales-Closing Period. Sale Balance-Financial year.

6.1.2 Unit Patterns

No.	Pattern Name	Description
2010	Correspondent-Transaction	Correspondent has many Transactions.
2020	Product-Stock-Place	Product and Stock Place Keep Stock. As Stock Place, Warehouse, Rack or Lot number are available.
2030	Employee Assignment	An Employee can be assigned multiple sections.
2040	Organization Structure	Organization can be kept in each period.
2050	Transaction-Transaction Line-Product	A Transaction can handle many Products, because of Product Line. Shipping Header-Shipping Detail-Product.
2060	Sales Order-Sales Order Line-Product	This is specialize 2050 for sales order.
2070	Purchase Order-Purchase Order Line-Product	This is specialize 2050 for purchase order.
2080	Closing	Many Transactions are summarized by one Time Bucket. And these summary becomes Closing.
2090	Paying or Receiving	Paying (Receiving) are related by Paying (Receiving) Date.
2100	Settlement	Closing and Paying (Receiving) are related by Period, and keep balance.
2110	Settlement for Receiving	Closing for sales. Invoice will be issued.
2120	Settlement for Paying	Closing for Purchase.
2130	Sales Order & Purchase Order (SS)	Sales Order and Purchase Order are related by same Product.
2140	Transport (SS)	Transportation from one place to another one place. Shipping, Goods Receiving, Moving between two Warehouses, etc.
2150	Direct Transport(SS)	Purchase Order will be created by sales Order. These shipping and Receiving will be related automatically.
2160	Product Information (SS)	Who supply, how to make order, etc.

2170	Daily Inventory (SS)	Balance of Input and Output for Product is summarized by each period.
2180	Settlement for Receiving & Paying(SS)	Paying and Receiving are related by one Period.
2190	Organization (SS)	Organization and Assignment by Period.
2200	Product Configuration (SS)	Set Product configuration.

6.1.3 Optional Patterns

No.	Pattern Name	Description
3010	Unit Price by Correspondent	Product Price is defined by Customer (or Supplier).
3020	Unit Price by Period	Product Price is defined by Date.
3030	Unit Price by Correspondent /Period	Product Price is defined by Customer (or Supplier) by Date.
3040	Employee-Role	Employee can have multiple roles.
3050	Employee Role Definition (SS)	Role has an object what is managed. Purchase by supplier. Sales by area. Support by Product.

6.2 Pattern Definition

6.2.1 Basic Patterns

6.2.1.1 Master & Detail

BFOP No.	1010
BFOP Name	Master & Detail
Inherited BFOP	none
Parameter	<Master>, <Detail>
Revision	Rev.2
Description	Master is referred by Detail, e.g. Area: Customer, License-Employee, Order type-Order

```
ddDetail (m:Master, d:Detail)
        pre        not m.detail -> includes(d)
        post       d.master.detail ->includes(d)
removeDetail(d:Detail)
        pre        d.master.detail -> includes(d)
        post       not m.detail -> includes(d)
```

6.2.1.2 Association

BFOP No.	1020
BFOP Name	Association
BFOP to be inherited	none
Parameter	<A>, <B>, <Association>
Revision	Rev.2
Description	One of the basic patterns. Widely applicable, for example, assignment of stock and duty and so on.

```
addAssociation(a:A, b:B, assoc:Association)
        pre        not a.association -> includes(assoc) and
                    not b.association -> includes(assoc)
        post       a.association -> includes(assoc) and
                    b.association -> includes(assoc)
removeAssociation(assoc:Association)
        pre        a.association -> includes(assoc)and
                    b.association -> includes(assoc)
        post       not a.association -> includes(assoc)and
                    not b.association -> includes(assoc)
```

6.2.1.3 Association-3

BFOP No.	1030
BFOP Name	Association-3
BFOP to be inherited	none
Parameter	<A>, <B>, <C>, <Association>
Revision	Rev.2
Description	Derived from 1020 (Association). Association is decided by relationship among 3 contents.

```
addAssociation(a:A, b:B, c:C, assoc:Association)
        pre         not a.association -> includes(assoc) and
                     not b.association -> includes(assoc) and
                     not c.association -> includes(assoc)
        post        a.association -> includes(assoc) and
                     b.association -> includes(assoc)and
                     c.association -> includes(assoc)
removeAssociation(assoc:Association)
        pre         a.association -> includes(assoc)and
                     b.association -> includes(assoc)and
                     c.association -> includes(assoc)
        post        not a.association -> includes(assoc)and
                     not b.association -> includes(assoc)and
                     not c.association -> includes(assoc)
```

6.2.1.4 Dynamic Hierarchical Structure

BFOP No.	1040
BFOP Name	Dynamic Hierarchical Structure
BFOP to be inherited	1060 DAG
Parameter	<Hierarchical Entity>, <Hierarchical Configuration>, <Configuration Effective Period>
Revision	Rev.2
Description	It is same as tree structure. Organization of company.

6.2.1.5 Directed Graph

BFOP No.	1050
BFOP Name	Directed Graph
BFOP to be inherited	none
Parameter	<Vertex>, <Edge>, <From>,<To>
Revision	Rev.2
Description	Graph structure composed of Vertex and Edge. Edge has a starting vertex and an ending vertex.

```
addEdge(from:Vertex, to:Vertex, e:Edge)
        pre         not from.out -> includes(e) and
                     not to.in -> includes(e)
        post        e.from.out -> includes(e) and
                     e.to.in -> includes(e)
removeEdge(e:Edge)
        pre         e.from.out -> includes(e)and
                     e.to.in -> includes(e)
        post        not from.out -> includes(e)and
                     not to.in -> includes(e)
```

6.2.1.6 DAG (Directed Acyclic Graph)

BFOP No.	1060
BFOP Name	DAG (Directed Acyclic Graph)
BFOP to be inherited	Directed Graph[Vertex/Vertex.Edge/Edge,FROM/FROM,TO/TO]
Parameter	<Vertex>, <Edge>, <FROM>,<TO>
Revision	Rev.2
Description	Derivation added a restriction of Acyclic to 1050 (Directed Graph). For example, children items never use their parent items on the expression of item structure.

6.2.1.7 Header & Detail

BFOP No.	1070
BFOP Name	Header & Detail
Inherited BFOP	none
Parameter	<Header>, <Detail>
Revision	Rev.2
Description	Header and Detail have a relationship of parent and children. e.g. Order and Order line, Bank and Bank branch, Division and section.

```
ddDetail (h:Header, d:Detail)
        pre         not .detail -> includes(d)
        post        d.header.detail ->includes(d)
removeDetail(d:Detail)
        pre         d.header.detail -> includes(d)
        post        not .header.detail -> includes(d)
```

6.2.1.8 Stock & Flow

BFOP No.	1080
BFOP Name	Stock & Flow
BFOP to be inherited	none
Parameter	<Source>, <Flow>, <Sink>
Revision	Rev.2
Description	A level of Stock quantity is decided by the difference of the accumulative input and output levels. Pattern expressed inventory control together with item's distribution on supply chains.

```
addFlow(i:Source, o
  pre  not i.flow -> includes(flow) and
        not o.flow -> includes(flow)
  post  flow.source.flow -> includes(flow) and
         flow.sink.flow -> includes(flow)and
         flow.source.accumlativeInputQuantity=
               flow.source.accumlativeInputQuantity@pre
                                          +flow.flowQuantity and
         flow.sink.accumlativeInputQuantity=
               flow.sink.accumlativeInputQuantity@pre
                                          +flow.flowQuantity
removeFlow(flow:Flow)
  pre  i.flow -> includes(flow)and
        o.flow -> includes(flow)
  post not low.source.flow -> includes(flow)and
        not  flow.sink.flow -> includes(flow)and
        flow.source.accumlativeInputQuantity=
        flow.source.accumlativeInputQuantity@pre
                                          -flow.flowQuantity and
        flow.sink.accumlativeInputQuantity=
                flow.sink.accumlativeInputQuantity@pre
                                          -flow.flowQuantity
```

6.2.1.9 Period

BFOP No.	1090
BFOP Name	Period
BFOP to be inherited	none
Parameter	<Period>, <From Date>, <To Date>
Revision	Rev.2
Description	Period has two types of dates, From Date and To Date, and has the difference of them as Duration. BFOP uses an alternative to Period (e.g. Period-Date) instead of Date in case Date is handled as Period. By doing so, you can easily sum from daily data to monthly and yearly. Referring the book listed below. BFOP assumes that all the instance needed beforehand exist on this pattern.

6.2.1.10 Transaction Record

BFOP No.	1110
BFOP Name	Transaction Record
BFOP to be inherited	1080 (Stack & Flow)
Parameter	<Participant>, <Transaction>, <Arbitrary Period>, <Arbitrary period Type>
Revision	Rev.2
Description	Transaction usually has two attributes. Occurrence Date and Participant. It can be connected to its Arbitrary Period through the Period Pattern of the Occurrence Date. Arbitrary Period is Period generated using Arbitrary Period Type as its rule. For example, in case "closing date is 25th every month", Period Type is defined as "from 26th to 25th of the following month" and Period has data from 032699 to 042599. A Sale transaction of 040199 is connected to its Period, then the Receiving Amount for the corresponding Arbitrary Period is decided.

```
addTransaction(o:Participant, d:Date, t:Transaction)
 pre   not o.transaction -> includes(t) and
        not d.association -> includes(t)
 post  t.participant.transaction -> includes(t) and
        t.date.transaction -> includes(t)and
        t.arbitraryPeriod.start<=t.occurenceDate<=
                                          t.arbitraryPeriod.end
removeTransaction(t:Transaction)
 pre   t.participant.transaction -> includes(t) and
        t.date.transaction -> includes(t)
 post  not o.transaction -> includes(t) and
        not d.transaction -> includes(t)
```

6.2.2 Unit Patterns

6.2.2.1 Correspondent-Transaction

BFOP No.	2010
BFOP Name	Correspondent-Transaction
BFOP to be inherited	1010 (Master & Detail)
Parameter	<Correspondent>, <Transaction>, <Payer or Receiver>
Revision	Rev.2
Description	Transaction occurs to Correspondent. Transaction means Sale Order, Purchase Order, Shipping, Returning goods, and Service Request and so on.

6.2.2.2 Product-Stock-Place

BFOP No.	2020
BFOP Name	Product-Stock-Place
BFOP to be inherited	1020 (Association)
Parameter	<Place>, <Stock Product>, <Stock>
Revision	Rev.2
Description	Sock of products are controlled for Place. Place can be replaced to Warehouse, Rack, or Lot number.

6.2.2.3 Employee Assignment

BFOP No.	2030
BFOP Name	Employee Assignment
BFOP to be inherited	1020 (Association)
Parameter	<Organization>, <Employee>
Revision	Rev.2
Description	An employee can be assigned to multiple sections.

6.2.2.4 Organization Structure

BFOP No.	2040
BFOP Name	Organization Structure
BFOP to be inherited	1040 (dynamic Hierarchical Structure)
Parameter	<Organization Entity>, <Organization Structure>
Revision	Rev.2
Description	Organization Structure is hierarchical structure. Since hierarchical structure has Period, Organization Structure on this pattern can have Effective Period. So, future changes of Organization and their tracks can be expressed.

6.2.2.5 Transaction-Transaction Line-Product

BFOP No.	2050
BFOP Name	Transaction-Transaction Line-Product
BFOP to be inherited	1010 (Master & Detail), 1020 (Header & Detail), 2010 (Correspondent-Transaction)
Parameter	<Transaction>,<TransactionLine>,<Product>, <correspondent>,<Payer or Receiver>
Revision	Rev.2
Description	This is a pattern described that a Correspondent has Transaction of multiple products. Can be used for estimate, sales order, and purchase order has no sales order connected.

6.2.2.6 Sales Order-Sales Order Line-Product

BFOP No.	2060
BFOP Name	Sales Order-Sales Order Line-Product
BFOP to be inherited	2050 (Transaction-Transaction Line-Product)
Parameter	<Product>,<Customer>,<Payer>, <Goods Receiver>
Revision	Rev.2
Description	A Customer puts an order of many products.

6.2.2.7 Purchase Order-Purchase Order Line-Product

BFOP No.	2070
BFOP Name	Purchase Order-Purchase Order Line-Product
BFOP to be inherited	2050 (Transaction-Transaction Line-Product)
Parameter	<Product>,<supplier>,<Receiver>
Revision	Rev.2
Description	Purchase Order of many products to a supplier.

6.2.2.8 Closing

BFOP No.	2080
BFOP Name	Closing
BFOP to be inherited	1100 (Transaction Record)
Parameter	<Correspondent>, <Transaction Occurrence Date>, <Closing Period>,<Closing Period Type>, <Transaction Type>,<Settlement for Receiving>
Revision	Rev.2
Description	Transaction summarized by time bucket is totaled for Transaction Type. For example, Sales in the Receiving Period can be divided into separate Receiving according to their contractual

	condition..

<participant
participant
<transaction
Transaction
[occurrenceDate /<transactionOccurenneDate>]
1100
transactionRecord
<closingPeriod
0.. n
1..1
<settlementForReceivin
-receiving Amout:Amount
0.. n
1..1
arbitraryPeriod
arbitraryPeriodType
<closingPeriodTyp
<transactionTyp
contract
default

6.2.2.9 Paying or Receiving

BFOP No.	2090
BFOP Name	Paying or Receiving
BFOP to be inherited	1100 (Transaction Record)
Parameter	<Payer or Receiver>, <Paying or Receiving>, <Paying or Receiving Date>,<Paying or Receiving Period>, <Paying or Receiving Period Type>
Revision	Rev.2
Description	Paying Or Receiving is identified its Paying Or Receiving Period by the Paying Or Receiving Date.

6.2.2.10 Settlement

BFOP No.	2100
BFOP Name	Settlement
BFOP to be inherited	2080 (Closing), 2090 (Paying or Receiving)
Parameter	<Closing Period Type>, <Paying or Receiving Period Type>, <Correspondent>,<Payer or Receiver>,<Transaction>, <Paying or Receiving>
Revision	Rev.2
Description	Closing Period is calculated by Closing Date of the Paying Condition.(e.g. closing on 20th, the end of month, 15th and the end of month, etc.) Paying Limit is calculated by Paying Date of the Paying Condition.(e.g. by the end of the next month, by 15th of the next month, by 10th of the month after next month, etc.) Paying Period (bucket) starts with the final date of the Closing Period and ends with paying Limit.

6.2.2.11 Settlement for Receiving

BFOP No.	2110
BFOP Name	Settlement for Receiving
BFOP to be inherited	2100 (Settlement)
Parameter	<Obvention>, <Sale>
Revision	Rev.2
Description	Totaling Sales on charge, it is charged to Payer. Checking the money received, Receiving Outstanding Balance is calculated..

6.2.2.12 Settlement for Paying

BFOP No.	2120
BFOP Name	Settlement for Paying
BFOP to be inherited	2100 (Settlement)
Parameter	<Settlement for Paying>
Revision	Rev.2
Description	Totaling Purchase on payment, it is paid to the Supplier. In case of transfer through bank, rate is calculated.

6.2.2.13 Sales Order & Purchase Order (SS)

BFOP No.	2130
BFOP Name	Sales Order & Purchase Order (SS)
BFOP to be inherited	2060 (Sales Order - Sales Order Line - Product) 2070 (Purchase Order - Purchase Order Line - Product)
Parameter	
Revision	Rev.2
Description	Product is referred by Sales Order and Purchase Order.

6.2.2.14 Transport (SS)

BFOP No.	2140
BFOP Name	Transport (SS)
BFOP to be inherited	1080 (Stock & Flow)
Parameter	<Transport Instruction>, <Transport>, <Transport From>, <Transport To>
Revision	Rev.2
Description	This pattern is an improvement considered logistics on 1080, Stock & Flow. Transport may be based on Transport Instruction. This Transport Instruction becomes Sales Order Line or Purchase Order Line after improving, and Stock/Shipping is connected to a transaction system.

6.2.2.15 Direct Transport (SS)

BFOP No.	2150
BFOP Name	Direct Transport (SS)
BFOP to be inherited	none
Parameter	
Revision	Rev.2
Description	Purchase Order connected to Sales Order is created. Purchase Order for many Sales Orders can be put an order. Sales Order may be divided into many Purchase Orders. Purchase Order connected to the Sales Order can be delivered directly.

6.2.2.16 Product Information (SS)

BFOP No.	2160
BFOP Name	Product Information (SS)
BFOP to be inherited	1020 (Association)
Parameter	<Product Information>, <Participant>
Revision	Rev.2
Description	Can store different information for each products, e.g. Supplier and its leadtime, Customer and its productNo, etc.

6.2.2.17 Daily Inventory (SS)

BFOP No.	2170
BFOP Name	Daily Inventory (SS)
BFOP to be inherited	2020 (Product-Stock-Place) 1090 (Period)
Parameter	<Daily Inventory>, <Daily Period>
Revision	Rev.2
Description	This is a pattern which is modeled on DailyInventory you often find on Sales or Production management system. Inventory expressed here is “Inventory of a product in a warehouse on a day”. Because Daily Inventory has no Date but Daily Period replaced by Period as an attribute, it’s easy to sum to monthly and yearly inventory.

6.2.2.18 Settlement for Receiving & Paying (SS)

BFOP No.	2180
BFOP Name	Settlement for Receiving & Paying (SS)
BFOP to be inherited	2110 (Settlement for Receiving), 2120 (Settlement for Paying)
Parameter	
Revision	Rev.2
Description	Because SettlementForReceiving and SettlementForPaying internally have prearranged executive data, cash flow is knowable.

6.2.2.19 Organization (SS)

BFOP No.	2190
BFOP Name	Organization (SS)
BFOP to be inherited	2030 (Employee Assignment), 2040 (Organization Structure)
Parameter	<Organization>
Revision	Rev.2
Description	Organization decides a management structure by connecting to other sub-systems.

6.2.2.20 Product Configuration (SS)

BFOP No.	2200
BFOP Name	Product Configuration (SS)
BFOP to be inherited	1060 (DAG)
Parameter	<Product>, <Product Configuration>,<Parent Product>, <Children Product>
Revision	Rev.2
Description	This is one of expressions for Product Configuration, and does not have Valid Period. So, it is impossible to beforehand store the configurational changes in future.

```
productStructure(product:Product)
     post  result = a set of productConfiguration
productExplosion(product:Product)
     post  result = a set of pair of product at the bottom level of
                product  structure  and  its  required  quantity,
                obtained  by  recursively  applying  the  product
                structure method.
```

6.2.3 Optional Patterns

6.2.3.1 Unit Price by Correspondent

BFOP No.	3010
BFOP Name	Unit Price by Correspondent
BFOP to be inherited	1020 (Association)
Parameter	
Revision	Rev.2
Description	Can create Unit Prices for each Correspondent.

6.2.3.2 Unit Price by Period

BFOP No.	3020
BFOP Name	Unit Price by Period
BFOP to be inherited	1020 (Association), 1090 (Period)
Parameter	
Revision	Rev.2
Description	Can create Unit Price with its valid period.

6.2.3.3 Unit Price by Correspondent and Period

BFOP No.	3030
BFOP Name	Unit Price by Correspondent and Period
BFOP to be inherited	1030 (Association-3), 1090 (Period)
Parameter	
Revision	Rev.2
Description	Can create Unit Prices for each Correspondent with their Effective Period.

6.2.3.4 Employee Role

BFOP No.	3040
BFOP Name	Employee Role
BFOP to be inherited	1020 (Association)
Parameter	<Assignment>, <Employee Role>
Revision	Rev.2
Description	Employee can be have multiple roles.

6.2.3.5 Employee Role Definition (SS)

BFOP No.	3050
BFOP Name	Employee Role Definition (SS)
BFOP to be inherited	1020 (Association)
Parameter	<Employee Role>, <Assignment>, <Employee Role Target>
Revision	Rev.2
Description	Role has an object what is managed. Purchase by supplier. Sales by area. Support by Product..

6.3 Overall Package Structure

4 PRODUCTS
A001
B001
3 BASIC MODEL
3050
Employee Role
Definition (SS)
3030
Price by
Correspondent/Period
B: Sales/Purchase/Stock
(SS)
3040
Employee Role
3020
Price by Period
A: Sales/Purchase (SS)
D:
Sales/Purchase/Stock-
Set Product (SS)
3010
Price by Correspondent
C: Sales/Purchase Set
Product (SS)
OPTION
2200
Product Configuration
(SS)
2130
Sales Order & Purchase
Order (SS)
2190
Organization (SS)
2160
Product Information
2170
Daily Inventory
2180
Settlement for
Receiving & Paying
2150
Direct Transport(SS)
2040
Organization Structure
2140
Transport (SS)
2070
Purchase Order-Purchase
Order Line-Product
2030
Employee Assignment
2110
Settlement for
Receiving
2060
Sales Order-Sales
Order Line-Product
2120
Settlement for Paying
2020
Product-Stock-Place
2050
Transaction-Transaction
Line-Product
2100
Settlement
2010
Correspondent-
Transaction
2090
Paying or Receiving
2080
Closing
2 UNIT
1 BASIC
1040
Dynamic Hierarchical
Structure
1030
Association-3
1070
Header & Detail
1060
DAG (Directed Acyclic
Graph)
1020
Association
1080
Stock & Flow
1050
Directed Graph
1090
Period
1010
Master & Detail
1110
Transaction Record

6.4 Sample Object Model for BFOP

References

[Bap ...bat, S. *Object-Oriented Networks Models for Arc... ...erations, and Management*, PT... 994.

[Blaha 98] Michael Blaha, William Premerl... *...ed Modeling and Design for Database Applications*. Prentice Hall, Uppe... River, N.J., 1998.

[Booch 91] Grady Booch. *Object-Oriented ...d Design with Applications, 1st ed.* Benjamin/Cu... ...rood Cit... 1991.

[Booch 94] Grady Booch. *Object-Oriented Analysis and Design with Applications, 2nd ed.* Benjamin/Cummings, Redwood City, Calif., 1994.

[Booch 96a] Grady Booch. *Obje... Managing the Object-Oriented*

Project. Addison-Wesley, Menlo Park, Calif., 1996.

[Booch 96b] Grady Booch. *Best of Booch: Designing Strategies for Object Technology*. SIGS Books, New York, N.Y., 1996.

[Booch 99] Grady Booch, James Rumbaugh, Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, Mass., 1999.

[Buschmann 96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. *Pattern-Oriented Software Architecture; A system of Patterns*. Wiley, Chichester, U.K., 1996.

[Coad 91] Peter Coad, Edward Yourdon. *Object-Oriented Analysis, 2nd ed*. Yourdon Press, Englewood Cliffs, N.J., 1991.

[Coleman 94] Derek Coleman, Patrick Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes, Paul Jeremaes. *Object-Oriented Development: The Fusion Method*. Prentice Hall, Englewood Cliffs, N.J., 1994.

[Cox 86] Brad J. Cox. *Object-Oriented Programming: An Evolutionary Approach*. Addison-Wesley, Reading, Mass., 1986.

[DeRemer 76] Frank DeRemer and Hans H. Kron. *Programming-in-the-Large versus Programming-in-the-Small*. IEEE Trans. on Software Engineering, SE-2(2):80-86, June 1976.

[D'Souza 99] Desmond F. D'Souza, Alan Cameron Wills. *Objects, Components, and frameworks with UML: The Catalysis Approach*. Reading, Mass., Addison-Wesley, 1999.

[Fowler 97] Martin Fowler. M. *Analysis Patterns: Reusable Object Models*. Reading, Mass., Addison-Wesley, 1997.

[Fowler&Scott 97] Martin Fowler, Kendall Scott. *UML Distilled*. Reading, Mass., Addison-Wesley, 1997.

[Gamma 95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Mass., 1995.

[Goldberg 83] Adele Goldberg, David Robson. *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley, Reading, Mass., 1983.

[Gosling 96] James Gosling, Bill Joy, and Guy Steele. *The Java™ Language Specification version 1.0*. Addison Wesley, 1996.

[Harel 98] David Harel, Michal Politi. *Modeling Reactive Systems with Statecharts: The STATEMATE Approach*. McGraw-Hill, New York N.Y., 1998.

[Jacobson 92] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Overgaard, *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, Wokingham, England, 1992.

[Jacobson 95] Ivar Jacobson, Maria Ericsson, Agneta Jacobson. *The Object Advantage: Business Process Reengineering with Object Technology*. Addison-Wesley, Wokingham, England, 1995.

[Jacobson 97] Ivar Jacobson, Maria Griss, Patrik Jonsson. *Software Reuse: Architecture, Process and Organization for Business Success*. Addison-Wesley, Harlow, England, 1997.

[Jacobson 99] Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unified Software Development Process*. Addison-Wesley, Reading, Mass., 1999.

[Kilov 96] Kilov, Haim and Ross, James. *Information Modeling: An Object-Oriented Approach*, Prentice-Hall, 1994. ISBN 0-13-083033-X.

[Martin 92] James Martin, James Odell. *Object-Oriented Analysis and Design*.

Prentice hall, Englewood Cliffs, N.J., 1992.

[Meyer 88] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, New York, N.Y., 1988.

[Reenskaug 96] Reenskaug et. al. *Working with Objects, the OORAM Software Engineering Method*. Manning 1996. ISBN1-884777-10.4

[Rumbaugh 91] James Rumbaugh, Michael Blaha, William Pemerlani, Frederick Eddy, William Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, N.J., 1991.

[Rumbaugh 96] James Rumbaugh. *OMT Insights: Perspectives on Modeling from the Journal of Object-Oriented Technology*. SIGS Books, New York, N.Y., 1996.

[Rumbaugh 99] James Rumbaugh, Ivar Jacobson, Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Reading, Mass., 1999.

[Selic 94] Bran Selic, Garth Gullekson, Paul T. Ward. *Real-Time Object-Oriented Modeling*. Wiley, New York, N.Y., 1994.

[Shlaer 88] Sally Shlaer, Stephen J. Mellor. *Object-Oriented Systems Analysis: Modeling the world in Data*. Yourdon Press, Englewood Cliffs, N.j., 1988.

[Shlaer 92] Sally Shlaer, Stephen J. Mellor. *Object Lifecycles: Modeling the world in States*. Yourdon Press, Englewood Cliffs N.J., 1992.

[Ward 85] Paul Ward, Stephen J. Mellor. *Structured Development for Real-Time Systems: Introduction and Tools*. Yourdon Press, Englewood Cliffs, N.J., 1985.

[Warmer 99] Jos B. Warmer, Anneke G. Kleppe. *The Object Constraint Language: Percise Modeling with UML*. Addison-Wesley, Reading, Mass., 1999.

[Wirfs-Brock 90] Rebecca Wirfs-Brock, Brian Wilkerson, Lauren Wiener. *Designing Object-Oriented Software*. Prentice Hall, Englewood Cliffs, N.J., 1990.

[Yourdon 79] Edward Yourdon, Larry L. Constantine. *Structured Design: Fundamentaals of a Discipline of Computer Program and Systems Design*. Yourdon Press, Englewood Cliffs, N.J., 1979.

[GRM] *ISO General Relationship Model*. ISO 10165-7 IEC

[ISO Latin-1] ISO 8859.1

[ISO/IEC FDIS 11179-1] *Information technology-Specification and Standardization of data elements Part 1: Framework for the specification and standardization of data elements*. ISO/IEC FDIS 11179-1

[ISO/IEC FDIS 11179-2] *Information technology-Specification and Standardization of data elements Part 2: Classification of data elements*. ISO/IEC FDIS 11179-2

[ISO/IEC 11179-3] *Information technology- Specification and Standardization of data elements Part3: Basic Attributes of Data Elements* ISO/IEC IS 11179-3, 1994.

[ISO/IEC 11179-4] *Information technology- Specification and Standardization of data elements Part4: Rules and guidelines for the formulation of data definitions*. ISO/IEC IS 11179-4, 1995.

[ISO/IEC 11179-5] *Information technology- Specification and Standardization of data elements Part5: Naming and identification principles for data elements*. ISO/IEC IS 11179-5, 1995.

[ISO/IEC 11179- *Information technology- Specification and Standardization of data*

6] *elements Part6: Registration principles for data elements.* ISO/IEC IS 11179-6, 1997.

[ISO/IEC FDTR 15452] *Information technology – Specification of data value domains.* ISO/IEC FDTR 15452

[MOF] *OMG Meta-Object facility specification.* OMG Document ad/97-08-14

[ODMG 93] Cattell, R.G.G., editor. *The Object Database Standard: ODMG-93.* Morgan Kaufmann Publishers, 1994.

[OMG Business Object Facility RFP] Common Facilities RFP-4, "*Common Business Objects and Business Object Facility*", OMG TC Document cf/96-01-04.

[OMG MOF RFP] Common Facilities RFP-5, "*Meta-Object Facility RFP*", OMG TC Document cf/96-05-02.

[OMG OA&D RFP] Analysis and Design RFP-1, "*Object Analysis & Design*", Rev. 1, June 1996. OMG TC Document ad/96-05-01.

[OMG OM] "*Object Models*", Draft 0.3, January 1995. OMG TC Document 94-01-13.

[OMG OMA] *Object Management Architecture Guide,* Revision 3.0.

[OMG CORBA] *The Common Object Request Broker: Architecture and Specification,* Revision 2.0, July 1995.

[OMG 95-01-02] *Common Facilities Architecture,* January 1995. OMG TC Document 95-1-2

[OMG 95-01-47] *Object Services Architecture*, Revision 1.1, January 1995. OMG TC Document 95-1-47

[OMG 95-12-05] *Systems Management*, Revision 2.0, December 6, 1995. OMG TC Documents 95-12-02 through 95-12-06.

[OMG 92-08-05] *Object Services Roadmap*, October, 1992. OMG TC Document 92-8-5

[OMG 95-03-31] *IDL Type Extensions RFP,* March 1995. OMG TC Document 95-1-35.

[OMG 95-03-31] *CORBAservices: Common Object Services Specification*, March 31,1995. OMG TC Document 95-03-31

[OMG 95-04-01] *BOMSIG white paper,* OMG TC Document 95-4-1

[OMG cf/96-05-01] *IBM Corporation. Common Facilities RFP2 Submission: Internationalization, Time Operations and Related Facilities.* May 6, 1995. OMG TC Document cf/96-05-01.

[OMG 99-03-10] Business Object Initiative RFP1 (OA&DTF RFP-6), "*UML Profile for Enterprise Distributed Object Computing Request for Proposal,*"

[OMG 99-03-11] Business Object Initiative RFP 2 (OA&DTF RFP-7), "*A UML Profile for CORBA Request for Proposal,*" OMG Document: ad/99-03-11

[OMG 99-08-01] *UML Profile for CORBA Joint Initial Submission.* Data Access Corporation, Genesis Development Corporation, Telelogic AB, UBS AG. Version 1.0, July 30, 1999. OMG Document ad/99-08-01.

[OMG 99-10-09] *OMG Object Analysis & Design Task Force (OA&DTF RFP-6) Submission, UML Profile for Enterprise Distributed Object Computing, Business Object Component Architecture (BOCA) Initial Proposal,* Data Access Technologies, Inc. and Electronic Data Systems(EDS), October 21, 1999, OMG Document ad/99-10-09

[RM-ODP] *Reference Model of Open Distributed Processing (RM-ODP),* ISO/IEC 10746.

Business Objects, Oliver Sims, McGraw-Hill, 0-07-707957-4

Object Advantage, Ivar Jacobson, Addison-Wesley, 0-201-42289-1

The Object-Oriented Enterprise, Rob Mattison, McGraw Hill

Business Engineering with object technology, David Taylor, Wiley,

1995

[SEMATECH CIM] *Computer Integrated Manufacturing (CIM) Application Framework Specification 1.2,* March 31,1995. SEMATECH. Technology Transfer 93061697E-ENG

[UML 98] *Unified Modeling Language Specification*. Object management Group, Framingham, Mass., 1998. Internet:www.omg.org.