ISO/IEC JTC 1/SC 32 N 0453

Date: 2000-02-15

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)
Administered by Pacific Northwest National Laboratory on behalf of ANSI

DOCUMENT TYPE	Other Document (Open)
TITLE	Framework for Classification and Identification of Enterprise Objects
SOURCE	Japan National Body (Hajime Horiuchi)
PROJECT NUMBER	
STATUS	Interested parties are invited to participate. Approved for Study Period by SC 32 at the Santa Fe meeting 2000-01-28 Resolution 8.
REFERENCES	
ACTION ID.	FYI
REQUESTED ACTION	Interested parties should contact the Japan National Body
DUE DATE	
Number of Pages	70
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 901 D Street, SW., Suite 900, Washington, DC, 20024-2115, United States of America

Telephone: +1 703 575 2114; Facsimile; +1 703 681 9180; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite http://bwonotes5.wdc.pnl.gov/SC32/JTC1SC32.nsf

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

SC32/WG2-SAF025

ISO/IEC JTC 1/SC32/ WG2 SAF025

Date: 2000/01/27

Framework for Classification and Identification of Enterprise Objects

---

Working Draft

Document type:
Document subtype:
Document stage:
Document language: E

Table of Contents

1 Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liasion with ISO and IEC, also take part in the work.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part3.

2 Introduction

This NWI proposal is intending to initiate a new standardization activity which focuses on the common view to the metadata for categorizing and structuring enterprise business objects, extending ISO/IEC 11179 Specifications.

The enterprise objects which consist of both information and process elements, should be used, not only in the development efforts of a particular organization, but also, in the interchanging and sharing them among different organizations by the means such as the Electronic Commerce (EC), Electronic Data Interchange (EDI).

The major issues of the standardization are as follows:

a) reference model of enterprise object architecture
b) rules and guidelines for categorizing atomic objects
c) rules and guidelines for categorizing business function objects
d) rules and guidelines for categorizing business function object patterns
e) rules and guidelines for the formulation of basic business function object patterns
f) naming and identifying principles for enterprise objects.
g) rules and guidelines for registering enterprise objects.

PURPOSE AND JUSTIFICATION

This proposal intends to establish a common reference model of the enterprise object architecture relying upon the object concept which enables consolidation of both data concept and software concept.

Major objectives of the reference model of the architecture:

a) to support establishment of common technical infrastructure for component based system.
b) to support promotion of the reusing and the distribution of enterprise business object component.
c)to support promotion of the standardization of undefinable atomic objects.
d)to support promotion of the standardization of basic enterprise business process model and business object pattern and framework

To cope with current trends on IT area, there must be a common infrastructure which enables effective and consistent collaborations among different organizations through the internet.

In the software development domain, the component based development manner became popular to built enterprise information systems with the standard fashion by standard components, to cope with effective collaboration with others in the changeable business environment.

Information system in a particular organization, should be developed with standard components which were derived by materializing predefined object patterns or frameworks, with the component assembling manner or purchase & plug-in manner.

In realizing those manner, there must be common platforms, standard component interface specifications and exchanging protocols. However, there must be also, a common architecture which enables all of system integrators, software venders, information contents venders, and end users, locate their products in the term of the business semantics.

In the EC area, consistent business collaboration among BtoB, BtoC,CtoC, have to be proceeded with common business semantics which are necessary to create business information and related software processes. There must be a common schema which enables consistent implementation of software processes triggered by the standard messages, specified by such as EDFACT. So far, no standard were specified for the business software process.

However, before specifying those standard, there must be common view for enterprise object architecture, which consist of both data aspect and process aspect of the enterprise and their collaborations.

The benefits of the reference model of the architecture :

The architecture could be useful in both information system constructions and sharing & exchanging software components. However, following aspect of the architecture should be reinforced.

- Provide common bases for exchanging information contents among different domains and different organizations across countries.
- Provide common bases for effective evaluation mechanism of proprietary components in both software and information content, by asking for each components to declare their ingredients with specifying standard elements.

- Provide common bases for evaluating business performance of the enterprise.
- Provide common bases for effective investigation of business modeling and its change which might cause information system changes.

Those benefits are mainly brought by providing common bases for both business peoples and IT peoples.

3 Scope

This proposal is an extension to the ISO/IEC 11179 Specifications and intend to standardize metadata for categorizing and structuring enterprise business objects. The scope of the proposal is focused on the standard common architecture which covers enterprise object for Electronic Commerce (EC), Electronic Data Interchange (EDI) and various business information systems in enterprises.

In this proposed standard, elements are classified into several conceptual layers: atomic object, enterprise function object, enterprise function object pattern and framework.

This proposal intend to specify a reference model of enterprise business object component architecture based on those conceptual layers, categorized elements and required attributes for registry in order to identify useful and reusable business object components. Actual conformant business object components derived from materializing a pattern or framework may be provided by software venders and built up into a specific system environment by system integrators.

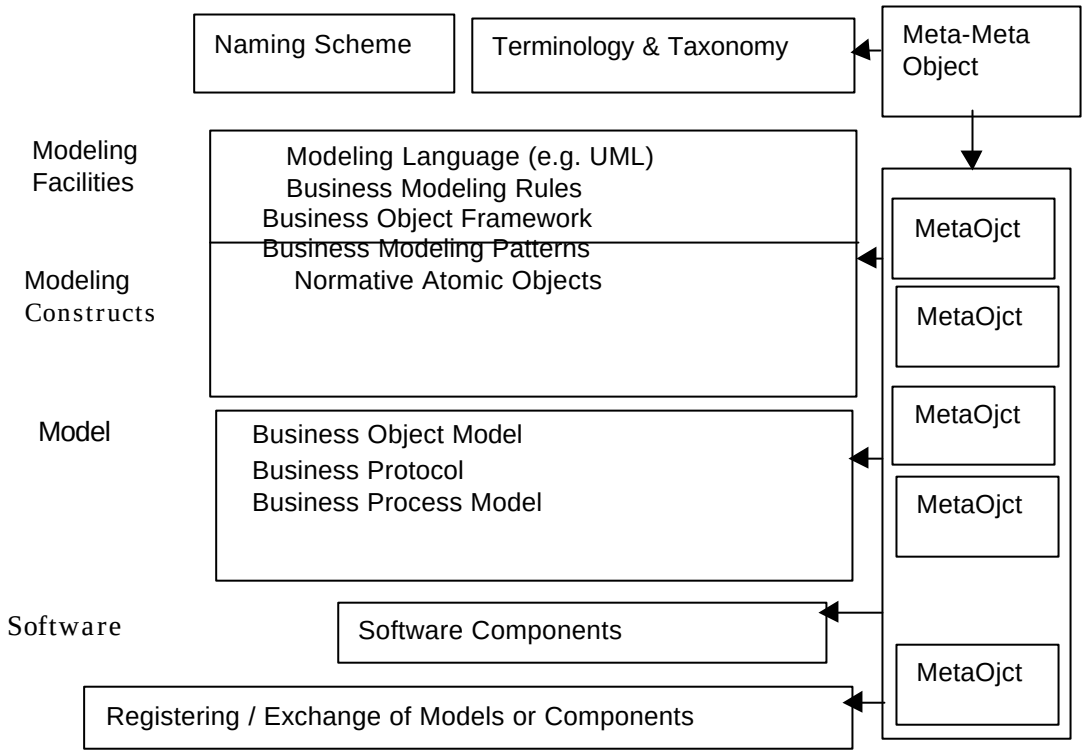
The major issues of the standardization are as follows:

a) reference model of component based enterprise business object architecture
b) rules and guidelines for categorizing atomic objects
c) rules and guidelines for categorizing enterprise business function objects
d) rules and guidelines for categorizing enterprise business function object patterns
e) rules and guidelines for the formulation of basic enterprise business function object patterns
f) naming and identifying principles for elements of enterprise business objects.
g) rules and guidelines for registering elements of enterprise business objects.

Overall architecture of the Framework of meta model

The meta model as the framework have to be consisted of following contents.

- Meta object for **Naming scheme**
- Meta object for **Terminologies and Taxonomy scheme**
- Meta object for **Identification scheme of various Business Objects**
- Meta object for **Modeling Facilities**
- Meta object for **Modeling Constructs**
- Normative **Atomic Objects**
- Normative **Value Domain**
- Normative **Modeling Patterns as profiles**
- Meta object for **Registering**



4 Conformance

<TBD>

5 Normative reference(s)

The following standards contain provisions which, through reference in the text, constitute provisions for this Part of the International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Part of the International Standard are encouraged to investigate the possibility of applying the most recent editions of standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standars.

ISO TR 9007:1987 Information processing systems –Concepts and terminology for the conceptual schema and the information

ISO 704:1987, Principles and methods of terminology

ISO 2382-4:1987, Information processing systems –Vocabulary

ISO Standards Handbook 10, Data Processing –Vocabulary, 1982

ISO 1087:1990, Terminology –Vocabulary

ISO 10241:1992, International terminology standards – preparation and layout

ISO/IEC FDIS 11179-1, Information technology –Specification and Standardization of elements Part 1:Framework for the specification and standardization of elements

ISO/IEC FDIS 11179-2, Information technology –Specification and Standardization of elements Part 2: Classification of elements

ISO/IEC 11179-3, Information technology – Specification and standardization of elements Part3: Basic attributes of elements components or Metadata registry structure, administration and content

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

ISO/IEC 11179-5:1995, Information technology –Specification and Standardization of elements Part 5: naming and identification principles for elements

ISO/IEC 11179-6:1997, Information technology – Specification and Standardization of elements Part 6: Registration of elements

ISO 11404: 1995, Information technology – Language-independent Datatypes

ISO/IEC FDTR 15452, Information technology – Specification of data value domains

6 Definition(s)

<The precise definition of following terminologies should be provided.>

administered component:
atomic object
attribute:
attribute value:
basic attribute:
concept:
conceptual data model:
data:
data element concept:
data element: A unit of data that in a certain context is considered indivisible.
data element dictionary:
data model:
domain specific object pattern:
enterprise object
domain specific entity object
domain specific plug-in object
domain process object
function object pattern
unction object framework
metadata registry.
Intity:
Identifier:
metadata:
object class:
permissible value:
property:
registration authority:
relationship:
schema:
shareable data:
value domain:

7 Classification

One of the purposes of this NWI is establishing a Common schema for classifying meta object in the common metamodel.

To cope with current requirements in the IT and business trends, meta objects in the common meta model might be classified as followed. The standardization on those classification have to be obtained.

In Fig.2, a framework of the meta model is illustrated. In this framework,, Meta object hierarchy, and meta objects that represent normative modeling facilities and modeling constructs were shown.

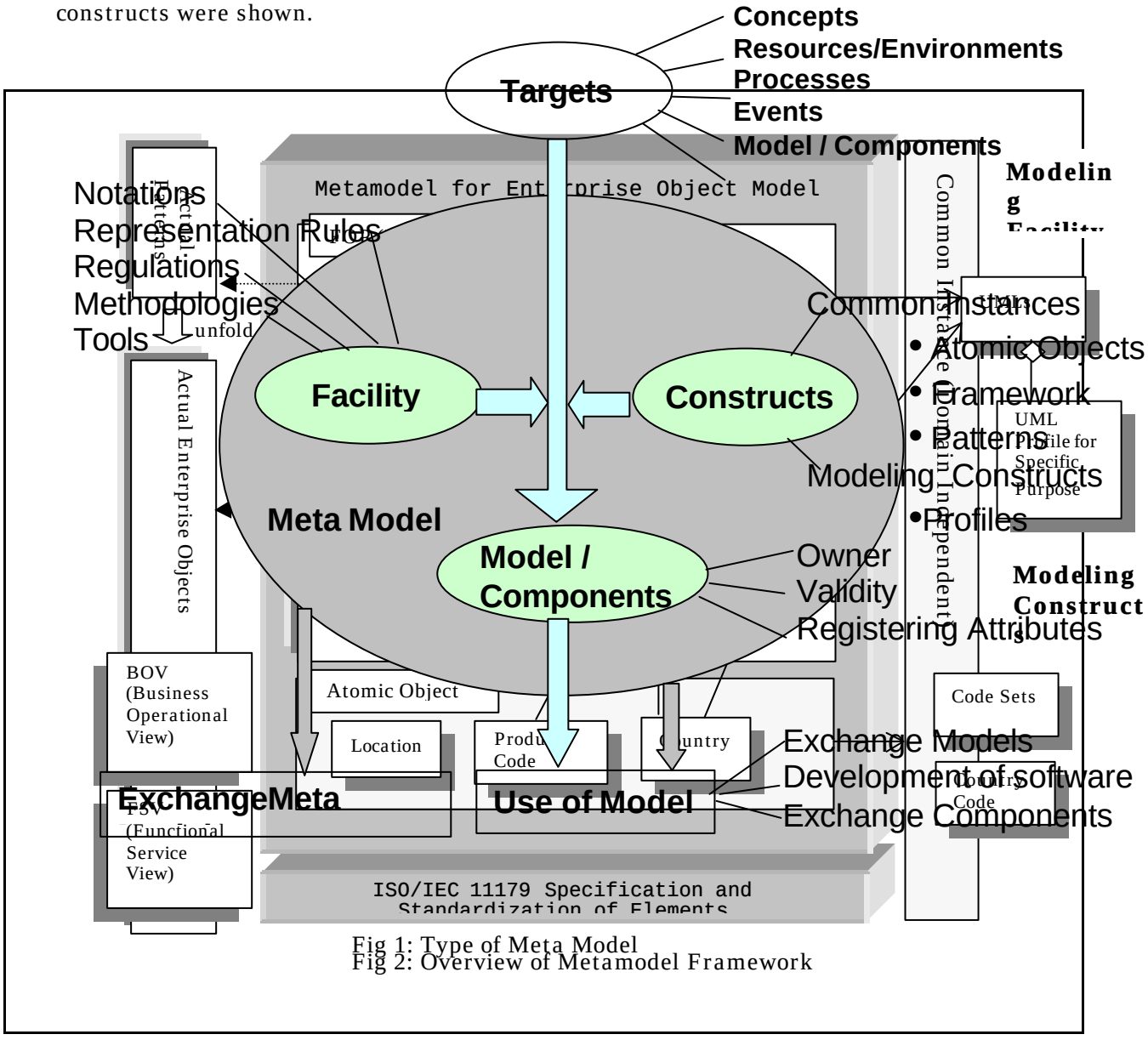


Fig 1: Type of Meta Model
Fig 2: Overview of Metamodel Framework

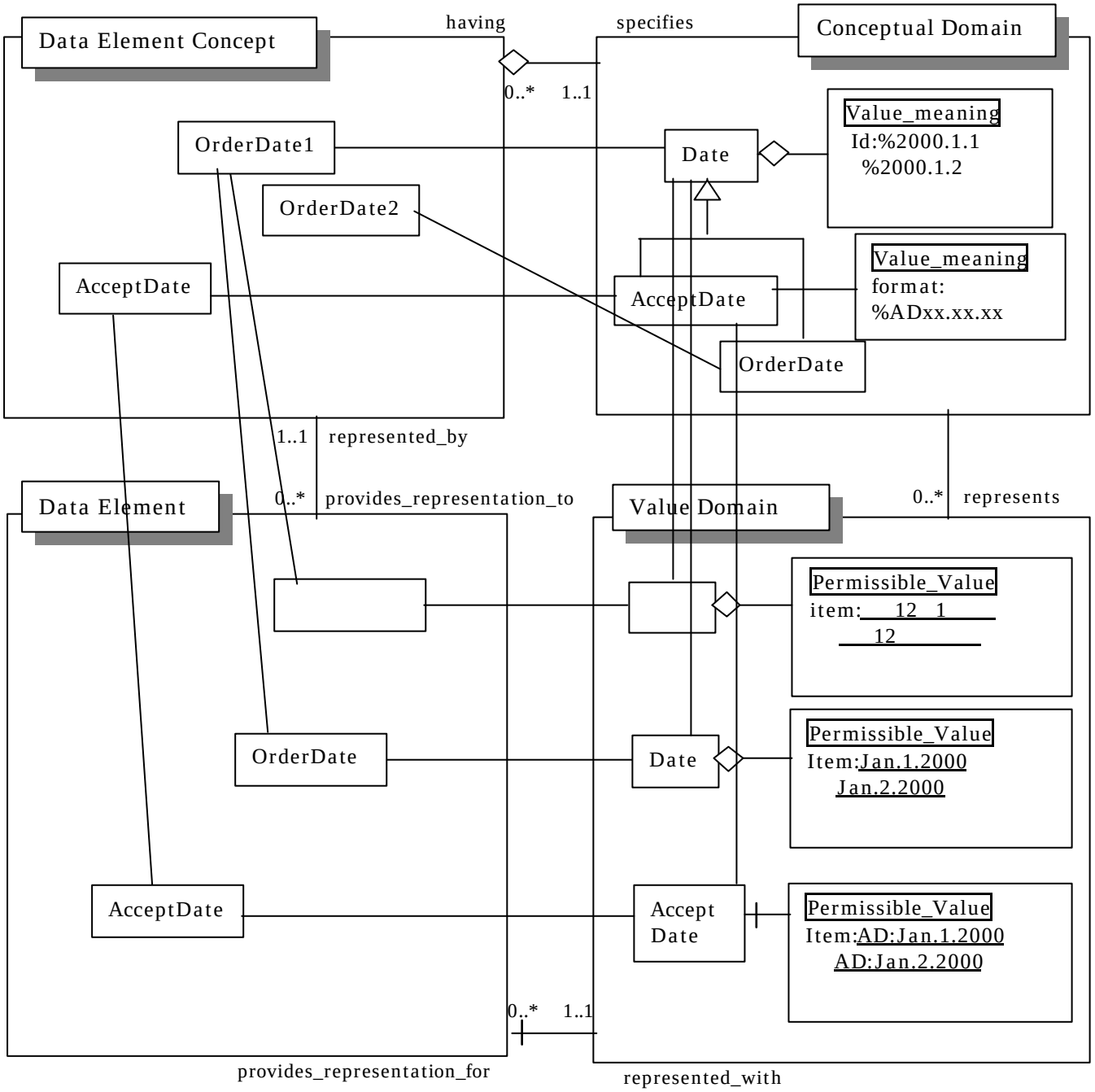
8 Identification

It is indispensable to provide a common schema which enable to global identification of various enterprise business objects, to cope with current EC, EDI environments.

The framework should address those requirements preparing a common identification mechanism which consist of ;

- Naming scheme
- Terminologies and Taxonomy scheme
- Encoded value domain for Business scheme
  - ✧ Identification of business enterprise scheme
  - ✧ Identification of business products scheme
  - ✧ etc

Those schema should be established in the meta model, facilitating the atomic object mechanism.


Data Element Concept
having
specifies
Conceptual Domain
0..*
1..1
Value_meaning
Id:%2000.1.1
%2000.1.2
OrderDate1
Date
OrderDate2
AcceptDate
AcceptDate
Value_meaning
format:
%ADxx.xx.xx
OrderDate
1..1
represented_by
Data Element
0..*
provides_representation_to
Value Domain
0..*
represents
Permissible_Value
item: 12 1
12
OrderDate
Date
Permissible_Value
Item:Jan.1.2000
Jan.2.2000
AcceptDate
Accept
Date
Permissible_Value
Item:AD:Jan.1.2000
AD:Jan.2.2000
0..*
1..1
provides_representation_for
represented_with

Annex A (Informative or TR) Concept of the Atomic Objects

Definition of the Atomic Objects

A normative set of atomic object should provide the bases for defining more complex objects which represent concepts in a particular domain, in order to establish share-ability of those objects.

The concept of the atomic objects could be defined, in the term of atomicity, as followed.

[1] Non Decomposable:

An atomic object is an elementary object which can not be decomposable any more. Accordingly, it has no structure in it.

[2] Un-definable(Needless to Define) concept:

The atomicity, also can be defined by it's nature of commonality against those, who use them. An atomic object can be recognized among those peoples or organizations without providing any definitions on its.

[3] Immutable:

Immutable means “no change” on its contents or values. An atomic object must be immutable, if contents of it have changed it become different object. Although atomic Objects have types (classes) and instances, instances of an atomic object type should not be changed.

[4] Self-contained & Independent object

An atomic object is a independent object which has no relationship with other object. Also, has no lifecycle process (CRUD) activated by events that should be caused on the relationship of more than two objects.

However, an atomic object has own constraints that govern its consistency or completeness in itself.

[5] Classification:

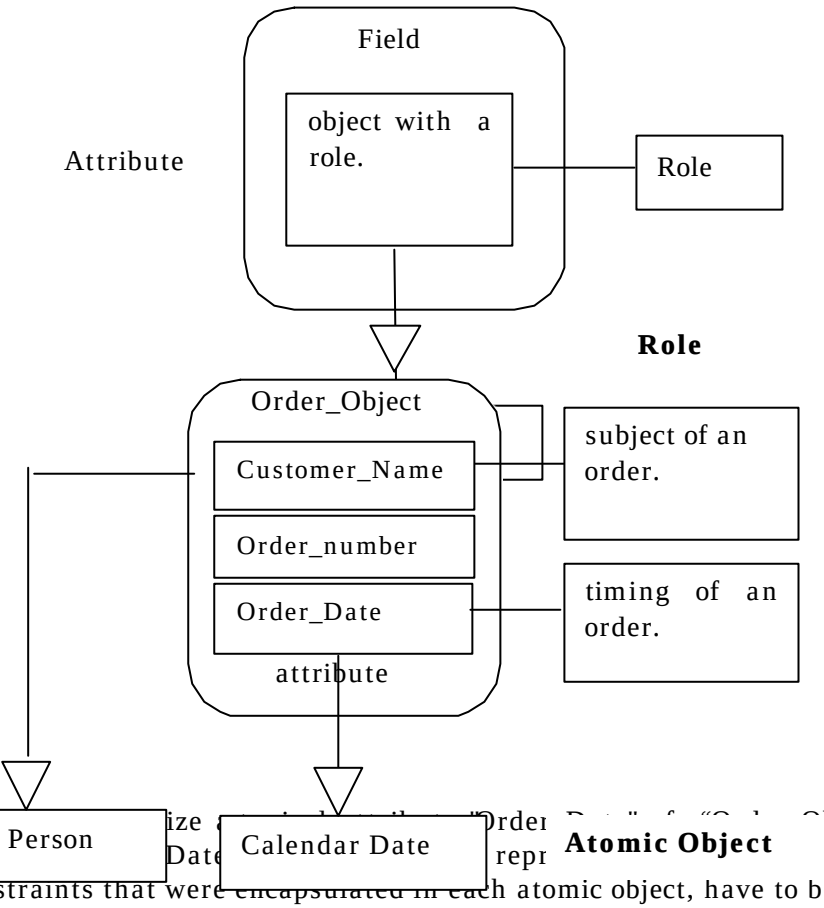
It might be allowed to define categories on a set of atomic objects. So, the specialization might be allowed on them. Although, no aggregation should not

allowed on them. It might be allowed to provide couple of attribute to an atomic object, in order to enable to specify constraints using those attributes, instead of using the property concept.

Use of Atomic Object

Atomic objects prepared for providing elementary object to be aggregated into other structured and attributed objects as ingredients. So, atomic objects have to be defined by investigating numerous object instances, distinguishing attributes of those structured and attributed objects from their values.

To use atomic objects effectively, an attribute should be recognized as a Role of an atomic object. The Role is a specific pattern which guides an aggregation manner.



For exa ize , an atomic o Date repr , " Order timing". Constraints that were encapsulated in each atomic object, have to be inherited by Structured objects.

For instance, constraints which should be encapsulated in AtomicObject "Calendar Date" are followings..

	[Form Constraint]	[Value Constraint]
●	YEAR= 4figures	2000
●	MONTH= 2figures	1 to 12

- DAY= 2figures 1 to 31

Those constraints should be inherited by object, such as “Order” to represent an attribute “OrderDate”. However, attribute “OrderDate” has own unique constraints , such as “Business day”, adding to inherited constraints.

Advantages of atomic object standardization

The advantages of the enumerating and the defining limited numbers of atomic objects as a standard, could be defined as followed.

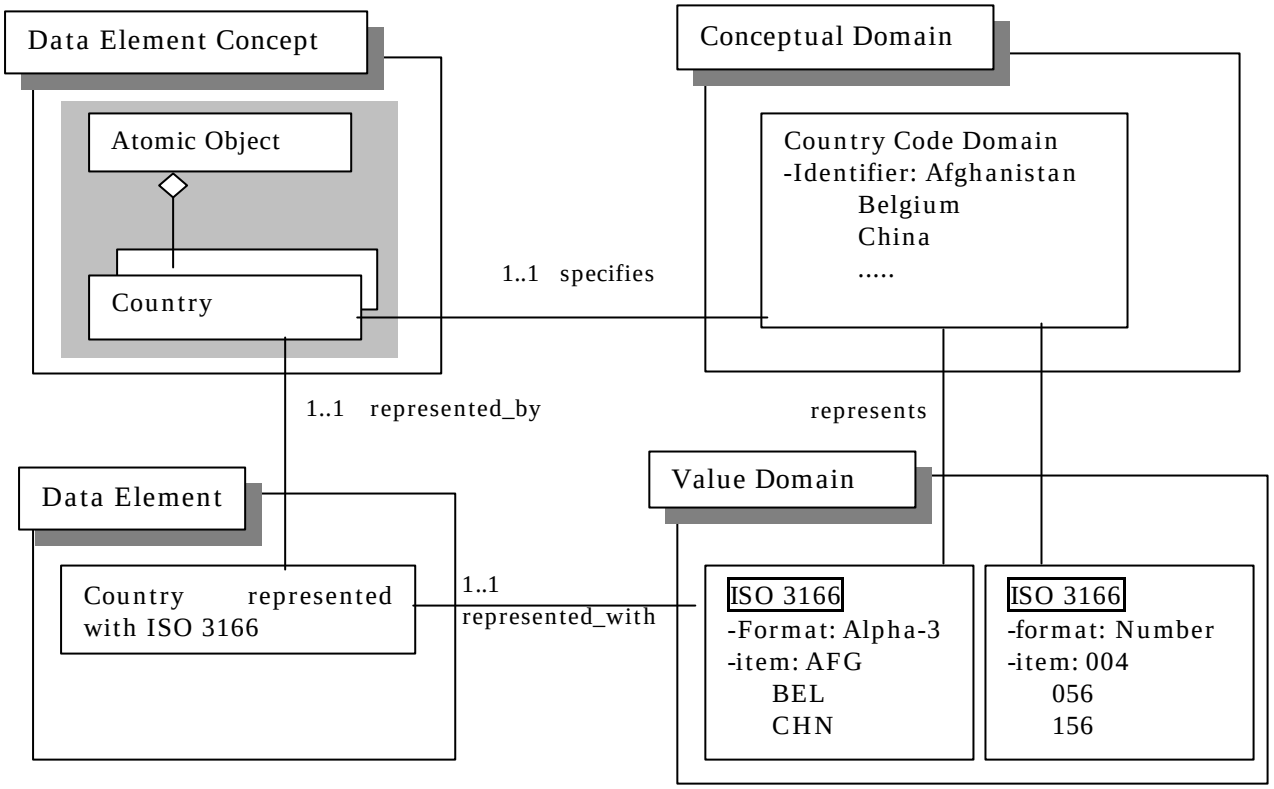
[1] Object independence

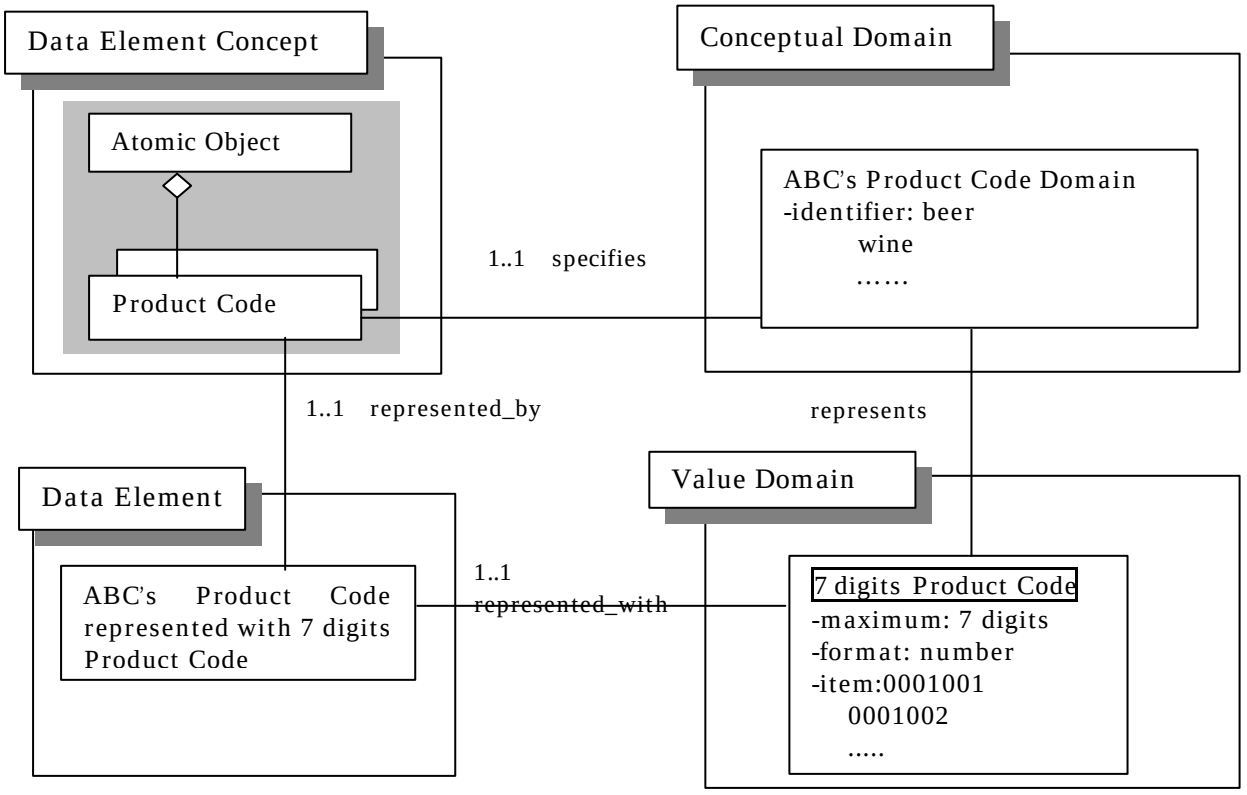
It could be possible to avoid duplications, which might be brought by defining and implementing a particular object in different objects, as a different object. It implies that by the reusing a same atomic object, it become possible to avoid duplication of processes.

[2] Increase Exchangeability and reusability

In order to share an object with same meaning, among different peoples or organizations, there must be common bases for defining a more complex object, which have to be aggregated with normative instances as contents.

[3] Implementation independence


Data Element Concept
Atomic Object
Country
1..1 specifies
Conceptual Domain
Country Code Domain
-Identifier: Afghanistan
Belgium
China
.....
1..1 represented_by
represents
Data Element
Country represented with ISO 3166
1..1
represented_with
Value Domain
ISO 3166
-Format: Alpha-3
-item: AFG
BEL
CHN
ISO 3166
-format: Number
-item: 004
056
156


Data Element Concept
Atomic Object
Product Code
1..1 specifies
Conceptual Domain
ABC's Product Code Domain
-identifier: beer
wine
……
1..1 represented_by
represents
Data Element
ABC's Product Code represented with 7 digits Product Code
1..1
represented_with
Value Domain
7 digits Product Code
-maximum: 7 digits
-format: number
-item:0001001
0001002
.....

Componet_Identifier/International_Code_Designator	1	2	3	4	5
Componet_Identifier/organization_identifier					
Componet_Identifier/organization_part_identifier(OPI)					
Componet_Identifier/OPI_source					
Componet_Identifier/data_identifier					
Componet_Identifier/version	1.0				

Administration_Identification/identifier:Component_Identifier				
Administration_Identification/record:Administration_Record				

Description/definition				

Designation/name				

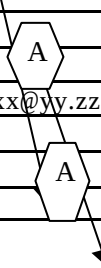
Context/description			
Context/language	Japanese		
Context/label			
Context/version	1.0		

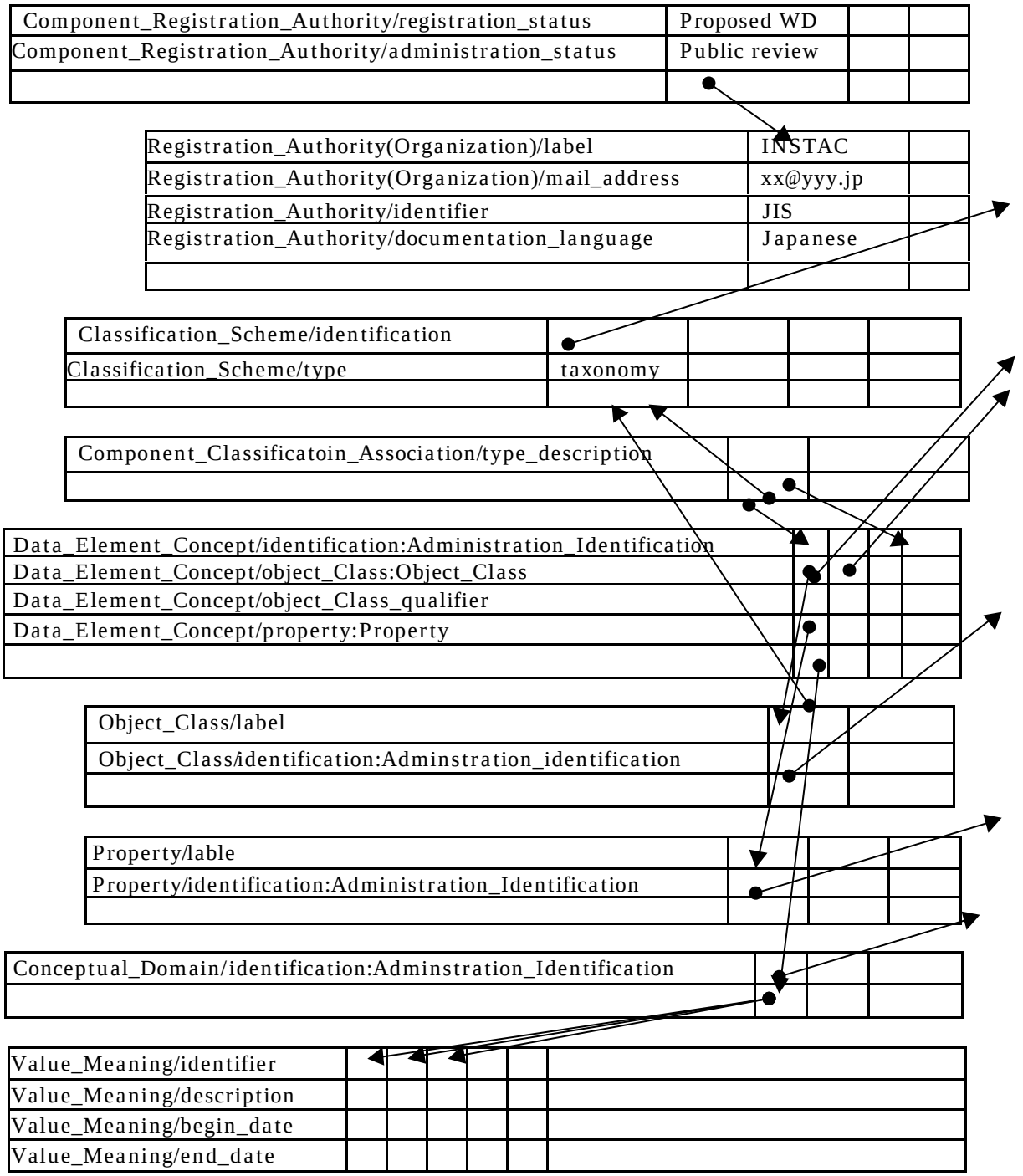
Administration_Record/creation_date	1999/12/24			
Administration_Record/change				
Administration_Record/note				
Administration_Record/unresolved_issue				
Administration_Record/orgin	CBOPs Atomic Objects			

Component_Submitting_Organization/contact_person:Person		

Componet_Responsible_Organization/contact_person:Person		

Person/name	Hajime Horiuchi		
Person/title			
Person/mail_address			
Person/phone_number			
Person/electronic_mail_address	xx@yy.zz.jp		
Person/fax_number			
Person/telex_number			





Componet_Identifier/International_Code_Designator

	Designation/name	**Description/definition**
1	CALENDAR_DATE	Date by the Gregorian calendar. CCYY/MM/DD
2	CALENDAR_GENERALIZATION_TYPE	Basic constraints of the date and time are based on the representation of a data and time ISO 8601:1988.
3	CALENDAR_MONTH	The month by the Gregorian calendar. CCYY/MM

4	CALENDAR_WEEK	Weekday. Sunday to Saturday.
5	CALENDAR_YEAR	Year by the Gregorian calendar. CCYY
6	CITY_CODE	Code for the representation of cities.
7	CODE_GENERALIZATION_TYPE	By being identified, the code clarifies structure of the inside of the object.
8	COUNTRY	Codes for the representation of names of countries. ISO 3166
9	CURRENCY	Currency.
10	DESCRIPTION	Description is used in order to explain abstract attribute, such as identifier or number or code.
11	FISCAL_MONTH	Fiscal month of the enterprise. It is not equal to the calendar month.
12	FISCAL_TERM	Fiscal term of the enterprise. First, Second, Third and Forth. It may express in the first half and the second half.
13	FISCAL_TERM_GENERALIZATION_TYPE	Conceptual domain for fiscal term of the enterprise.
14	FISCAL_WEEK	The number of the weeks of a progress term is expressed in an fiscal term, using the beginning of the term as the 1st week.
15	FISCAL_YEAR	Fiscal year of the enterprise.
16	FLAG	Flag is the logical switch which the event was started or not started. Flag can take only two values yes/no or truth/fowls.
17	GREGORIAN_CALENDAR	The calendar which corrects the error of the Julius calendar, had defined in 1582. By the Gregorian calendar, it will be 365 days the normal year and 366 days the leap year and divide into a series of 12 month 1 year.
18	HOUR	Hour
19	IDENTIFIER_GENERALIZATION_TYPE	Identifier generalization type is classified to identifier, number and code.
20	IDENTIFIRE	Identifier, it identifies the object.
21	IMPLEMENT_CODE_for_BUSINESS_MANAGEMENT	Code for business management performs a work, it specifies the codes which should be managed, such as “item/service”, “accounting subject”. These codes are specified as the sub-type of the “Atomic Object”. These codes are used in the case of the implementing

		to the computer system.
22	IMPLEMENT_CODE_for_COMPUTER_PROCESSING	The code used in order to specify the judgment standard of setting in a computer processing, such as 'look-up code', 'category code', 'classification(type) code', 'status code', 'password' and 'reason code'. These codes are specified as the sub-type of the "Atomic Object". These codes are used in the case of the implementing to the computer system.
23	IMPLEMENTABLE_CODE_GENERALIZATION_TYPE	Implementable code generalization type.
24	INSTITUTION_CODE_GENERALIZATION_TYPE	Institution code, is the object as which the existence is specified by the artificial system, the law, the standard, etc. Regulation on geography, such as country, prefecture/state, city, region. The codes which the public agency determined, such as postal code, phone number. The code which each country have as a common element.
25	LOCATION	The "location" expresses the conceptual position on geography.
26	MEASUREMENT_UNIT	Representation of SI and other units in systems with limited character sets. ISO 2955
27	MINUTE	Minute
28	MONEY	Generalization type of Money.
29	NAME_OBJECT_GENERALIZATION_TYPE	The generalization types of all names.
30	NUMBER_GENERALIZATION_TYPE	Number has arbitrary value, it has neither internal structure. Number consists of a non-meaning code number.
31	NUMERICAL_VALUE_INTEGER	Numerical value Integer.
32	NUMERICAL_VALUE_with_DECIMAL	Numerical value with decimal.
33	PERSON_NAME	Person name.
34	PHONE_NUMBER	Phone number.
35	POSTAL_CODE	The postal code which the postal-administration of each country determined.
36	PREFECTURE_or_STATE_CODE	Code for the representation of States or Prefectures.
37	SECOND	Second
38	TIME_GENERALIZATION_TYPE	Basic constraints of the date and time are based on the representation of a date and time ISO 8601:1988.

39	TIME_STAMP	Original expression for the computer system expressing the details of the data and time.
40	TIME_TERM	The progress time of date or time.

Example of attribute naming convention

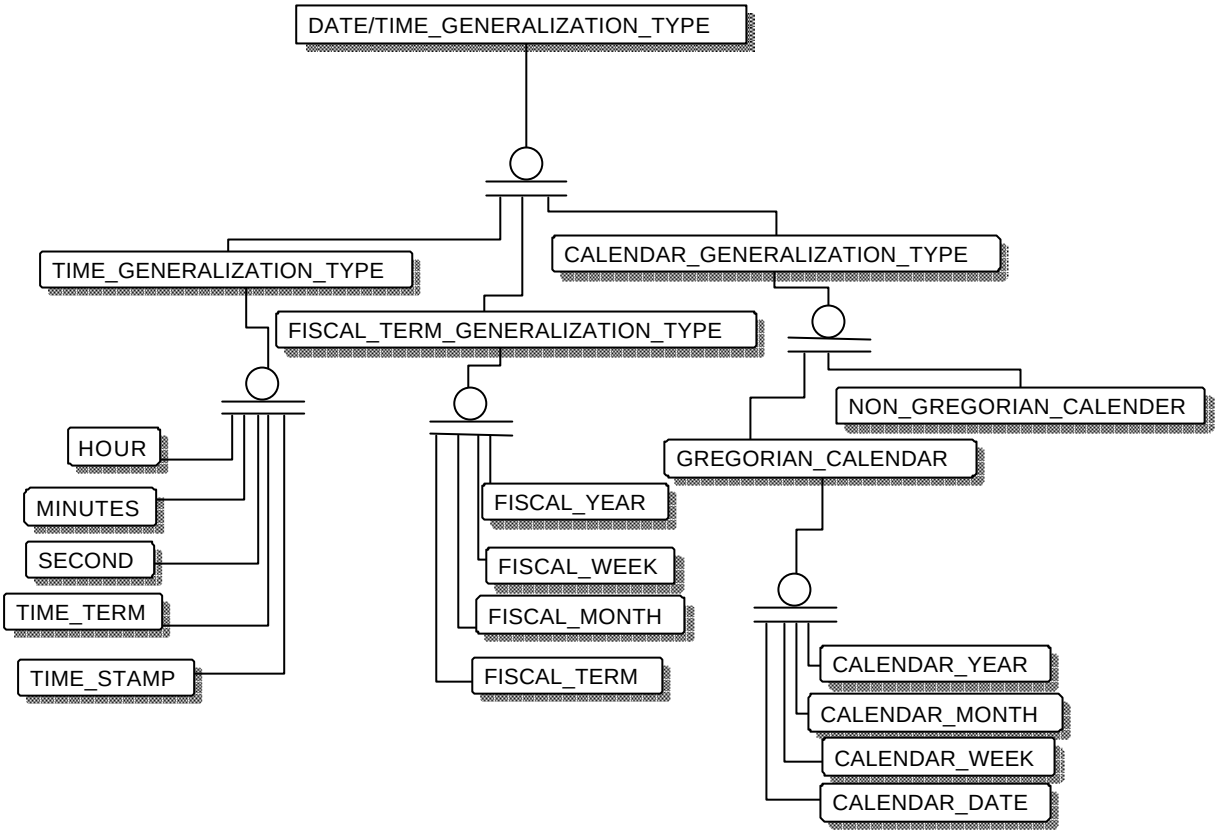
This is an example of naming convention attribute based on “Atomic Objects”.

	Base Domain (Atomic Objecs)	Domain Name Using Atomic Objects
1	BLOB	BLOB
2		Photo_image
3		Video_image
4	Code	Code
5		Category_code
6		Lookup_code
7		Password
8		Product_code
9		Reason_code
10		Status_code
11		Type_code
12	Description	Description
13		Address
14		Comment
15		e-mail_address
16		Mesure_unit
17	Name_object	Name_object
18		First_name
19		Kana
20		Last__name
21	Flag	Flag
22	Identifier	Identifier
23		Detail_line
24		Number
25		Number_auto_increse
26	Institution_code	Institution_code
27		City_code
28		Country_code
29		Currency
30		Phone_number
31		Postal_code
32		Prefecture_code
33		Region_code
34		Teritory_code
35	Numerical	Numerical
36		Factor
37		Percent
38		Quantity_integer
39		Transaction_count
40	Money	Money
41		Account_of_money
42		Cost
43		Price
44	Date/time	Date/time
45	Calendar	Calendar
46		Calendar_date
47		Calendar_month
48		Calendar_week_day

49		Calendar_year
50	Fiscal	Fiscal
51		Fiscal_month
52		Fiscal_term
53		Fiscal_week
54		Fiscal_year
55	Time	Time
56		Hour
57		Minitue
58		Second
59		Time_stamp
60		Time_term

Data model of "Atomic Object" by IDEF1X

Sample model of "DATE/TIME" part.



Java description of the atomic object and its constraints
Sample Description of "DATE/TIME" part

Atomic object list
This result is the list of that which encode "the atomic object list" by Java.
Package CBOP.COMMON
The common function in CBOP.atom is provided. Usually the class relating with a programmer is Bundle_ja_JP. and others are a class which an atomic object uses internally.
Bundle_ja_JP is a module for locale which is internationalization of Java. When using English in US, the class which registered the standard code of the code table about conatraints of an atomic object using Bundle_en_US is created.

Class Summary

Bundle_ja_JP	The class for code table determination for country =ja (Japan) and language =JP (Japanese)
Date	A function required for dates is provided.
Format	The function for normalized eXpression is provided. This class uses "GNU.regeXp.".
Modulars	The modulus-related function checkdigit is provided.
Nums	The function of a relation with "Number" is provided.
Table	The function for accessing the table of Ms ACCESS by JDBC+ODBC is provided.
ValueEXception	The function called when value is eXtraordinary value.

Package CBOP.atom
The function of an atomic object is provided.

Class Summary of Atomic Object

CalendarDate	CalendarDate Atomic Object
CalendarMonth	CalendarMonth Atomic Object
Calendars	Calendars Atomic Object DateTime Class
CalendarYear	CalendarYear Atomic Object
City	City Code Atomic Object
Code	Code Class
Country	Country Code Atomic Object
DateTime	DateTime Class Atomic Object
Description	Description Atomic Object
EmailAddress	E-mail address Atomic Object
FirstName	FirstName Atomic Object
FiscalMonth	FiscalMonth Atomic Object
FiscalTerm	FiscalTerm Atomic Object
FiscalWeek	FiscalWeek Atomic Object
FiscalYear	FiscalYear Atomic Object
Flag	Flag Atomic Object
Hour	Hour(HH) Atomic Object
LastName	LastNam Atomic Object
MeasureUnit	MeasureUnit Atomic Object
Minutes	Minutes(MM) Atomic Object
Money	Money Atomic Object
NumberAutoIncrement	Number Auto Increment Atomic Object
Numbers	Numbers Atomic Object
PhoneNumber	Phone Number Atomic Object
PostalCode	Postal Code Atomic Object
Prefecture	Prefecture Code Atomic Object
Territory	Territory Atomic Object
Time	Time Atomic Object

The atomic object definition by Java

The following list is an excerpt of code line written by Java on the basis of "the atomic object list" and "the constraints list of atomic objects".

Package CBOP.COMMON

Class bundle_ja_JP

Description: The class for code table determination for country =ja (Japan) and language =JP (Japanese)

Upper Class: java.util.ListResourceBundle

Form: public class bundle_ja_JP
eXtends ListResourceBundle

Sample:

```
static final Object[][] = {
                    {“country” , “X0304”},
    {“Industry” , “X0403”}};
```

Semantics:

In country =ja (Japan) and language =JP (Japanese), the standard of an atomic object "a country code" is JIS X0304, and "an industrial classification code" is JIS X0403. A code table can refer a name by the code and the key of a standard.

```
//-----------------------------------------------------------------
// Object ID      : cbop.common.bundle_ja_JP
// Object Name    : jp_JP for resource bundle
// Upper Class    : ListResourceBundle
// Object Contents : Determined the identifier for search talbe to ja_JP
// Author         : CAC. Yoshihiko Yamada
// Date           : 1999/01/08
//-----------------------------------------------------------------
package cbop.common;

import java.util.ListResourceBundle;

public class bundle_ja_JP  eXtends ListResourceBundle {

 public Object[][] getContents() {
  return contents;
 }

 static final Object[][]  contents = {
  {"Country"        , "X0304"},   // Country code
  {"Prefecture"     , "X0401"},   // Prefectures code
  {"City"   , "X0402"},   // City code
  {"Industry"   , "X0403"},   // Industrial classification code
{"Industry_ja_dai" , "X0403-1"}, // Classification_industrial_code_Japan_class_large
{"Industry_ja_chuu" , "X0403"},  // Classification_industrial_code_Japan_class_middle
{"Industry_ja_shou" , "X0403"}, // Classification_industrial_code_Japan_class_minor
 {"Industry_ja_sai" , "X0403"}, // Classification_industrial_code_Japan_class_detail
  {"Occupation"  , "X0404"},   // Oocupation_Classification_code
 {"Occupation_ja_dai" , "X0404-1"}, // Oocupation_Classification_code_class_large
 {"Occupation_ja_chuu" , "X0404"}, // Oocupation_Classification_code_class_middle
```

```
 {"Occupation_ja_shou" , "X0404"},   // Oocupation_Classification_code_class_minor
  {"CommodityClassification" , "X0405"},   // Commodity Classification
  {"MeasureUnit" , "X0124"},   // Measure Unit
 };
}
```

Class Dates

Description: A function required for dates is provided.

Upper Class: None

Form: public class Dates

Metho:

Public static int getDays(int iMonth)	
Public static int getDays(int iYear, int iMonth)	
Public static int isLeapYear(int iYear)	

```
//---------------------------------------------------------------
// Object ID      : cbop.common.Dates
// Object Name    : Date object
// Upper Class    : Code
// Object Contents : Date object
// Author         : CAC. Yoshihiko Yamada
// Date           : 1999/01/08
//---------------------------------------------------------------
package cbop.common;

import cbop.common.ValueEXception;

public class Dates {

 static int month_days[] = new int[12];
 static {
  month_days[0] = 31;
  month_days[1] = 28;
  month_days[2] = 31;
  month_days[3] = 30;
  month_days[4] = 31;
  month_days[5] = 30;
  month_days[6] = 31;
  month_days[7] = 31;
  month_days[8] = 30;
  month_days[9] = 31;
  month_days[10] = 30;
  month_days[11] = 31;
 }

 //---------------------
 // Static Method
 //---------------------
  public  static int getDays(int iMonth) throws ValueEXception {
```

```
    if (1<= iMonth && iMonth <=12)
      return month_days[iMonth-1];
    throw new ValueEXception(String.valueOf(iMonth));

  }
   public  static int getDays(int iYy, int iMonth) throws ValueEXception {
    if (isLeapYear(iYy) == true)
      if (iMonth == 2)
              return 29;
    if (1<= iMonth && iMonth <=12)
      return month_days[iMonth-1];
    throw new ValueEXception(String.valueOf(iMonth));

  }
  public  static boolean isLeapYear(int iYy) {
    if (iYy%4 == 0 && iYy%100 !=0)
      return true;
    if (iYy%400 == 0)
      return true;
    return false;
  }
}
```

Annex B (Informative or TR) Representation of Business Function Object Patterns

---

Introduction

This section presents Business Function Object Patterns (BFOP) defined in the UML Profile. 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.

Background Facts

The proposed UML Profile 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.

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.

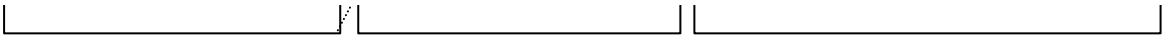


Figure 4.2-1 Overview of Architecture

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

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.

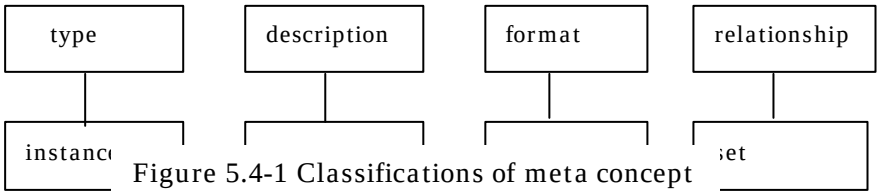


Figure 5.4-1 Classifications of meta concept

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.

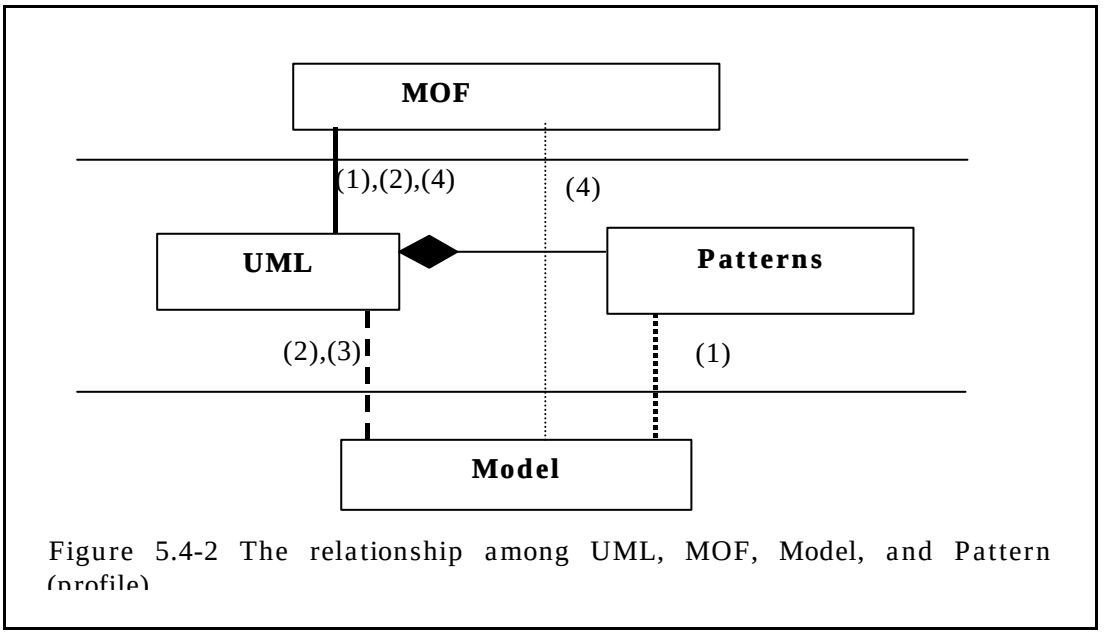


Figure 5.4-2 The relationship among UML, MOF, Model, and Pattern (profile)

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.

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.

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.

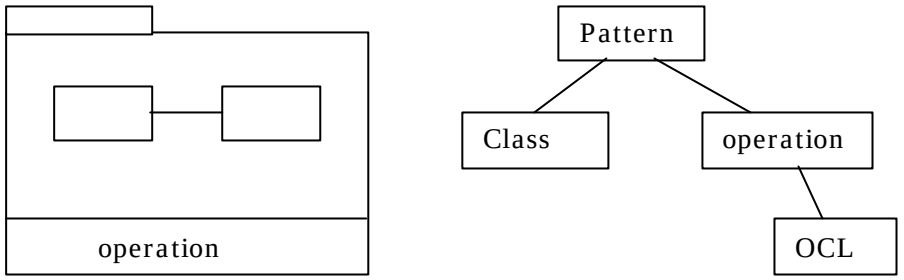


Figure 5.4-3 The notation of Simple Pattern and its

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.

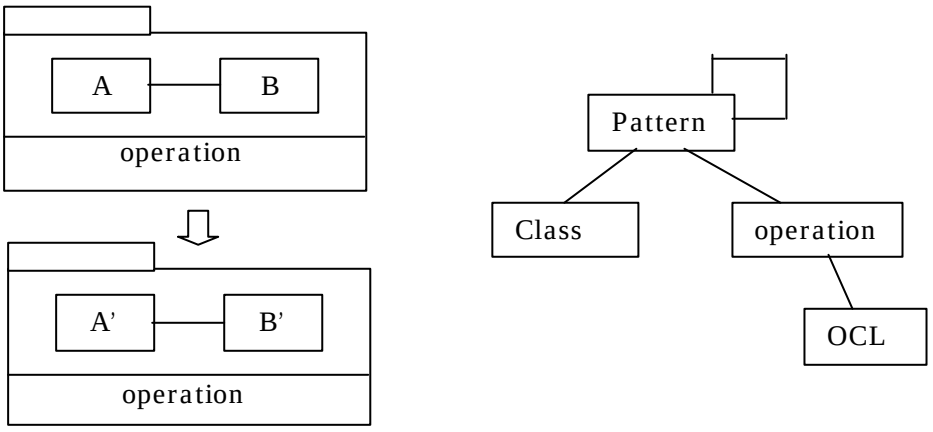


Figure 5.4-4 The notation of Pattern Inheritance and its metamodel

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.

Relationship among Pattern Mechanisms

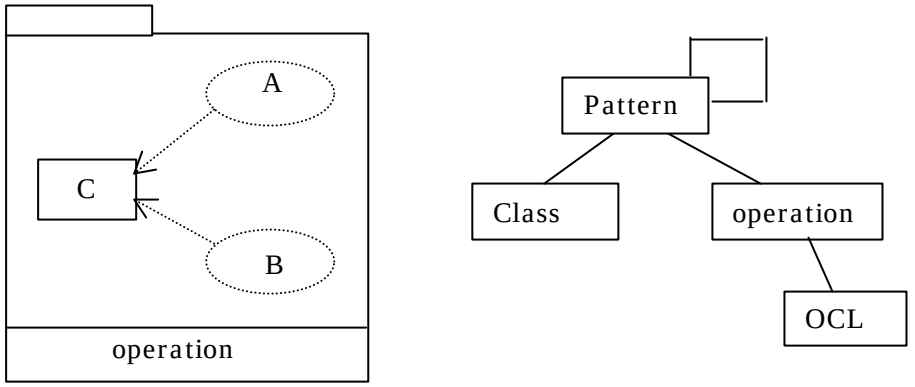


Figure 5.4-5 The notation of Pattern Composition and its Metamodel

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.

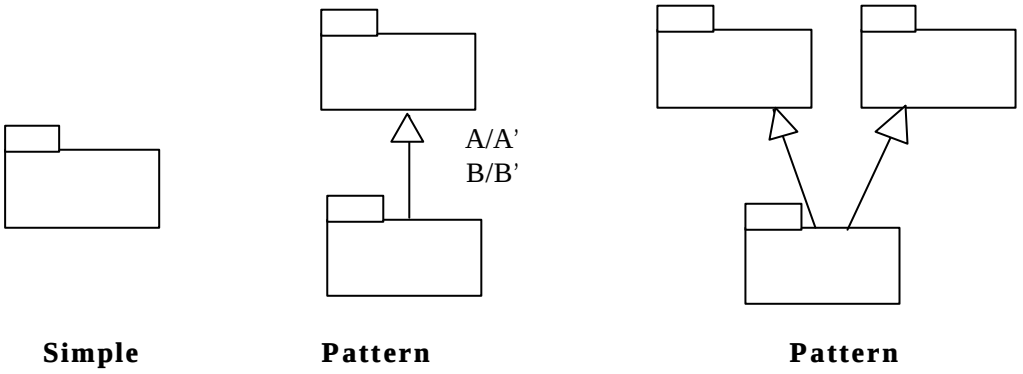


Figure 5.4-6 The relationship for pattern mechanisms

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.

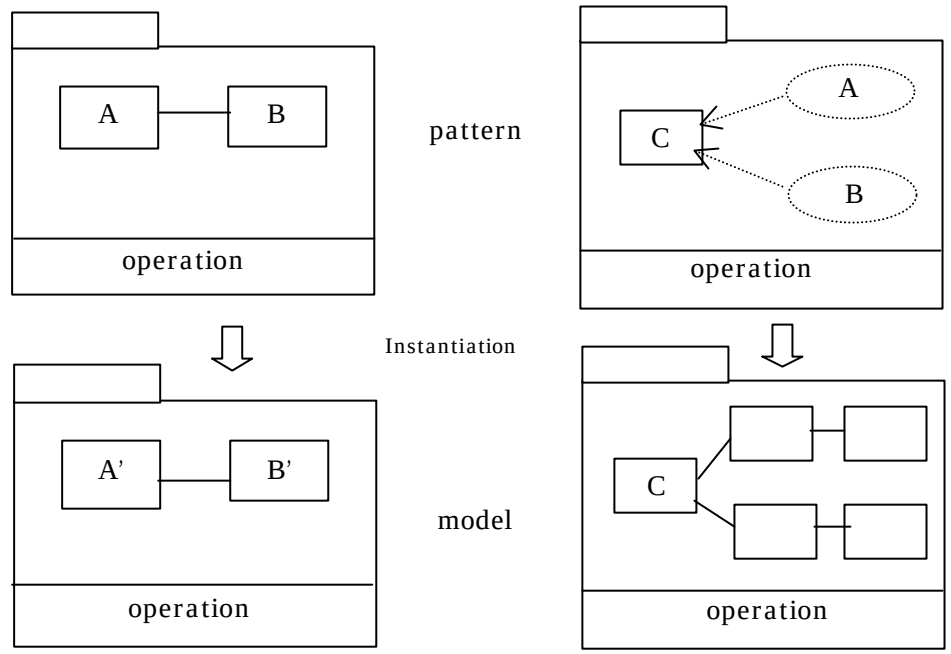


Figure 5.4-7 Instantiation of Patterns

UML Profile for BFOP

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

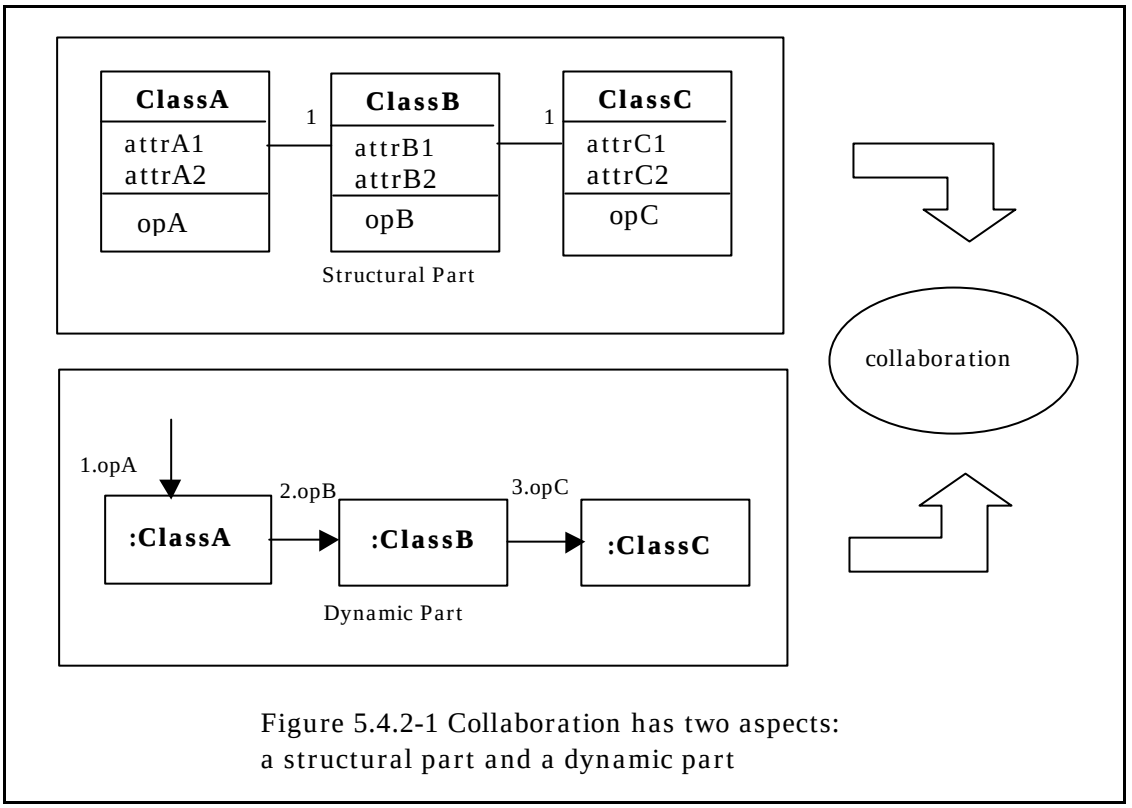


Figure 5.4.2-1 Collaboration has two aspects:
a structural part and a dynamic part

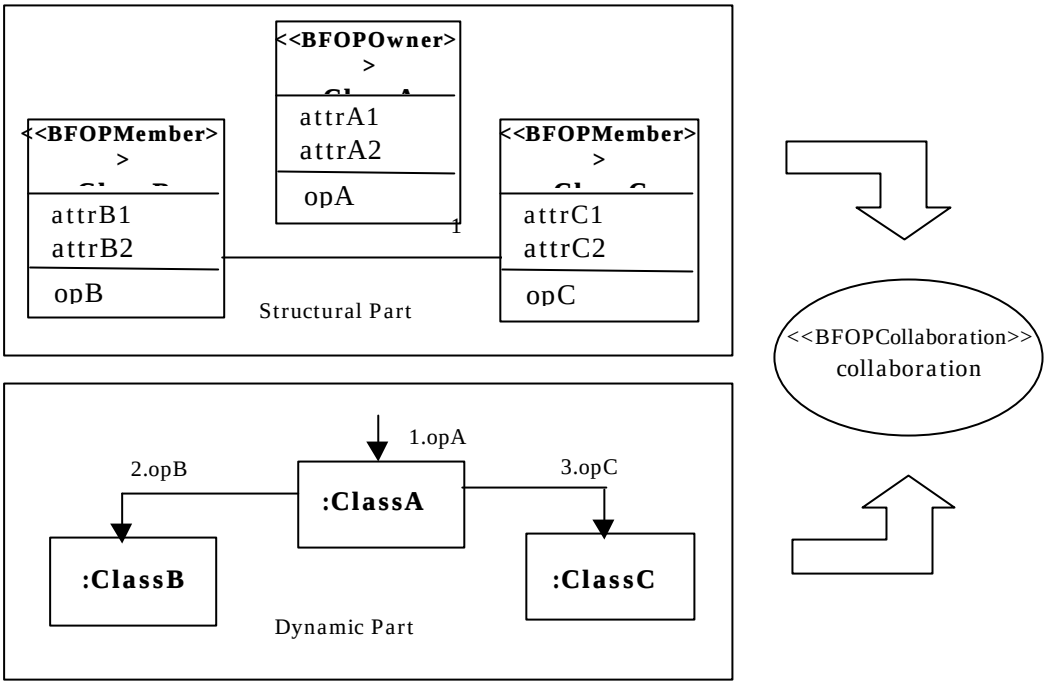


Figure 5.4.2-2 Collaboration also has two aspects: a structural part and a dynamic part. However, BFOP owner class has no association with BFOP member classes and BFOP member class instance does not interact with other member instances.

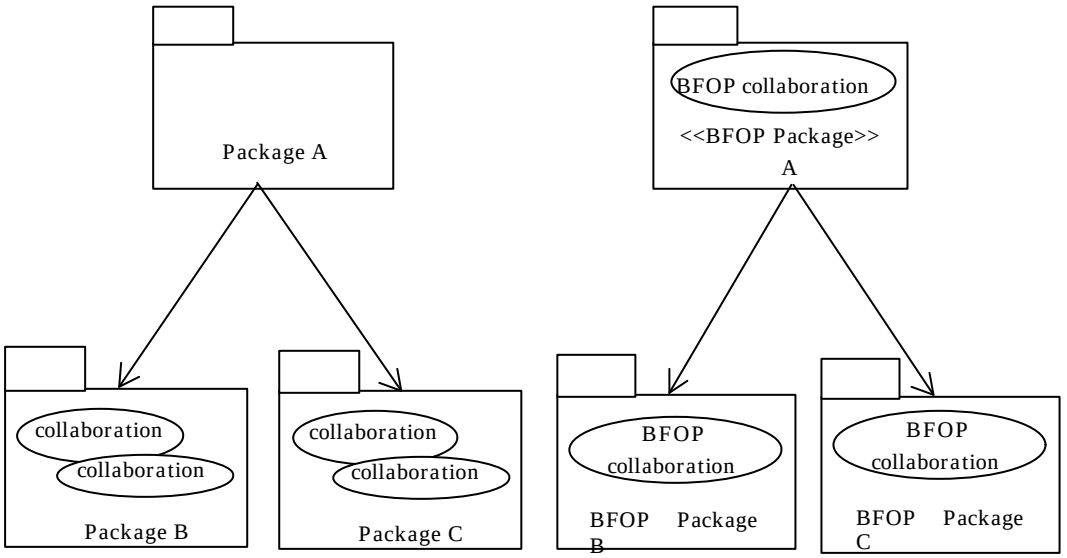


Figure 5.4.2-3 BFOP Package always contains only one BFOP collaboration.

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.

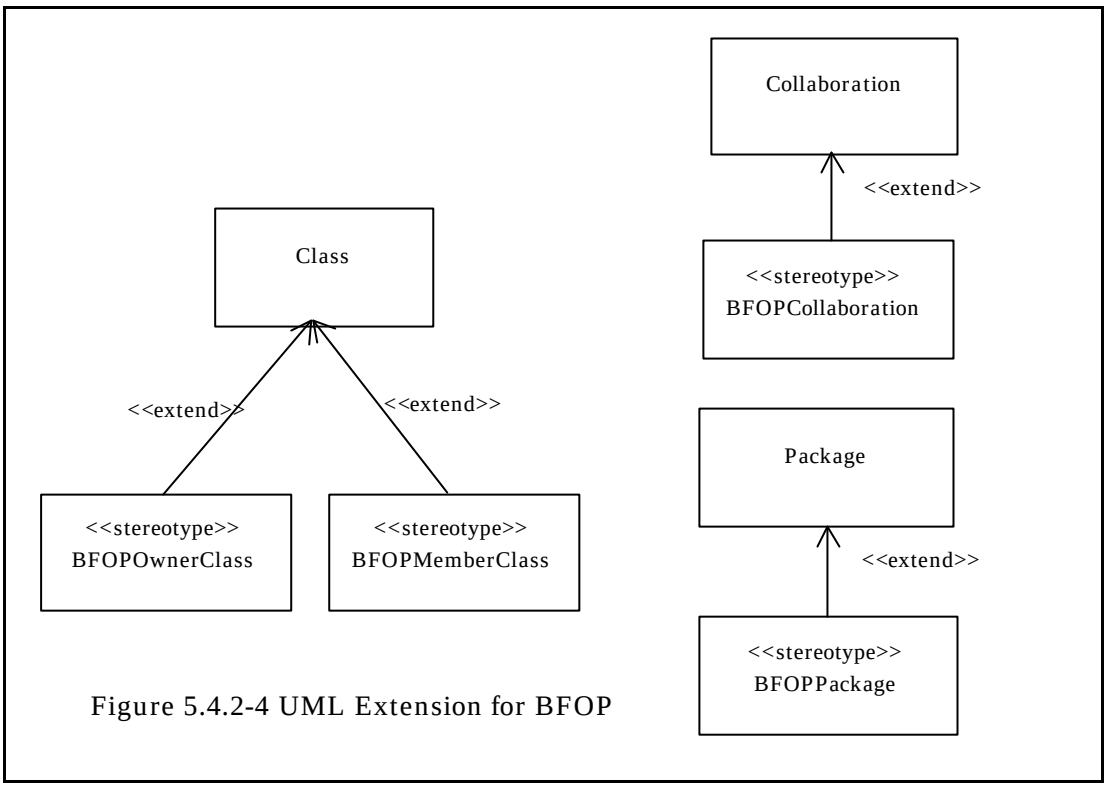


Figure 5.4.2-4 UML Extension for BFOP

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.

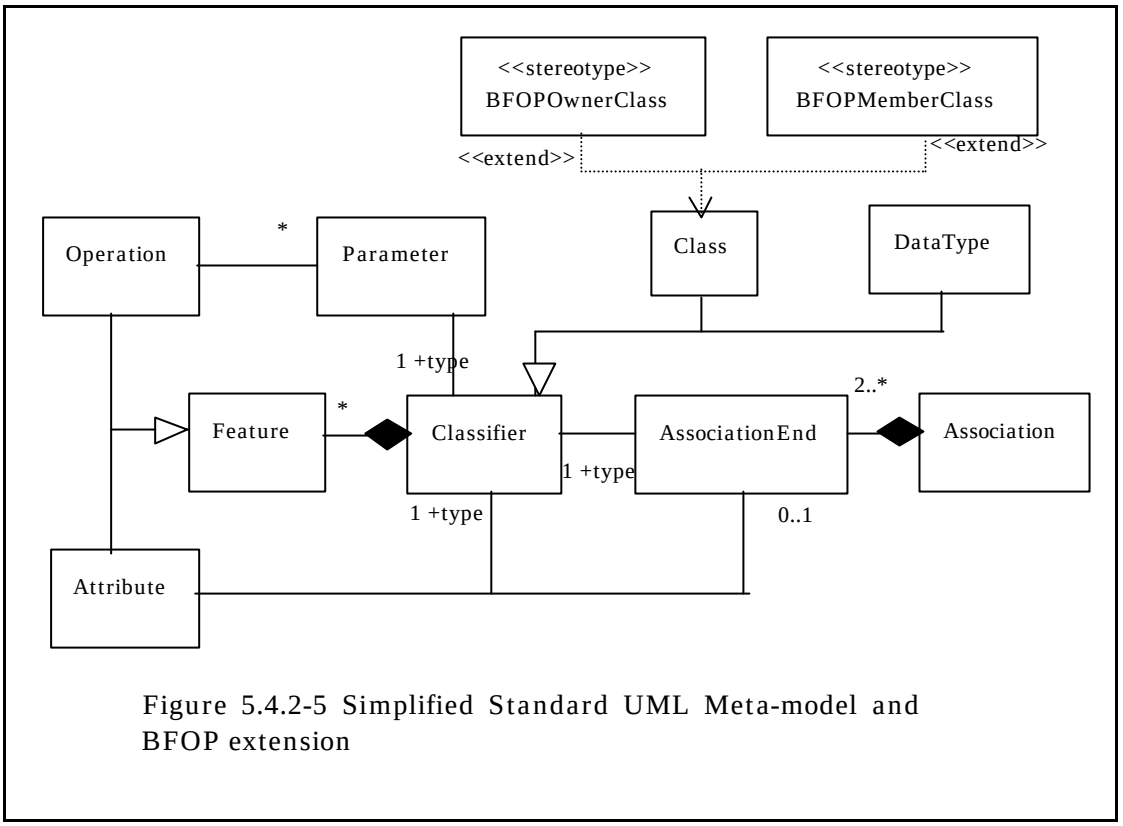


Figure 5.4.2-5 Simplified Standard UML Meta-model and BFOP extension

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..*

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

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.

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.

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	Product Price is defined by Customer (or

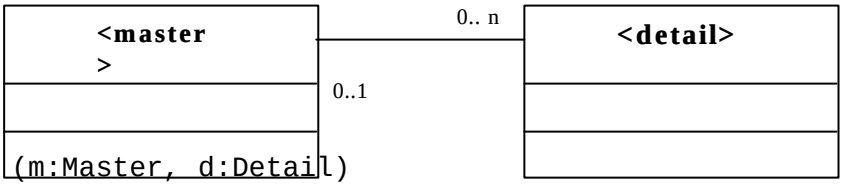
	Correspondent /Period	Supplier) by Date.
3040	Employee-Role	Employee can have multiple roles.
3050	Employee Role Definition (SS)	Role has an object what is managed. Purchase by supplier. Sales by area. Support by Product.

Pattern Definition (excerpt of examples)

Basic Patterns

Master & Detail

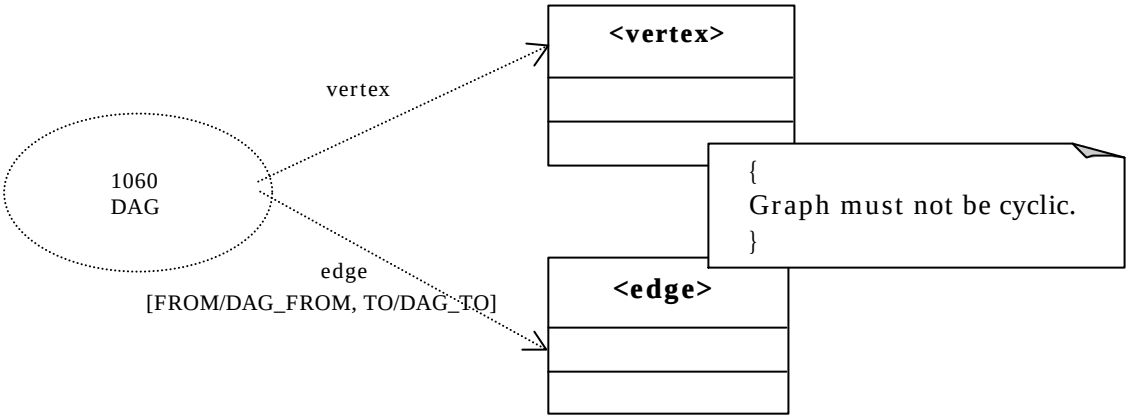
BFOP No.	1010
BFOP Name	Master & Detail
Inherited BFOP	none
Parameter	<Master>, <Detail>
Revision	Rev.2
Description	Master is referred by Detail, e.g. Area: Customer, License-Employee, Order type-Order




```
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)
```

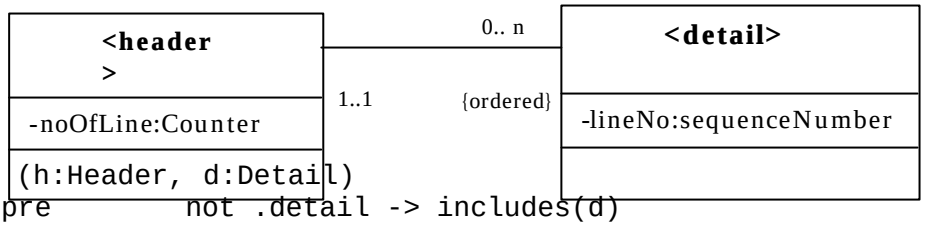
DAG (Directed Acyclic Graph)

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.



Header & Detail

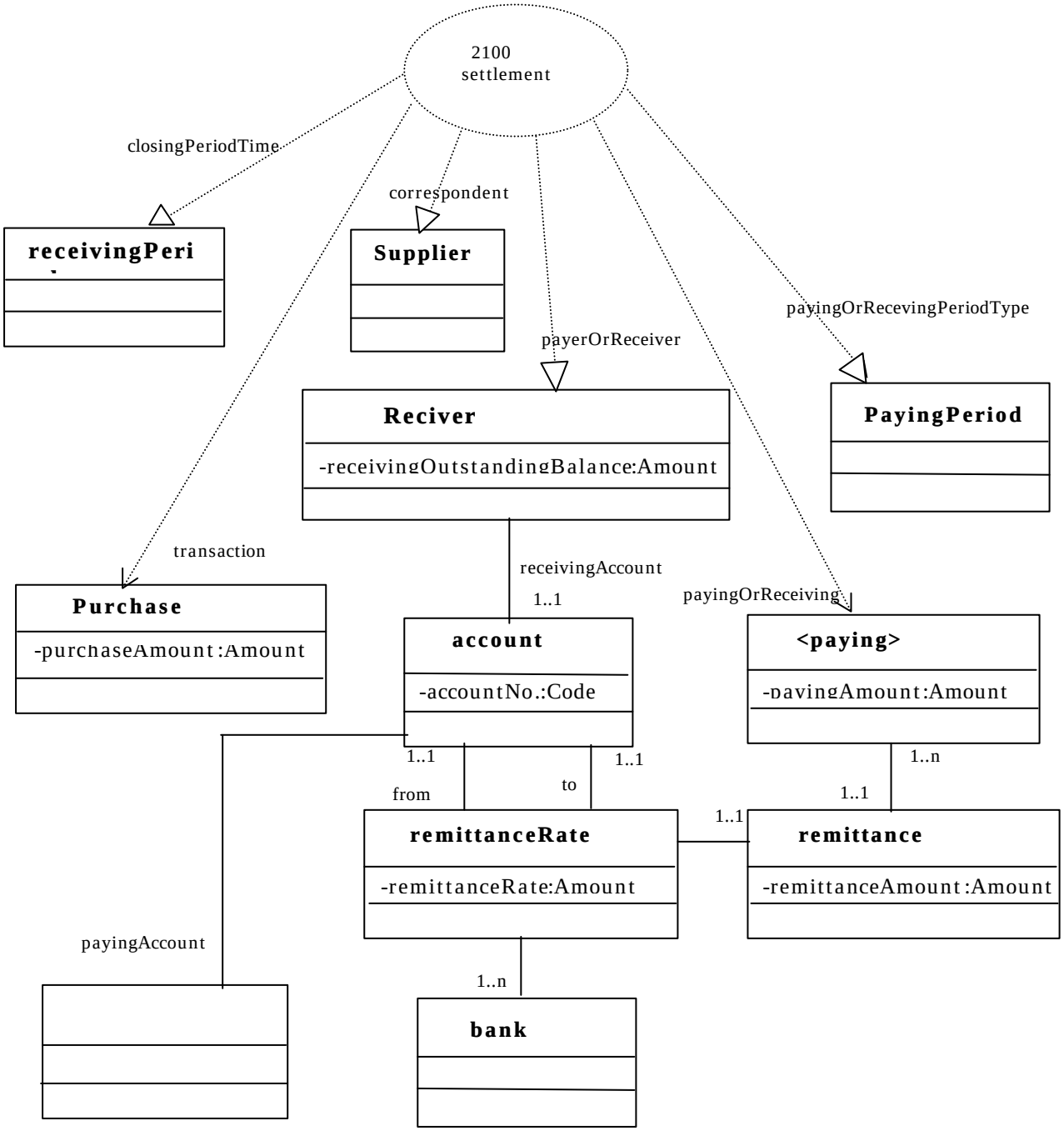
BFOP No.	1070
BFOP Name	Header & Detail
Inherited BFOP	none
Parameter	<Header>, <Detail>
Revision	Rev.2
Description	Header and Detail have a relationship of parent and children. e.g. Order and Order line, Bank and Bank branch, Division and section.



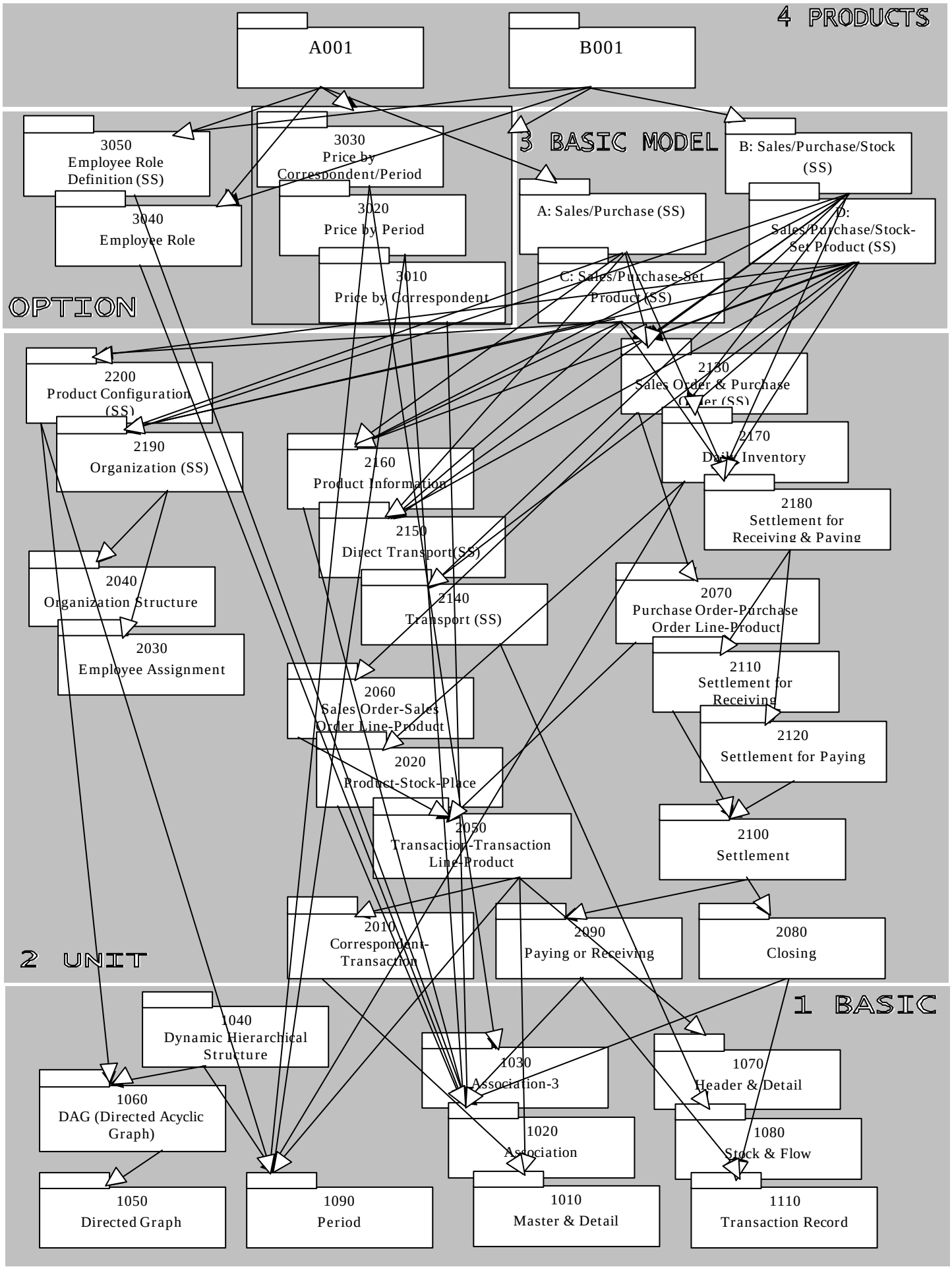

```
ddDetail (h:Header, d:Detail)
        pre         not .detail -> includes(d)
        post        d.header.detail ->includes(d)
removeDetail(d:Detail)
        pre         d.header.detail -> includes(d)
        post        not .header.detail -> includes(d)
```

Settlement for Paying

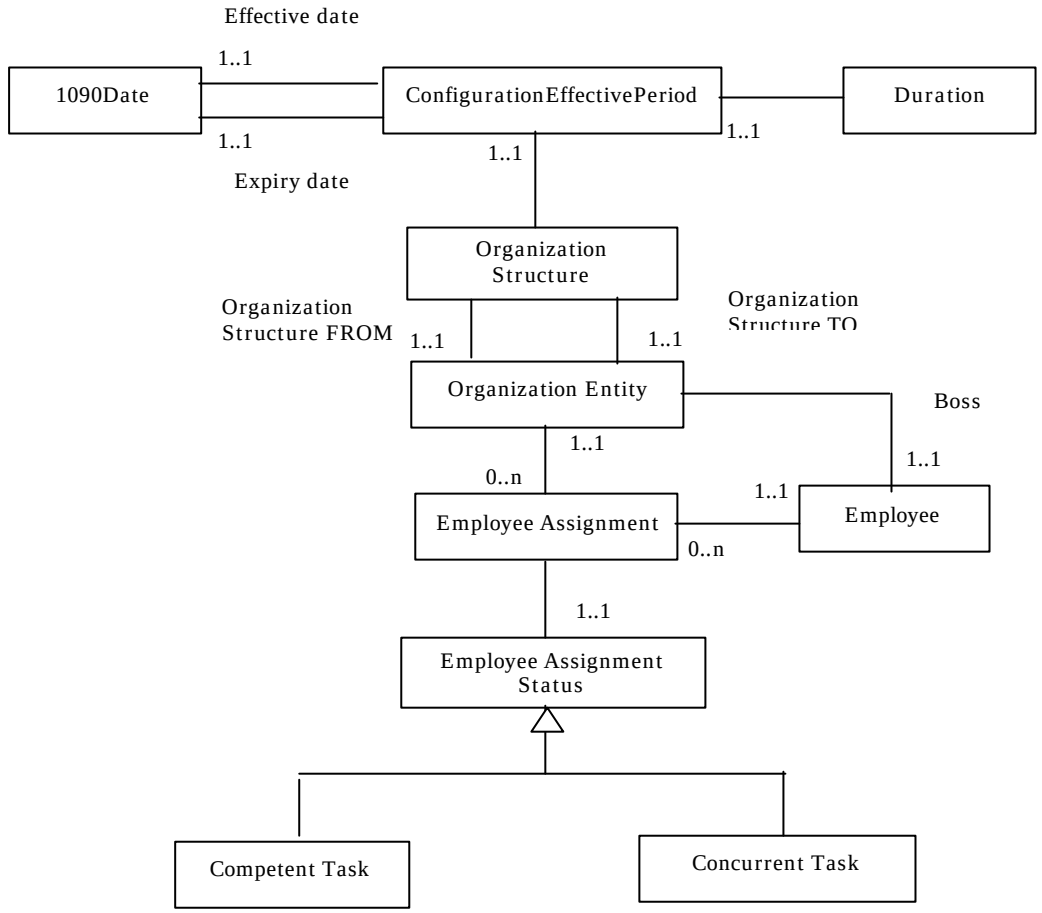
BFOP No.	2120
BFOP Name	Settlement for Paying
BFOP to be inherited	2100 (Settlement)
Parameter	<Settlement for Paying>
Revision	Rev.2
Description	Totaling Purchase on payment, it is paid to the Supplier. In case of transfer through bank, rate is calculated.

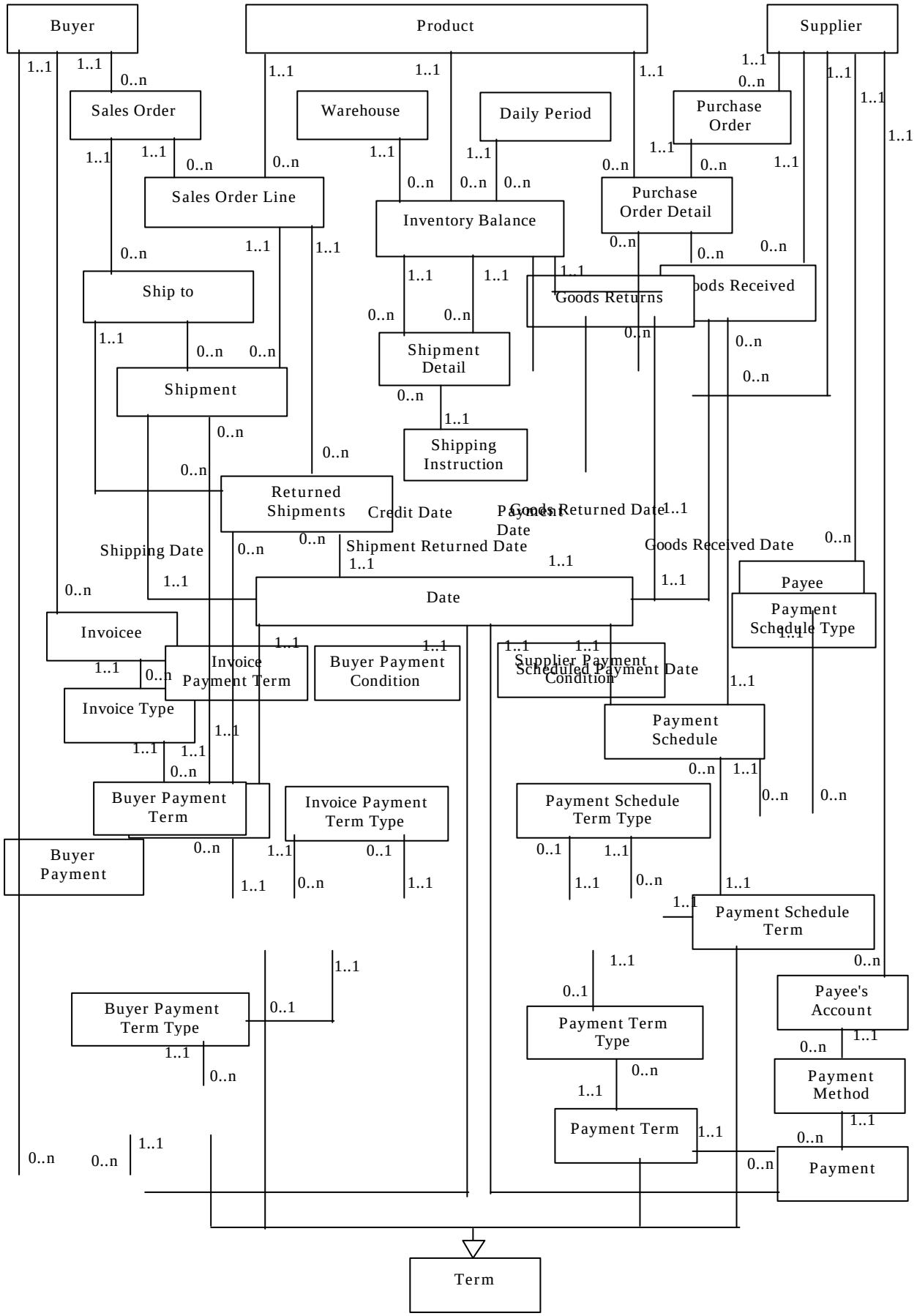


Overall Package Structure


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

Sample Object Model for BFOP




Buyer
Product
Supplier
Sales Order
Warehouse
Daily Period
Purchase Order
Sales Order Line
Inventory Balance
Purchase Order Detail
Ship to
Goods Returns
Goods Received
Shipment Detail
Shipment
Shipping Instruction
Returned Shipments
Credit Date
Payment Date
Goods Returned Date
Shipping Date
Shipment Returned Date
Goods Received Date
Date
Payee
Payment Schedule Type
Invoicee
Invoice Payment Term
Buyer Payment Condition
Supplier Payment Condition
Scheduled Payment Date
Invoice Type
Payment Schedule
Buyer Payment Term
Invoice Payment Term Type
Payment Schedule Term Type
Buyer Payment
Payment Schedule Term
Buyer Payment Term Type
Payment Term Type
Payee's Account
Payment Method
Payment Term
Payment
Term

Annex C (informative) An Example of Modeling A Conceptual Bicycle

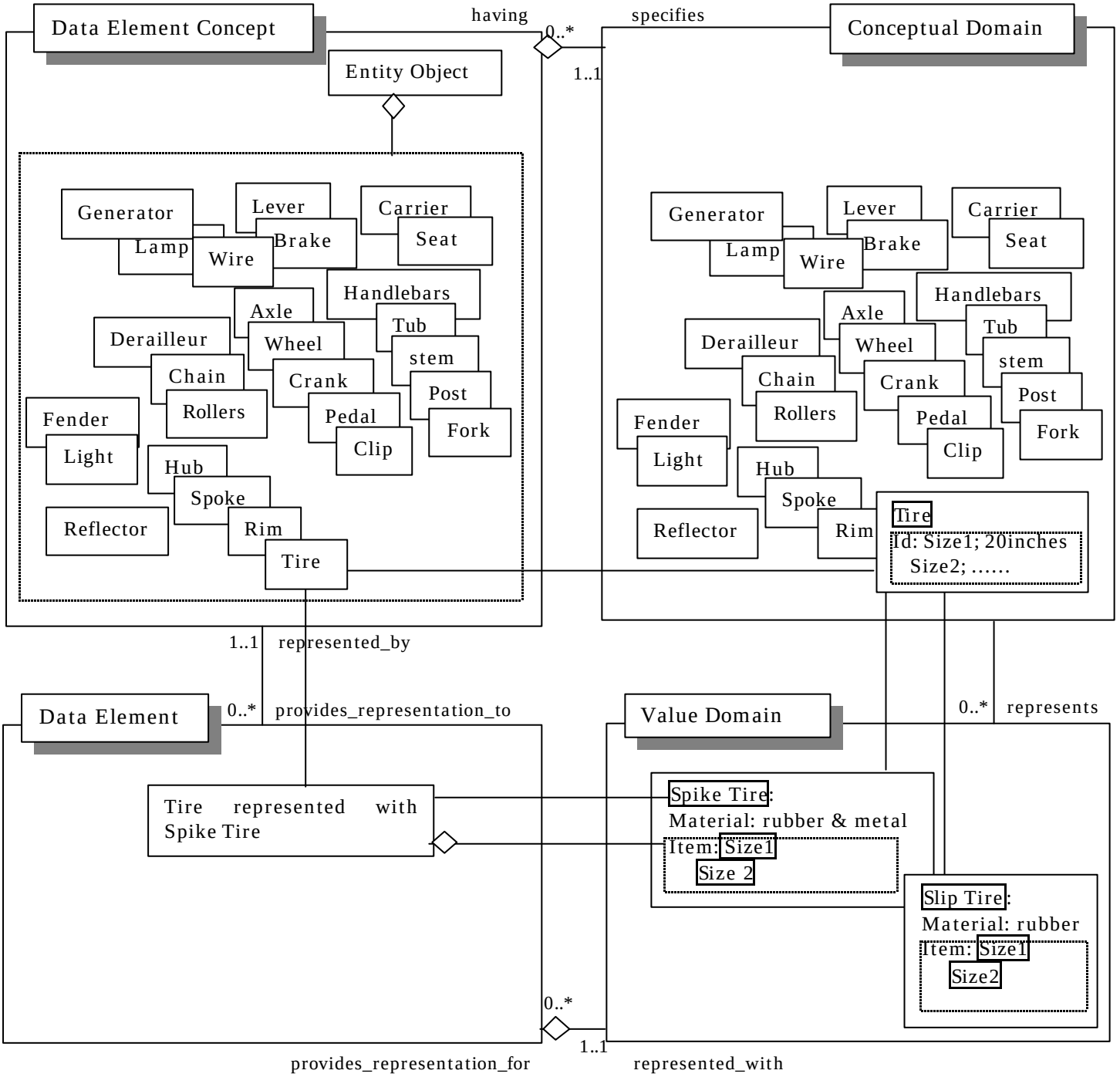


Fig 2: An Example of Entity Object: ”Tire” in Modeling A Conceptual Bicycle

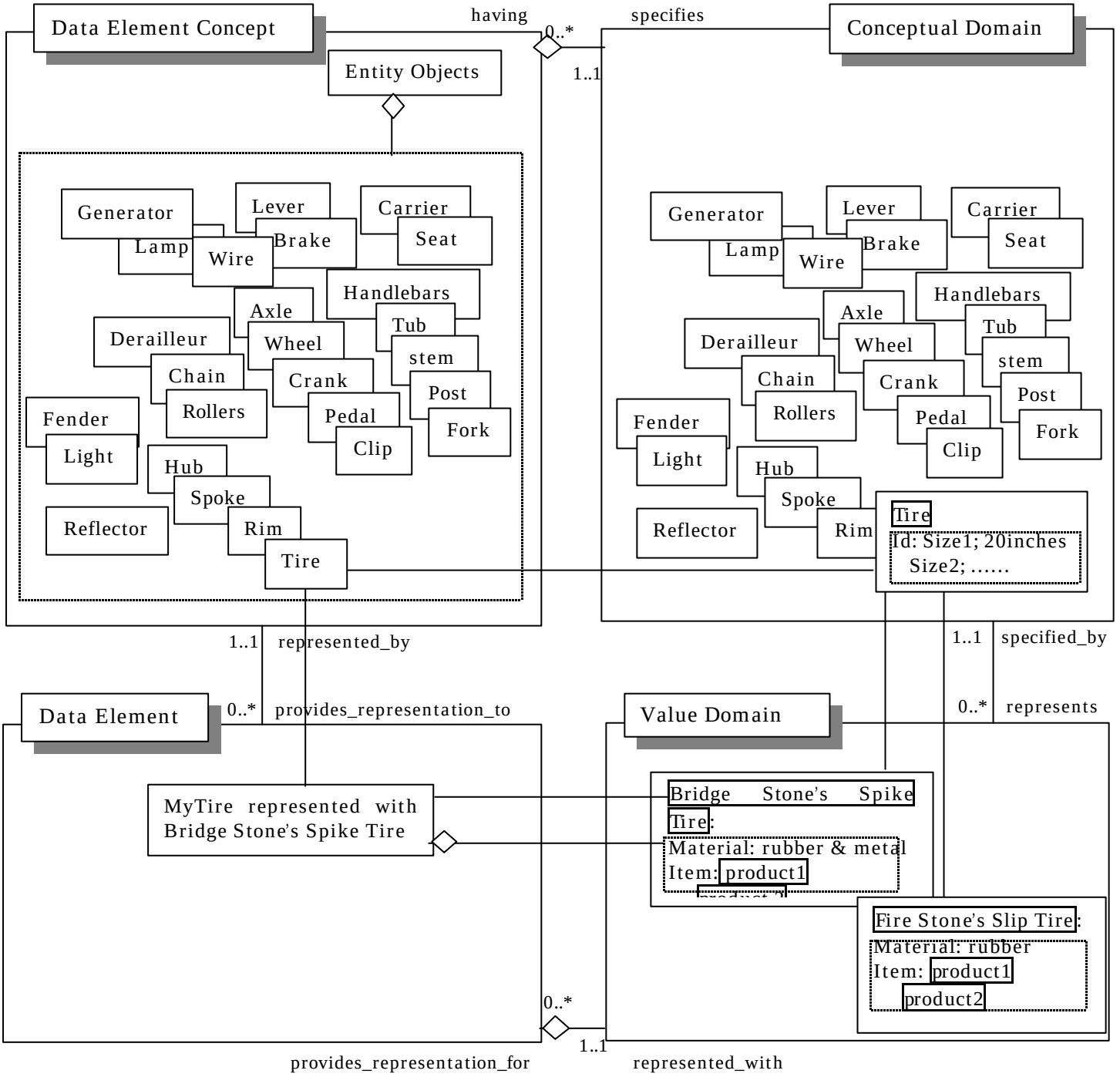


Fig 3: An Example of Entity Object: ”Tire” in Modeling A Bicycle

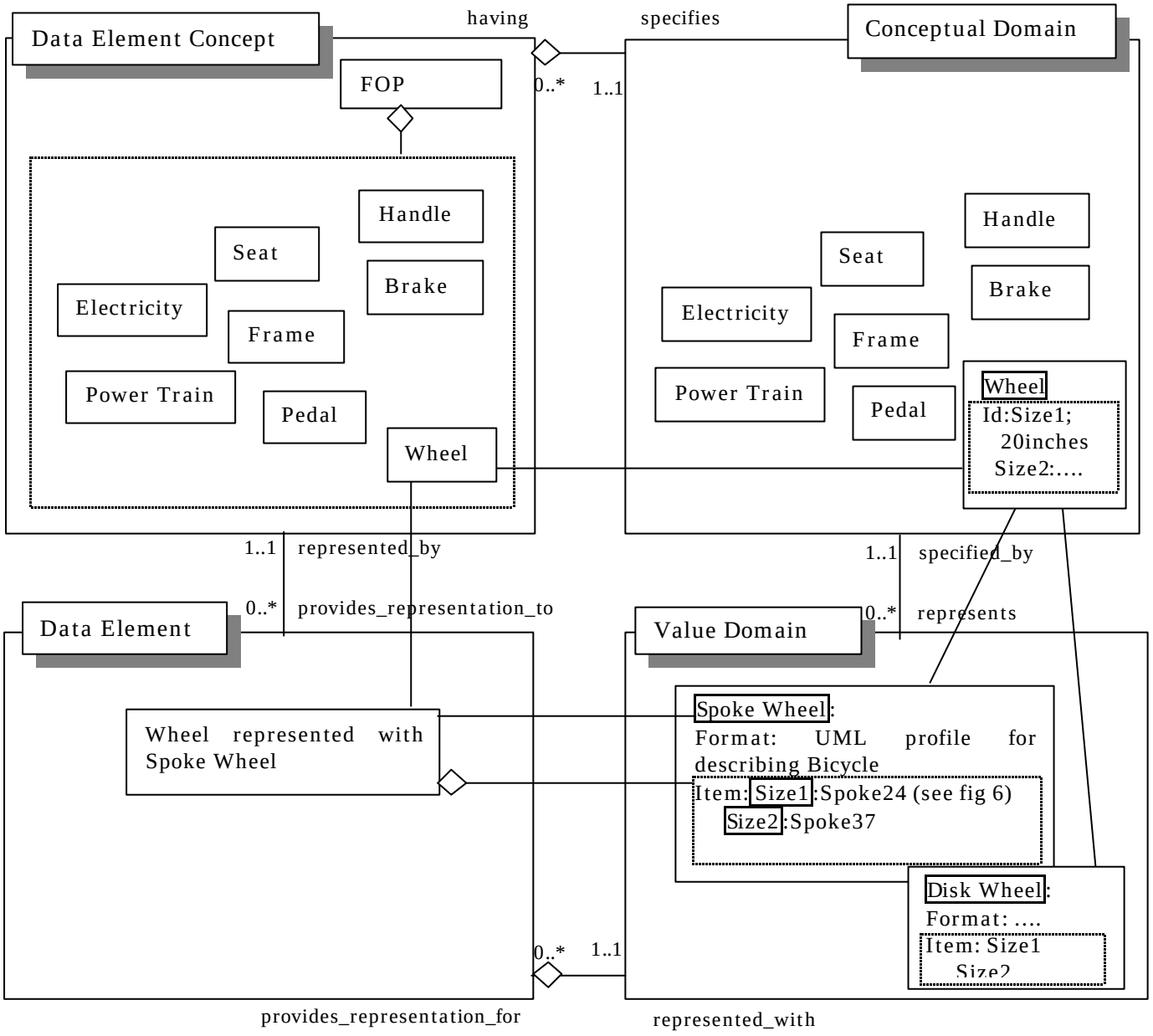


Fig 4: An Example of FOP: ”Wheel” in Modeling A Conceptual Bicycle

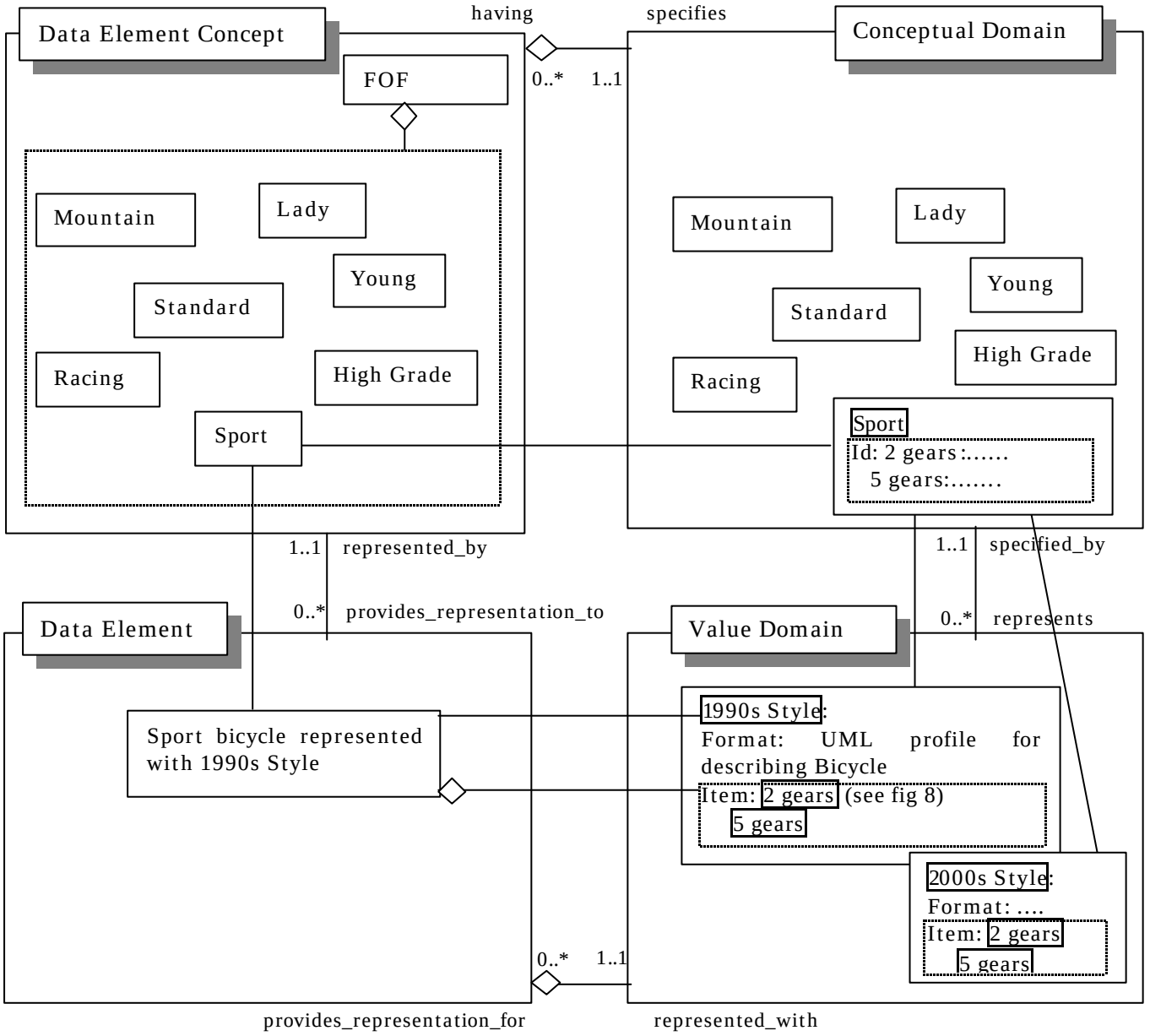


Fig 5: An Example of FOF: ”Sport” in Modeling a Conceptual Bicycle

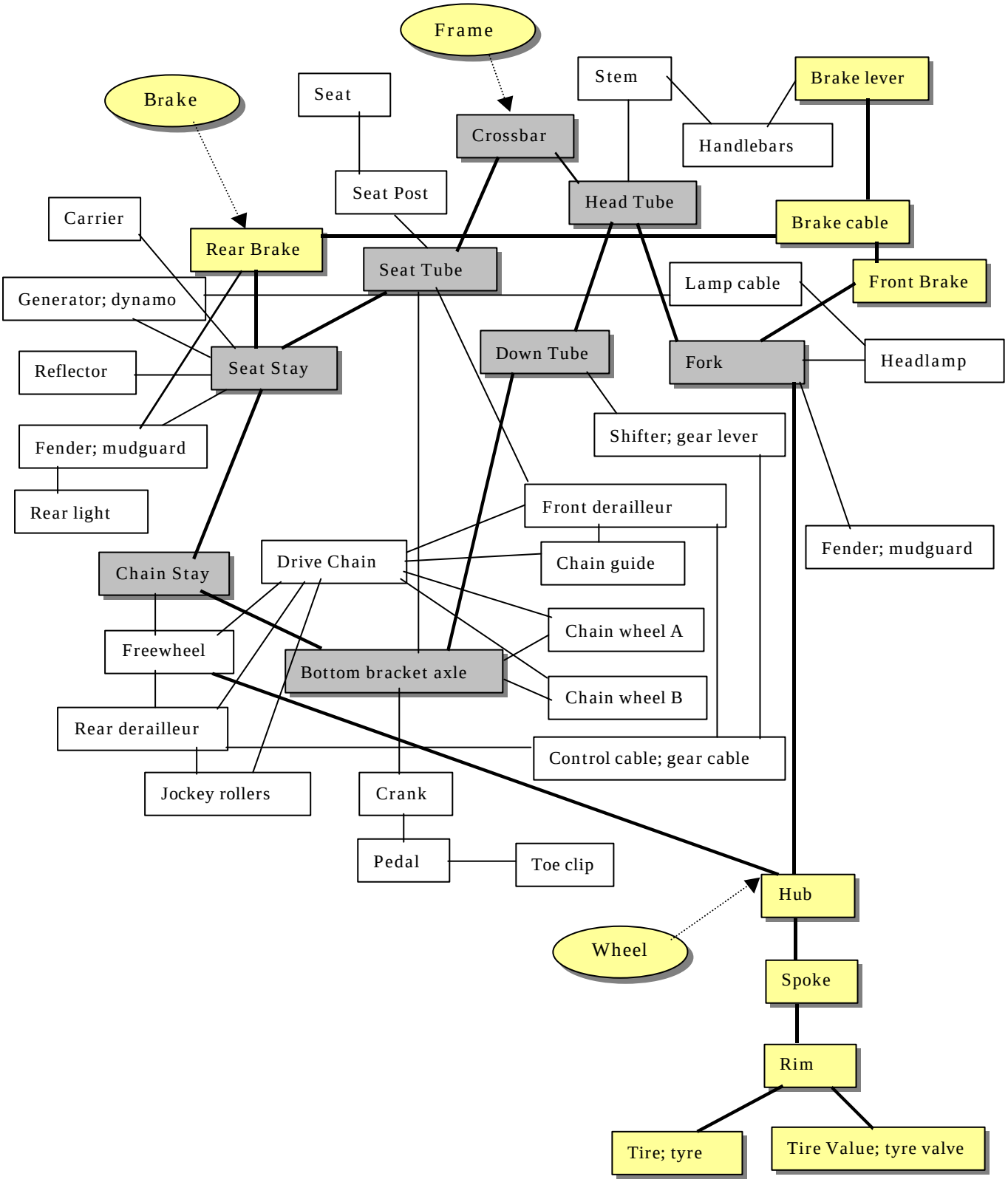


Fig 6: An Example of concrete Patterns for FOPs : ”Frame”, “Brake”, ”Wheel” in Modeling a Conceptual Bicycle

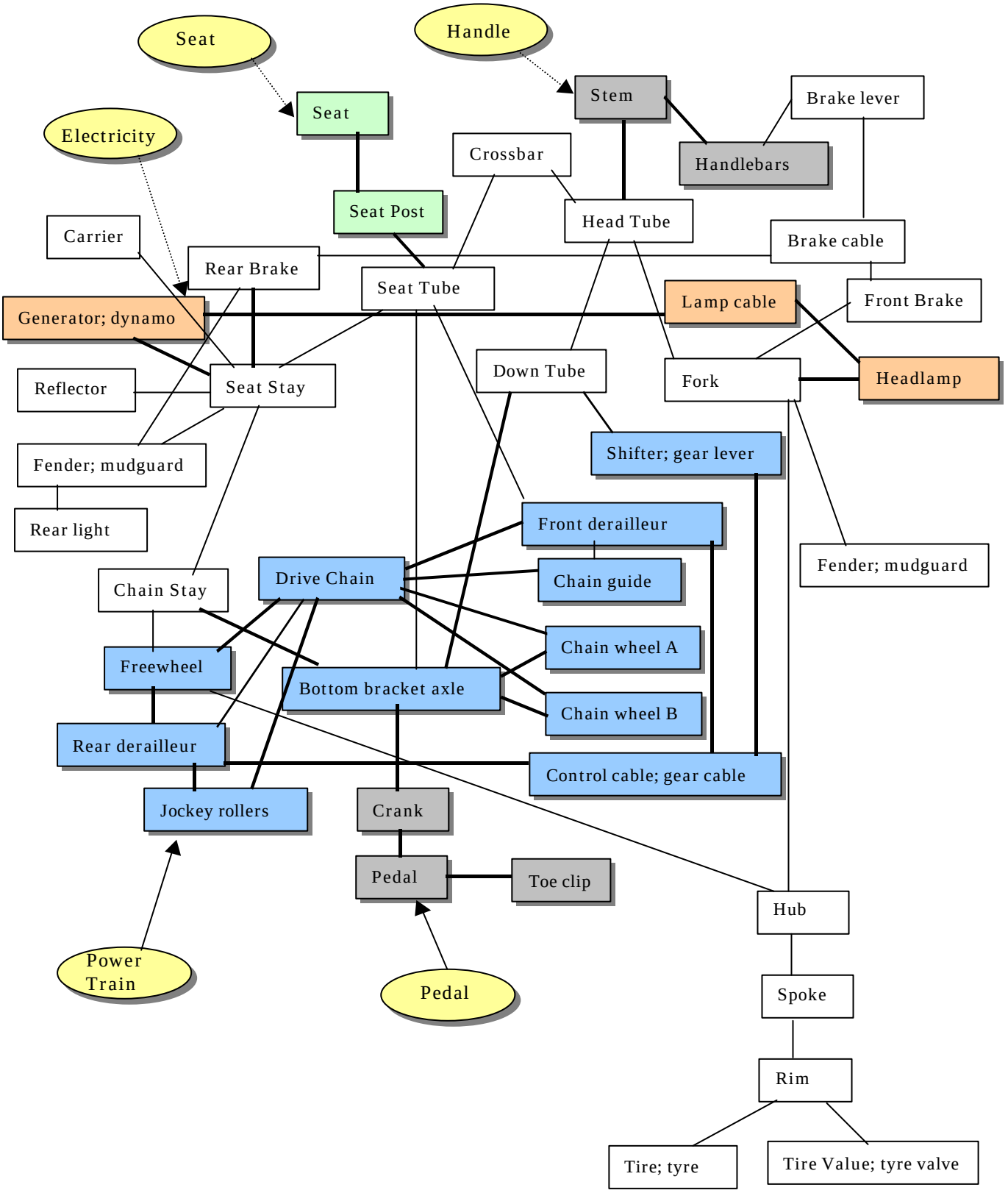


Fig 7: An Example of concrete Patterns for FOPs : ”Handle”, “Seat”, ”Electricity”, “Power Train”, “Pedal” in Modeling a Conceptual Bicycle.

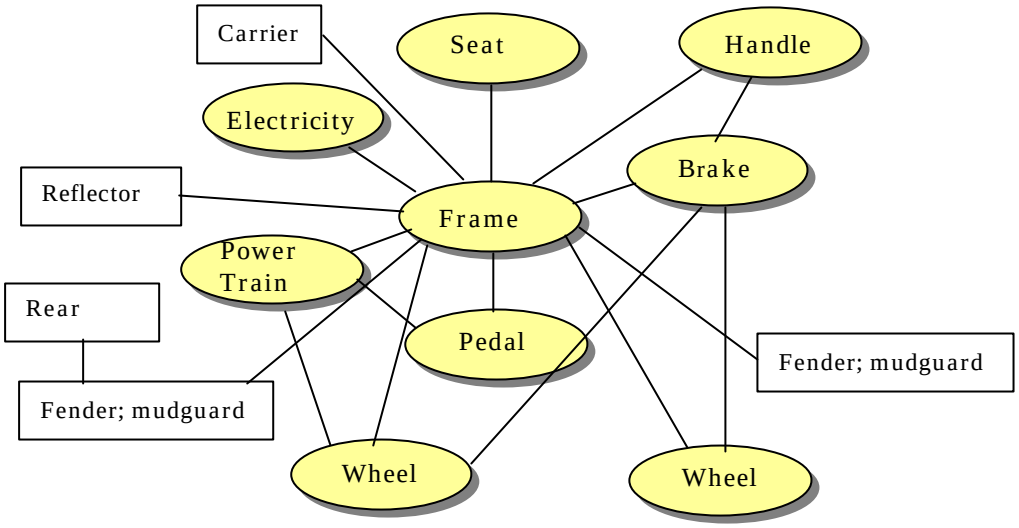


Fig 8: An Example of concrete frameworks for FOF: ”Sport” in Modeling a Conceptual Bicycle

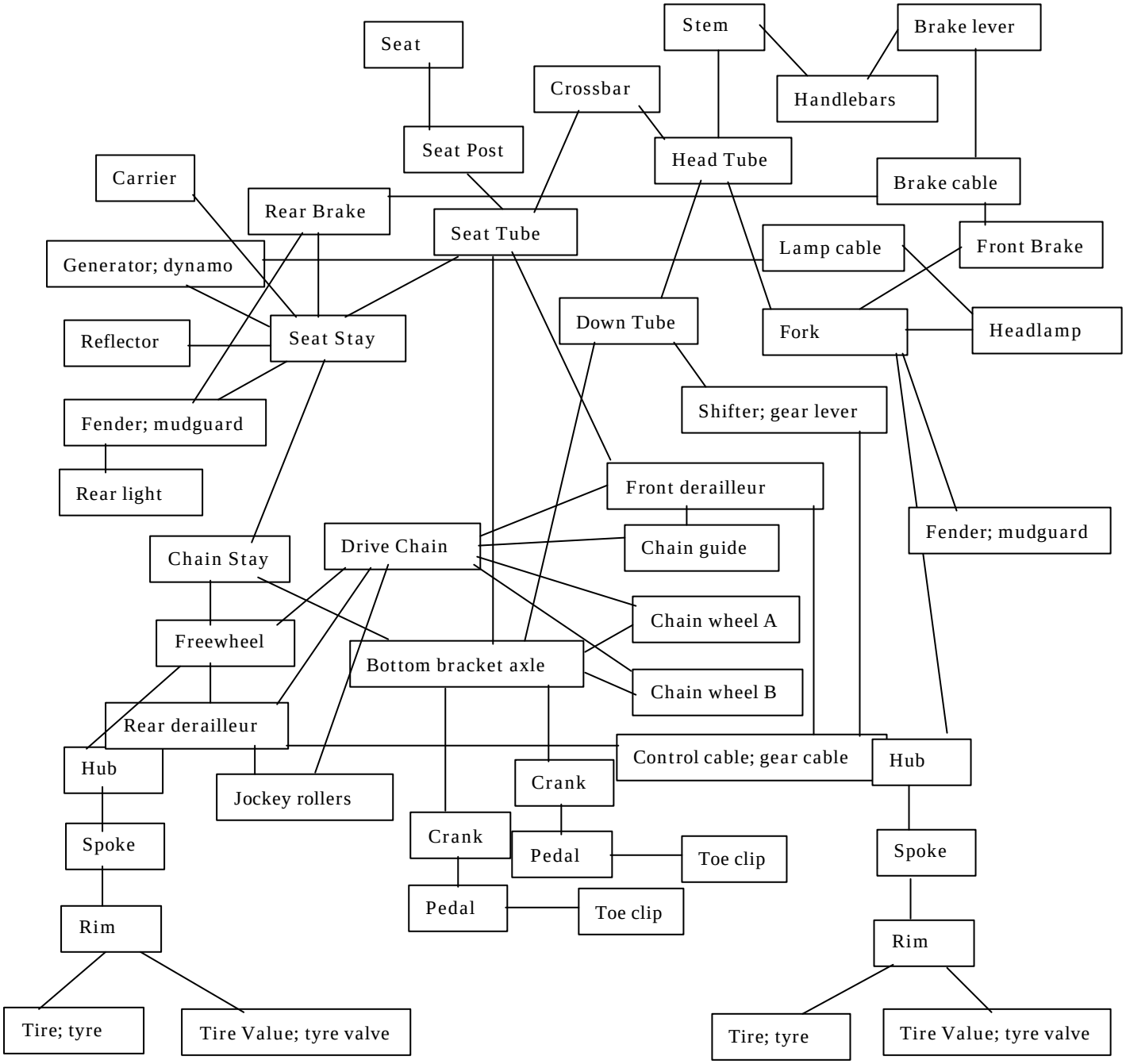


Fig 8: An Example of concrete enterprise objects for “1990s Style Sport bicycle” in Modeling a Conceptual Bicycle

References

[Bapat 94] Bapat, S. *Object-Oriented Networks, Models for Architecture, Operations, and Management,* PTR Prentice Hall, 1994.

[Blaha 98] Michael Blaha, William Premerlani. *Object-Oriented Modeling and Design for Database Applications.* Prentice Hall, Upper Saddle River, N.J., 1998.

[Booch 91] Grady Booch. *Object-Oriented Analysis and Design with Applications, 1st ed.* Benjamin/Cummings, Redwood City, Calif., 1991.

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

[Booch 96a] Grady Booch. *Object Solutions: 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 elements Part 1: Framework for the specification and standardization of elements.* ISO/IEC FDIS 11179-1
[ISO/IEC FDIS 11179-2]	*Information technology-Specification and Standardization of elements Part 2: Classification of elements.* ISO/IEC FDIS 11179-2
[ISO/IEC 11179-3]	*Information technology- Specification and Standardization of elements Part3: Basic Attributes of Elements* ISO/IEC IS 11179-3, 1994.
[ISO/IEC 11179-4]	*Information technology- Specification and Standardization of 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 elements Part5: Naming and identification principles for elements.* ISO/IEC IS 11179-5, 1995.
[ISO/IEC 11179-6]	*Information technology- Specification and Standardization of elements Part6: Registration principles for 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.