Deontic Analysis Patterns

Paul Johannesson & Petia Wohed

Dept. of Computer and Systems Sciences
Stockholm Uniersity/KTH
Electrum 230, 164 40 Kista, Sweden
email: {pajo, petia}@dsv.su.se
url: http//www.dsv.su.se/

Abstract

In order to reduce the costs for systems development, methods for the reuse of specification knowledge have been developed. One approach is to build libraries of reusable analysis patterns, i.e. abstract models describing the generic features of a type of situation that may occur in many different domains. In this paper, we propose a novel analysis pattern based on a deontic perspective. The basic components of this pattern are object types describing obligations, the parties involved in these obligations and their respective roles, and the speech acts that create and manage the obligations. We argue that this pattern captures specification knowledge at an appropriate level of abstraction, has a wide applicability, and effectively supports designers in the construction of models. Furthermore, we show how deontic objects at different levels are interrelated.

1. Introduction

When constructing large information systems, the key to success lies in eliciting and representing requirements. Experience has shown that these activities are difficult as well as time consuming. Even with the use of CASE tools, capturing and representing requirements remain one of the major costs in building information systems. One reason for this is that requirements and domain knowledge are still often captured from scratch for each new system to be built, even for systems within the same general area. This duplication of effort results in high costs and hinders the construction of larger and more knowledge-intensive systems. To overcome these problems, systems analysts and software engineers must find ways of sharing, reusing, and extending systems.

There are many senses in which the knowledge contained in a system can be shared and reused. One form of reuse is code reuse, which can be realised through modules invoking each other as procedures from a function library. Code reuse can also be realised through the inclusion of source specifications, i.e. the content of one module is copied into another module at design time. Another form of reuse is through the exchange of techniques, meaning that the content of a system module is not directly used; instead, the solution approach behind the module is communicated in a way that facilitates its re-implementation.

Essential to all these forms of reuse is the build-up and maintenance of a library of reusable modules. Such a library can be utilised in many different ways. It can be searched through keywords that retrieve and select modules based on their functionality, as suggested in [Burton87]. However, keywords describing the functionality of systems cannot effectively support reuse across different applications. Faceted classification schemes, [Prieto-Diaz87], overcome some of the problems of simple keyword retrieval by describing non-functional features of modules and by providing a lexicon to support differences in terminology. The contents of the modules in a library may vary widely, from source code to generic objects and models. The latter are abstract models that describe the generic features of a type of situation that may occur in many different contexts; these abstract models are commonly known as *patterns*.

Patterns come in a large number of varieties. One of these is the *design pattern*, [Gamma95], which has received much attention in the software engineering community. A design pattern is a description of communicating objects and classes that are customized to solve a general design problem in a particular context□, [Gamma95]. Thus, a design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design. While a design pattern addresses the design stage in systems development, an *analysis pattern* concerns the analysis and specification stage. An analysis pattern consists of an application independent model of a domain structure, e.g. a model of time or causality at a higher level of abstraction, or a model of library systems at a lower level. An analysis pattern describes, at an arbitrary level of abstraction, a set of real-world objects, their interrelationships, and the rules that govern their behaviour and state. Some examples of analysis patterns are domain abstractions as discussed in [Maiden92], the analysis patterns in [Fowler97], and data model patterns as introduced in [Hay96]. The notion of analysis patterns is similar to that of ontologies in the knowledge representation community, [Neches91]. An ontology is commonly defined as an explicit specification of a conceptualisation, where a conceptualisation consists of □objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them□, [Gruber95].

An important quality requirement on an analysis pattern is that it be sufficiently general, i.e. it should be sufficiently abstract to be applied to many different application models. Another quality requirement is that analysis patterns should minimise the cognitive distance, i.e. it should be easier to understand and apply the analysis pattern than to model a part of the application from scratch. Clearly, these two quality requirements are in conflict as highly abstract patterns may be quite difficult to apply. The difficulty to construct patterns satisfying both these requirements is probably a major reason why analysis patterns are not yet widely used. An important research task is, therefore, to identify analysis patterns at an appropriate level of abstraction.

The purpose of this paper is to show how deontic concepts can be used for modelling fundamental aspects of social reality in the context of information systems design. Deontic concepts have been applied in different areas of computer science; for a survey, see [Wieringa93]. We believe that deontic concepts have an important role to play also in enterprise modelling, and that they can be used for extending existing ontologies in this area, such as TOVE, [Gruninger97]. In order to show this, we introduce a novel analysis pattern. The basic components of this pattern are object types describing obligations, the parties involved in these obligations and their respective roles, the speech acts that create and delete these obligations, the subject matters of and the reasons for the obligations. We claim that this analysis pattern is both sufficiently general and has a low cognitive distance, which means that it will be useful in a large number of areas as well as simple to apply. The paper is organized as follows. Section 2 introduces the modelling formalism and notation used in the rest of the paper. Section 3 presents a basic deontic pattern; the components of this pattern are described as well as its typical variants. Section 4 extends this deontic pattern by introducing different types of deontic objects and discussing the different kinds of roles parties can play with respect to a deontic object. Finally, Section 5 summarises the paper and gives suggestions for further research.

2. Modelling Formalism

The basic concept in conceptual modelling approaches is the object. Objects that are similar are grouped together into object types, such as Person and Department. Objects have different properties, which are modelled by attributes or relationships. In our graphical notation, see Figure 1, object types are represented by rectangles, attributes are represented by bulleted lists inside the object types and relationships are represented by labelled lines. Both a relationship and its inverse are represented with the same line. The name of a relationship is placed closest to the object type that is its domain. The object type on the other side of the line representing the relationship is called its range. The graphical notation can only represent cardinality constraints and generalisation relationships. The generalisation relationships are shown by drawing the subtypes of an object type inside the rectangle modelling this object type. The cardinality constraints specify for each relationship if it is single-valued, injective, total and surjective. A relationship is single-valued when each instance of its domain is connected with at most one instances of its range. A relationship which is not single-valued is multi-valued and is depicted by a fork connecting the line representing the relationship with its range. A relationship is total when each instance of its domain is connected to at least one instance in its range. A relationship that is not total is partial, which is shown by a dotted line on the half of the line that is closest to its domain. A relationship is injective (surjective) when its inverse is single valued (total).

3. A Basic Deontic Pattern

In this section, we introduce a basic deontic pattern that describes obligations among agents. We start by classifying the object types occurring in different pattern in two categories: concrete and abstract object types. Concrete object types have instances that are physical objects. BUILDING, for instance, is a typical example of a concrete object type. Every object that is not concrete is an abstract object. An example of an abstract object type is ORGANISATION. Note that Stockholm University could appear as an instance of both BUILDING and ORGANISATION, but there still are two different objects, the building in the first case and the institution in the second, both being denoted by the same term.

An important characteristic of certain abstract objects is that they entail obligations. For instance, Peter’s employment at Stockholm University entails that Peter is obliged to work a number of hours for the university, which in turn is obliged to pay him salary. In contrast, the existence of the institution Stockholm University does not by itself entail any obligations. Abstract objects that entail obligations are called DEONTIC OBJECTs. Examples of deontic object are employments, work orders, bookings, and marriages. A general deontic pattern is shown in Figure 1. In the right upper corner of each object type an abbreviation of the name is given. The deontic analysis pattern contains the following components:

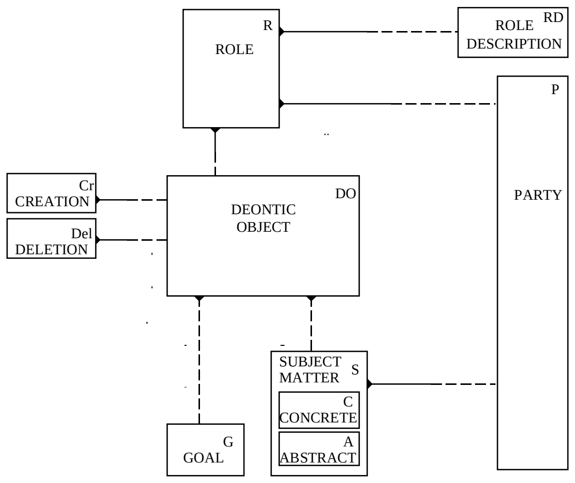

Figure 1: A Basic Deontic Pattern

DEONTIC OBJECT

Deontic objects are objects that group together obligations for different parties. Typical properties for a deontic object are the period for which it exists and a number of parameters representing important information about the created obligations. For instance, the attributes start date, percent and salary for an object type EMPLOYMENT represent information connected to the entailed obligations, namely the start date for the obligations, the working time an employee is required to do, and the salary the employer has to pay.

ROLE and ROLE DESCRIPTION

A deontic object usually concerns more than one party, as most obligations are directed from one party to another, [Herrestad95]. As a consequence of this, DEONTIC OBJECT is always associated to the object type PARTY with at least two different relationships. In an employment, these relationships are the employee and the employer. In a lending, they are the lender and the borrower. However, a deontic object may involve more than two parties as well. To build a general model, which can represent a flexible number of parties involved in a deontic object, we use the object types ROLE and ROLE DESCRIPTION. ROLE DESCRIPTION brings general information about each particular ROLE, while ROLE shows who is playing a role and for which time.

SUBJECT MATTER

The subject matter of the obligations of a deontic object are represented by the object type SUBJECT MATTER. Typically, the subject matter of a deontic object is an activity that is to be carried out, for example to return a borrowed object.

CREATION and DELETION

The object types CREATION and DELETION model the creation and deletion of deontic objects. Consider for instance a Marriage pattern where a deontic object type MARRIGE is connected to a type WEDDING - the object type representing the creation of a marriage. Moreover a marriage can be ended up by a divorce, which is represented by the corresponding object type. The creation and deletion of a deontic object are performed by means of speech acts. In order to model people performing such speech acts, additional relationships to PARTY are required.

GOAL

The purpose of the object type GOAL is to represent the reason for the existence of a deontic object. The existence of a deontic object needs to be motivated when it is created under certain conditions. For instance, when a head teacher books an assistant for her course, the reason for doing so is to make it possible to carry out some activity. In such cases it is common to keep information not only for a booking but also about the activity which is the reason for the booking. Below, we give examples of modifications of the deontic pattern, which results in more concrete analysis patterns.

Employment Pattern

In Figure 2, an analysis pattern for employment is given, where the object type EMPLOYMENT is associated to an ORGANISATION that is the employer, a PERSON who is an employee and one or several POSITION ASSIGNMENTs. A POSITION ASSIGNMENT shows the share of an EMPLOYMENT to a POSITION. For instance, the model has the capacity to represent that a person is employed full-time at a company from a specific date, and that the employment is shared in 40% as project leader and 60% as consultant. An employment gives rise to a large number of obligations, e.g. that the employee should work a specified amount of time and that the employer should pay a salary.

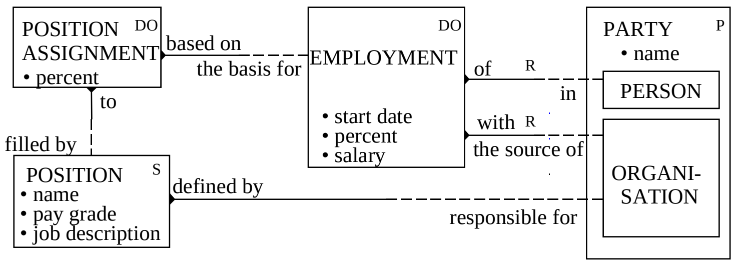

Figure 2: Employment pattern

Work Order Pattern

An analysis pattern for work orders is given in Figure 3. An example of a WORK ORDER is the order that a certain course shall be given by a department during a specified period of time. The department is then responsible to see to it that the course is given during this time period. Furthermore, a number of parties can be involved in a work order and they can play different roles. Examples of different WORK ORDER ROLE TYPEs are head teacher and assistant, where a head teacher is also the examiner and an assistant helps with teaching and administration the head teacher. The object type WORK ORDER ROLE is introduced to show which role type a party has in a work order.

Furthermore, a work order can give authorisation for a number of activities. An ACTIVITY is for instance a specific lecture. Finally, the object type ACTIVITY ASSIGNMENT is introduced to show who is going to perform a particular activity.

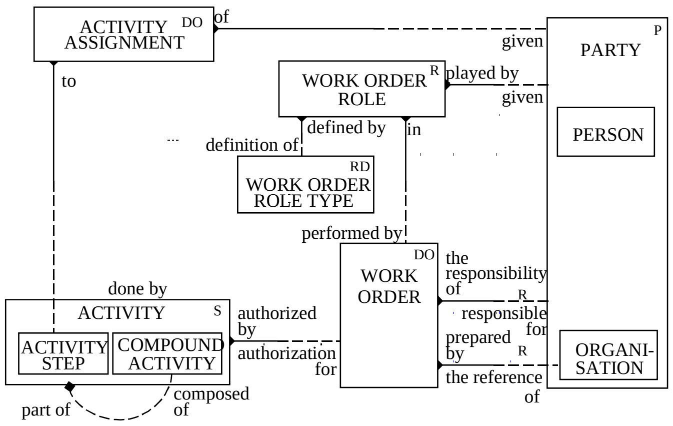

Figure 3: Work order pattern

When constructing a concrete analysis pattern from the deontic pattern, it is common to omit some components or collapse one component into another one. Some of the most common omissions are the following:

- **ROLE** omitted. The object types ROLE and ROLE DESCRIPTION are often omitted and replaced by relationships between DEONTIC OBJECT and PARTY. This omission is common when there is only a small and fixed number of established roles, e.g. wife and husband in a marriage, where these roles can be modelled by means of relationships wife and husband. When the number of roles may vary or when information is needed about the roles, the object types ROLE and ROLE DESCRIPTION are required.
- **PARTY** omitted. In some cases the role of some party is omitted. An example of this is given in Figure 3, where only one role in ACTIVITY ASSIGNMENT is modelled - the performer's role. However, there exists also another role, that of the party who has assigned the activity. The reason that this role is not modelled is that it can be derived from the description for a WORK ORDER ROLE TYPE who is responsible for the performance of a work order and may thereby delegate activity assignments.
- **SUBJECT MATTER** omitted. The object type SUBJECT MATTER is omitted when the deontic object does not primarily entail obligations for a particular action or object. For example, a marriage entails a large number of obligations in many different circumstances, and it is not possible to single out a specific obligation as more important than all the others. A work order, on the other hand, entails primarily the obligation to carry out a particular activity, while the other obligations in this context are less important; so the object type ACTIVITY is included in the WORK ORDER pattern.

The analysis patterns in Figures 2 - 3 are very similar to those found in the pattern literature, e.g. [Fowler96]. Most of the patterns in these references are presented as isolated models without any relationships to each other. However, there obviously exist different types of relationships between the patterns and an important task is to make these explicit. Roughly speaking, there are at least two types of relationships between deontic objects. First, one deontic object can be the motivation for another deontic object, e.g. a booking of a room can be motivated by a work order to carry out some activity that requires a room. Secondly, a deontic object can be established according to, or in the context of, another deontic object. In the rest of the paper, we will discuss how the basic deontic pattern can be extended in order to express relationships among deontic objects.

4. An Extended Deontic Pattern

In order to express relationships between deontic objects, it is necessary to provide a more detailed analysis of the different types of deontic objects that exist. In Figure 4, an extension of the basic deontic pattern from Figure 1 is shown. To manage the complexity, which such an extension brings to the model, we similarly to [Fowler97] distinguish between two levels, namely a knowledge level and an operational level. The knowledge level models the kinds of objects that exist, the relations between these, and the general rules in a domain, whereas the operational level captures the every day events and the objects involved in these. For example, objects like Peter, Stockholm University, Peter's employment at Stockholm University are instances of the object types PARTY and DEONTIC OBJECT at the operational level, whereas the corresponding objects person, university and employment are instances at the knowledge level of the object types PARTY TYPE and DEONTIC OBJECT TYPE. One more difference between these two levels is that much of the information at the operational level is time sensitive, i.e. it is important to keep information about when certain events occur or the time interval for validity periods. Summarising, the knowledge level captures the rules established by the culture or the law system that a universe of discourse has to obey, while the operational level keeps information about occurrences following these rules. It is then not surprising that the operational level is almost symmetrical to the knowledge level, which means that most of the object types/relationships at the former level have a corresponding object type/relationship at

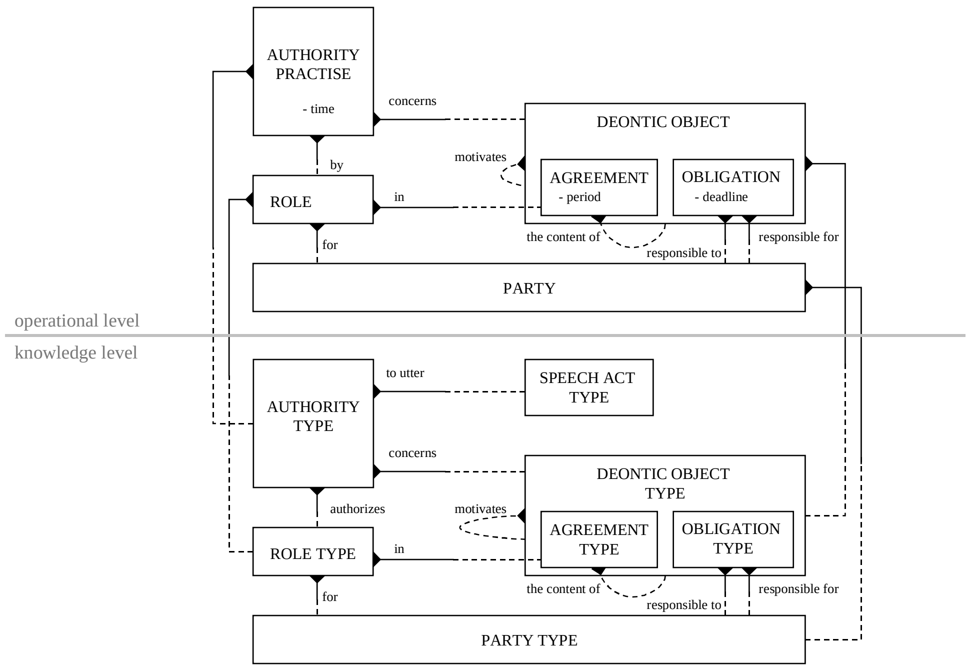

Figure 4: An Extended Deontic Pattern

the latter level. In the rest of this section, we are describing the object types from the operational level and when necessary clarifying them with the help of their corresponding object types from the knowledge level.

Turning now to the pattern in Figure 4, one of its extensions is the distinction between two classes of deontic objects: obligations and agreements. An object is an instance of OBLIGATION if it is a deontic object entailing a single obligation, or a small number of obligations. In contrast, an object is an instance of AGREEMENT if it is a deontic object where it is not possible to specify a small number of obligations that constitute the object. Instead, an agreement provides a context for other deontic objects by regulating how they can be created, destroyed, and modified, and by specifying rules telling how violations of obligations should be handled. An example of an instance of an OBLIGATION is an ACTIVITY ASSIGNMENT from Figure 3, which entails the obligation to perform a certain activity step that cannot be divided into smaller steps, which means that it obliges to a single action. An example of an AGREEMENT is an EMPLOYMENT (Figure 2), which regulates the rights and duties of the employee and the employer. Furthermore, an employment provides a context for a WORK ORDER (Figure 3) from the employer to the employee, where the work order entailing the obligation to produce something or provide a service is a deontic object itself. This is modelled by the relationship context for. There is also another relationship between deontic objects that has to do with the reason, or motivation, for creating a deontic object. For example, a room can be booked (one deontic object) in order to fulfil the obligation of giving a lecture (another deontic object). This relation covers the object type GOAL from Figure 1.

One more difference between an OBLIGATION and an AGREEMENT is the way they are related to the object type PARTY. For an obligation there is always one party who is responsible for fulfilling the obligation, and there is another party for whom the obligation is to be fulfilled, which is represented by the attributes responsible for and responsible to respectively. Normally, the party who requested the deontic object is the same as the one for whom the associated obligation is to be fulfilled, and the party who approved of the creation of the object has the responsibility to fulfil the associated obligation. In contrast, for an agreement a number of roles for different parties are created and the responsibilities are not connected to only one of those roles but are distributed among the different roles. For example, in an employment agreement there is an employee and an employer role and both roles imply responsibilities. The object type ROLE TYPE at the knowledge level and ROLE at the operational level represents this, where head of the department is an instance of the first one and Peter as the head of a specific philosophy department is an instance of the second one. Note that ROLE TYPE corresponds to the ROLE DESCRIPTION in Figure 1.

A role can be viewed as a bunch of authorities for performing certain actions. Authorities are modelled by means of AUTHORITY TYPE, which given a role type and a deontic object type specifies one speech act type meaning that someone who has the role is authorised to perform that type of speech act for that deontic object type. To clarify this, we give a few examples concerning one deontic object type - position establishment. A project manager at a laboratory at a department has the authority to request a new position to his project. Each request for a new position is then approved by the head of the laboratory and finally accepted by the head of the department. Afterwards the secretary registers it. Request, approval, acceptance and registration are all instances of SPEECH ACT TYPE. When the approval speech act is uttered by a role holder with the right authority a new deontic object is created – a position. Since the role allocations are results from agreements, agreements can be viewed as regulating the creation, modification, and deletion of different deontic objects. Finally, an example of AUTHORITY PRACTISE is the fact that Ann in the role of project leader at Stockholm University requests the 1 June 98 one more research assistant position to her project. Note that on the knowledge level, AUTHORITY TYPEs specify the authorities associated with certain roles, while at the operational level, AUTHORITY PRACTISEs describe actions that are carried out using certain authorities.

5. Summary and Further Research

The deontic pattern introduced in this paper can serve as a very abstract analysis pattern in a library of reusable modules. The other modules will contain specialisations of the deontic pattern, such as lending, booking, and sale. The deontic pattern will serve as an entry point to the library and as a template for structuring the other patterns. The deontic pattern may be used in schema specification in many different ways. First, one of its specialisations can be integrated directly into a schema without any modifications. Secondly, a fragment of a schema can be compared to the deontic pattern in order to check the correctness and completeness of the schema. Utilising the deontic pattern in these manners can improve the quality of schema specification in several respects. In particular, the completeness of a schema can be improved, as comparing a schema fragment to a pattern will assist a designer in identifying aspects of the application that have been left out in the specification. Furthermore, the stability of a schema can be increased by adding components from the deontic pattern, which are not strictly necessary for the current application, but can make later requirements easier to accommodate. Finally, the deontic pattern can facilitate the documentation of a schema by providing templates for the different types of obligations that may occur. Instantiating these templates for a particular application can be a most effective way of constructing a comprehensive documentation.

The analysis pattern introduced in this paper is described solely in terms of objects and relationships, and it therefore provides only a superficial representation of deontic structures. In order to provide a deeper representation, the deontic structures must be described by means of complementary formalisms. Adequate formalisms for this purpose may range from DEMO models, [Dietz92], to illocutionary and deontic logics, [Assenova96], [Johannesson98b], [Dignum95]. Utilising these formalisms, some of the vague notions in the deontic pattern, in particular subject matter and goal, can be made more precise. Furthermore, it will become possible to analyse in greater detail the different variants of the deontic pattern, and thereby construct a systematic structure of these variants, which will help to build a library of reusable specifications. Another line of research is to empirically investigate the usefulness of the deontic pattern. Such an investigation would focus on two distinct issues. First, the applicability of the deontic pattern should be measured by studying how frequently it occurs in applications from different domains. Secondly, one should investigate how well the deontic pattern supports a systems designer in the specification task - this can be done by comparing designers that are familiar with the deontic pattern with those that are not with respect to results and ways of working.

References

[Assenova96] P. Assenova and P. Johannesson, "First Order Action Logic - An approach for Modelling the Communication Process between Agents", *in First International Workshop on Communications Modelling - The Language/Action Perspective*, Ed. J. D. F. Dignum, E. Verharen and H. Weigand, London, Springer Verlag, 1996.

[Burton87] B. Burton, R. Aragon, S. Bailey, K. Koehler and L. Mayes, "The Reusable Software Library", *IEEE Software*, pp. 25 - 33, 1987.

[Dietz92] J. Dietz, "Modelling Communication in Organizations", in *Linguistic Instruments in Knowledge Engineering*, Ed. R. v. d. Riet, pp. 131 - 142, Elsevier Science Publishers, 1992.

[Dignum95] F. Dignum and H. Weigand, "Modelling Communication between Cooperative Systems", in *CAiSE*, pp. 1995.

[Fowler97] M. Fowler, *Analysis Patterns - Reusable Object Models*, Addison-Wesley, 1997.

[Gamma95] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns*, Addison-Wesley, 1995.

[Gruber95] T. Gruber, “Toward principles for the design of ontologies used for knowledge sharing”, *International Journal of Human Computer Studies*, No 43, pp. 907 Ð 928, 1995

[Gruninger97] M. Gruninger, “Integrated Ontologies for Enterprise Modelling”, url: http://www.ie.utoronto.ca/EIL/tove/onto-sum.fm.html

[Hay96] D. C. Hay, *Data Model Patterns*, Dorset House Publishing, New York, 1996.

[Herrestad95] H. Herrestad and C. Krogh, “Deontic Logic Relativised to Bearers and Counterparties”, in *Anniversary Anthology in Computers and Law*, Ed. J. Bing. and O. Torrund, pp. 453 - 522, 1995.

[Johannesson95] P. Johannesson, “Representation and Communication - A Speech Act Based Approach to Information Systems Design”, *Information Systems*, vol. 20, no. 4, pp. 291 - 303, 1995.

[Johannesson98a]P. Johannesson and P. Wohed, “Deontic Specification Patterns - Generalisation and Classification”, *International Conference on Formal Ontologies in Information Systems*, ed. N. Guarino, Trento, Italy, 1998.

[Johannesson98b] P. Johannesson and P. Wohed, “Modelling Agent Communication in a First order Logic”, (to appear in) *Accounting, Management and Information Technologies*, 1998.

[Maiden91] N. A. Maiden and A. G. Sutcliffe, “Analogical Matching for Specification Reuse”, in *6th Annual Knowledge-Based Software Engineering Conference*, pp. 108-116, Syracuse, New York, IEEE Computer Society Press, 1991.

[Maiden92] N. A. Maiden and A. G. Sutcliffe, “Exploiting Reusable Specifications through Analogy”, *Communications of the ACM*, vol. 35, no. 4, pp. 55-64, 1992.

[Neches91] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator and W. Swartout, “Enabling Technology for Knowledge Sharing”, *AI Magazine*, vol. 12, no. 3, 1991.

[Prieto-Diaz87] R. Prieto-Diaz and P. Freeman, “Classifying Software for Reusability”, *IEEE Software*, vol. no. 1, pp. 6 - 16, 1987.

[Wieringa93] R. Wieringa and J. Meyer, “Applications of Deontic Logic in Computer Science: A Concise Overview”, in *Deontic Logic in Computer Science*, eds. R. Wieringa and J. Meyer, Wiley, 1993.