A Basis for Integration within Enterprise Modelling

Remigijus GUSTAS[1]

Swedish Institute for Systems Development
Electrum 212, S-164 40 Kista, Sweden
e-mail: gustas@sisu.se

1. Introduction

Ideally, wishes of various enterprise stakeholders could be described as objectives which constitute a driving force for further enterprise reengineering activities. Recently, this idea was raised in several papers [Bub93], [SBWJ94]. The main difficulty in the process of concurrent enterprise modelling is that an equivalent objective can be interpreted and understood by several persons differently, i.e. within the concurrent engineering the visibility of activities and semantic dependencies across several perspectives can differ. This distinction is due to different objectives of the enterprise stakeholders. Quality Function Deployment (QFD) is a method to enhance communication between developers of different views[JaKe94]. It is based on the special form entitled House of Quality. The quality tables mirroring the mappings (dependencies) of customer requirements from different perspectives into product characteristics and development processes. Nowadays QFD implementations are mostly quality tables editors [JaKe94]. This is not surprise, because software developers have no deeper understanding of types of dependencies among enterprise actors, activities, products and goals of customer. The new generation of QFD tools can be based on systematic study of several kinds of dependencies which are represented by so-called houses of quality. In general, two directions are suggested for bridging goals(informal pragmatic statements of requirements) to enterprise semantic constraints and dependencies:

1) Refinement goals in terms of states which are essentially defined as logical expressions [DLF93]. Language of predicate logic is powerful enough for definition of semantics of any goal, but logical representations are highly inconvenient to human beings from comprehensive point of view. The complexity and unreadability of logical formulas is the main reason why managers, designers and analysts do not like to use logical specifications [SoZa92].

2) Refinement of goals into business rules which are defined on a basis of ER [Chen76] kind of representations [Tempora92].Abstraction level of primitives of semantic models [HuKi87], [PeMa88], [RBPEL91] or some knowledge representation methods [Brach79] is much more higher compared to traditional logic formalisms. They have been developed to capture some meaning as well as structure of data. Characteristic feature of rules defined in business modelling level[Gust94a] is that they abstract from any particular logical data representation aspect, while concentrating on epistemological [Brach79]aspects which are common to all logical representations [Dietz91]. Although business modelling level rules are much more readable and understandable for human beings, there is no complete understanding of rationales behind goals.

Business rules are of interest because they capture semantics of goals. Goals are important [DLF93] in several aspects:

1) goals may motivate completeness of specified changes of requirements, or may provide some help in detection overspecifications, i.e. to discover requirements which have no motivation,

2) goals justify and explain the presence of enterprise components which are not necessarily comprehensible when they are separated from each other,

3) goals may be used to understand responsibilities of the enterprise actors,

4) structure of goals provide basic elements for detecting and resolving conflicts that arise from different perspectives. Goals lead to integration of enterprise elements.

A goal hierarchy could be formed of interconnected goals of different levels of abstraction ranging from higher level business objectives to lower level operational goals [Wang93]. The structural

---

[1]The author is on leave from InformationSystems Dept., Kaunas University of Technology, Lithuania;
e-mail:remigijus.gustas@if.ktu.lt

decomposition and refinement of such pragmatic categories may be understood in correspondence to the hierarchical decomposition of processes performed in structured analysis [Yourd89]. The goals are broken dawn into lower level business functions each of which is intended to perform some specific task. The decomposition goes until a level is reached where the function is sufficiently well understood, i.e. can be specified in terms of semantic constraints. The example of a goal decomposition hierarchy is depicted in Fig. 1.1.

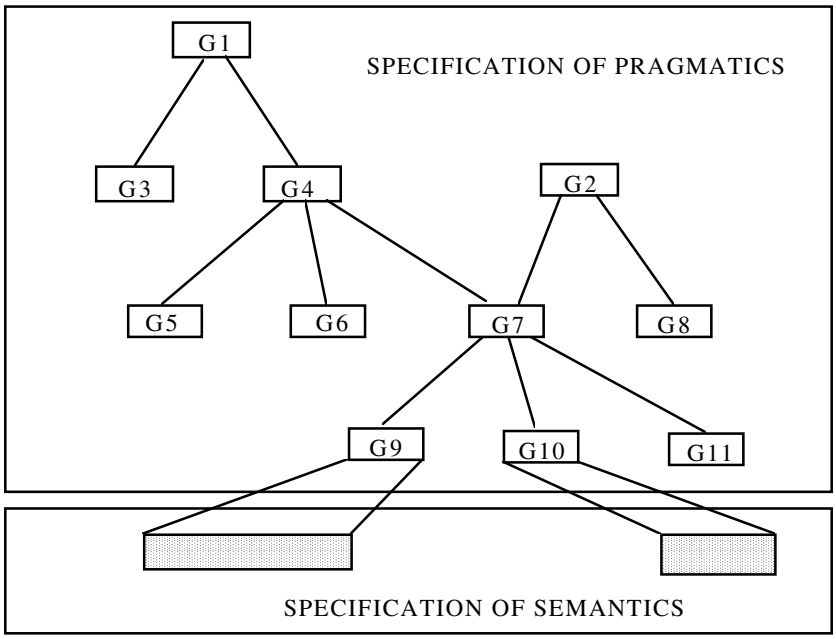


Fig.1.1. The goal decomposition hierarchy

Several goal decomposition hierarchies can be interpreted as the reasons for the specification of several enterprise or information system semantic level dependencies. At the lower level of a goal hierarchy, the goals are much easier to formalise than the more vague higher level business objectives. The reason is quite simple -the lower level goals consist of lesser number of dependencies and constraints.

The three dimensions of requirements engineering [Pohl93] illustrates the three major aspects of the problem of goal modelling. On a specification dimension we are dealing with understanding of semantics of goal. Representation dimension has to support the meaning of goal by expressing it in terms of some formal notation. Finally, the agreement dimension is based on understanding of goals from different viewpoints. It has to support trade offs among several goals, i.e. negotiation process of distributed enterprise modelling. Enterprise models [Bub93], [F3_94] have been developed to make strict distinction between essential business oriented aspects (specification dimension) and logical data processing aspects (representation dimension). Nevertheless, still remains the fundamental problem of description of semantics of goal in terms of static and dynamic descriptions of the enterprise.

Nowadays, the concept of context becoming unimportant issue in the process of enterprise integration. Recently, it has been addressed by several researchers[JaPo93]. Context deals with a partition of requirements according to several viewpoints. Modelling of the context is very important as it constitute the basis for definition of goal boundaries. Although several enterprise models have been studied quite intensively, in general, there is no complete agreement upon definition of context. The ultimate objective of this paper is to illustrate possibility of description of some pragmatic level elements in terms of semantic dependencies of an enterprise, including the most primitive cases of modelling in context.

2. Enterprise Modelling

The framework of Enterprise Modelling [F3_94] includes a number of interrelated submodels (Fig.2.1). The semantic and pragmatic links established in these submodels are reflecting rationale and motivation for the specific syntactic level parts, i.e. specification of the enterprise information system parts. The enterprise model reflects essential business entities, activities, and how they interact.

Information system model determines the data elements and functions that represent business entities and processes [SoZa92]. So, the enterprise model deals with understanding and agreement perspective while information system model corresponds to representation perspective [Pohl93].

The concept of enterprise here should be interpreted in a very wide sense [Bub93]. It could actually mean description of qualitative aspects of business of the whole enterprise or some small part of it. In the context of information system development, the concept of enterprise rather denote a limited area of activity of the organisation which is concerned by the application designer. The rest of this section describes the different 'worlds' of enterprise model.

The enterprise submodels [F3_94] are based each on a number of pragmatic and semantic dependencies which are typical for that submodel. Each submodel concentrates on specific aspect of the enterprise. The basic components of enterprise model are depicted in Fig.2.1.

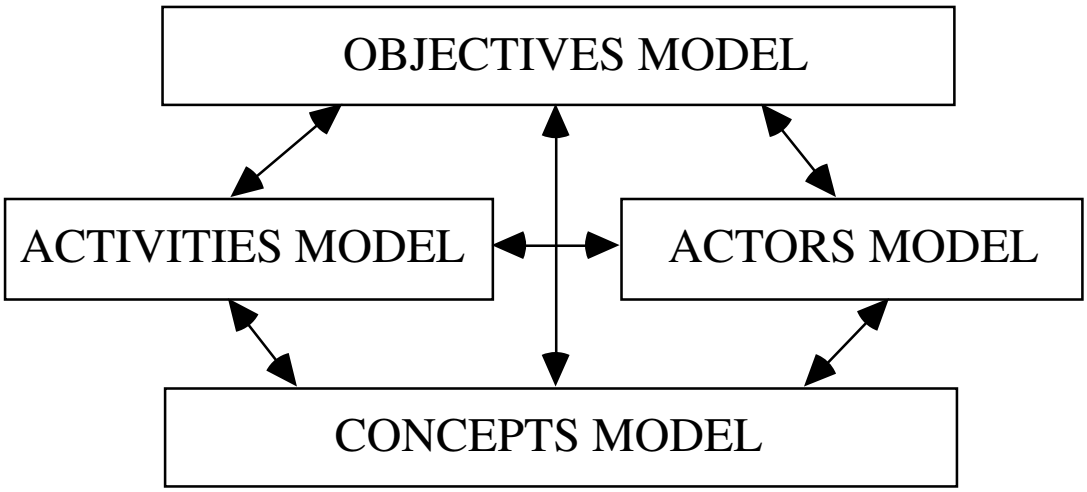


Fig.2.1. Components of the Enterprise Model

There are four general enterprise modelling components which can be used for the description of business: Objectives submodel, Activities submodel, Actors submodel, Concepts submodel. Notice that enterprise modelling doesn't mean specification of the information system at the same time. Models of business are one thing, while models of information systems are something different [Will93], [SoZa92].

Each enterprise submodel focuses on the specific business modelling aspect. The objectives are motivating and driving the development of other parts of enterprise model. Sometimes, enterprise components can be easily stated before goals are well-understood. Elaboration of objectives submodel is then done in bottom-up way, by asking the reason for their existence. This part of enterprise model addresses the 'why' perspective. The activities submodel describes the 'how' perspective and contains organisational activities, i.e. actions or tasks of an enterprise. Activities are able to make changes to states of affairs, which are expressible in terms of concepts (resources[Will93]). The actor submodel addresses the 'who' perspective. It defines the strategic dependencies [Yu94] between actors in achieving of their goals by means of message passing and triggering actions at the same time. The semantics of each message is expressible in terms of concepts. The concepts submodel defines static constraints among concepts. It addresses the 'what' we are talking about. It should be noted that using static constraints and some conceptual operations, the 'where' and the 'when' perspectives are expressible in terms of concepts model as well.

Enterprise modelling is a concurrent process of capturing of the pragmatic and semantic aspects [LLS94] of requirements. The dependencies of concept, activities and actors submodels describe semantic aspect of the enterprise. The semantic dependencies constitute the basis for definition of pragmatic dependencies, i.e. the objective submodel relations in terms of static and behavioural constraints. The more specific explanations of semantic and pragmatic dependencies among typical elements of the enterprise 'worlds' are presented in the next two chapters.

3. Modelling of Pragmatic Dependencies

Ultimate goal of objectives modelling is to identify, to document and to analyse the enterprise in terms of pragmatic level entities such as goals, problems, causes of problems, opportunities. The pragmatic relationships among categories can be specified by several actors in parallel. The objectives model addresses the 'why' perspective and it is motivating and driving the development of other submodels. One of the main problems here is unambiguous description of pragmatic level entities on the basis of semantic level modelling artefacts such as dependencies of concepts, actors and activities submodels. In this chapter, the overview and description of pragmatic relationships of the objectives model is presented.

The objective submodel components specify the pragmatic aspect [LLS94] of the enterprise. The typical goal can be expressed on the basis of two specifications: the specification of actual situation and the specification of desired situation. The actual situation defines the static and behavioural constraints of actual specification, and desired situation - the expected specification. The explanation of a relation between two situations using the typical dynamic element is depicted in Fig.3.1.

Fig.3.1. Relationship between initial and desired specification

The development action describes activity that has to be performed in order to achieve a predefined set of goals. The activity forms a set of development tasks that have to be carried out [Bub93]. Let us consider the special case when the specification has to be developed for one particular goal of the enterprise. In this case, an actual situation may be specified by the problem statement, while the semantics of a desired situation can be refereed by a goal statement. Thereby, the development action takes us from the problematic to the expected situation, specified by the goal.

The opposite of a goal is a problem. *A problem describes a situation which is not desirable*. This situation can be specified by set of static and dynamic constraints. *A specification of a problem* is defined as a semantic difference between a set of semantic constraints of the actual situation and a set of constraints of the desired situation. So, the semantics of a problem can be regarded as part of the specification of product at the actual situation. It is very important that *the problem can not exist without stating the goal*. If the designer has no predefined goal then the problem does not make sense. A goal describes a *desirable situation*. *Specification of a goal* is unambiguously defined as part of the semantic description of a desired situation which is deduced as a semantic difference between a set of constraints of the desired situation and the set of constraints of the actual situation. So, according to this understanding, the specification of goal is the opposite side of specification of problem. At the same time, the specification satisfyce [Simon84] the goal or the problem and is regarded as the *extension* of this pragmatic element.

Any two goals can be *contradictory*, i.e. one goal can influence negatively or hinder to the achievement of the other goal. It means that interpretations of the notion of a goal and a problem are relative and dependent on actor objectives. The same description could be interpreted as a goal for one actor, and as a problem for another actor. To specify contradictions between several goals of agents, the negative influence dependency [F3_94] is introduced. An influence from A to B (A $\xrightarrow{\cdots}$ B) dependency means that situation denoted by B is affected by situation A. The negative influence pragmatic relationship($\xrightarrow{-}$) from one goal to another indicates that the first hinders to the achievement of the second. The positive influence dependency ($\xrightarrow{+}$) would mean, that the achievement of some goal, automatically contributes to the achievement of the other.

The *pragmatic dependency of specialisation* of A by B is denoted A $\prec$ B if and only if the specification of B is a subset of the specification of the pragmatic element A. The 'motivation' dependency [F3_94] is usually used instead of specialisation to refine a goal. The 'cause' dependency can be used in the case of specialisation of a problem. So, the consistency of decomposition of complex pragmatic entities to simple ones can be verified as far as their structures are specified by semantic constraints. Graphical notations of types of pragmatic relationships are presented in Fig.3.2.

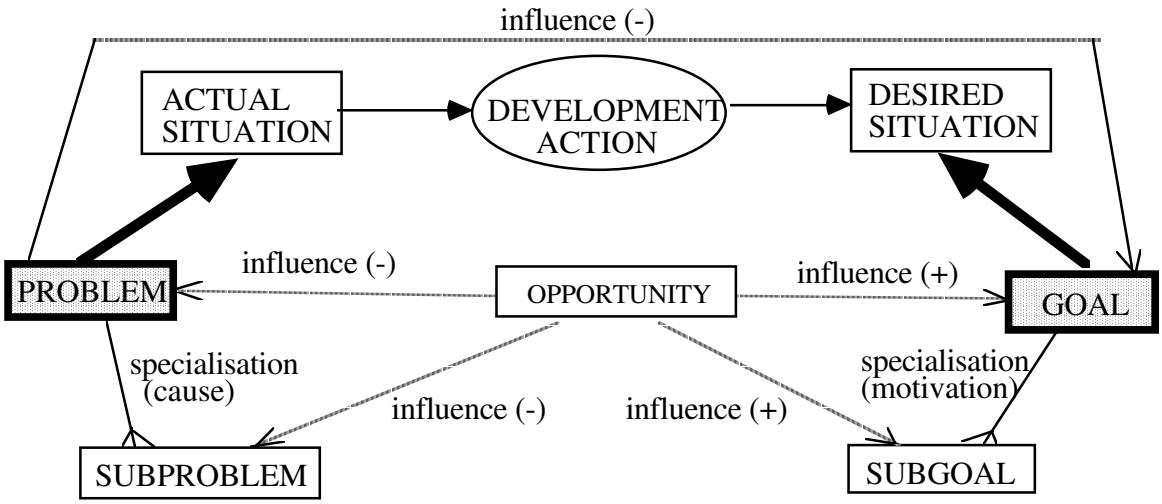


Fig.3.2. Types of pragmatic dependencies. Here ➡ is the inheritance (ISA [Brach79]) dependency.

Using the above described primitives, we can now explain notions of an opportunity and a cause of a problem introduced in the area of enterprise modelling [F3_94]. *Opportunities* express situations that may be used to improve actual situations. It should be noted that the notion of an opportunity is relative and dependent on the methodology [F3_94], [Will93]. Sometimes, opportunities are used for description achievable situations not regarded as final, i.e. situations that we may want to take advantage of. If so, they could be interpreted as intermediate goals. Opportunities can be specified using representations of the semantic level.

A cause of a problem can have different interpretations as well. One of the most simple definitions of a cause could be based on the notion of a situation which is the reason for a problem to exist. Causes are usually used for specification the explanations of problems and regarded as extensions of problems. Every specification of a subproblem is a cause for problem, and thereby, a subproblem causes a problem. The cause of the problem could be formed by taking the union of causes of subproblems. If the semantics of each cause can be expressed by exactly one constraint rule, then it shows higher quality of structuring in the phase of goal modelling.

Modelling primitives applied for specification of semantic constructs of an enterprise are usually based on Entity - Relationship like (ER) notations[Tempora92], [F3_94]. Although these notations are much more readable and understandable for human beings compared to logical formalisms, there is no uniform theoretical basis and even agreement whether the concept should be specified as entity, or as relationship, attribute, action, actor, goal, etc. Usually, the same name of concept can have all those several roles, but just indifferent context of interpretation. One of the main difficulties of ER representations are that they require to distinguish sharply among such meta categories, i.e. the designer is enforced to make decision whether the concept will be interpreted as an entity, or as a relationship, or as an action, etc. Strict interpretation of categories can cause different representations for the semantically equivalent enterprise elements, even if the same modelling constructs are used. Because of different interpretations of the same concept, the designer is able to define structurally different, but semantically equivalent representations. As a consequence of strict interpretation of semantic categories, is impossible to achieve uniform descriptions of goals defined independently by several actors. The examples of various interpretations of the same concept JOB in three different semantic representations is depicted in Fig.3.4.

So, ER kind of notation is representation dependent. Abstraction principle[Griet82] means that conceptual model must refer exclusively to rules of the universe of discourse, while sharp classification of categories imply the rules which are dependent on the representation dimension [Pohl93]. The motivation behind this principle is the need to simplify the process of concurrent enterprise modelling by concentrating only on conceptually relevant aspects and disregarding everything else.

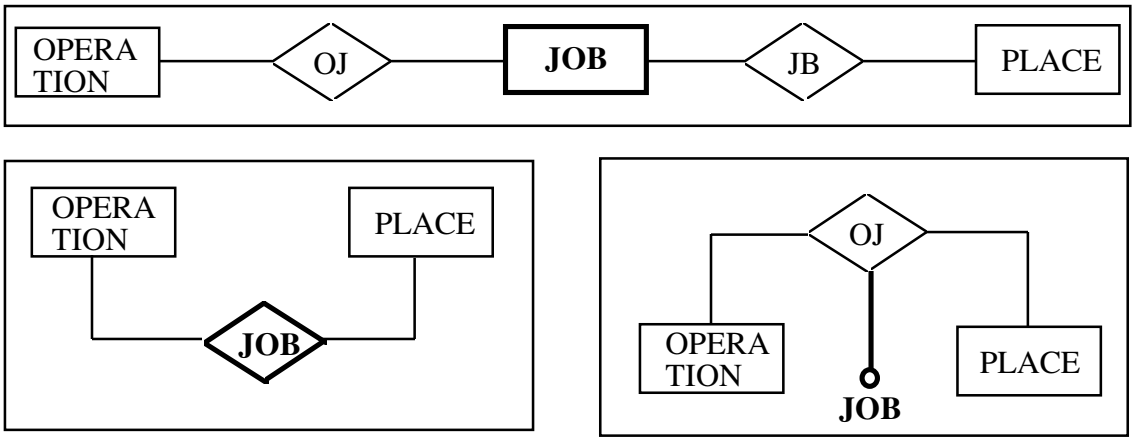


Fig.3.4. Illustration of relativeness of interpretation of categories

The enterprise specifications must possess number of desirable qualities[IEEE84], [Louc92], [Griet82] such as independence, abstraction, formality, traceability, unambiguity, completeness, consistency and so on. The specifications needs to be analysed in order to determine whether they are ambiguous, incomplete or inconsistent. One subset of natural ambiguities is known as naming conflicts [BaLe86].The roots of those ambiguities are quite natural. They arise when the same name is used for the specification of concepts with the different meaning, or visa versa - different names specify the same concept . The graphical illustration of naming conflicts is presented in Fig.3.5.

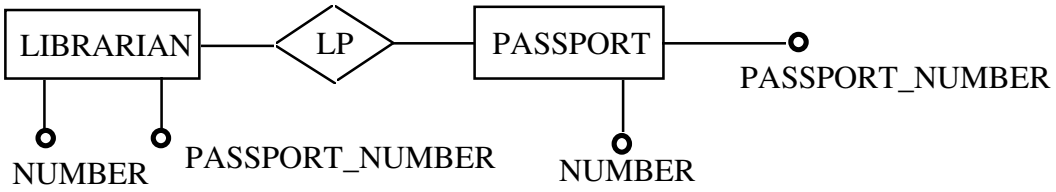


Fig.3.5. Illustration of naming conflicts

Naming conflicts cause ambiguities and misunderstanding when characteristics of enterprise products are described concurrently by several actors. In order to eliminate those natural ambiguities, the new kind of qualitative and structural dependencies of enterprise concepts should be introduced. Usually, the naming conflicts can resolved on a basis of enriched enterprise modelling technique.

The dependencies of enterprise submodels have to build a unified basis for integratability of descriptions caused by different origins of enterprise constraints. Usually, the consequences of using of several models, such as ER diagrams and data flow diagrams [Yourd89], for description static and dynamic aspects of an enterprise are that interaction between those specifications is lost. It means that enterprise experts becoming unable to trace the specification as a whole.

4. Analysis of Behavioural and Static Dependencies

Each 'world' in the enterprise model has own laws and specific types of primitives for modelling their semantics and reasoning. The dependencies introduced in this chapter can be regarded as a the common basis for enterprise integration. Furthermore, they constituted the most primitive building elements for interpretation of categories (meta-types) of various modelling techniques exploited in the area of enterprise modelling such as data flow diagrams, state transition diagrams and entity-relationship diagrams. Simple dependencies among concepts constitute the basis for definition of more complex dependencies between goals in the objectives model, or between actions in the activities model. In the next subsection the behavioural dependencies will be described. Their roots are situated in the activities and actors submodels. The static dependencies of concept model will be presented in the second subsection.

4.1 The Behavioural Dependencies

The behavioural dependencies of an enterprise model describe semantics of most primitive dynamic relationships. In this part of submodel, the particular organisational activity is defined in terms of actions, states between actions, and information or material flows between agents. Action represents impact directed from an actual state to a possible state. If the action describes a simple goal, then the possible state corresponds to the desired state. Graphical notation of the most primitive construct for specification of dynamic dependency between states of the activities submodel is depicted in Fig.4.1.1.

Fig. 4.1.1. Dynamic element of activities submodel

Another kind of dynamic relations are known as strategic dependencies[Yu94]. They exist as material or information flows between several actors of the enterprise. These dependencies are relating the elements of actors submodel. An actor is an active element that carries out specific action to achieve some goal. Each flow dependency link specifies that one actor - recipient depends on the another actor - agent that may attain some goal by performing specific action. Graphical notation of flow dependency is presented in Fig.4.1.2.

Fig. 4.1.2. Dynamic element of flow dependency between actors

Clearly, behavioural dependencies are motivated by elements of objective submodel. Every most primitive goal can be described using notions of an action, a state, an agent and a flow between several agents. A flow is represented graphically by an arrow into and out of a process. The flow is used to describe the movement of chunks of information or material flow from one actor to another. Thus, the flows represent dynamics of the enterprise, whereas the states represent changes in the intensional structure of flows. The states are referring to the concept submodel components. The actions are typically carried out by components of the actors submodel. Actors can be represented by individuals, groups, job positions or artefacts. The semantic relationships among actors are specified by using concept model dependencies. Graphical notation of a typical combined dynamic relationship among two actors, a flow, an action and two states is depicted in Fig.4.1.3.

Here, change of state is controlled by external agent. The transition from actual state to possible state will be performed by action, if conditions stated by actual state expression are fulfilled and the flow from agent to recipient is sent. After the activation of the specific action, the conditions defined by possible state become fulfilled, the actual state conditions cease to exist and the flow directed from the agent is received by the recipient. This understanding of dynamics of actions is in correspondence with well-known modelling technique namely Data Flow and State-Transition Diagrams [Yourd89]. States can be specified on a basis of the concept model as derived compound entities [Gust94b]. The action shows a part of the system that transforms the inputs into the outputs.

The dynamic dependency between actors is describing goals and desires of an agent. Recipient by depending on agent is able to achieve goals that it was not able to be done by agent. At the same time, the recipient becomes vulnerable [Yu94]. If agent fails to deliver flow, the recipient would be unfavourably affected in its capability to achieve its goals. By matching the dependencies from agents to actors in actors submodel, one can explore opportunities that are available to these actors. It should be noted that static dependencies among actors such as aggregation and generalisation [Smith77] which facilitate reasoning about opportunities and vulnerabilities of recipients are defined by the concept submodel. The recipient becomes vulnerable to the failure of delivering the message or the material flow sent by agent. In this case the recipient would be concerned about viability of a particular flow dependency.

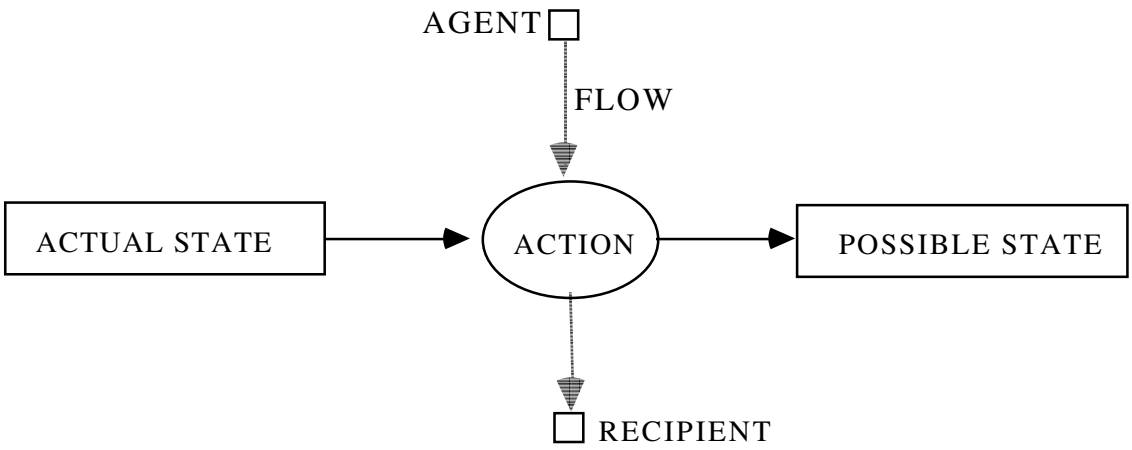


Fig. 4.1.3. Typical Dynamic Element

4.2 The Static Dependencies

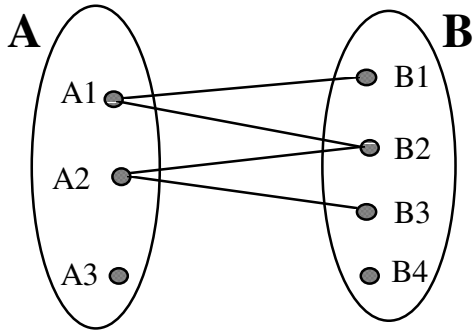
Static constraints among concepts constitute the basis for definition and derivation of new semantic relationships of the enterprise. Although semantic [HuKi87] and conceptual models have been studied quite intensively, in general, there is no well-understood theory and, sometimes, even agreement upon definitions of such widely accepted constructs as generalisation, aggregation, classification, inheritance, etc. Besides these well-known abstractions, a number of additional types of relationships have been identified by various researchers [Stor93] in some neighbouring disciplines. Some abstract non primitive dependencies (e.g.: inheritance) are defined on the basis of more simple ones. It allows interpretation of each construct in terms of most primitive semantic relationships. This kind of formality of the models let us have the interplay among several types of dependencies. The overall objective of this chapter is to introduce homogeneous basis for modelling of different semantic relationships among concepts. It could contribute for understanding of building constructs defined as several types of semantic categories in the area of semantic data modelling [HuKi87], [Tempora92].

Semantics of binary relationships at the enterprise can be defined as cardinalities represented by minimum and maximum number of occurrences of the concept B that can exist for each occurrence of the concept A, and vice versa [TsiLo82]. In the initial phase of concept modelling, the following constraints can be specified for any binary relationship r(A, B):

(1,1;0,1) - Injection dependency, it will be denoted by A==> B ,
(1,1;1,1) - Bijection dependency ($\longleftrightarrow$),
(1,1;0,*) - Total Functional dependency ($\longrightarrow$),
(1,*;1,1) - Surjection dependency ($\longleftarrow\!\!\!\!\rightarrow$),
(1,*;0,1) - Surjective partial functional dependency (==>>),
(1,*;1,*) - Mutual multivalued dependency ($\Longleftrightarrow$),
(1,*;0,*) - Total Multivalued dependency ($\longrightarrow\!\!\!\!\rightarrow$)
(0,1;0,1) - Partial injectional functional dependency (|==>) ,
(0,1;0,*) - Functional (partial) dependency ($\vdash\!\!\rightarrow$),
(0,*;0,*) - Multivalued (partial) dependency ($\vdash\!\!\rightarrow\!\!\!\!\rightarrow$).

The semantics of constraints is very important due to capability to form an intensional structure of concept. Not every of the above described dependencies has the same degree of usefulness in building of the intensional environment for the specific concept, because of their non primitiveness. By non primitiveness is meant that some of dependencies can be derived by using basic (or in other words most primitive) dependencies. It should be noted, that is very convenient to employ an arbitrary number of static constraints, but designer must be aware with consequences and complications which he will face in the following steps of enterprise modelling by using uncontrolled set of dependencies. If all kinds of dependencies are allowed as basic constraints, one could meet difficulties entitled as holes of semantics[Gust94b].

The extension of binary relationship between A and B can be represented by type (A Δ B) which denotes all meaningful associations between instances of A and B, and, if there is no association between the instance of concept A and any instance of B, then additionally - associations with empty instances($\perp$). Here: Δ - aggregation operation [Smith77].The example of binary relationship and it's extension is presented inFig.4.2.1.



(A Δ B)	
A1	B1
A1	B2
A2	B2
A2	B3
A3	$\perp$
$\perp$	B4

Fig.4.2.1. The example of binary relationship and extension (A Δ B)

The concept A in the relationship (A Δ B) is represented by set of instances. Concept A in the (A Δ B) contains a semantic hole, if for any instance of the concept B does not exist any association to at least one instance of the concept A. According to the presented example (Fig.4.2.1) , both concepts A and B contain semantic holes in the relationship (A Δ B). If one of any concept of the relationship contains a semantic hole then the relationship owns a hole as well. The holeless relationship is represented just by meaningful associations of instances. In the following, by (A Δ B) only meaningful associations of instances of A and B will be meant.

The semantic holes cause incompleteness, ambiguity and inconsistency of requirements. They can be eliminated by defining higher level constraints in terms of most primitive dependencies. So, non primitive constraints can be interpreted as the driving force in order to define the requirements as exactly as possible by elimination of holes of semantics. Of course, the problem of the minimal set of basic dependencies still remains.

The first five types of presented static dependencies are characterised as the set of basic (most primitive) binary constraints. They represent a set of relationships which somehow allow the designer to avoid holes of semantics. Some of them ($\Longleftrightarrow$, $\Longleftrightarrow\!\!\!\!>$) can be used for specification of semantic holeless relationships. The next three dependencies (==>, $\longrightarrow$, ==>>), allow simple elimination of the semantic hole by operation of prefixing[Gust94b]. The cases defined by the rest subset of dependencies are more complicated. Usually, those dependencies require redefinition of constraints.

The presence of constraints which are representatives of the holeless relationships, characterise higher quality of conceptual specification. It could be considered as a basis for concurrent reengineering of the enterprise because of unambiguity of the specification. By unambiguity is meant that the same specification is interpreted by several independent domain experts in the same way.

Higher order static relationships can be specified by using two kinds of abstractions: aggregation and generalisation [Smith77]. Abstraction of aggregation is based on the presented set of binary dependencies and special aggregation operations among concepts. The basis for a generalisation abstraction constitute the inheritance (ISA [Brach83]) dependency. The inheritance dependency can be described in terms of binary dependencies. Several generalisation constraints, which are identified in the area of semantic data modelling [HuKi87], [PeMa88], can be defined by using inheritance dependency, and two intensional(sum and product [Kan90]) operations of union and intersection [Gust94b].

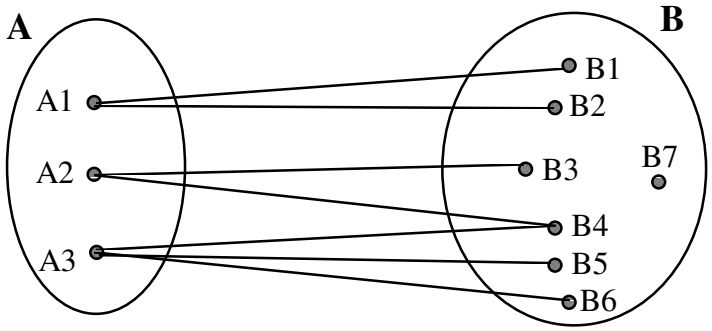
4.2.1 Abstraction of Aggregation

Aggregation relationships represent their extensions and semantics of them can be defined by the set of constraints which are valid regarding extensions of the relationships. Our assumption is that aggregation constraints can be specified using basic binary dependencies and some special aggregation operations among concepts. For example, if any instance of C is in association with one and only one

meaningful pair of instances of aggregation relationship (A Δ B) then this fact could be specified as follows: C $\longrightarrow$ (A Δ B) . Actually, the last expression demonstrate the criterion of aggregation.

The aggregation relationship is stable if it is dependent from another concept by one of the basic dependencies. Using stable relationships, it is possible to derive [Gust94c] unstable relationships. Unstable relationships can be derived according to the special inference rules. For example, if C $\longrightarrow$ (A Δ B) then B $\longmapsto$ A and A $\longmapsto$ B. The derived partial functional dependencies are representing two equivalent relationships. It should be noted that C $\longrightarrow$ (A Δ B) if and only if C $\longrightarrow$ A and C $\longrightarrow$ B . If the designer have identified unstable relationship, then he should be able to find another concept or even conceptual structure (set of dependencies) in order to make it stable by eliminating the semantic holes. Reengineering of requirements due to this rule is decreasing incompleteness and ambiguity of the enterprise model. At the same time, deleting of semantic holes, improve the quality of the specification. In principle, this reengineering strategy, allows the designer to define semantics of any relationship [Chen76] which could be specified or derived relation among entities, relationships and attributes. The straightforward way of the definition relationship C between two entities A and B could be by using the injection dependency, i.e. C ==> (A Δ B) . The aggregation criterion in this case, is an existence of natural language expression denoted by C. For modelling of more complicated situations, let us introduce more specific aggregation operations such as association (:) and grouping ({}). The operations give a possibility to define semantics for the notions of collection and grouping of classes which have been exploited in the area of semantic data modelling [HuKi87], [AbHu87].

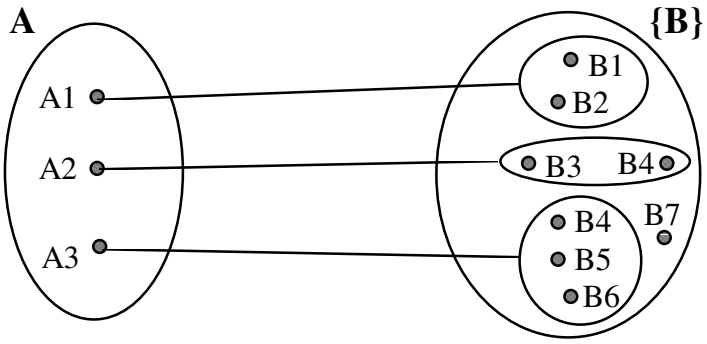
The criterion for specification the operation of grouping of B by A is identification of the total multivalued dependency($\longrightarrow\!\!\!\!\!\rightarrow$) between any two concepts. The example of total multivalued dependency and it extension between concepts A and B is presented in Fig.4.2.1.1.



(A Δ B)	
A1	B1
A1	B2
A2	B3
A2	B4
A3	B4
A3	B5
A3	B6

Fig. 4.2.1.1. The example of multivalued dependency

Notice that if A $\longrightarrow\!\!\!\!\!\rightarrow$ B then A $\longrightarrow$ {B} . Here: A $\longrightarrow$ {B} - the dependency of grouping of concept B according to the instances of A. The graphical interpretation of operation of grouping in this case is depicted in Fig.4.2.1.2.



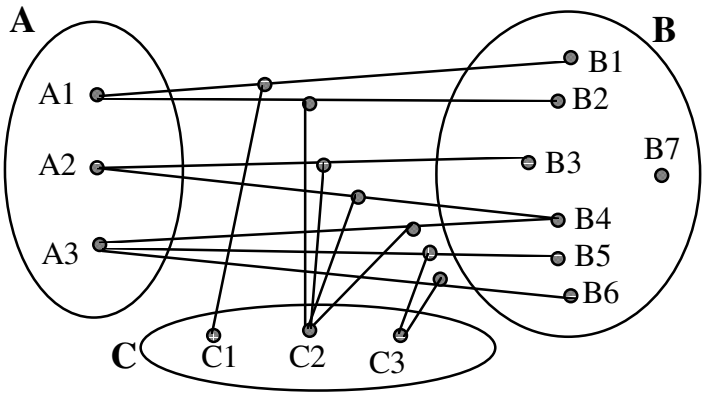
(A Δ {B})	
A1	B1 $\sqcap$ B2
A2	B3 $\sqcap$ B4
A3	B4 $\sqcap$ B5 $\sqcap$ B6

Fig. 4.2.1.2. The example of grouping B by A

If A $\longleftrightarrow$ {B} , one could derive simplest case of 'part-of' dependency, exploited in the area of knowledge representation [Stor93], i.e. if A $\longleftrightarrow$ {B} then B -part of-> A , where -part of-> denote the 'part-of' dependency between concepts. Notice that A $\longleftrightarrow$ {B} if and only if A $\longrightarrow$

{B} and{B} $\longrightarrow$ A . For example, if LIBRARY $\longleftrightarrow$ {BORROWER} then BORROWER -part of-> LIBRARY .

Let A:B denotes all meaningful associations of relationship (A Δ B) in the case A -part of-> B. The criterion for introducing of association operation of A in the context of B (denoted by A:B) is that B is in part-of dependency from A (A -part of-> B)and the existence of third concept C such that C is totally dependent from every meaningful association of the aggregate (A Δ B) , i.e. (A:B) $\longrightarrow$ C . On an other hand, if (A:B) $\longrightarrow$ C then every meaningful association of the instance of concept A with the instance of B is related to one and only one instance of C. Furthermore, concept B is in the ownership semantic relationship [Stor93] of A with the specific property C, i.e. B owns A with C. The graphical interpretation of association operation is depicted in Fig.4.2.1.3.



(A : B) Δ C			
A1	$\sqcap$	B1	C1
A1	$\sqcap$	B2	C2
A2	$\sqcap$	B3	C2
A2	$\sqcap$	B4	C2
A3	$\sqcap$	B4	C2
A3	$\sqcap$	B5	C3
A3	$\sqcap$	B6	C3

Fig. 4.2.1.3. The example of association of A with C in the context of B

For example, if BORROWER:LIBRARY $\longrightarrow$ TIME then LIBRARY owns BORROWER for some period of TIME . If STUDENT:COURSE $\longrightarrow$ GRADE then COURSE owns STUDENT with one and only one specific GRADE .

Several combinations of operations of association and grouping with basic dependencies allows us to describe the semantics of more complicated constructions. For example, it is possible to define the semantics such categories as time stamped entity and time stamped relationship [Tempora92].

4.2.2 Abstraction of Generalisation

Inheritance (ISA) dependencies [Brach83], [HuKi87], [PeMa88] are of various kinds. Although the inheritance constructs are well-understood [GuOr94],there is no complete agreement of interplay between the inheritance dependency and other types of basic constraints. For example, some of the researchers understand the inheritance in the same way as the inclusion dependency. To eliminate ambiguity, we will define the inheritance dependency in terms of other basic constraints.

Let $B \in X_i$, if X ==> B. Let $B \in X_f$, if X $\longrightarrow$ B. Let $B \in X_s$, if X==>>B. The inheritance dependency from concept X to Y is defined by X $\blacktriangleright$ Y if and only if X ==> Y , $Y_f \subseteq X_f$, $Y_i \subseteq X_i$, $Y_s \subseteq X_s$. Inheritance here is defined in terms of two abstractions: intensional and extensional [Sowa84]. If X $\blacktriangleright$ Y then the set of instances of concept Y must be injectionally dependant on concept X and the structure (intension) of concept Y must be included as part of concept X intensional structure. On the other hand, if the inheritance dependency from concept X to Y is specified then X automatically inherits the structure of Y with all the properties and constraints predefined on the elements of that structure. For example, let us have the following between concepts: PERSON $\longmapsto$ BOOK, PERSON ==> PASSPORT_NUMBER . Refinement of partial functional dependency from PERSON to BOOK could be done by introducing of new inheritance dependency BORROWER $\blacktriangleright$ PERSON and by specification of Borrower's dependency such as BORROWER $\longrightarrow$ BOOK . It's not difficult to realise that the meaning of the dependency PERSON $\longmapsto$ BOOK could be derived by combination of these two additionally introduced dependencies. So, the existence of redundant dependency PERSON $\longmapsto$ BOOK doesn't make sense anymore. At the same time the concept BORROWER inherits the intensional structure of PERSON (see definition of inheritance). So, the dependency BORROWER ==> PASSPORT_NUMBER is not specified, but could

be derived by default. The inherited dependencies are derived according to special inference rules[Gust94c]. It should be stressed that the inheritance dependency is a semantic relationship introduced here as a combination of the concept subsumption [Nilss94] and a kind of concept substructure relationship which is similar to the relationship based on syntactical graph forms [Sowa84], [FLDC86].

Synonymic type of the semantic ISA relationship[Stor93] is just another kind of the inheritance. It will be entitled as mutual inheritance dependency. The mutual inheritance dependency is defined by A $\longleftrightarrow$ B if and only if A $\longrightarrow$ B ,B $\longrightarrow$ A. For example: PERSON $\longleftrightarrow$ HUMAN . Several inheritance constraints among concepts such as partial and total overlapping, partial and total disjointness [GuOr94] of concepts Band C in comparison with more general concept A can be defined by union($\sqcup$) and intersection ($\sqcap$) of concepts. Operations can be specified among any number of concepts. Operation of union is defined as union of extensionals and intersection of intensionals of concepts B and C . So, the case of partial overlapping of Band C in comparison with more general concept A can be specified by($B \sqcup C$) $\longrightarrow$ A . For example, PERSON $\longleftarrow$ (STUDENT$\sqcup$ LECTURER$\sqcup$ MANAGER) .

Intersection of concepts is defined as intersection in an extensional and in an intensional sense as well. Interpretation of an intersection of concepts at the extensional dimension is understood as intersection of extensionals and in intensional sense - as union of intensionals of B and C. Intersection of concepts, in the case of multiple inheritance of concepts Band C by more specific concept A (A $\longrightarrow$ B and A$\longrightarrow$ C) can be specified by A $\longrightarrow$ ($B \sqcap C$) . For example, GIRL $\longrightarrow$(CHILD$\sqcap$ FEMALE).

Total overlapping [GuOr94] of concepts B and C in comparison with more general concept A can be defined by A $\longleftrightarrow$ ($B \sqcup C$) if and only if A $\longrightarrow$ ($B \sqcup C$) , A$\longleftarrow$ ($B \sqcup C$) . Here, every instance of concept A is also B or C instance, or neither B nor C. For example, PERSON$\longleftrightarrow$ (CITIZEN$\sqcup$ FOREIGNER) .

The partial overlapping is the special case of total overlapping and semantics of it can be specified by A$\longleftarrow$ ($B \sqcup C$) . Here, every instance of concept A is also B or C instance, or neither of them. For example, PERSON$\longleftarrow$(LIBRARIAN$\sqcup$ STUDENT).

Total disjointness [GuOr94] of concepts B and C in comparison with more general concept A can be defined by A $\longleftrightarrow$ ($B \nabla C$) if and only if A $\longleftrightarrow$ ($B \sqcup C$) and ($A \sqcap B$) = $\perp$. Here: ∇ - operation of generalisation, $\perp$ - the empty element which means 'nothing'. In this case, every instance of concept A is also either B or C instance, but not both of them. While partial disjointness constraint can be specified by A $\longleftarrow$ ($B \nabla C$) if and only if A $\longleftarrow$ ($B \sqcup C$) and ($A \sqcap B$) = $\perp$. Here every instance of concept A is also either B or C instance, or neither of them. It must be noted, that the notion of union of B and C is consistent with their least common generalisation ($B \sqcup C$) and the notion of intersection of B and C - with their greater common specialisation($A \sqcap B$) . For example: CHILD $\longleftrightarrow$ (GIRL ∇ BOY) , PERSON $\longleftarrow$ (GIRL ∇ BOY).

4.3 Bridging Static and Dynamic Representations

The behaviour of the universe of discourse can be specified by set ofactions which are able to perform changes on states when the special conditions are satisfied. Actual (input) and possible (output) states ofactions specify dynamic schema of flow in movement. Each flow is interpreted as a dynamic concept and states can be regarded as different roles of that concept [Lind95]. The problem here is to bridge the states of the dynamic concept to static schema of concepts. The static schema of concepts is represented by static dependencies and operations on concepts. Every action can be interpreted as static relationship where the flow is involved as an entity playing some role in that relationship. Graphical explanation of dynamic dependency between states of the concept (playing different roles) in the action is depicted in Fig.4.3.1.

Fig. 4.3.1. States and Roles of the Concept in the Action

The notions of the concept, the role, the state and the action are relative. It means that the same name can be interpreted by any of those categories. For example, the concept PAPER can be interpreted as the actual state (role1)and the concept BOOK - as the possible state. The most important aspect here is that the semantic dependencies among those elements must be specified unambiguously. The elements (states, roles, concepts in the action) are specified unambiguously, if and only if they are holeless concepts in the context of the static relationship of this action. If the concept is expected to be used as the representative of state, then the semantic hole must be eliminated in it. Notice that only holeless concepts can be used for definition of states of actions. Semantic holes can be eliminated by conceptual operations of prefixing, restriction and negation [Gust94b].

Operation of prefixing of B in the context of A is denoted by A. B if and only if concept B is dependent (by any kind of conceptual dependency) from A. Compound concept A.B identifies the subset of concept B instances which are associated with at least one instance of A. Operation of prefixing is characterised by the following property: if $A \mapsto\!\!\!\!\rightarrow B$ then $A.B \rightarrow B$. The graphical illustration of prefixing of concept B in the context of A is presented in Fig.4.3.2.

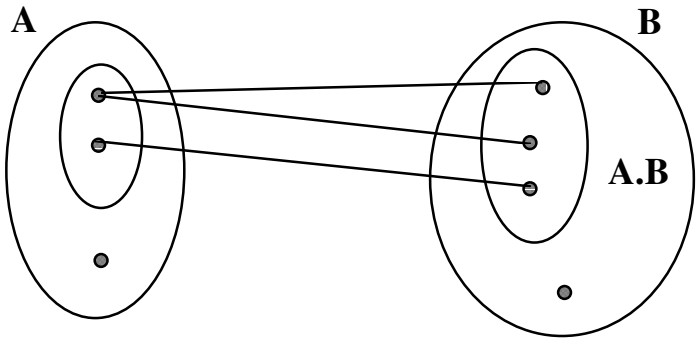


Fig. 4.3.2. The example of prefixing of B in the context of A

For example, if BOOK ==>> COPY then BOOK.COPY $\rightarrow$ COPY. Compound concept BOOK.COPY can be used for unambiguous identification of the entity instead of more general concept COPY (seeFig.4.3.4). Notice that if A ==>> B then $A \mapsto\!\!\!\!\rightarrow B$.

Operation of restriction allows the enterprise expert to define the presence of the concept with the same name in different roles [Lind95] or contexts. At that time, the same expert is able to specify several rules which would be interpreted as contradictory statements in the single context. Operation of restriction of A in the context B can denoted by A[B] if and only if $A \check{S} B$. Here: $\check{S}$ - the subsumption dependency. If A is the instance of some concept X which is totally dependent from B, i.e. $B \rightarrow X$ and $A \in X$,then $A \check{S} B$. Graphical illustration of restriction of concept A in the context of B is presented in Fig. 4.3.3.

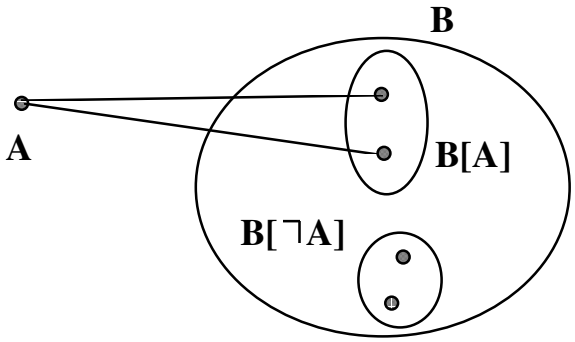


Fig. 4.3.3. The examples of restriction and negation

If $A \check{S} B$ then A can be interpreted as the role or even (under special conditions) as the instance of B. The graphical example of restriction is presented in Fig.4.3.4. Operation of restriction is very natural and important in the case of concurrent enterprise modelling. It allows to handle contradictory statements for several roles of the same object.

Operation of negation (¬) can be used exceptionally for restriction and prefixing of concepts. The meaning of negation is defined as follows:

A[¬ B] = A - A[B] , A[¬ B] ⟶ A , ¬ A.B = A - A.B , ¬ A.B ⟶ B .

Graphical illustration of negation of concept A in the context of B is presented in Fig. 4.3.4. For example, let us have the following dependencies: BOOK ==>> COPY , LOANED Š BOOK .

According to these dependencies, we are able to derive the set of compound concepts which are depicted in Fig.4.3.4.

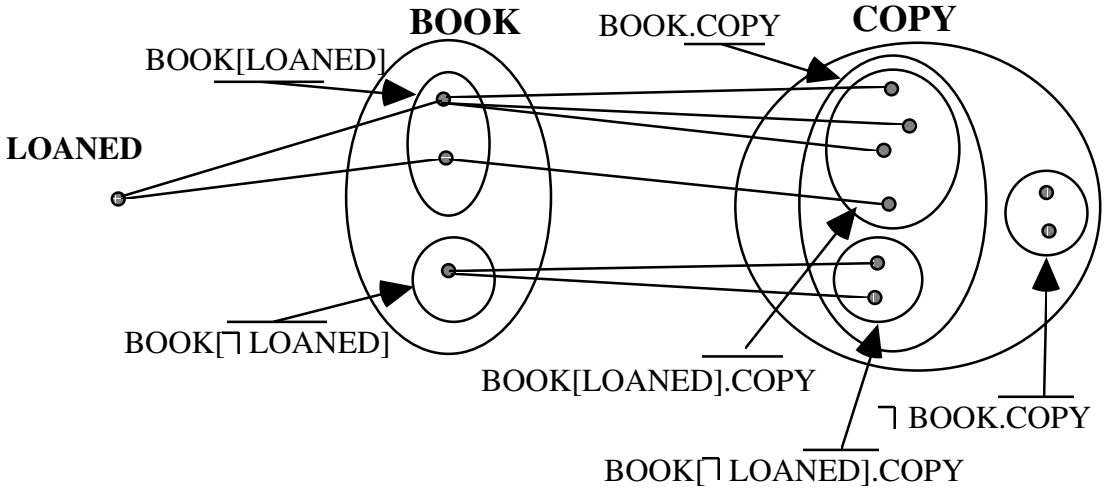


Fig. 4.3.3. The examples of prefixing, restriction and negation

Using operations of negation, restriction and prefixing, the domain expertis able to specify the holeless compound concepts. At the same time, by identifying and eliminating holes of semantics, the designer derive unambiguous specifications of complicated concepts which can denote states of actions, or roles of objects in the dynamic specification of the enterprise. The procedure of creation of compound concepts, makes the concurrence of enterprise modelling possible. It's happen due to deletion of ambiguity in denotations of states, roles, actions and flows of the enterprise. The compound concepts can be provided by new dependencies, i.e. be involved in a new even more complicated relationships. So, by the same procedure it is possible to specify an intensional structure of these complex concepts.

5. Concluding Remarks

Modelling primitives applied for specification of semantics and pragmatics[LLS94] of the enterprise are usually based on Entity - Relationship like(ER) notations [Tempora92], [F3_94]. Although these notations are much more readable and understandable for human beings compared with logical formalisms, there is no uniform theoretical basis and even agreement whether the concept should be specified as entity, or as relationship, attribute, action, actor, goal, etc. Usually, the same object can have all those several roles, but just in different context of interpretation. One of the main difficulties in using of ER representations is that they require to distinguish sharply among the meta-categories, i.e. the designer is enforced to make decision whether the concept will be interpreted as an entity, or as a relationship, or as an action, etc. Strict interpretation of categories can cause different representations for the same part of the enterprise, even if the same modelling constructs are used. Because of different interpretations of the same concept, the designers are able in parallel to specify semantically equivalent, but structurally different representations [BaLe86]. Several descriptions with the same meaning makes very complex the process of concurrent enterprise modelling. The possibility of integration of different representations in several levels of decomposition simplifies the parallel development of the enterprise model by independent experts.

Abstraction principle [Griet82] means that the enterprise model must refer exclusively to rules of the universe of discourse, while sharp classification of categories imply the rules which are dependent on the representation dimension [Pohl93]. The motivation behind this principle is the need to

simplify the process of requirements modelling by concentrating only on conceptually relevant aspects and disregarding everything else. ER kind of notation is representation dependent. Lack of relativeness of interpretation of basic categories of several representations, injure the main principles of conceptualisation [Griet82]. The enterprise requirements specification must possess number of desirable qualities[IEEE84], [Louc92] such as independence, abstraction, formality, traceability, unambiguity, completeness, consistency and so on. We expect, that approach to the concurrent enterprise modelling, could positively influence all these properties. For example, the ambiguity could be eliminated by creating unique compound names according to the concept model dependencies. The completeness could be achieved by decomposition of several pragmatic and semantic relationships to the most primitive ones. The risk of inconsistency could be lowered by comparison of two structures of synonymic objects and discovering contradictions among the basic constraints. Some of the conceptual dependencies specify the most elementary atomic pieces of semantic constructs. On the basis of very primary constraints, it is possible to compare more general schemes, so that one could detect the similarity or equivalency of the specifications. Some abstract non primitive dependencies, such as inheritance, motivation, influence, etc., are defined on the basis of more simple ones. It allows interpretation of each semantic or pragmatic construct in terms of most primitive semantic relationships. This kind of formality of the models let us have the interaction among several dependencies and, so, interplay among different submodels is automatically present. Positive side of this complicated building of dependencies is that consequences of changes in the viewpoint of one enterprise expert would be possible to trace into another.

The dependencies constitute a unified basis for integratability over representation caused by the enterprise constraints of different origins. Usually, the consequences of using of several models, such as ER diagrams and data flow diagrams [Yourd89], for description static and dynamic aspects of the enterprise are that interaction between two specifications is lost. It means that designers becoming unable to trace the specification as a whole. Furthermore, it is impossible to achieve uniform specification of the objectives structure if the designer can not distinguish among the most primitive dependencies of static and behavioural parts. The purpose for introducing of the homogeneous basis for enterprise modelling is that developed specification could be unambiguously understood among concurrent developers and that it could be used for assessment of quality of the requirements specification.

In this paper has been presented an approach for interpretation of several semantic and pragmatic relationships in different classes of models. The idea was illustrated using the framework of enterprise modelling which covers large scope of several information systems modelling aspects. The enterprise submodels are based each on a number of dependencies and relationships of different types. They represent different 'worlds' of requirements and addresses the 'why', the ‘how', the 'who' and the 'what' perspectives in the enterprise modelling. Additionally, using of the basic conceptual dependencies, allows interpretation of the 'where' and the ‘when' perspectives as well. It should be noted, that we are not proposing the new model for specification of requirements, but we are discussing the set of basic dependencies which could be used for describing of several semantic and pragmatic relationships indifferent classes of models. Of course, the question of identification the minimal set of them is still open.

The dependencies among concepts, constitute the uniform basis for enterprise modelling. So, they should be specified as precisely as possible. If the designer would use weaker primitives, then incomplete description of semantics of relationships can cause incompleteness and ambiguity in the specification of requirements. Basic constraints are of interest because they capture semantic detail about structure of relationships and control the procedure of definition of compound conceptual expressions, which are assumed as definitions of states of actions, derived concepts, relationships and business rules in terms of most primitive elements. By introducing uniform basis for modelling semantic and pragmatic dependencies of the enterprise, the artificial ambiguities of representations could be eliminated. Other subset of ambiguities are known as naming conflicts. They usually arise when the enterprise modelling is performed independently by parallel domain experts. The origin of those ambiguities is quite natural. In order to avoid the subset of natural ambiguities, the names of concepts must be extended according to the special rules, using conceptual operations of prefixing, specialisation and negation. The criterion for the extension of names of objects, is elimination of semantic holes in the static relationships of concepts.

The ultimate objective of this paper is to introduce the homogeneous basis for understanding of semantic and pragmatic dependencies in different classes of models. The uniform approach to modelling of basic categories of enterprise models could contribute for improving desirable qualities

such as independence, traceability, unambiguity, completeness and consistency of the enterprise representations developed in parallel by independent domain experts. The goal for introducing of such homogeneous understanding is that the specifications could be unambiguously interpreted among various stakeholders.

Acknowledgements

This work was partially funded by ESPRIT Basic Research Action 6353 entitled NATURE which is concerned with Novel Approaches to Theories Underlying Requirements Engineering. NATURE partners are RWTH Aachen (Germany), University of Paris I (France), City University (London, UK),FORTH Research Institute (Crete, Greece) and Swedish Institute for Systems Development (Kista, Sweden).

7. References

[BaLe86]
C Batini, M Lenzerini, S B Navathe. A Comparative Analysis of Methodologies for Data Base Schema Integration. ACM Computing Surveys, Vol. 18, No.4, pp. 323 - 363, 1986.

[Brach79]
R Brachman. On the Epistemological Status of Semantic Networks. Associative Networks, Academic Press, pp. 3-50, 1979.

[Brach83]
R Brachman. What IS-A is and isn't: An analysis of taxonomic links in semantic networks. Computer,16(10), pp. 30-36, 1983.

[Bub93]
J A Bubenko. Extending the Scope of Information Modelling. Fourth International Workshop on the Deductive Approach to Information Systems and Databases, Polytechnical University of Catalonia, pp. 73-97, 1993.

[Chen76]
P Chen. The Entity-Relationship Model: Towards a unified view of data. ACM Transactions on database systems 1, 1, 1976, pp. 9-36.

[Dietz91]
J L G Dietz. The Essential System Model. Third Int. Conf. on Advanced Information System Engineering CAiSE'91, Springer-Verlag, pp.441-460, 1991.

[DLF93]
A Dardenne, A van Lamsweerde, S Fickas. Goal Directed Requirements Acquisition. Science of Computer Programming, 20, pp. 3-50, 1993.

[F3_94]
F3 Consortium. 'F3 Reference Manual (Esprit III Project 6612)', SISU, Kista, Sweden, 1994.

[FLDC86]
J Fargues, M C Landau, A Dugourd, L Catach. Conceptual graphs for Semantics and Knowledge processing. In: IBM J. Res. Develop., vol.30, No 1, pp. 70 -79, 1986.

[Griet82]
J J van Griethuisen (ed.). Concepts and Terminology for the Conceptual Schema and Information Base, Report ISO TC97/SC5/WG5, No 695, 1982.

[GuOr94]
R Gustas, T Orci. Database Design Methodology of ORES. Royal Institute of Technology, Stockholm, Sweden, 1994.

[Gust94a]
R Gustas. From Conceptual to Business Modeling. Proc. of Baltic Workshop on National Infrastructure Databases - Baltic DB'94, Vilnius, Lithuania, 17-20 May, pp.218-229, 1994 .

[Gust94b]
R Gustas. Towards Understanding and Formal Definition of Conceptual Constraints. Information Modeling and Knowledge Bases, IV, IOS Press, pp.381 - 399, 1995.

[Gust94c]

R Gustas. Towards Foundations of Database Schema Independent Modelling of Concepts. Proceedings of the Second International East-West Database Workshop (Austria), Springer-Verlag, 1994.

[HuKi87]
R Hull, R King. Semantic Database Modelling: Survey, Applications, and Research Issues. ACM Computing Surveys, Vol.19, No.3, pp.201-260, 1987.

[IEEE84]
IEEE. An American National Standart. IEEE Guide to Software Requirements Specifications. The Institute of Electrical and Electronics Engineers, Inc., ANSI/IEEE std 830-1984, New York, 1984.

[JaPo93]
M Jarke, K Pohl. Establishing Visions in Context: Toward a Model of Requirement Engineering. 14th Int. Conf. on Information Systems, Orlando, USA, 1993.

[JaKe94]
S Jacobs, S Kethers. Improving Communication and Decision Making within Quality Function Deployment. Proc. of Concurrent Engineering: Research and Applications, Pittsburgh, Pennsylvania, pp. 203 - 212, 1994.

[Kan90]
H Kangassalo. Foundations of Conceptual Modelling: A Theory Construction View. In Information Modelling and Knowledge Bases, IOS Press,1990, pp.19-35.

[Lind95]
P Lindgren. Anything, Everything and Things Playing Roles. International Conf. on Information System Concepts, Marburg, 1995.

[Louc92]
P Loucopoulos. Conceptual Modeling. Conceptual Modeling, Databases and Case: An Integrated view of Information Systems development, John Wiley,pp.1-26, 1992.

[LLS94]
O I Lindland, G Sindre, A Solvberg. Understanding of quality in conceptual modelling. IEEE Software, pp.42-49, 1994.

[Nilss94]
J F Nilsson. An Algebraic Logic for Concept Structures. In: Information Modelling and Knowledge Bases V, IOS Press, pp. 75 - 84,1994.

[PeMa88]
J Peckham, F Maryansky. Semantic Data Models. ACM Computing Surveys,Vol.20, No.3, pp.153-189, 1988.

[Pohl93]
K Pohl. The three Dimensions of Requirements Engineering. Int. Conf. on Advanced Information System Engineering CAiSE'93, Springer-Verlag, 1993.

[RBPEL91]
J Rumbaugh, M Blaha, W Premerlani, F Eddy, W Lorensen. Object-Oriented Modelling and Design, Prentice Hall Int., 1991.

[SBWJ94]
A Sutcliffe, J A Bubenko jr., B Wangler, S Jacobs. Goal Driven Requirements Engineering (submitted paper).

[Simon84]
H A Simon. The Sciences of the Artificial. MIT Press, 1984.

[Smith77]
J M Smith, D C P Smith. Data Base Abstractions: Aggregation and Generalization. ACM Transactions on Database Systems, Vol.2, No2, pp.105-133, 1977.

[Sowa84]
J F Sowa. Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publ. Co., 1984.

[SoZa92]
J F Sowa, J A Zachman. Extending and Formalizing the Framework for Information Systems Architecture. IBM Systems Journal, 31(3), pp. 590 -616, 1992.

[Stor93]
V C Storey. Understanding Semantic Relationships. VLDB Journal, Vol.2,pp.455-487, 1993.

[Tempora92]
TEMPORA Consortium. 'Tempora (P2469) Project Manual', BIM Information Technology, Belgium, 1992.

[TsiLo82]
D Tsichritzis, F Lochovsky. Data Models. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Yu94]
E S K Yu. Modelling Strategic Relationships for Process Reengineering. PhD theses, Dept. of Computer Sciencies, University of Toronto, Canada,1994.

[Yourd89]
E Yourdon. Modern Structured Analysis, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[Wang93]
B Wangler. Contributions to Functional Requirements Modelling. PhD theses, Dept. of Computer and Systems Sciencies, Royal Institute of Technology, Stockholm, Sweden, 1993.

[Will93]
H Willars. TRIAD Modelleringshandbok: Business Modelling Overview, SISU, 1993.