

# **MAG's prebuilt data model (business level)**

## **A universal prebuilt data model in order to avoid reinventing the wheel**



### **#1 - IMPORTANT NOTE REGARDING THE SCOPE OF THIS DOCUMENT**

This version is limited to the 'REALITY', 'HUMAN RESOURCES' and 'UTILITIES' Domains of Business Objects as defined in the MAG's data enterprise architecture document (also freely downloadable from the MAG's website). Further versions of the prebuilt data model will describe the other domains and will refine existing ones.

You must also take into consideration the fact that the data models described in this document are dedicated to the business level, that is to say without handling the Data Categories as defined in the MAG's data enterprise architecture document mentioned above.

In order to meet the need of the Master Data modeling, further publications will propose the derivation of the MAG's business prebuilt data models to a set of logical data models compliant with the MAG's derivation rules.

As said in other MAG's publications, the MDM prebuilt data model relies on global enterprise data architecture that is not reliant or specific to the MDM field. A sustainable MDM modeling requires resilient data architecture at the level of the enterprise. As this type of modeling is not easy to handle from scratch, the MAG community proposes data modeling rules and prebuilt data models.

### **#2 – DISCLAIMER ABOUT THE TRANSLATION FROM FRENCH TO ENGLISH**

The author of this document is not a native English speaker. Therefore, UK members are welcome to contribute with their corrections of any area of misunderstanding. Do not hesitate to contact either the author or to post proposals on the MAG's public group.

Version: September 09, 2008

Authors:

Pierre Bonnet – [pierre.bonnet@orchestranetworks.com](mailto:pierre.bonnet@orchestranetworks.com) – Consultant – Orchestra Networks.

## About the authors

### Pierre Bonnet



Pierre Bonnet has worked for more than 15 years as a consultant in the IS architecture field and IT project management.

After several publications of best practices in the SOA and web services field, he participated in the launching of the non-profit organization Praxeme Institute and became co-author of the enterprise methodology Praxeme with Dominique Vauquier (recognized methodologist and main author of Praxeme) and Philippe Desfray (cofounder of Softeam/Objecteering and member of

OMG in particular co-author of the UML notation). For more information about Praxeme see:

<http://www.praxeme.org>.

At the end of 2007, Pierre Bonnet launched the Sustainable IT Architecture community (S-IT-A) that rallies companies around three key principles securing IS overhaul in a progressive way, for more sustainable information systems and software: SOA maturity matrix and the associated governance (Cosmetic SOA, Overhaul SOA, Extended SOA), the ACMS (Agility Chain Management System), and the criteria for evaluating enterprise methods in the context of the IS overhaul. Those principles are also described in the S-IT-A's book, publisher ISTE – Wiley. For more information see: <http://www.sustainableitarchitecture.com/home>.

In 2000, with two associates, he co-founded Orchestra Networks, a software company in the MDM field (Master Data Management). For more information see: <http://www.orchestranetworks.com>.

### Dominique Vauquier



Dominique Vauquier is an experienced modeler and methodologist. As a methodologist, he launched the initiative for a public method and created Praxeme, the enterprise methodology. The initiative is backed by many enterprises and organizations (SAGEM, SMABTP, French Defense, AXA Group, etc.). See: [www.praxeme.org](http://www.praxeme.org).

As a modeler, he led several missions in various contexts (defense, insurance, industry, energy, pharmacy, distribution) and expressed his expertise in methodological guides.



The current version of this paper benefited from the semantic modeling approach of Praxeme. Next versions will take advantage of existing models. Dominique established for enterprises and will more rigorously apply the modeling principles.

This first version focuses on the information, as a result of the structural modeling. The following versions will add actions and transformations that the functional and contractual modeling techniques explore. See: "Guide of the semantic aspect".

## Other information about the MDM Alliance Group

On the MAG's website you can get other information including a complete pack of slides for delivering a one-day course around the MDM modeling procedures and the complete MAG's Guidebook that describes the MDM modeling procedures. You can also find out information about delivering prebuilt reference data models.

**URL to the MAG's website:**

<http://www.mdmalliancegroup.com/>

**URL to the MAG's public group for exchanging about MDM modeling:**

<http://groups.google.fr/group/mdmalliancegroup-public>

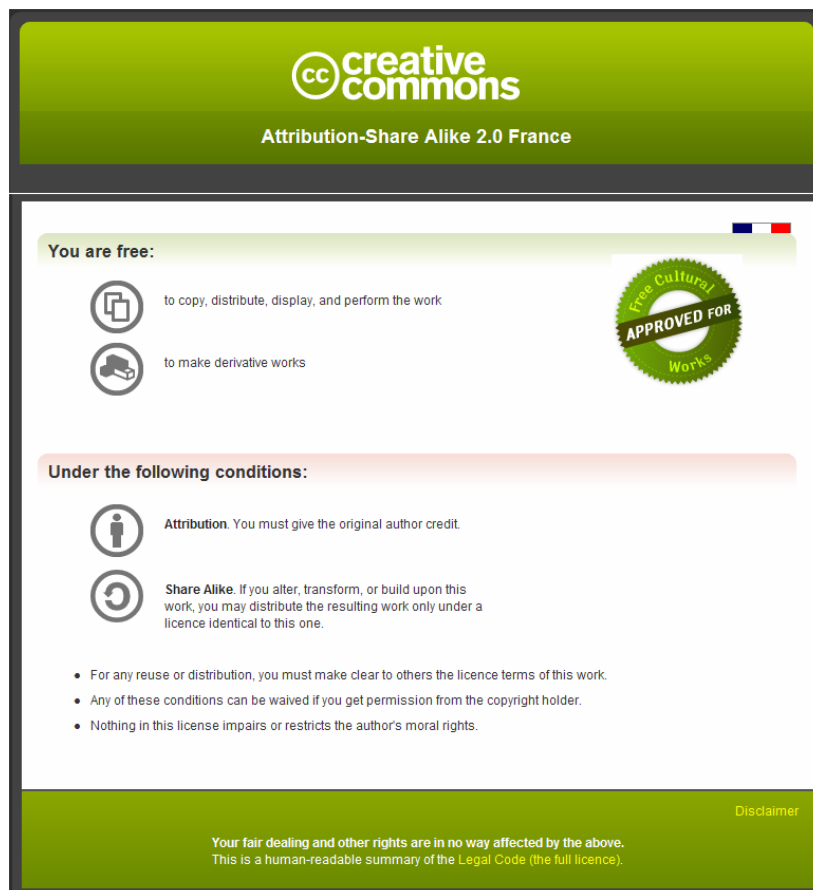
## Version notes

Date	Author	Note
September 09, 2008	Pierre Bonnet (Orchestra Networks) and Dominique Vauquier (Praxeme Institute)	First version published onto the MAG's web site

## Conditions of use

You can use this document and prebuilt data models for your own needs and publications under the condition to quote the source (author credit):

**'Pierre Bonnet, Orchestra Networks, MAG's prebuilt data model'**. This document and prebuilt data models are protected by a **Creative Commons**, described below. This document is freely downloadable via the MAG's website: <http://www.mdmalliancegroup.com/>. MAG stands for MDM Alliance Group.



The image shows a Creative Commons license banner for Attribution-Share Alike 2.0 France. It features the Creative Commons logo and the text "Attribution-Share Alike 2.0 France". Below this, it lists the freedoms granted: "You are free: to copy, distribute, display, and perform the work" and "to make derivative works". It also lists the conditions: "Attribution. You must give the original author credit." and "Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one." A green seal on the right says "Free Cultural Works APPROVED FOR Works". At the bottom, there is a disclaimer: "Your fair dealing and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full licence)." and a "Disclaimer" link.

# TABLE OF CONTENTS

- 1. Objectives ..... 5**
  - Other restrictions ..... 5
  - Related documents ..... 6
  - Reading notes ..... 7
- 2. Reminder about the business objects domain ..... 8**
- 3. Recording the history of the data changes..... 9**
- 4. ‘Utilities’ domain ..... 10**
  - 4.1. Semantic of Period..... 10
  - 4.2. Semantic of Classification (Taxonomy)..... 12
  - 4.3. Semantic of Event ..... 13
  - 4.4. Semantic of Thesaurus ..... 14
- 5. ‘Reality’ domain..... 16**
  - 5.1. Semantic of Party Role ..... 16
  - 5.2. Semantic of Asset and Party ..... 19
  - 5.3. Semantic of Party (Organization and Person) ..... 20
  - 5.4. Semantic of Party and its associations ..... 21
  - 5.5. Semantic of Asset ..... 22
  - 5.6. Semantic of Address and Country ..... 23
  - 5.7. Semantic of Contact..... 26
  - 5.8. Utilization of events..... 27
- 6. ‘Human Resources’ domain..... 28**
  - 6.1. Semantic of Position and Project ..... 28
  - 6.2. Semantic of Skill and Course..... 30
  - 6.3. Semantic of Pay ..... 31
  - 6.4. Semantic of Employee ..... 33
- 7. Appendices ..... 34**
  - 7.1. Lookup tables ..... 34
    - 7.1.1. UTILITIES domain..... 34
    - 7.1.2. REALITY domain ..... 34
    - 7.1.3. HUMAN RESOURCES domain ..... 35

## 1. Objectives

This document presents a proposal for business data models for three central domains of business objects as defined in the MAG's modeling procedures: 'REALITY' domain, 'HUMAN RESOURCES' domain and 'UTILITIES' domain. These data models are sufficiently generic to be universal. For example, we don't model specific data structures for Customer, Personnel, Organization... but a more generic representation that relies on the concepts of 'Party' and 'Party Role'. Using this approach, we avoid setting up siloed reference databases which are unfortunately often noted when using specific functional approaches such as PIM (Product Information Management) and CDI (Customer Data Integration) modeling.

These business data models provide the basis for modeling your own business data models without reinventing the wheel.

As indicated on the first page, these data models are located at the business level; this is a semantic modeling that describes information without taking into account the logical derivation rules required to obtain an XML Schema at the end. It is the reason why the business model relies only on associations and not on nested data types, as is possible at the logical level. Consequently, the logical concept of 'Data Category' needed to set up boundaries between data structures and ensure an efficient mechanism for the loosely coupled data is not necessary (see also our MAG's modeling procedures, freely downloadable from the MAG's website).

You can build the derivation from the business model to logical model by using the derivation rules that are defined in our MAG's modeling procedures and then build the Data Category for MDM.

As the data models are located at the business level, they can't be used directly to generate a physical data model such as XML Schemas or DDL in SQL. A further logical modeling will be required so as to take into account both upgradeability and performances goals. The derivation rules from the business level to the logical level are described in the MAG's MDM modeling procedures.

Even if you can't reuse directly our prebuilt data models, it could be interesting in a first time, to use them in order to benchmark your own data models. For example, the prebuilt data model for recording the Address object is very generic and tries to handle many types of format. Obviously, if the information system needs a subset of these types, a simplification of the initial model remains possible.

### Other restrictions

At this stage of our work, the prebuilt data models deal with the static view of the modeling. They don't encompass elementary<sup>1</sup> and extended<sup>2</sup> operations as defined in the MAG's modeling procedures. Those operations and business objects' life-cycles will be defined later either in the MAG's community or in the context of each company.

The prebuilt data models are more dedicated to propose a universal data structure rather than an exhaustive list of attributes. The added-value of these prebuilt data models resides mainly in the relations that are defined between business concepts.

---

<sup>1</sup> The elementary operations are CRUD operations (Create, Read, Update, Delete). Those operations are directly implemented with help from the default CRUD brought by the MDM tool.

<sup>2</sup> The extended operations stem from business objects' life-cycles. Their implementation doesn't rely on the default CRUD brought by the MDM tool.

## Related documents

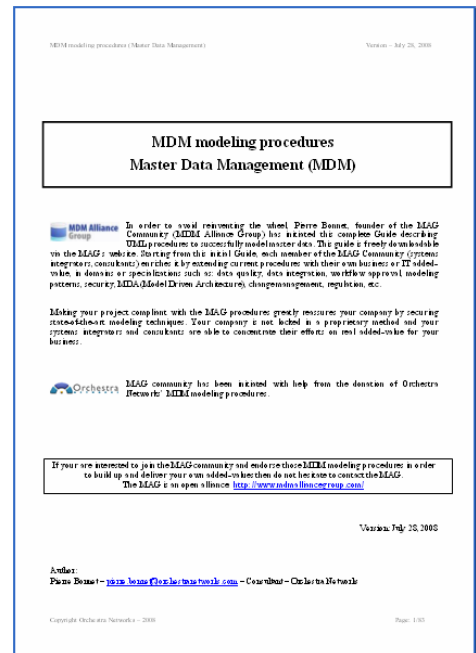
### THE MAG'S MODELING PROCEDURES GUIDEBOOK

Companies are regularly facing with bad quality of their master data. In addition to being shared between several business data silos, these data are often holding a content directly driven by the business case or context. Such a context is particularly growing today in a continuous manner to follow IT systems enlargement and progress: various adaptations by channels, partners, subsidiaries, environments, etc.

The challenge around data standardization and mastering becomes more significant when taking into account business requirements and policies (SOX and Solvency II) and furthermore existing best practices in term of "IT governance" (COBIT, ITIL standards, ...). These concerns are particularly relevant since they require revealing how master data update and interaction log processes are documented and tool based.

To address quality issues of these data, several software solutions exist nowadays focused on a new domain called MDM (Master Data Management). Most advanced ones allow a unified management of the data regardless the area it is related to. It could either concern business (Product, Party, Accounting...), organizational (Directory, Structure...), functional (setting...) or technical (configuration...). This unified management allows, more easily, pooling of governance processes acting for data manipulation/stewardship and associated log events administration.

Exclusive of tool consideration, the success of a MDM project requires deploying a tailored approach in term of modeling and organization. It is a matter of controlling information modeling procedures, particularly master data models design and related management processes, including those concerning data integration between systems. In this guidebook, we will see that processes as mentioned above are based on design best practices as established since long time ago and used in several domains: Entity-Relationship modeling, object oriented approach, configuration management (versioning, filling by contexts<sup>3</sup>), design by contract, etc.



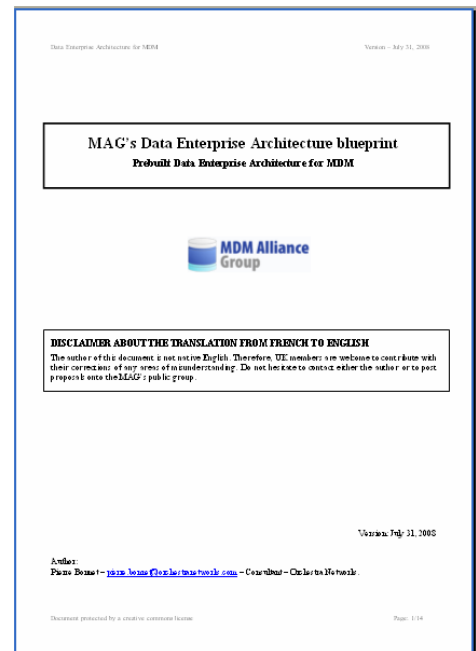
Freely downloadable from the MAG's website

### THE MAG'S DATA ENTERPRISE ARCHITECTURE

The MAG Data Enterprise Architecture provides a very simple and strong global data architecture at the level of **Domains of business objects** and **Data categories** as defined in the MAG's modeling procedures. From an UML point of view, each Domains of objects and Data category is represented by a package. Inside those packages, business objects and administrative objects are gathered. **Consequently, master data are described by taking into account that data architecture.** The detailed descriptions of business objects and administrative objects are not defined in the MAG Data Enterprise Architecture blueprint. Those data structures must be bespoke modelled and/or bought on the market.

According to the MAG modeling procedures applied to logical models, nested data types are not possible between Data categories. Each Data category is a building block and follows the loose coupling principle. The relationships between those Data categories are handled via relational mechanisms including multivalued attributes (list of Foreign Keys). With help from those modeling procedures and the MAG Data Enterprise Architecture blueprint, companies can start their data modeling without reinventing the wheel and in a more efficient way by reducing the risks of data modeling.

When using prebuilt data models with detailed data structures for business and administrative objects, companies have a great and strategic interest in ensuring their integration inside the MAG Data Enterprise Architecture Blueprint. Without taking into account that goal, companies will face with huge risks of useless complex data models relying on bad nested data types and duplications of information because Data categories and Domains of Business Objects / Administrative Objects are not respected. This approach is also strongly recommended and relevant when software packages are used. Indeed, in order to enhance the independence from software packages it is required to model and manage reference data via a data model that doesn't stem from the software packages' databases (ERP, CRM...).



Freely downloadable from the MAG's website

<sup>3</sup> Also named 'Variants' in other publications.

## Reading notes

- Understanding the prebuilt data models requires a basic knowledge of UML modeling, especially the use of qualifier attributes and association classes.
- The UML diagrams presented here can be easily read via a PDF reader by using zoom effects.
- We will highlight the data models with help from several detailed views.
- As the prebuilt data models are located at the business level, the object identifiers (OID) are not mentioned and will be treated when deriving to the logical data models: Only business identifiers are presented when needed.
- Note about the color coding. The yellow color is a way of identifying classes located in other domains of business object. The thistle color is used for identifying the lookup tables. Those tables are presented only to facilitate the understanding of the models. An appendix, at the end of this document, presents the exhaustive list of those tables with some examples for possible values.

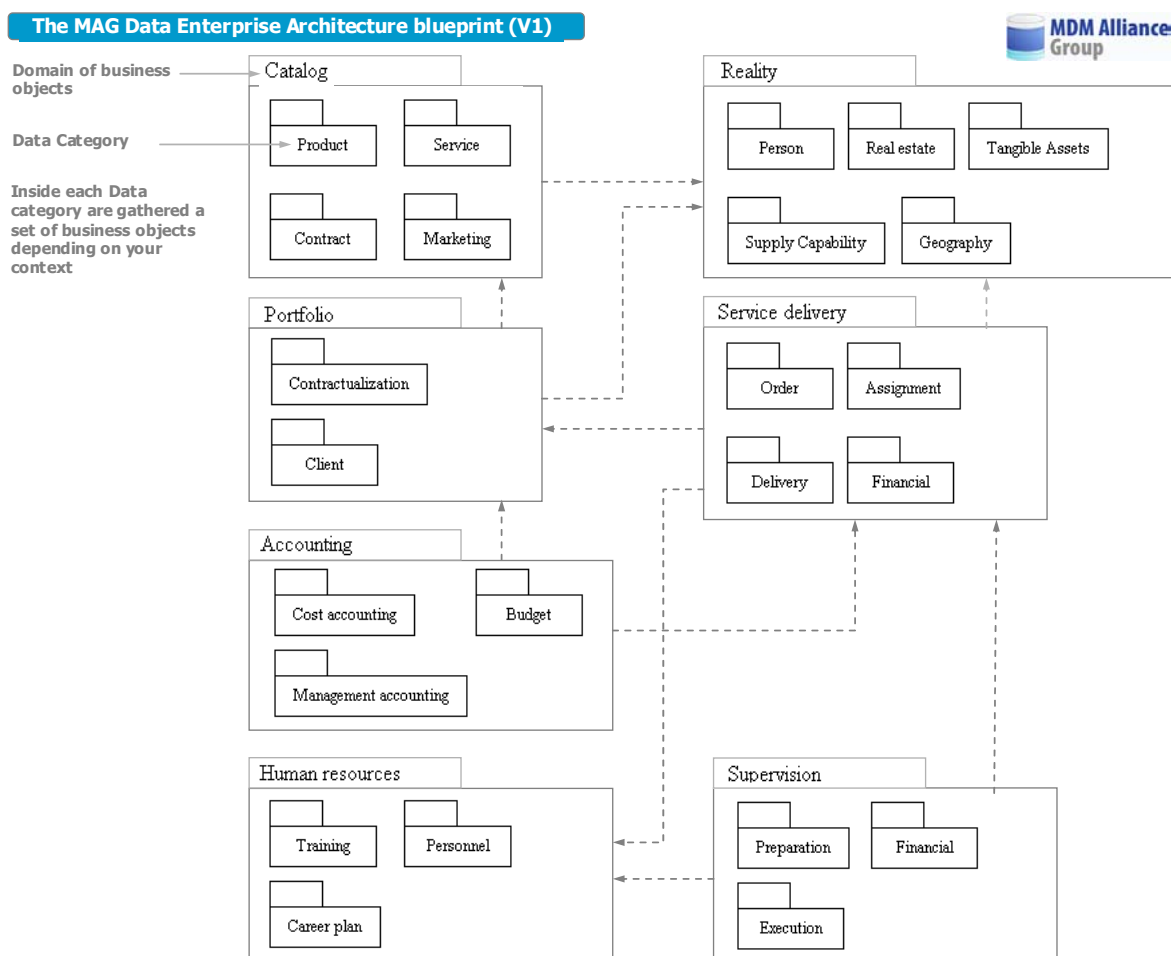
## 2. Reminder about the business objects domain

The data modeling is based, first of all, on a first level of structuring of information, based around the concept of business objects domain. The modeled data will be gathered as business objects within these domains.

A business-object is built from a main semantic class associated, if required, to complementary classes of a shorter semantic range. For example, a business object “Order” may be modeled with a header part (main class) and several order items (associated class).

The domains are valid for modeling master data and furthermore for the whole information system modeling. It is therefore a structuring of the business knowledge. **This is an object oriented approach, at high level, that remains stable over time.**

Each domain is represented as a package in UML notation. The dependency links between these packages summarize the links between information. The reduction of the dependencies between the business objects domains is required. The domains will form the first level of access to the MDM administration tool. The prebuilt domains of business objects and data categories are as follows:



The dependency links between the domains of business objects can be changed so as to take into account your own context. Those links depend on the detailed content of business objects inside each domain.

In the scope of this document we only describe the content of the REALITY and HUMAN RESOURCES domains at the level of business, in other words without defining the boundaries of Data Categories since this is a concern at the logical level. We add a ‘UTILITIES’ domain so as to describe transversal needs such as the Period (concepts related to the time), the Classification and some basic data types.



### 3. Recording the history of the data changes

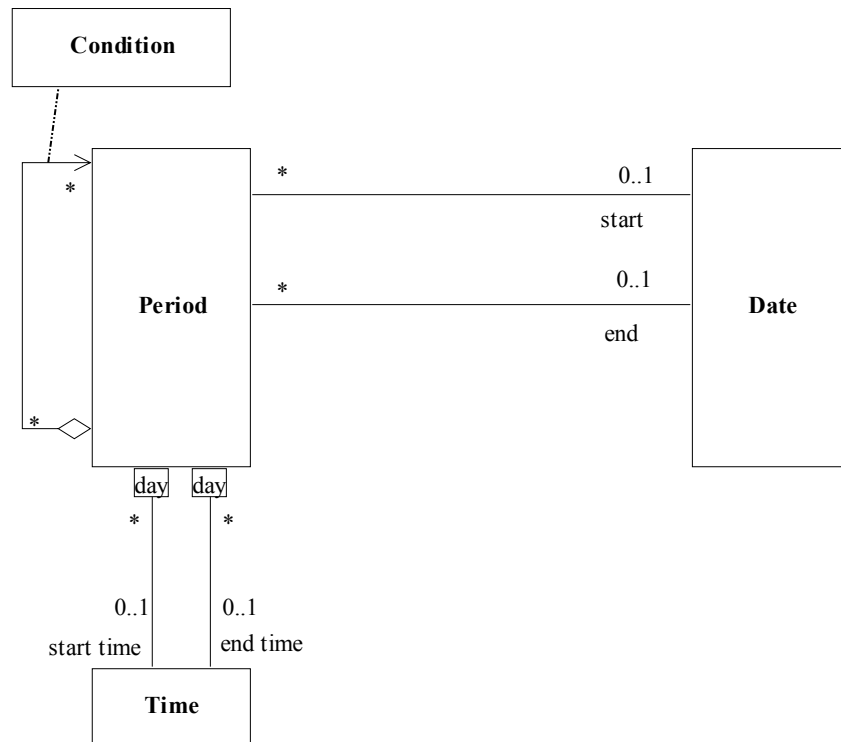
A further specific paper about the management of the time will be proposed. For the present work, we describe a dedicated concept allowing the management of the time, called 'Period' (see the Utilities domain). In order to provide the reader with some examples of the utilization of this 'Period' concept, the data model shows its utilization for some business entities.

In some companies, all information must be recorded with their history (*Update* is forbidden; only *Insert* is possible). Consequently, it could be very awkward to connect the concept 'Period' with each business entities. In that context, it is clever to consider, at the level of the business modeling, that all information is recorded with their history. It is only at the level of the logical modeling that a generic mechanism is applied, for example via a generic feature brings by the MDM tool (considering the scope of the master data) or another specific implementation of the 'Period' concept.

## 4. 'Utilities' domain

### 4.1. Semantic of Period

The Period concept is used to record the change of data over time.



A period can be defined as an infinite period (always valid) when the end date is not fed.

The condition class allows for registering constraints between periods, such as subtraction (a period with blank periods inside) or a combination of several periods.

The 'Time' class allows for recording the time for a period, for example to declare the opening time.

In a further paper, a proposal about a complementary 'record date' will be discussed. The 'record date' is the date when we knew the information. This date may be different from the actual start and end dates. For example, a pay term may be changed the January 15th (record date) and must be valid from the January 01st. With help from both actual date and record dates it becomes easier to ensure a complete traceability of transactions and response to questions such as: how a bill (pay, financial report, etc.) was calculated?

Consequently, some possible use cases are as follows:

- 'record date '=' actual start date'; this is the usual use case.
- 'record date '>' actual start date'; this is a correction in the past.
- 'record date '<' actual start date'; this is a recording for the future.

When a transaction is launched (for example a bill) it must be possible to precise a specific date of the bill that is not mandatory the current date, may be in the past or in the future (simulation). From this specific date the system must retrieve what it is supposed to know at this date. The information that must be used is given by using the record date.

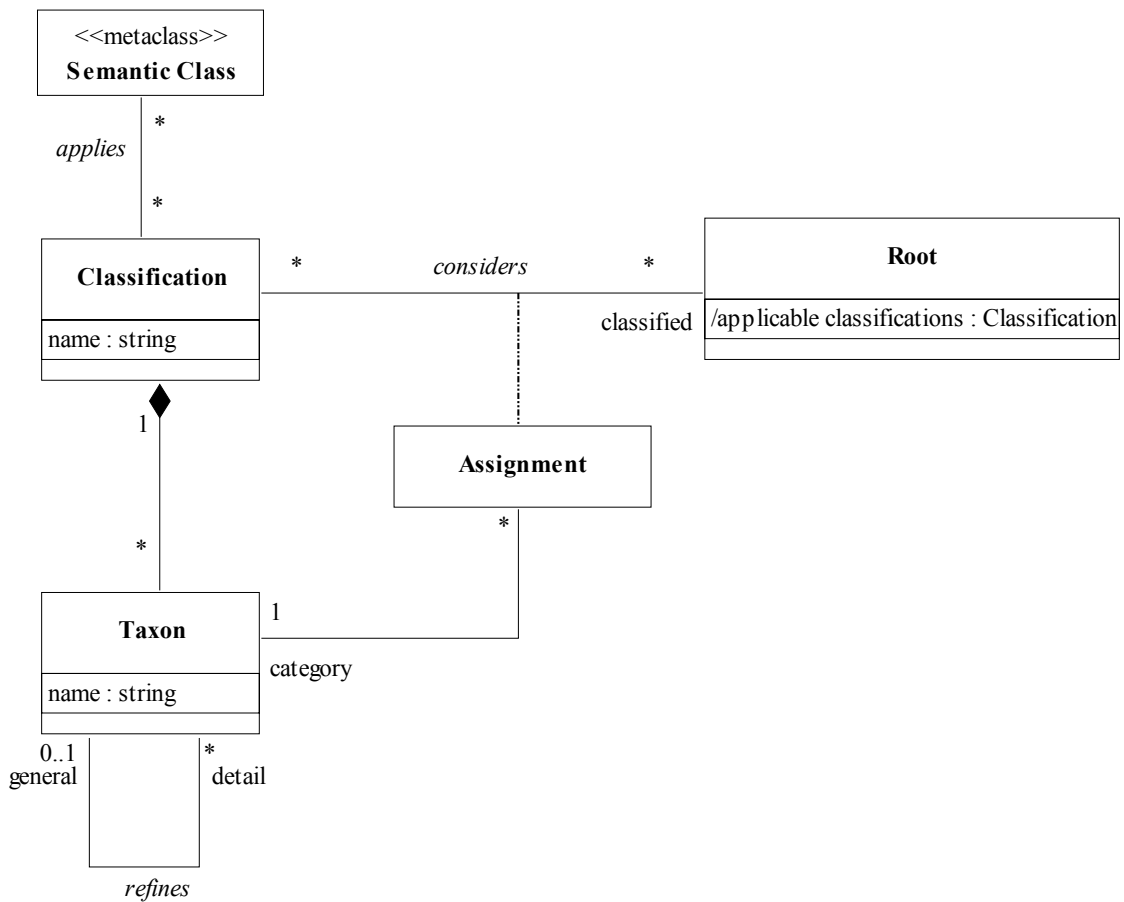
Example data changes history:

Record date	Actual date	Value
01/01	01/01	500
15/01	01/01	1000 – This is a correction in the past because 500 was incorrect

If the current date is comprised between [01/01, 14/01] then the value is 500.

If the current date is higher than 14/01 then the value is 1000.

## 4.2. Semantic of Classification (Taxonomy)



The 'Classification' Class allows for registering the various classification needed, for example: 'Product group', 'Marketing offer', 'Medical principles'...

Since the classification mechanism is applied to the 'Root' class, each business entity of our data model can benefit from the classification. The derived attribute '/applicable classification' (class Root) allows for defining the possible classifications attached to a specific entity. This configuration about the binding between entities and classifications will be managed via the MDM.

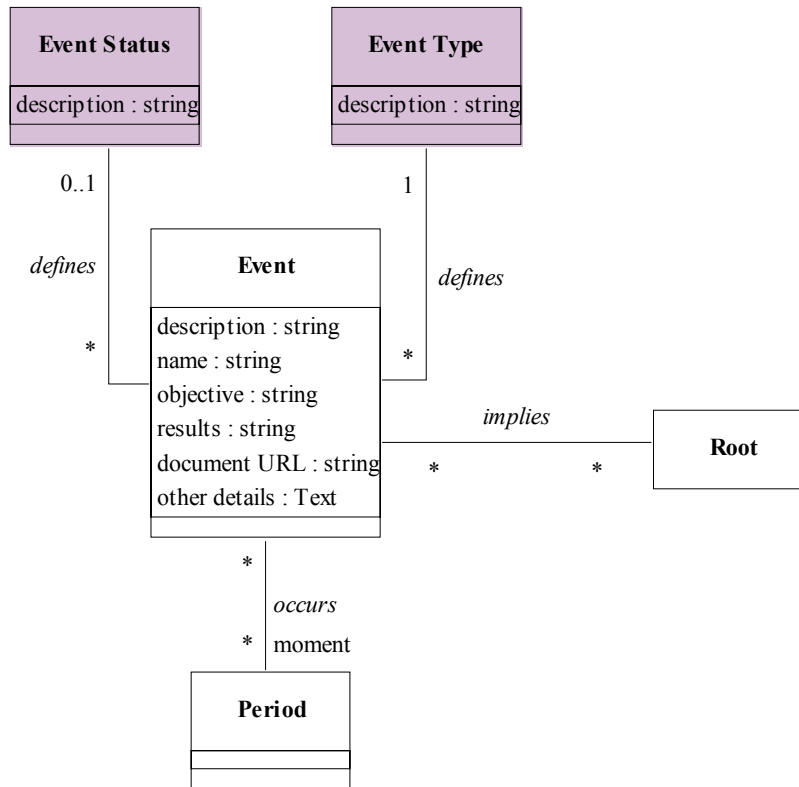
The 'Taxon' class defines the taxonomy in the scope of a particular classification. It is possible to define the taxonomy via a tree by using the 'refines' association.

The class 'Assignment' allows for binding an object with a Taxon in the context of a classification. With this modeling, an object only belongs to one category for a given classification.

### 4.3. Semantic of Event

As for the 'Period' concept, each business entity may be associated with events. This is the association 'implies' between the Root and the Event class. However, beyond this predefined association, the model can set up supplementary associations, for example in order to record which parties are responsible for the management of events (see the semantic of 'Utilization of Events' in the Reality domain).

In a further paper, we will discuss about the interaction between this event management and the events described and handled by the business state machines.



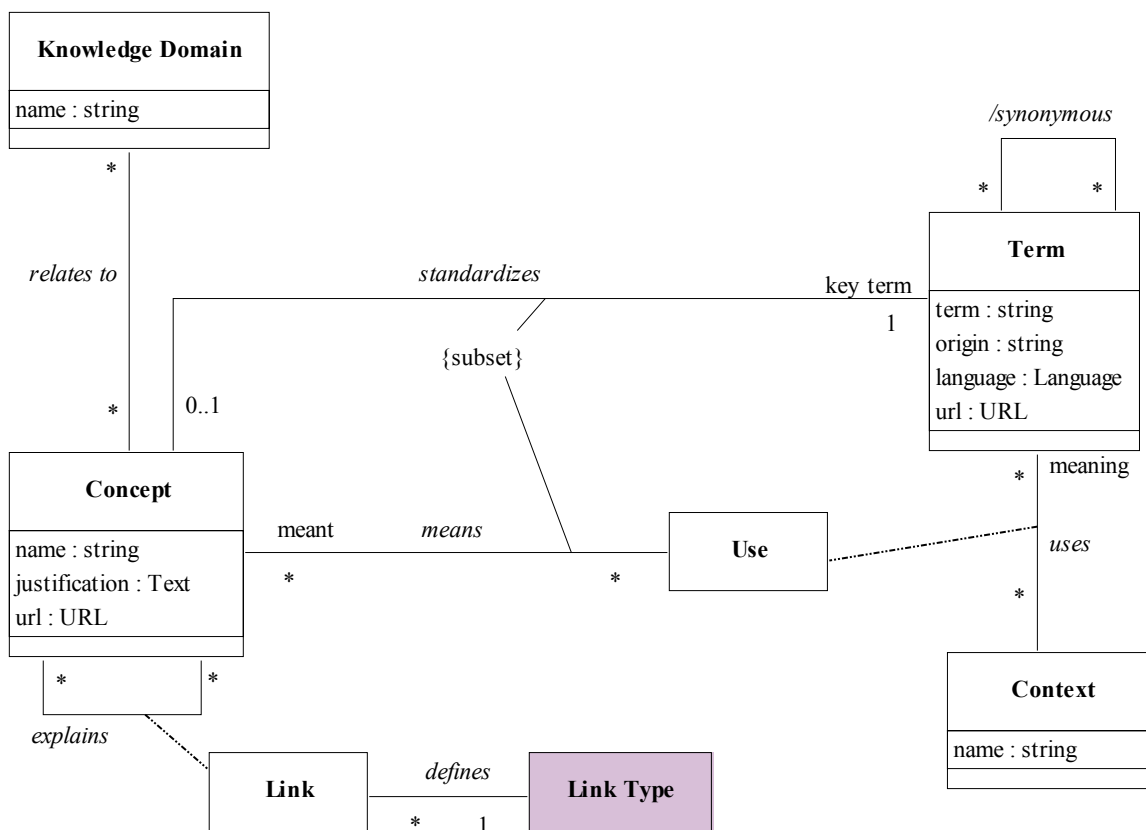
Example of possible values for 'Event Status': Cancelled, Confirmed, Provisional, Close, On going, To be checked...

Example of possible values for 'Event Type': Meeting, Contact, Regular contact, Marketing campaign, Claim management, Machine repair...

### 4.4. Semantic of Thesaurus

The thesaurus is a mechanism used to record and retrieve knowledge expressed by terms. In particular, the thesaurus allows for defining synonyms. As the data model shows, the synonyms are not defined directly between terms but through a more generic mechanism called 'Concept'. For example a concept may be 'Business Object' and the terms that define it may be 'Class', 'Entity', 'Business class', 'Object', etc. This is synonymous terms. Furthermore, concepts may have relationship between them in order to precise information such as: a concept is a generalization or specialization of another concept; a concept is linked to another concept (usual 'see also' in the thesaurus) ; a concept is related to another one ; etc. Rather than define these relations directly between all terms (including the several languages for a term) it is more relevant to handle them at the level of the 'Concept'.

The use cases of the thesaurus are very large. For example, it allows for defining the knowledge for a 'Help system' with a clever user online assistant (synonyms, polysemies, etc.). It may be used for setting up an online diagnostic system that allows a business user to describe, with informal terms, a use case and obtain the relevant parts of a business regulation depending on the user context.



**‘Knowledge Domain’ class :**

The knowledge domain defines a scope of interest. It could be recorded for organizational procedures (claim management, customer relationship, etc.), business goals (business regulation such as SOX, SolvencyII...), business domains (Product, Customer, Third Party...).

**‘Context’ and ‘Use’ classes:**

A term holds a unique meaning for a specific context. For example, in the context of IT field, the term ‘class’ refers to the concept of ‘Object oriented approach’ whereas in the context of Education, the term ‘class’ refers to the concept of ‘Course’.

With help from the ‘Use’ associative class, the concept attached to the term is defined.

**'Term' and 'Concept' classes:**

A term holds a 'meaning' regarding a context. A unique 'Concept' may be defined through several terms (the association 'means' between 'Concept' and 'Use' is a many-to-many). Among these terms, one is stated as the 'key term' for the concept (the constraint 'among' is expressed via the UML constraint 'subset').

A unique concept may be defined by several terms. These are synonymous terms. For example, the concept 'Business Object' is defined by various terms such as 'Class', 'Entity', 'Business class', 'Object', 'Category class', etc. Conversely, a unique term may mean several concepts. For example, the 'class' term is related to the concepts 'Business object', 'IT object', 'Utilities object', etc.

**Association '/synonymous' on the Term class:**

This association is derived because the group of terms that defines a unique concept are retrieved from the 'means' association between 'Concept' and 'Use' classes.

**Association 'Explains' on the 'Concept' class, 'Link' and 'Link Type' classes:**

The concepts are bound together in order to precise usual types of relation between concepts such as: broader concept, narrower concept, related concept, generalization concept, specialization concept, etc. These types of relations are recorded by using the 'Link Type' class. They allow for classifying concepts in the thesaurus.





### **Association 'interacts' of the Party class**

The ternary called 'interacts' between Party and 'Party Relationship Type' allows for describing which roles are played between Parties. This ternary is completed with the 'Party Relationship' class so as to describe specific information regarding the period of relation. The 'role 1' and 'role 2' are defined via the 'Party Relationship Type' that is linked with the 'Party Role Type' through the same 'role 1' and 'role 2'.

Some examples:

- Brian (A person) has the role 'Husband' and Stone (Another Person) has the role 'Wife' in the context of a party relationship type 'Married'.
- Brian has the role 'Customer' and Tato (a legal Organization) has the role 'Seller' in the context of a party relationship type 'Customer relationship'.
- Stone has the role 'Buyer' and Cosmo (a legal Organization) has the role 'Provider' in the context of a party relationship type 'Buyer relationship'.
- Tato has the role 'Employer' and Brian has the role 'Employee' in the context of the relationship type 'Employment relationship'.

### **'Party Relationship' class**

The 'Party Relationship' class is only used in order to record some information about the period of time of the relation. 'Transactional' information such as orders, employment contracts... are defined without setting up direct association the 'Party Relationship'. For example, the 'Human Resources' domain encompasses an 'Employment Contract' class that is not connected with the 'Party Relationship' because this latter only describes a type of relation between two parties regardless transactional information describing this relation. Obviously, constraints could be defined for example to state that an 'Employment Contract' cannot be recorded if an 'Employment' party relationship doesn't exist.

### **'Party Role Type' class**

The possible values for 'Party Role Type' are as follows (these values are filtered depending on the type of the Party):

- Prospective Customer, Customer, Employee, Employer, Provider, Lodger, Buyer, Member, Chairman, Chief Executive Officer, CFO, VP, Manager, Staff member, Doctor, Donor...
- Head Quarter, Divisional Office, Branch Office, Department, Business Unit, Subsidiary, Team
- Located at (e.g.: an employee works in the office X)
- Wife, Husband, Father, Mother, Sister, Brother, Daughter, Son, Grandfather, Grandmother...

If needed, an aggregation could be added in order to set up a rollup structure. For example, the Head Quarter role could be a composition of other roles such as Divisional Office, Business Unit. With help from this composition, it becomes possible to define a Party Relationship Type that acts for several types of roles. This is surely needed for the Party Relationship Type 'Customer Relationship' that could be available of several roles of organization.

### **'Party Relationship Type'**

Some possible values for the 'Party Relationship Type': 'Employment relationship' to define the relationship between the roles Employee and Employer, 'Customer Relationship' to define the relationship between the roles Customer and Organization, 'Married', 'Apartment Rental', etc.

Be aware that a Party Relation Type is defined for two identified Party Role Types. Obviously a set of constraints must be defined in order to state, for example, that a 'Married relationship' is mandatory bound with a role 'Husband' and a role 'Wife'. See also the remark regarding the algebra for the management of the roles (class Party).

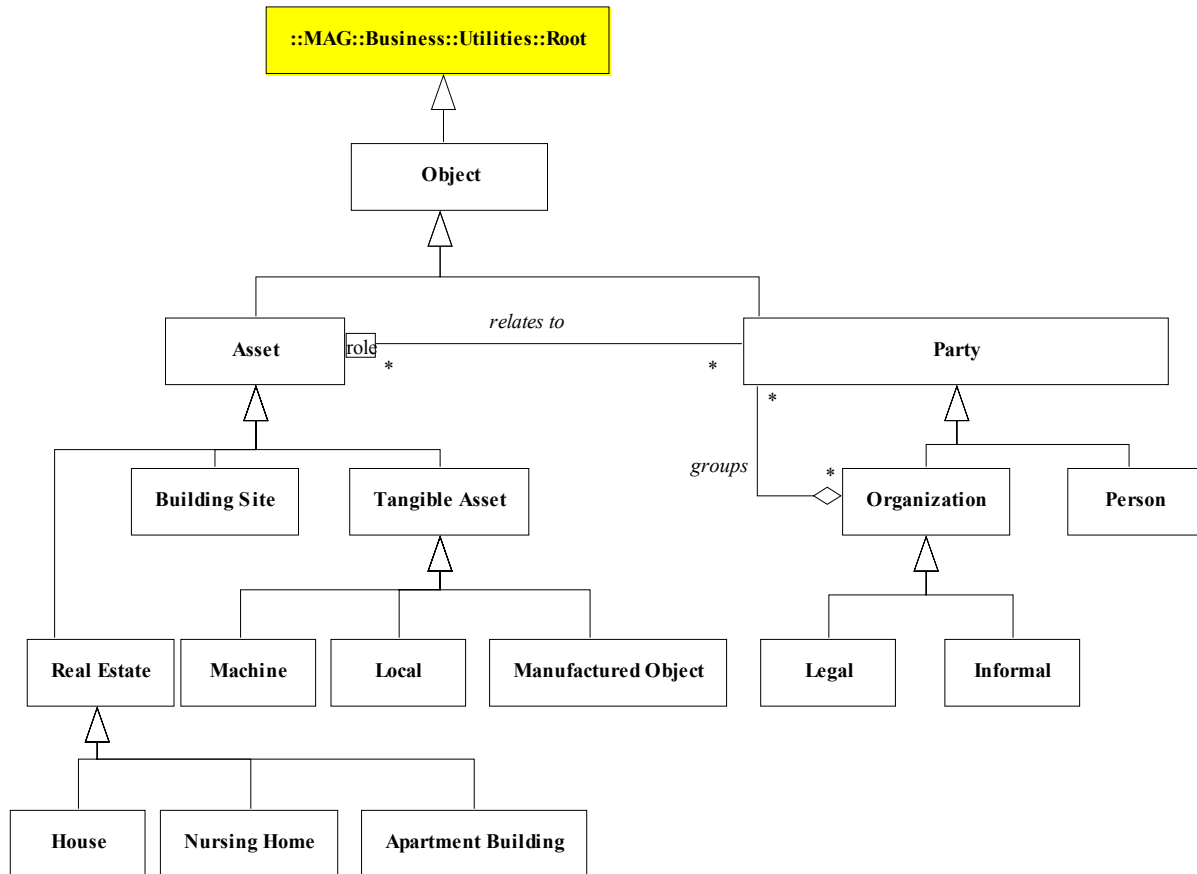
**‘Asset’ class:**

An Asset is any tangible asset, real estate... but also patents, software, etc.

**Association between Asset and Party**

The qualifier attribute 'role' attached to the association between Asset and Party classes ('relates to') allows for defining the involvement of parties regarding the assets: Owner, Manager, Real Estate User, Real Estate Architect IT responsible, Investor, etc.

## 5.2. Semantic of Asset and Party



### Aggregation between ‘Party’ and ‘Organization’:

The aggregation between Party and Organization allows for defining group of organizations and persons. Be aware that the roles of each organization or person in a group are defined via the ternary between Party and 'Party Relationship Type'. For example, an informal organization 'Family' is first of all a group of persons and each person plays a specific role. Those roles (father, mother, sister...) are defined through the ternary association mentioned above.

### ‘Informal’ class:

Example of possible values for the Informal Class: Family, Club, Group...

### ‘Object’ class:

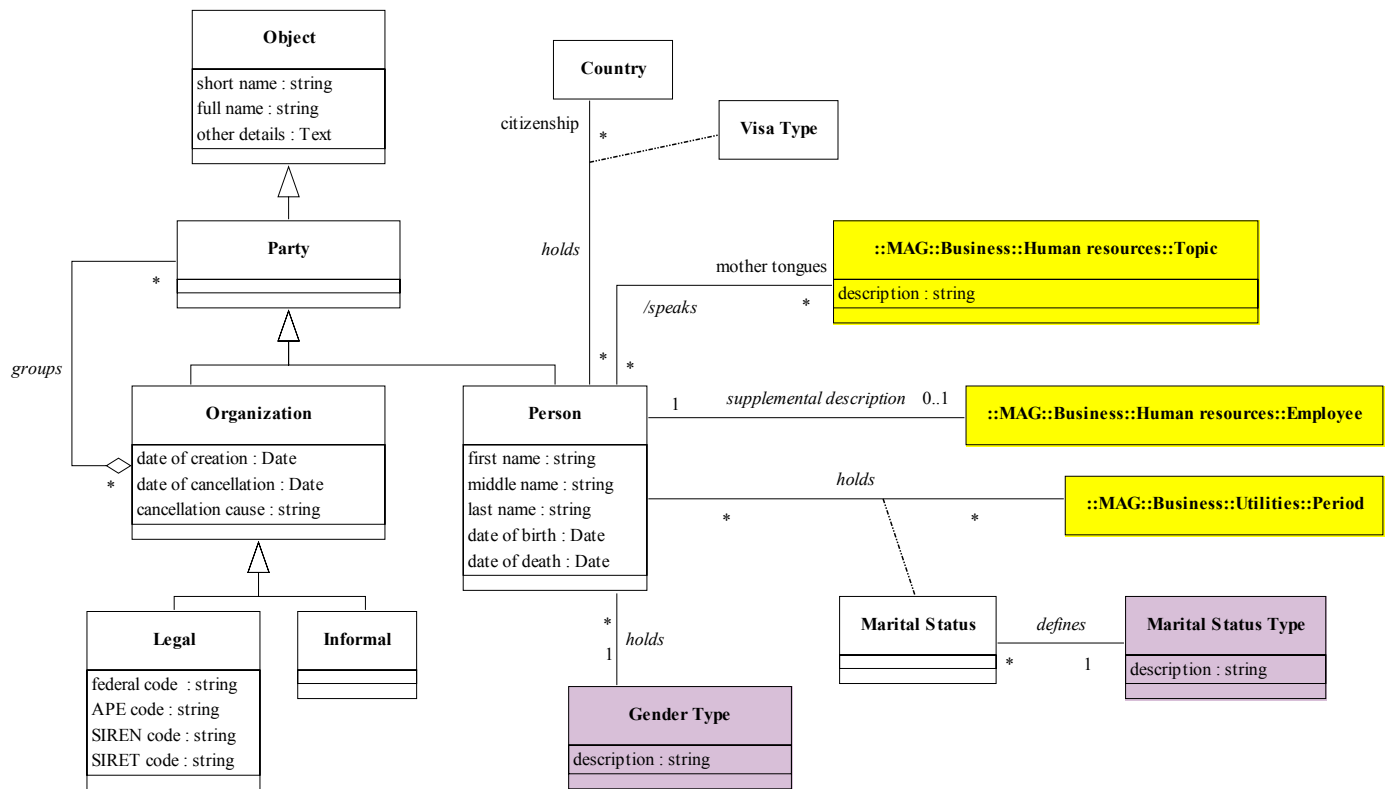
The 'Object' concept is the root of the description of Asset and Party. With help from this root, other key concepts such as Address can be generalized (see other parts of the data model).

### ‘Root’ class:

The class 'Root' is used in order to obtain a generalization class available for all classes, not only those that are already gathered by the 'Object' class. The model doesn't show all the inheritance links with the Root class so as to avoid useless complexity of reading.

Using the 'Root', the data model generalizes some keys concepts, for example the 'Classification' mechanism that allows for classifying any type of semantic entity defined by the data model (see the Utilities Domain to obtain more information about the Classification mechanism).

### 5.3. Semantic of Party (Organization and Person)



#### ‘Person’ class

The skills of Person are described via the Topic class in the HR domain associated to the Party class. Therefore, in the Person class only the native languages are described via the association to ‘Topic’ located in the HR domain. Other languages abilities are described via the Topic class mentioned above.

#### Association between Person and Period classes:

The 'holds' association between Person and Period is a good example of the mechanism used in order to manage the notions related to time. The 'Period' class is described in the Utilities domain. The associative class 'Marital Status' is bound with Person and Period so as each Marital Status is recorded for a period of time.

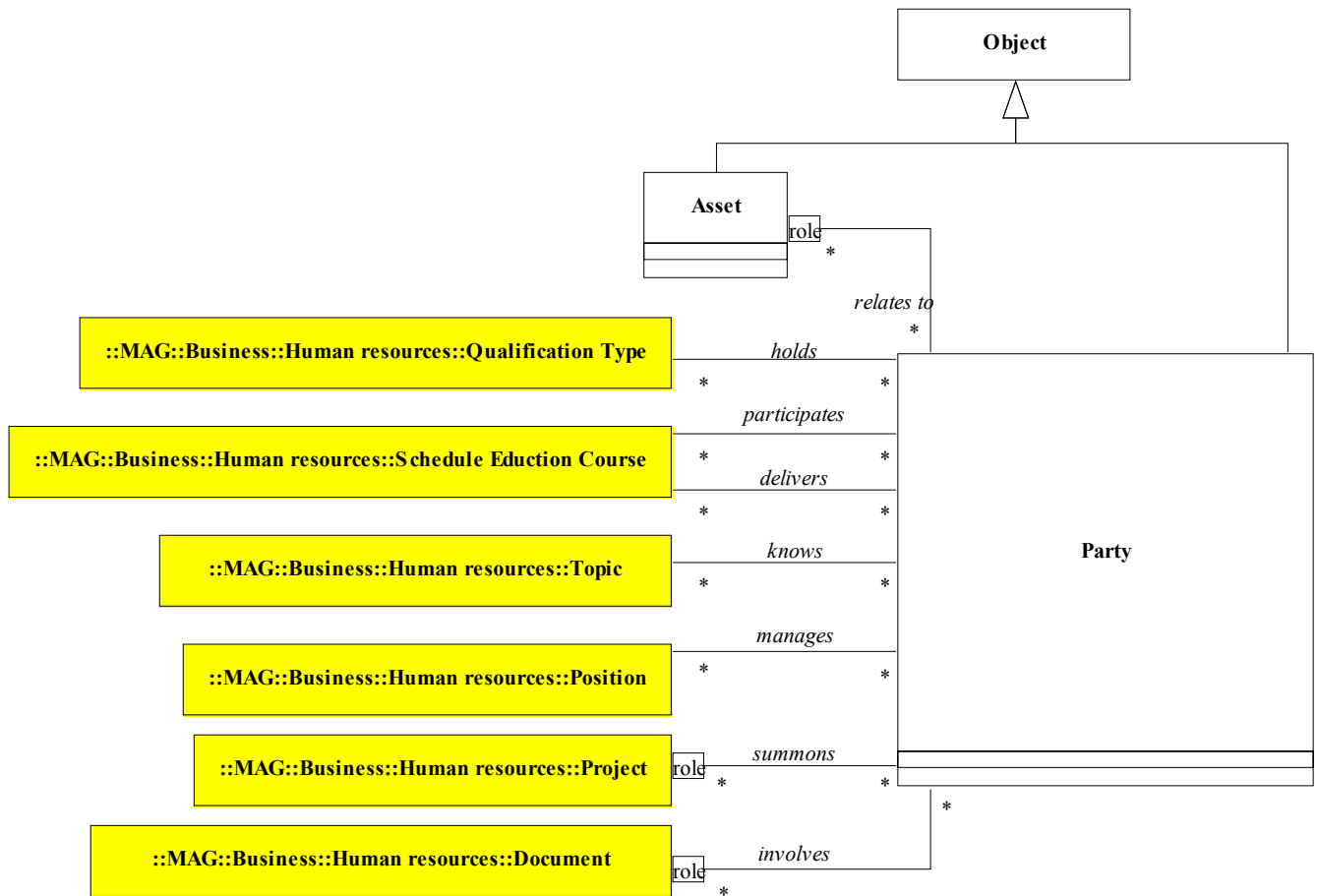
#### ‘Employee’ class and other facets attached to the Person

The Employee class allows for describing supplemental information that are not already described in the generic Person Class and its superior class that is to say the class Party. For example, we may have special information regarding the legal working identification of each employee. You must keep in mind that Employee is a type of role applied to a Person. If a unique Person works for several employers then be cautious that this Employee Class remains unique. Specific information regarding both Employee and Employer are described via the ‘Employment contract’ in the HR domain.

Depending on your context, you can create new classes that describe other facets regarding the Person either in the HR domain or other domains (e.g.: A Customer facet defined in the Portfolio domain).

## 5.4. Semantic of Party and its associations

In order to define the involvements in the projects and skills, each party may have several associations with various objects that are located at the Human Resources domain.



### Associations from the HR domain:

The associations from the HR domain to Party are not oriented whereas it should be in order to be compliant with the dependencies between the domains of business objects. Most probably, the direction is from the HR domain to the REALITY domain that is to say from the HR classes to the Party Class.

Example of possible values for the qualifier attribute 'role' attached to 'summons' association between the Project class and Party class: Owner, Sponsor...

Example of possible values for the qualifier attribute 'role' attached to the association 'involves' between 'Document' and Party classes: Owner, Author, Manager...

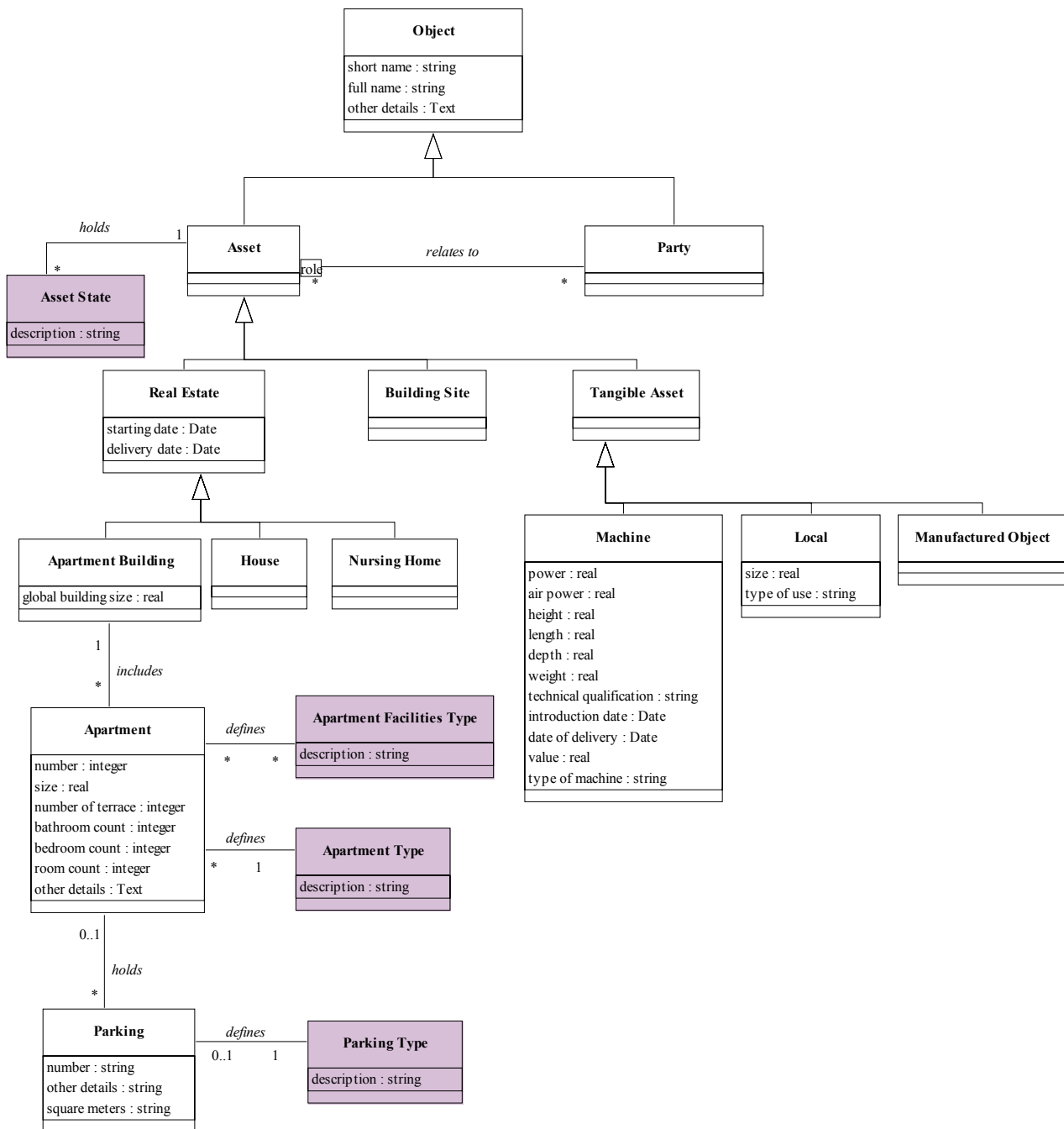
### Other important points already shown before:

Other associations from Party are not described here (see other parts of the data model, in particular the semantic of Contact and the semantic of Event).

This model doesn't present the mechanism used so as to handle roles and relationships between roles. This mechanism is already described in the 'Asset and Party mechanism' data model.

Reminder: all business entity can be associated with a classification mechanism by using the 'Classification' defined in the Utilities domain. Consequently, the Party can be classified, if needed.

## 5.5. Semantic of Asset



In this limited attempt of modeling a Real Estate data structure and other assets, the more important point to consider has already been explained previously. It concerns the relation between Asset and Party through the qualifier attribute 'role' attached with the association 'relates to' (see semantic of Asset and Party mechanism). With help from this attribute, all relationships between assets and parties can be expressed.

As for the others Asset, don't forget that Real Estate is linked with Party through the qualifier attribute 'role' associated with the 'relates to' association. With help from this mechanism, it becomes possible, for example, to mention who are the apartment's lodger and owner... and using the 'Party Relationship' class (see the semantic of Asset and Party mechanism) it is also easy to describe the relationships between parties.

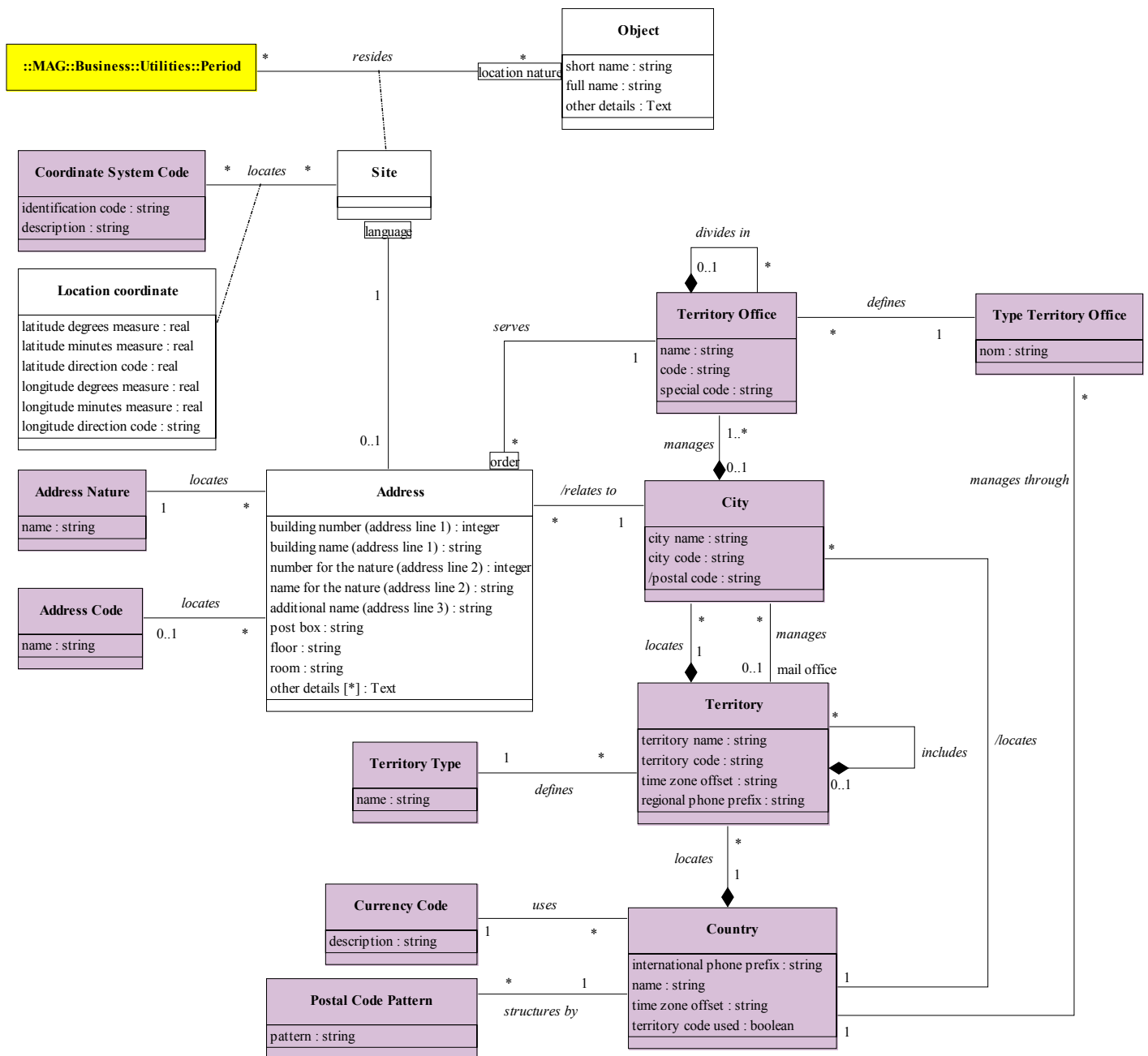
Possible values for 'Asset State': Current State, Delivery State, Future Project, etc.

## 5.6. Semantic of Address and Country

The data model 'Address and Country' should be sufficiently universal to manage any type of address for any country. The 'Territory Office' mechanism allows for describing the administrative organization that manages the mail distribution. Obviously, this organization depends on the country. On the other side, the 'Territory' description is not reliant on the mail distribution; this is a geographical description of the country. Here some examples of possible use of the data model:

- FRANCE: 69130 – Territory code='69' & Territory Office code='130'.
- US: 90001 – Territory code='900' & Territory Office code='01'.
- UK: M1 1AA – Territory Office code='M' (Postal Zone) + '1' (Postal District) + '1' (Sector) + 'AA' (Unity).  
As indicated, for the UK country, the postal code doesn't rely on the Territory code.

Reminder: this is a business data model. Depending on your actual needs and the IT target, the logical data modeling will allow you to optimize the model.



**'Period' class:**

The period is shown in this part of the data model in order to illustrate the use of this mechanism (see also the part about the history of the data changes). Obviously, the addresses are valid for a period of time. The associative class between the 'Period' and the 'Object' allows for describing this concern. It is useful to note that a Period can be defined as an infinite period of time (always valid).

**'Site' class:**

The concept of 'Site' allows for defining several addresses for a unique location depending on the language. The case occurs in multilingual countries. This is the goal of the qualifier attribute 'language' attached to the association between the Site class and the Address class. This mechanism could be used for further needs such as the recording of electronic postal address (to be defined in a further version).

Independently the language, a Site can have location coordinates compliant with a Coordinate System Code (GPS, Galileo...).

**'location nature' qualifier attribute on the relation 'resides':**

The 'location nature' qualifier attribute attached to the association 'resides' of the 'Object' class is used in order to state the utilization of the address. Examples of possible values are as follows: Usual (most of the time this nature is sufficient, except if others are declared), Business, Residential, Second Home, Headquarter, Delivery (where the goods are delivered), Invoicing (where the invoice must be sent), Return (for a customer, the return link designates the site where the goods are to be returned in case of problem), Vacation, etc.

**'Territory Office' class:**

The 'Territory Office' is for example the 'Postal Office' in France or the 'Postal Zone' in the US.

In France, the 'Territory Office' code is composed with three figures. For example the French City 'Orléans' has two postal offices: first for the North (000) and second for the South (100). By prefixing these codes with the Territory code (department) then the complete postal code is obtained (45000 and 45100). In the US, the 'Territory Office' code is composed with two figures.

However, in other countries the Territory Office could be different. For example, in the UK this is the concatenation of the Postal Zone (order=1), the District Zone (order=2) inside a Postal Zone, the Sector (order=3) and finally the Unity (order=4). Reminder: the 'order' is given by the qualifier attribute 'order' attached to the association 'services' between the class Address and the 'Territory Office'.

The type of each 'Territory Office' is recorded with help from the 'Type Territory Office' class

The 'special code' attribute could correspond to the CEDEX code in France (Courrier d'Entreprise à Distribution Exceptionnelle). This code is not mandatory and is added at the end of the complete address. In the US, it could correspond to the four figures that complement the basic ZIP Code (called ZIP+4).

**'Type of Territory Office' class:**

Possible values depending on the country: 'Postal Office' (France), 'Postal Zone' (US), 'Postal Zone' & 'District Zone' & 'Sector' & 'Unity' (UK), etc.

**Qualifier attribute 'order' of the Address class:**

The qualifier attribute 'Order' of the Address class is used when several Territory Offices are used to compose the complete territory code. For example, in the UK this is the concatenation of the Postal Zone (order=1), the District Zone (order=2) inside a Postal Zone, the Sector (order=3) and finally the Unity (order=4).

**'/related to' association between Address and City:**



The city attached to the Address is obtained from one of the Territory Office codes defined via the 'serves' association. Note that the symbol '/' before the name of the association ('/relates to') indicates a derived association, in other words an association that is calculated with help of other associations in the data model.

#### **‘/postal code’ attribute of City class:**

Most of time, the 'postal code attribute' is a concatenation of the code stemming from the Territory and a 'Territory Office' code. The syntax of this attribute depends on the country. The class 'postal code pattern' provides the right syntax for each country.

However, in the UK, for example, the postal code is obtained by the Territory mechanism only (Postal Zone + Postal District + Sector + Unity). In order to know if the territory code must be used or not, the 'territory code used' attribute in the Country class is recorded.

#### **Attribute 'territory code used' of the Country class:**

The value of this Boolean attribute is 'true' if the Territory code is used to compute the postal code otherwise the postal code is obtained only with the mechanism of Territory Office.

#### **‘Territory Type’ class:**

The values of 'Territory Type' may depend on the geographic boundaries. For France: Département, Région. For USA: State, County.

#### **‘manages’ association between City and Territory**

The association 'manages' is recorded when the computation of the postal code must be done with a special Territory.

For example, in France, the city 'Laveyrune' is located in the territory of 'Ardèche (07)' whereas its postal code is composed with the 'Lozère (48)' territory. Consequently the postal code of Laveyrune is 48250 even if its actual territory is '07'.

#### **‘Address Nature’ and ‘Address Code’ classes:**

Example of possible values for the 'Address Nature' class: Street, Road, Lane, Boulevard, Way, Building plot... The values are reliant on the language of the Address (should be managed directly via the MDM).

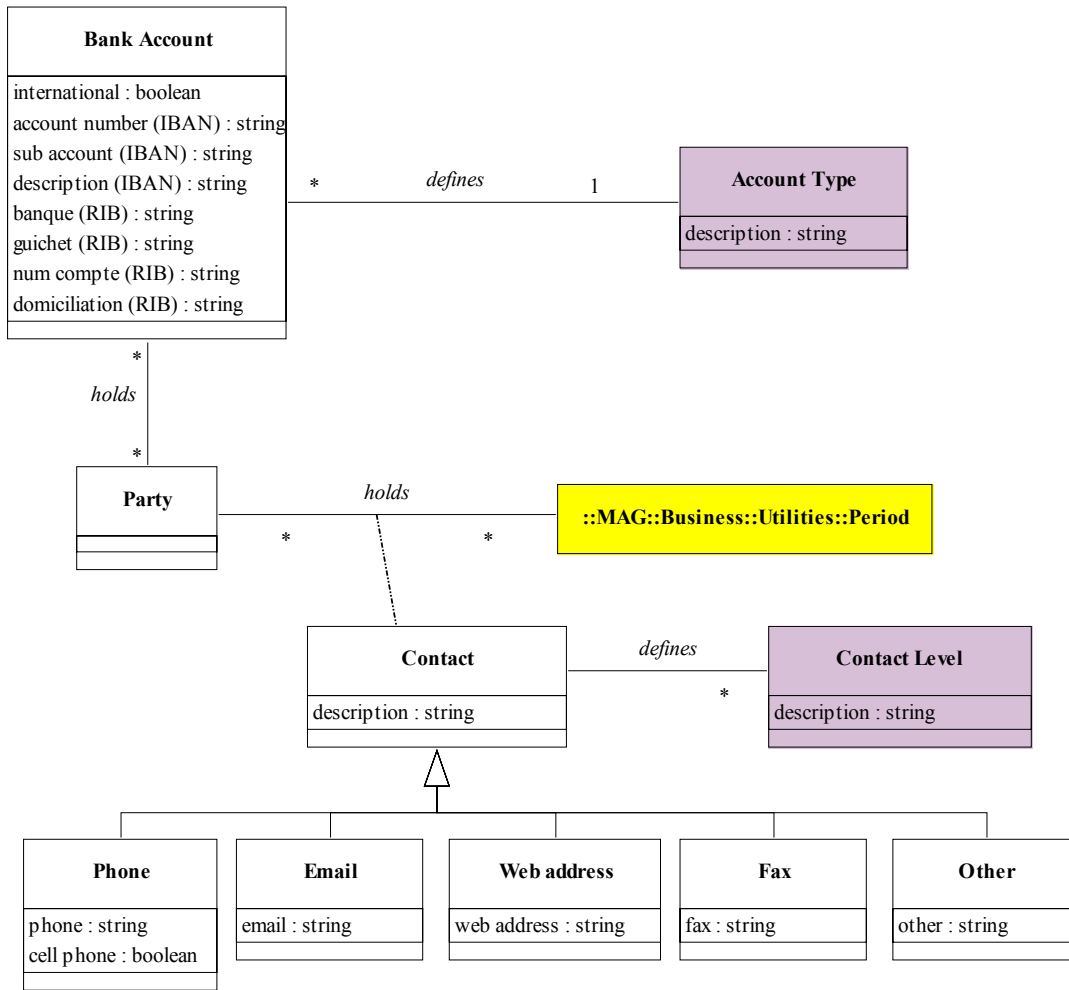
Example of possible values for the 'Address Code' class: Bis, Quinte, Quater, Ter...

#### **‘Postal Code Pattern’ class:**

The 'postal code pattern' class allows for describing the syntax of the postal code (see City class) depending on the country. For example, the length of the territory code composing the postal code is: France (3 positions), US (2 positions), Belgium (2 positions). And the length of the complete postal code is: France (5 positions), US (5 positions), Belgium (4 positions).

In the UK the 'postal code pattern' may have several structures. This variability is treated by the many to many between Country and 'Postal Code Pattern' classes.

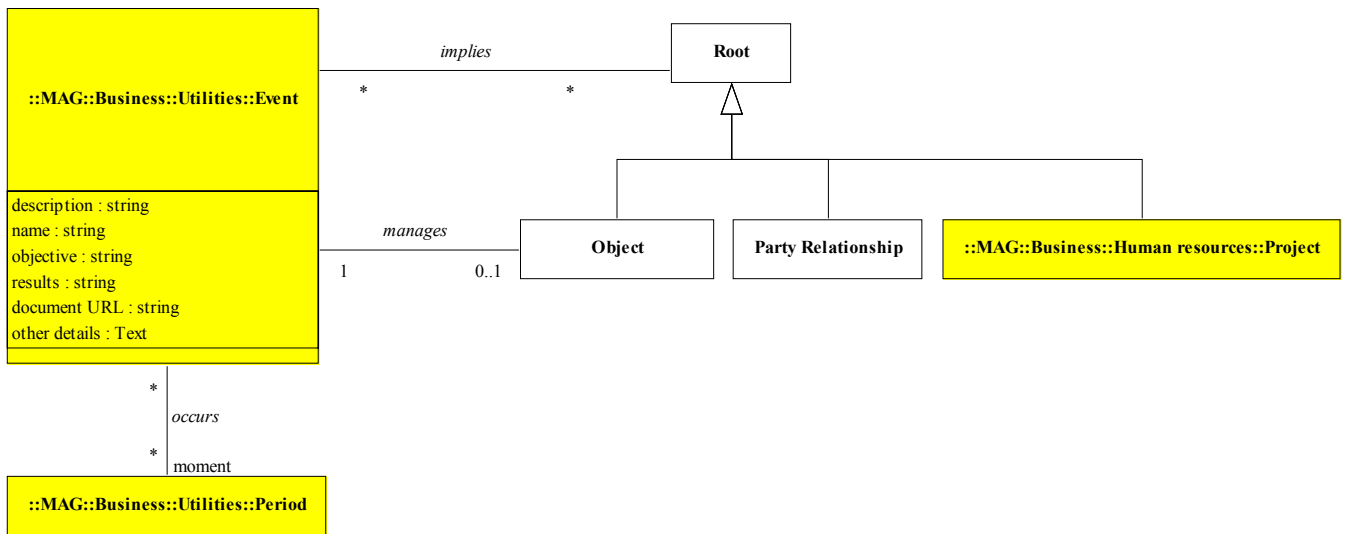
## 5.7. Semantic of Contact



Assets are not directly connected with Contact Mode. We consider that contacts are available only for Party. For example, if we want to define a phone for a building (type of Asset), it is required to precise the Party who owns this phone; the qualifier attribute 'role' attached to 'relates to' association (see semantic of Asset) allows for describing this type of relation between assets and parties. In this case, we may define a Party who manages the building. This party has a Contact Mode in order to store and retrieve the phone number.

Example of possible values for the class 'Contact Level': Office, Private, Travel...

## 5.8. Utilization of events



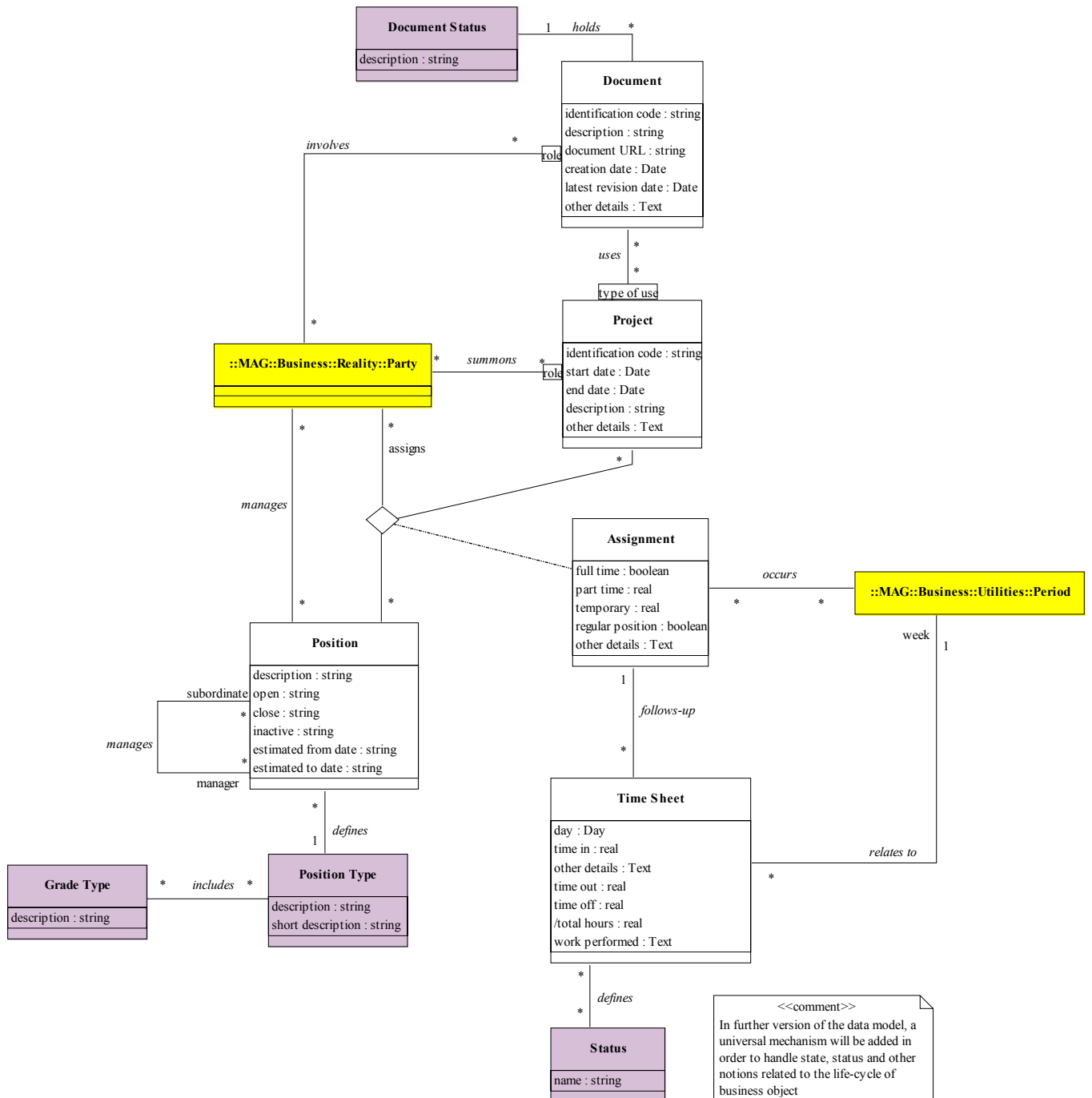
The Event management is described in the Utilities Domain. You must note that the associations ‘implies’ are inherit with help from the association ‘implies’ from ‘Event’ to ‘Root’ class (see Utilities domain). All objects (here Object, Party Relationship, and Project) inherit from the ‘Root’ class.

The ‘manages’ association is added in order to record the parties that are responsible for the management of the events.

In a further version of our prebuilt data models, we will detail the relationship between this event modeling and the modeling of state machines belonging to the business objects.

## 6. 'Human Resources' domain

### 6.1. Semantic of Position and Project



The position is a job slot. It can be occupied by one or more persons over time. A job slot is managed by one or several parties including these responsible for the recruitment management. When a Party is appointed to a position it may require an updating of its Party roles. For example, when a Party is appointed as a CFO then this party has definitely a Party role entry associated with a CFO Party role type. Therefore, synchronization between the 'Position Type' Class and the 'Party Role Type' may be ensured. Both classes are needed. The estimated dates are given so as to keep information about the forecast. The actual dates of assignments are set up in the Assignment class.

The ternary 'assigns' between Party, Position and Project allows for defining the assignments over time.

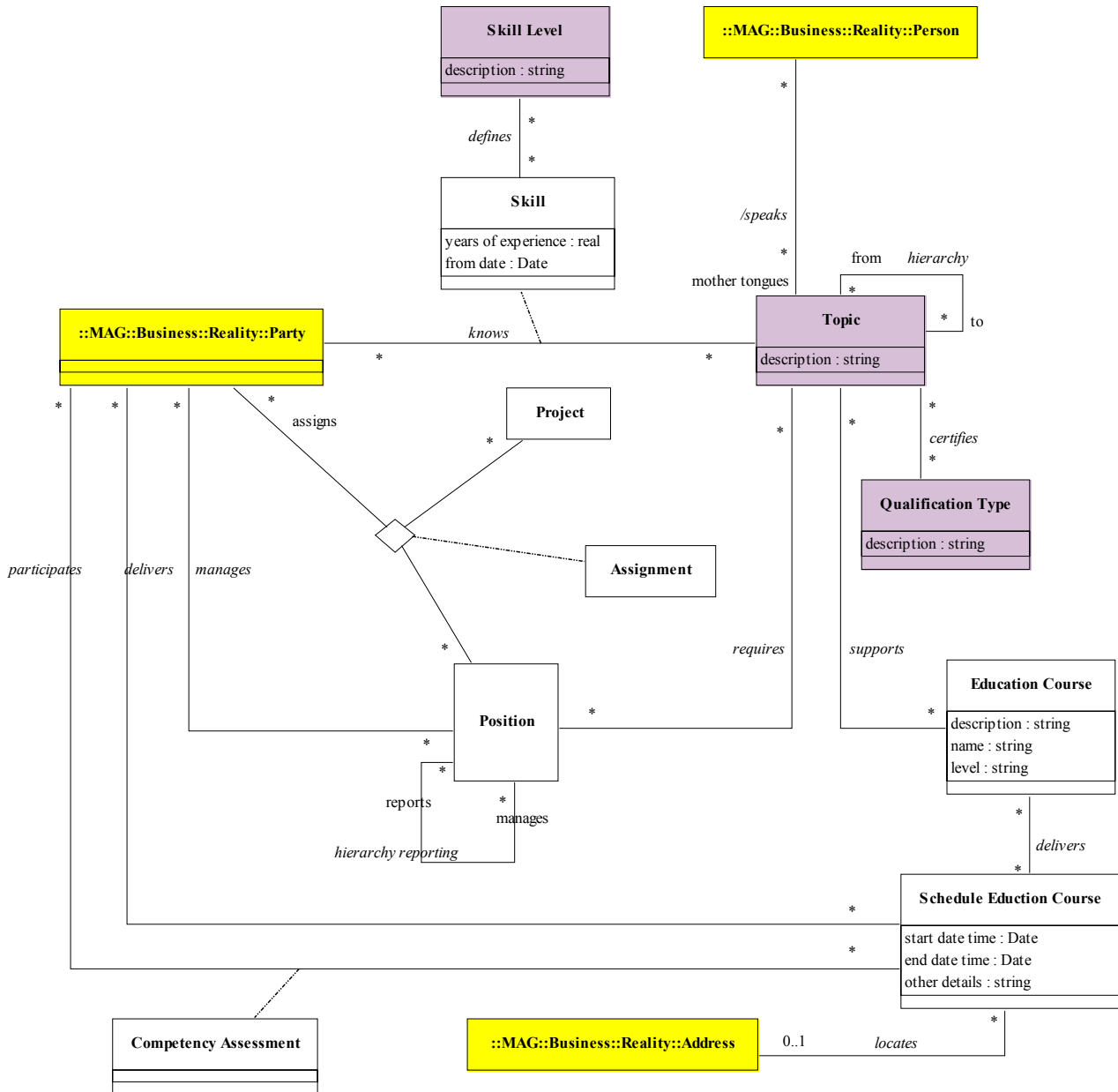
Example of possible values for the qualifier attribute 'role' attached to 'involve' association between the Project class and Party class: Owner, Sponsor...

Example of possible values for the qualifier attribute 'role' attached to the Project class: Owner, Sponsor...

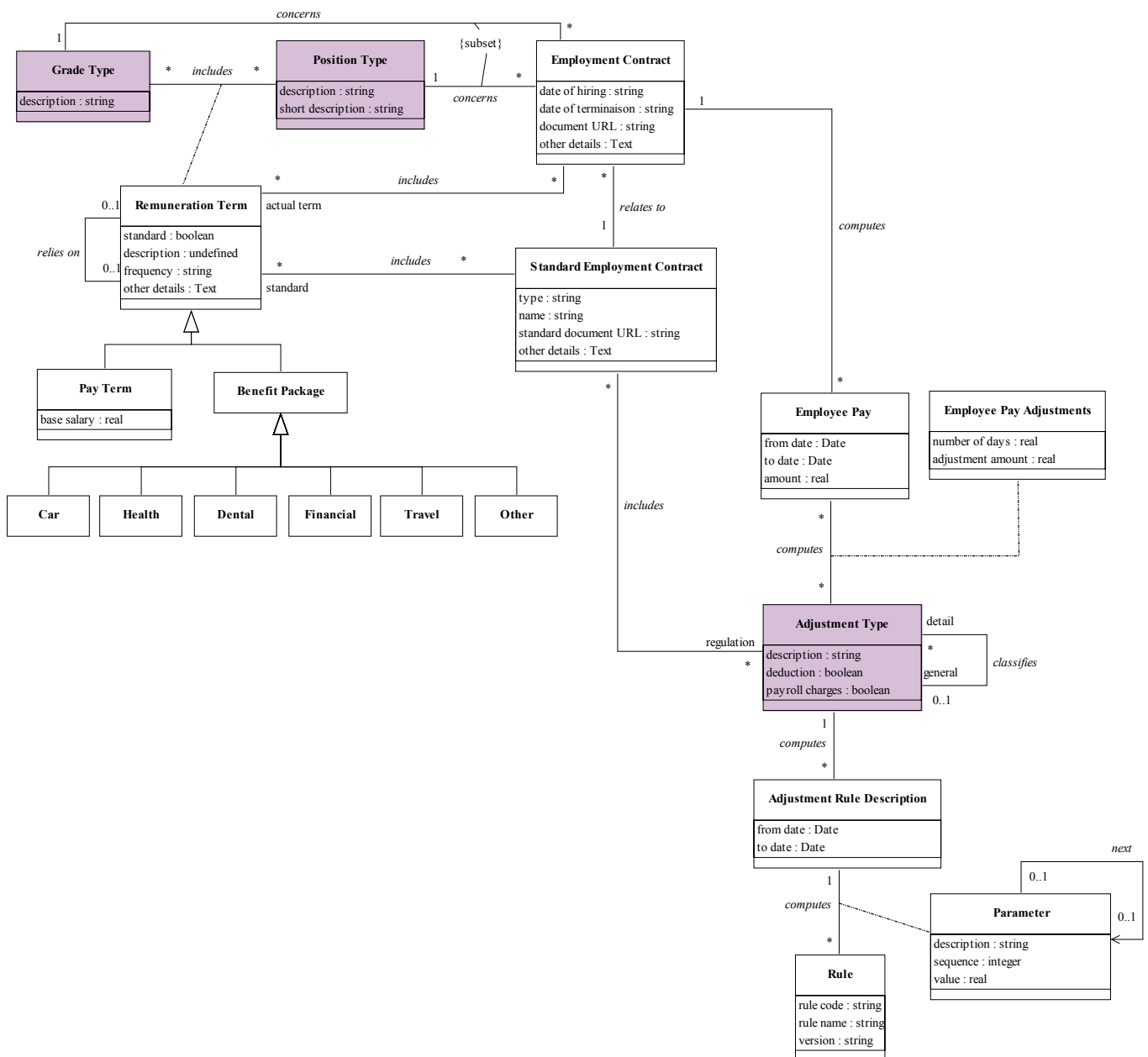
Possible values for the qualifier attribute 'type of use' attached to association between 'Project Document' and Project classes: Utilization, Creation...

## 6.2. Semantic of Skill and Course

This attempt for modeling Skill and Course relies on two central concepts already described. First of all, the Party class that allows for identifying various associations such as 'participates', 'delivers', 'manages' and 'assigns'. Secondly, the Position class which is described through the topics needed for the job.



### 6.3. Semantic of Pay



**‘Employment Contract’ class:**

The 'Employment Contract' describes the information of employment agreement between the parties involve in a relationship, most of time, an Employee and an Employer. Both are described as a Party, either as an organization or a Person (see Reality Domain).

With help from the 'Employment Contract' all pay information are structured. Three central mechanisms are set up. First of all, the configuration of standard pay terms and packages, which are not reliant on a specific Employment contract, are defined. This configuration is done for each Position Type and Grade Type. Secondly, actual pay term and benefit packages are declared in the context of a specific Employment contract. Finally, the computation of the employee pay relies on a parameterization of the pay adjustments needed for each pay (addition and deduction). This parameterization also includes the referencing of the rules involved in the computation.

**'Remuneration Term' class:**

The 'standard' attribute of the 'Remuneration Term' class states whether the term is standard (not reliant on a specific Employment Contract) or overloaded for a specific Employment Contract.

When overloading a remuneration term, the originate term used is recorded via the association 'relies on'.

**'Adjustment Type' class:**

Using 'classifies' association, adjustment types can be organized via a tree structure.

Example of possible values for the Adjustment Type: CSG, Retirement, Allowance, Benefit, Bonus, Danger money, Sick leave...

**Other explanations:**

'Adjustment Rule Description', 'Rule' and 'Parameter' classes allows for identifying formal rule name and description. Formal means that the IT platform will be able to launch the right rule from that name, either with a BRMS (Business Rules Management System) or other.

Rules referenced via the Rule class can be handled via a rule engine (generic or dedicated to payroll management) and are parameterized via the MDM (see the associative class named 'Parameter').

The class 'Parameter' allows for defining the values of each parameter that are included in the rules.

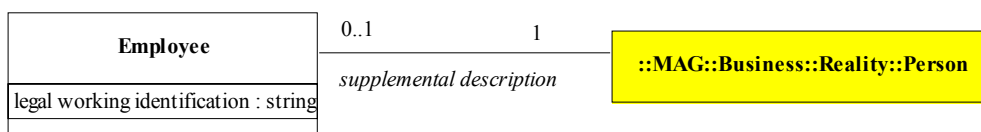


## 6.4. Semantic of Employee

Regarding the scope of the employee description, we have already defined the main information. Indeed, since our data architecture is based on a 'centered role' approach (see the semantic of Asset and Party) rather than an 'centered actor' approach, the Employee class is only used to describe supplemental information dedicated to Employee under condition that information is not already described via the classes Party and Party Relationship. For example, we may have special information regarding the legal working identification of each employee

Don't forget that 'Employee' is first of all a role and not an actor. The actor is a Person whose role is 'Employee' amongst many other possible roles.

Consequently, the data model for the Employee topic is very simple and generic. All information regarding the addresses, the skills, the assignments and projects, the pay terms... are already defined either in a generic form or attached to the 'Employment Contract' class.



If a Person works for several employers then be cautious that this Employee Class remains unique. Specific information regarding both Employee and Employer are described via the Party Relationship mechanism, for instance the Employment contract because a unique person may have several Employment contracts for one or many employers (see the semantic of Asset and Party mechanism).

General note beyond the 'Employee' domain: depending on your context, you can create new classes that describe other facets regarding the Person either in the HR domain or other domains (e.g.: A Customer facet defined in the Portfolio domain).

## 7. Appendices

### 7.1. Lookup tables

A lookup table is a reference table containing key attribute values that can be linked or related to other business data in order to classify them and translate labels. Lookup tables constitute the first layer of reference data managed by the MDM. The other layers are formed directly by central business objects such as Person, Party, Employee, etc.

#### 7.1.1. UTILITIES domain

Class name	Description	Example of values
Event Type		Meeting, Contact, Regular contact, Marketing campaign, Claim management, Machine repair...
Event Status		Cancelled, Confirmed, Provisional, Close, On going, To be checked...

#### 7.1.2. REALITY domain

Class name	Description	Example of values
Taxinomy	Applied to Party Classification	Business, Large business, Financial sector, Pharmaceutical sector, Insurance sector, Health care sector, Administration, Government agency, IT sector, Non profit organization, Recruitment Agency, Community, Charity, School, Family, Team...
Party Role Type	These values are filtered depending on the type of the Party.	<p>The possible values for 'Party Role Type' are as follows (these values are filtered depending on the type of the Party) :</p> <ul style="list-style-type: none"> <li>- Prospective Customer, Customer, Employee, Employer, Provider, Buyer, Seller, Member, Chairman, Chief Executive Officer, CFO, VP, Manager, Staff member, Doctor, Donor...</li> <li>- Head Quarter, Divisional Office, Branch Office, Department, Business Unit, Subsidiary, Team</li> <li>- Located at (eg: an employee works in the office X)</li> <li>- Wife, Husband, Father, Mother, Sister, Brother, Daughter, Son, Grandfather, Grandmother...</li> </ul> <p>If needed, an aggregation could be added in order to set up a rollup structure. For example, the Head Quarter role could be a composition of other roles such as Divisional Office, Business Unit. With help from this composition, it becomes possible to define a Party Relationship Type that acts for several types of roles. This is surely needed for the Party Relationship Type 'Customer Relationship' that could be available of several roles of organization.</p>
Party Relationship Type	Describes the type of a relation between two or more parties.	<ul style="list-style-type: none"> <li>- 'Employment relationship' to define the relationship between the roles Employee and Employer.</li> <li>- 'Customer Relationship' to define the relationship between the roles Customer and Organization.</li> <li>- 'Married'</li> <li>- 'Apartment Rental'</li> <li>- Etc.</li> </ul> <p>Be aware that a Party Relation Type is defined for two identified Party Role Types. Obviously a set of constraints must be defined in order to state, for example, that a 'Married relationship' is mandatory bound with a role 'Husband' and a role 'Wife'. See also the remark regarding the algebra for the managementg</p>

		of the roles (class Party).
Location nature (qualifier attribute attached to association 'resides' of the Object)	The location nature is often important to decide which address is the right one depending on a functional context such as sending a mail, assigning an intervention at the customer's site...	Usual (most of the time this nature is sufficient, except if others are declared), Business, Residential, Second Home, Headquarter, Delivery (where the goods are delivered), Invoicing (where the invoice must be sent), Return (for a customer, the return link designates the site where the goods are to be returned in case of problem), Vacation, etc.
Country		Usual country list
Territory Type		The values may depend on the geographic boundaries. For France: Département, Région. For USA: State, County.
Address Nature	The values are reliant on the language of the Address (should be managed directly via the MDM).	Street, Road, Lane, Boulevard, Way, Building plot...
Address Code		Bis, Quinte, Quater, Ter...
Currency Code		EUR, GBP, JPY, USD...
Account Type		Transaction accounts, Savings accounts, Basic bank account...
Marital Status Type		Single, Married, Separated, Divorced, Widowed, Engaged, Annulled, Cohabiting, Deceased
Gender Type		Male, Female, Undisclosed
Contact Channel Type	Depending on these values the Contact Class's attributes may be fed or not. Inheritance is voluntary not used in order to simplify the modeling and foster upgradeability when new contact modes are required	Email, Phone, Web, Other
Contact Level		Office, Private, Travel...
Apartment Type		Studio, Alcove studio, Duplex, Triplex, Quadruplex, Loft, Maisonette...
Apartment Facilities Type		Broadband, Cable TV...
Parking Type		Standard, Double, Parking garage, Parking lot, Parking place, Valet parking, Private parking, Public parking, Parking for bicycles, Multi story car park...

### 7.1.3. HUMAN RESOURCES domain

Class name	Description	Example of values
Position Type		Country Manager, CIO, CFO, CEO; IT consultant...
Grade Type	Allows for declaring several levels of remuneration for a unique position type. Remuneration is defined via two classes: 'Pay term' and 'Benefit package'.	Level1, Level2, Level3...
Adjustment Type	Needed in order to calculate the salaries.	CSG, Retirement, Allowance, Benefit, Bonus, Danger money, Sick leave...
Topic		Marketing, Sales, Automotive Sales, IT, Object Modelling,

		Java programming, Driving license... The root defining the list of languages is 'Language': English, French, Spanish, German... This root is useful to access the subset of codes regarding the languages, for example from the Person Class that used the Skill Type to declare the native language of the Person
Skill Level		High, Advanced, Medium, low, Native, Writing, Reading, Speaking
Qualification Type		International Bacculaureate diploma, MBA, Computer science high school diploma, Internal company diploma...
Document Status		Valid, Invalid, Under construction, To be checked...

End of the document