Diplomarbeit

Design and Implementation of an Ontology for Knowledge Assessment

Stefan Ukena
stefan.ukena@gmx.de, Matrikelnr. 5021131

April 4, 2005

University of Hamburg,
Department of Informatics,
Software Engineering Group

Prof. Dr. Christiane Floyd
(primary supervisor)

Dr. Carola Eschenbach
(secondary supervisor)

Acknowledgments

I would like to thank Prof. Dr. Christiane Floyd, Dr. Carola Eschenbach and Dr. Rüdiger Klein for their advice, time and their support. I would also like to thank Martin Dotter for the time he spent explaining aircrafts and aircraft design to me. Finally, I would like to thank Prof. Dr. Kristel Kumbruck for establishing the contact to the KMOD team which made it possible for me to participate in the project.

Table of Contents

0 Introduction

0.1 The subject of this thesis[1]

This thesis is based on my work for the research and development project of Airbus Bremen, Germany, and DaimlerChrysler Research & Development, Berlin, Germany. I focus on two aspects of that project: the ontology we created and the way we proceeded to create it. This includes an introduction to the concept of ontology in informatics[2], a description of the specific ontology and the process of its creation, as well as an evaluation of the project's results. I will suggest that the process of ontology creation can be improved by learning from application-oriented software development.

Prof. Dr. Christiane Floyd and Dr. Carola Eschenbach were my primary and secondary advisers at the University of Hamburg, respectively. Dr. Rüdiger Klein was my supervisor at DaimlcerChrysler Research & Development Berlin.

The KMOD project

The KMOD project serves as the case study of this thesis. The acronym *KMOD* stands for *Knowledge Management Overall Diagnosis*. KMOD was a research project of EADS Airbus in cooperation with DaimlerChrysler Research & Development. KMOD's goal was the creation of an ontology-based information system for knowledge assessment the so-called *KMOD system*. The underlying ontology was to be used to evaluate the knowledge of Airbus' departments.

The project started within a single section of Airbus but is expected to be successively propagated through all other sections. The department to participate in the KMOD project was EGA[3]. It has members in all four Airbus national companies[4], and is closely involved in the process of developing new aircrafts, a central activity of any aircraft building company. Activities include designing airplanes and testing the design. The project lead was in the hands of Martin Dotter from Airbus, an expert in IT-based knowledge management related topics within the Airbus company.

DaimlerChrysler Research & Development was a contractor in the KMOD project with Dr. Rüdiger Klein as a member of the KMOD project team. Dr. Rüdiger Klein is also the company's supervisor of this thesis. He is an expert in the field of IT-based knowledge management for engineering as well as the knowledge-management-related technologies that were used in the project.

---

1 The official German term for this kind of thesis is *Diplomarbeit*.

2 I will use the term *informatics* rather then *computer science*, *science of computing*, etc. because it most closely resembles the German term *Informatik*.

3 The name EGA is not an acronym and does not have any meaning.

4 Airbus has plants in the following four countries: UK, France, Spain, and Germany.

My role in the project

I joined the project team as a student for DaimlerChrysler Research & Development during the second half of 2004. When I joined the project it had already been running for some time. My job was to create an ontology in cooperation with my company supervisor, based on work that had previously been done in the project. The creation includes the identification of relevant concepts and relations from interview documents, the design of an ontology, and its implementation with Protégé-2000 and Flora-2. This ontology was called the *KMOD ontology*.

The KMOD ontology

The KMOD ontology, together with an F-Logic-based query-engine, is the central part of the KMOD system. The KMOD system is a tool for middle and upper management. It is expected to answer a question like "*Which critical knowledge areas are effected by the retirement of experts within the next five years?*".[5]

An important design criterion for the KMOD ontology was the separation of the *knowledge* about the *domain* from the knowledge about *knowledge assessment.* Therefore the KMOD ontology can be viewed as consisting of two parts:

- a domain-independent part (called the ***knowledge assessment ontology***)
- a second part for a specific domain

The idea was to create a completely domain-independent, reusable ontology for knowledge-assessment. This *knowledge assessment ontology* should be modeled in such a way that it can easily be reused with supposedly any domain by creating or reusing an existing domain ontology. The twofold design is meant to ensure that the KMOD ontology can be used in other sections of Airbus by adjusting or replacing the ontology for one domain by an ontology for another domain.

0.2 The structure of this thesis

This thesis is divided into four main chapters. Chapter 1: *Ontology and Ontologies* is an introduction to the concept of ontology in both philosophy and informatics. The term *ontology* and related concepts are explained as far as this is necessary for the understanding of the rest of this thesis. Emphasis is put on introducing ontologies as artifacts (in the context of informatics).

Chapter 2: *The KMOD project* details the project, the process of creating the KMOD ontology, and the tools which were used.

The project result is described in the chapter 3: *The KMOD ontology*. This chapter includes an informal description of the KMOD ontology as well as a detailed list of the most important classes.

---

5 A second objective was to create a guide for new employees. It was supposed to be a starting point to get an overview of EGA's structure, its processes, and the knowledge involved. This objective was abandoned during the course of the project.

Chapter 4: *Evaluation* is an evaluation of the KMOD ontology, the tools, and the process that was used to create the ontology. The chapter concludes with a number of suggested improvements.

The final chapter, chapter 5: *Summary and outlook*, gives a short summary of the results of the thesis, its conclusion, and an outlook on possible future work.

Part 1

Ontology and ontologies

Part 1 introduces the three meanings of the term *ontology*: a philosophical discipline, an artifact in philosophy and an artifact in informatics.

After a brief introduction to the term's (philosophical) history the chapter concentrates on ontology artifacts in informatics, including the notion of *formal* and *situated* ontologies which is used throughout this thesis.

1 Part 1: Ontology and ontologies

The creation of the KMOD ontology was the central aspect of the KMOD project. Generally, in informatics ontologies are used for the formal specification of semantics, expecting to enable computers to process documents in a way that is more meaningful to the user than it is today. The introduction of ontologies is expected to move an application's "understanding" of documents from the syntax- to the semantics-level. Semantics is here understood solely in the sense of *formal* semantics, or interpretation, of a logical theory: formal semantics relates the syntactically defined symbols of the theory to the theory's model.

The first part of this chapter takes a look at the philosophical origin of the term *ontology* as far as this is necessary to understand its recent use in informatics. The second part of the chapter investigates how the term is used in informatics. I will contrast the notion of *formal ontologies* that is used in the context of communication of software agents, with the notion of *situated ontologies* which is applicable in the context of knowledge sharing between humans. I also suggest a preliminary definition of the notion of a *situated ontology* which is used throughout this thesis. The chapter ends with a look at two aspects of ontologies in informatics that are relevant for the description and evaluation of the KMOD project, respectively: different types of ontologies and ontologies as *boundary objects.*

1.1 The origin of the term *ontology*

1.1.1 Ontology as a philosophical discipline

The word *ontology* stems from the Greek words ον (pronounced *on*), which means "being", and ιόγος (*logos*, both ο's pronounced short, like in *log* and *boss*), which means "study" or "discipline". In philosophy, *Ontology*[6] designates the *study of being as such.*[7] If Ontology is the study of being as such, then the *subject* of Ontology is being itself. But what does that mean?

A good way to grasp the concept of Ontology is by taking a look at its history.

According to a dictionary of philosophical terms ([Prechtl and Burkard 1996]) the first appearance of the word Ontology can be traced back to the 16th century German scholar Rudolf Goclenius, also known as Rudolf Gockel. He was the first to

6 *Ontology* with a capital "O" will designate the philosophical discipline. This spelling is used to distinguish the different meanings of the word *ontology,* and was introduced by Guarino (1998).

7 Instead of *study of being as such* it is also sometimes called *study of being in general, study of being in the abstract* or *study of the nature of being.*

distinguish Ontology as a discipline of its own. What Goclenius called Ontology had before been regarded as only an aspect of another discipline: metaphysics.[8]

The philosophical discipline of metaphysics goes back to Aristotle, though he did not use that name. He spelled out his program of metaphysics in a series of fourteen books. The series as a whole had no title at the time but later came to be known as *ta meta physika* or simply Metaphysics. [Prechtl and Burkard 1996]

In these fourteen books Aristotle describes a discipline that in his mind was to be considered the first and highest among all philosophical disciplines. That is why he referred to it by the name of "first philosophy". [Prechtl and Burkard 1996]

This *first philosophy* is concerned with what all the other specialized sciences do simply take for granted. It is concerned a) with the preconditions of being and b) with being itself. While Aristotle himself perceived his *first philosophy* as a whole it was later divided into two separate disciplines: a) theology, which is concerned with the preconditions of being and b) Ontology which is concerned with being itself. [Prechtl and Burkard 1996]

The qualifier "as such" indicates what sets Ontology apart from all other disciplines: whereas scientists like physicists are concerned with things that exist, an Ontologist is concerned with existence itself, i. e. being itself. A physicist is concerned with how two objects interact, which forces act upon them etc. The physicist does not question the nature of the existence of objects or forces she observes. This is what the Ontologist does. She might ask herself how the existence of the objects differs from the existence of the forces that act upon them. Do the forces exist in a way that is independent of the objects? Or are the forces merely a property of the two objects?

If the former is true, then forces would be regarded as *first-class objects*: they exist on their own, their existence does not depend on other objects. If the latter is true, the force would only exist as properties of the two objects. This would give rise to even more questions: Is the force a property of one of the objects? Or maybe a property of both of them?

These questions are all concerned with the *ontological status* of something, in this case the ontological status of the forces that act upon two objects.[9]

1.1.2 Ontologies as conceptual artifacts

The term *ontology* is not only used to refer to the philosophical discipline of Ontology but also to refer to the artifact that is the subject of Ontology:

---

8 It should be noted that the meaning of the term *metaphysics* has undergone many changes. The meaning described here is that of Aristotle's *first philosophy.*

9 It should be noted that the observer plays no role in any of these ontological questions. Ontology is not at all concerned with how we come to know whether the objects, forces etc. exist or not. These kind of questions ("*Do we know* if the two objects exist?") are the subject of epistemology.

> **Ontology** Either the part of metaphysics concerned with the nature of existence, or [...] the entities (things, processes, properties) postulated by a particular scientific theory or conceptual scheme.
>
> *[Curd & Cover, p. 1303]*

Different Ontologists have different theories with regard to ontological questions. They have different theories about the nature of being, or, to put it yet another way, *they advocate different ontologies.* An ontology in this sense is a systematic account of existence.

In this context an ontology is an artifact: a theory of existence. Because different philosophers will have different theories it makes sense to speak of *ontologies*, in plural, as well.

Thus, two different meanings of "ontology" can be distinguished:

- ***Ontology:*** The name of a philosophical discipline.
- ***ontology, ontologies:*** A theory (or theories) about the nature of being; a systematic account of existence. The goal of Ontology.

Speaking of the existence of ontologies in the plural can be misleading with regard to what the original goal of Ontology was. Ontology does not look for several accounts of existence which *may* be true, but it looks instead for the one true ontology that is the only complete account of existence. Therefore the different ontologies must be regarded as *competing* ontologies for this status: the status of being the one true ontology.

1.1.3 From Ontology to ontologies in informatics

Today, the main areas of informatics which are concerned with ontologies are artificial intelligence and knowledge representation on the one hand, and information system design and system development on the other.

The interest in philosophical Ontology started in the field of artificial intelligence and knowledge representation, namely among knowledge engineers [Guarino 1995]. Until the early 1990's, knowledge engineers were mainly concerned with modeling how people think. A shift in perspective turned the focus to modeling "systems in the world" [Clancey 1993, 34].

This shift in perspective emphasizes the importance of modeling the *environment* of a system, i. e. the *problem domain.* The problem domain is assumed to be part of an objective reality. The underlying assumption seems to be that there is only one such objective reality which is universally accepted and which can be represented. This is where philosophical Ontology is expected to be of help. Philosophical Ontology is

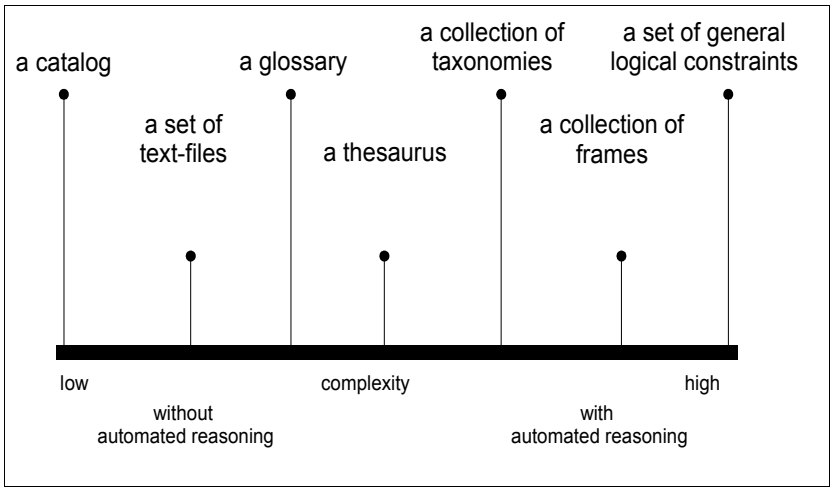

Figure 1
What is an ontology? (Based on [Smith and Welty 2001, foreword, page v])

concerned with an account of reality. Therefore, knowledge engineers turned to Ontology for insights about the modeling of reality. [Guarino 1995]

With the increasing importance of knowledge related technologies, the term *ontology*[10] is now commonly used in other areas of informatics as well, namely in information system design and in system development. Though the term *ontology* remains the same, it is often used with different meanings.

Even within one community different meanings of the term *ontology* can be observed. At the national conference of the *American Association for Artificial Intelligence* Welty, Lehmann, Gruninger, and Uschold identified among themselves a number of different usages of the term *ontology*. Their note for the slide titled "What is an Ontology?" (cf. figure 1) reads:

> "Answers to the question, 'What is an Ontology' vary. Rather than even try to achieve consensus among ourselves, we identified a spectrum of possible definitions along the axis of axiomatization."
> *([Welty et al. 1999], notes for slide three "What is an Ontology")*

As figure 1 shows, the term "ontology" is used to refer to a wide range of entities. An ontology can be anything from a catalog of words to a logical theory expressed as a set of general logical constraints. Depending on the complexity of the representation, this may encompass the possibility of automated reasoning.

Others, like Guarino (1998), presuppose an ontology to always be rigorously defined in terms of a logical theory.

10 For the rest of this thesis the term *ontology* will be used to refer to its meaning in informatics, not philosophy. The latter will henceforth be referred to as *a philosophical ontology* or *an ontology in philosophy*.

The common feature of all these different meanings of "ontology" lies in the subject of representation: All ontologies in one way or the other represent concepts and their relations.

Turning from formal to functional aspects, another distinction of the term's usage may be observed: This distinction has to do with an ontology's function as a means of communication between different agents. The term *agent* is used by Gruber (1993) and Guarino (1998) to refer to computer agents .

Others explicitly refer to both humans and computer agents and assume that in the context of ontologies, humans and computer agents can be treated as being equal ([Maedche and Staab 2001] and [Noy and McGuinness 2001]).

1.2 Ontologies and conceptualization

In his seminal paper, Gruber defines an ontology as "an explicit specification of a conceptualization." [Gruber 1993, 908]

He describes a conceptualization as "the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them."[11] [Gruber 1993, 908] If he would leave it at that, a conceptualization would literally consist of objects, concepts, etc. and their relations. Instead he continues: "A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose." [Gruber 1993, 908] Thus, a conceptualization is not identical with the objects and relations themselves but is instead an abstract and simplified view of these objects, concepts, etc. and relations among them. It resides therefore on a more abstract level than a philosophical ontology which is concerned with the objects themselves.

Another common definition of ontologies in informatics comes from Guarino (see, for example [Guarino 1998]). For Guarino the term *conceptualization* refers to the philosophical meaning of ontology. It is merely a different name to avoid confusion [Guarino 1998].

Zúñiga gives yet another explanation in [Zúñiga 2001]. She argues, that computer scientists may *think* that the term *conceptualization* refers to the philosophical meaning of ontology, but that it in fact does not.

Obviously the term *conceptualization* is used very differently by different authors, just like the term *ontology*. Let us for the moment assume Gruber's position and return to the original question of defining an ontology in informatics.

Gruber's definition of an ontology in informatics together with his notion of conceptualization results in the following definition:

11 He attributes this notion of conceptualization to [Genesereth and Nilson 1987].

> *(1) An ontology (in informatics) is an explicit specification of one or more person's abstract and simplified view of the objects, concepts, and other entities and their relations in a domain.*

Gruber does not define the term *specification* but he states: "Formally, an ontology is the statement of a logical theory." [Gruber 1993, 909] Thus, a specification in Gruber's sense clearly means a *formal* specification.[12] This leaves us with the following definition:

> *(2) An ontology (in informatics) is a formal specification in the form of a logical theory of one or more person's abstract and simplified view of the world or of part of the world.*

Guarino wants to "refine" Gruber's definition when he defines an ontology as "a logical theory accounting for the intended meaning of a formal vocabulary, i. e. its ontological commitment to a particular conceptualization of the world." [Guarino 1998] He also emphasizes that a philosophical ontology (what he calls a conceptualization) is language-independent, while an ontology in informatics is language-dependent.

Guarino, Gruber, and Zúñiga seem to share the assumption that there is such thing as an objective reality which can be represented by some means of representation in an objective way. This notion of ontology is suitable in the context of automated reasoning.

I will call this the *formal* notion of ontology or simply refer to it as the concept of *formal ontologies*, to avoid confusion with a different notion of ontology that is suitable for the context of knowledge sharing between humans, which I will introduce in the next section.

1.3 Formal vs. situated ontologies

The notion of *formal ontologies* in the sense just described, includes a set of common assumptions which have already been mentioned in the previous sections:

- There is such a thing as an objective reality which is universal for everyone.
- This reality is objectively represented by philosophical ontologies which are the basis for a conceptualization and ontology in informatics.
- Human agents are equivalent to computer agents in the context of communication with the help of ontologies.

Another assumption that arises directly from the "objective reality"-assumption is implicit in the notion of reusing ontologies, which is a common goal in the field of ontology (in informatics):

12 Though he does allow room for "human-readable text" [Gruber 1993, 909] with informal descriptions of the ontology's concepts.

- There is such a thing as an "eternal domain" which, once truthfully represented, can be reused indefinitely.

Together, these assumptions paint a picture of formal ontologies which are based on a philosophical ontology to capture reality and enable the de-contextualization of knowledge. In the light of Zúñiga's characterization of philosophical Ontology as something that "is not concerned with how people know things in a particular sphere, nor about how they experience these things, or what language they use to refer to them" [Zúñiga 2001, 195], these assumptions should be challenged. If a philosophical ontology does not allow for any kind of view that a user might have, then an ontology in informatics cannot be a specification of such an ontology.[13]

In the context of knowledge sharing between humans, these assumptions may be replaced with a different set of assumptions resulting in a completely different picture. First, one may assume that there is not just a single objective reality but a multitude of realities. These realities cannot be represented by an *objective* representation. Instead, a group of people may be able to agree upon *some* representation that reflects their particular view. To emphasize the non-objective, consensual nature of this representation, I will call a *conceptual view* as opposed to a conceptualization.

In this context the equivalence of humans and computer agents, that is often implied or explicitly stated (cf. [Oppermann, Schnurr and Studer 2001], [Guarino 1998]), does no longer make sense. A person has a conceptual view of the world but a computer agent does not. Instead, a person imposes her conceptual view onto the software agent. Thus, Mahesh and Nirenburg (1995) introduced the notion of a *situated ontology*:

> "A situated ontology is a world model used as a computational resource for solving a particular set of problems. […] World models (ontologies) in computational applications are artificially constructed entities."
>
> *[Mahesh and Nirenburg 1995, 1]*

This leads to a revised set of assumptions which appears to be more suitable in a social context:

- There is not one reality but a multitude of realities.
- A group of people can agree on a shared *conceptual view* that reflects a shared view of the world.
- Humans impose their conceptual view onto computer agents.

One may reject the idea of multitude of realities. This does not have to be taken literally. With respect to natural language processing, Hobbs (1985) states: "There's too much of a mismatch between the way we view the world and the way the world

13 The same is true for software agents, because they, too, have a perspective that is not neutral in the sense a philosophical ontology is neutral.

really is." [Hobbs 1985, 68] He suggests to choose a representation based on how we talk about the world, not on how the world really is.

Based on these new assumptions the notion of a *situated ontology* may be defined:[14]

> (3) *A **situated ontology** is a description of the conceptual view of a person or a group of persons that the person or persons choose to represent for some purpose.*

This definition allows ontologies to come in different forms. It excludes neither an informal description in the form of a text nor a rigorous specification using first order logic. While for the purpose of a computer system a formal specification will be desirable to support computations like automated reasoning, an informal description may be useful as well; especially during the design of the ontology and for communication with people that are not experienced in formal representation languages.

The definition also makes explicit that a situated ontology in informatics is not directly related to a philosophical ontology by emphasizing that it is the view of a person that is being specified. This does not mean that computer scientists cannot learn from Ontology when designing their ontologies. But they should be aware of the different epistemological status of ontologies in informatics and philosophy: the former specify a certain *view of the world*, the latter a true account of existence, independent of any observer.

1.4 Situated ontologies in communities of practice

Acknowledging the situatedness of ontologies as artifacts calls for a different view on the design and use of ontologies. *Communities of practice* form a suitable frame of reference when studying the use of computer artifacts. Seeing ontologies as *boundary objects* between *communities of practice* enables us to view them as situated and plastic artifacts that change over time and in the hands of different people.

The concept of *boundary objects* can best be explained with reference to the concept of *community of practice.* Communities of practice can be characterized as "shared histories of learning" [Wenger 1998, 103]. They cut across organizations, but their members all share a commitment for a specific domain of interest, share a sense of community, and actually engage in some form of shared practice. Certain groups, like claims processors at an insurance company or programmers at a software company, do not automatically constitute communities of practice. But if the programmers regularly meet to exchange ideas and learn from each other, then they have formed a community of practice. In this sense we are all members of a number of different communities of practice [Wenger 1998].

The concept of *boundary objects* was introduced by Star (1989). Boundary objects are "those objects that both inhabit several communities of practice *and* satisfy the

14 This definition is based on the definition of a system by Nygaard (1986).

informational requirements of each of them." [Bowker and Star 1999, 16] The interesting thing about boundary objects is their ability to "travel" across borders of different communities of practice: They can be accommodated to the special needs of each community of practice while maintaining an identity of their own [Bowker and Star 1999, 16].

Classifications, like a thesaurus, are a good example of boundary objects. A thesaurus is a controlled vocabulary that has a certain structure. Librarians commonly use a thesaurus to classify books. A student will use the same thesaurus for a different purpose: to locate a book. Here, the thesaurus is used both to classify books as well as to locate them. The thesaurus can be seen as a nexus were different practices meet.

Apparently, the student would not be able to locate the book if the librarian and the student do not share a certain degree of understanding of the thesaurus' purpose and the meaning of the thesaurus' terms. This shared understanding is not inherent to the thesaurus but arises from the practice of using it for a purpose.

Bowker and Star point out that boundary objects cannot easily be engineered.[15] Instead, boundary objects "grow" from a common practice. This may happen by using them like tools or as the basis for a certain practice. The same is true for situated ontologies. Their usefulness and acceptance for knowledge sharing depends on their ability to be used as tools or serve as the basis for practice of the people involved. A simple engineering approach to create an ontology is not enough to ground it in a community of practice. This is acknowledged by Uschold and Jasper (2003) in their report on the Boeing knowledge management project:[16]

> "[...] we were faced with the realization that people will resist imposition of a global vocabulary, and therefore ways must be developed to reap the advantages of a standard vocabulary while allowing individuals to continue to use their own terms locally."
>
> *[Uschold and Jasper 2003, 235]*

Four characteristics are expected to support the formation of boundary objects (see, for example [Bowker and Star 2000]; the examples for each property are taken from [Wenger 1998]):

1. *Modularity*, in the sense that a newspaper contains a diverse collection of articles: every reader may attend to articles which are of interest from their specific point of view while ignoring others.
2. *Abstraction*, in the sense that a topographic map reflects only certain properties of the terrain, while abstracting from other properties (like the kind of vegetation)

15 As example of attempts to engineer boundary objects in the context of informatics, Bowker and Star cite the design of information systems that allow for access by people with very different point of views [Bowker and Star 1999, 305].

16 This project will be introduced in the next chapter.

3. *Accommodation*, in the sense a building can accommodate to the specific needs of its tenants, caretakers, owners etc.
4. *Standardization*, in the sense a library thesaurus states how the terms ought to be used for classification.

Obviously a situated ontology is a candidate for becoming a boundary object because it has the potential to exhibit all of these four characteristics. A situated ontology will not necessarily have all four characteristics. Nor will a situated ontology automatically become a boundary object. But the odds can be increased by designing situated ontologies based on classifications which are already shared by different communities of practice.

This section offered an alternate view on ontologies. Viewing ontologies as boundary objects enables one to focus on communication and learning, drawing the attention to the role ontologies may play in a social context. This is not to say that the formal aspects are not important. But the role of ontology artifacts in a social context is equally important.

In the chapter "Evaluation" I will return to the subject of ontologies as boundary objects, suggesting that techniques from software design might help in the designing of ontologies as potential boundary objects.

1.5 Computer-implemented ontologies

While ontologies do not necessarily have to be computer-implemented, this becomes a central concern in the context of automated knowledge sharing and reuse between software agents. This is also a central concern in the literature about ontology artifacts in informatics. For the KMOD ontology, the reuse of existing ontologies was considered as well.[17]

To facilitate reuse of existing ontologies, Guarino (1998) suggests a classification of four types of ontologies based on their content. This classification divides types of ontologies into three levels of generality. These four types of ontologies are distributed among the three levels of specialization as follows:

1. level: **Top-level** ontologies are the most general ontologies. They should define very basic concepts like time and space.
2. level: **Domain** ontologies and **task** ontologies are more specialized than top-level ontologies but are more general than application ontologies. They should define general concepts related to a generic domain (like airplanes) or task (like constructing) respectively.
3. level: **Application** ontologies are the most specific ontologies. They usually specialize upon concepts from both domain and task ontologies.

17 This will be discussed in section 2.2 "Related work" below.

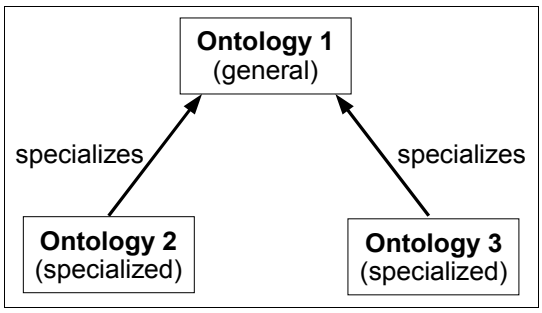

Figure 2
Example: Two special ontologies (2 and 3) specialize upon the same general ontology (1). The arrows represent the specialization-relation.

Reuse may be achieved in the following way (cf. figure 2): The ontologies "Ontology 2" and "Ontology 3" specialize upon the same (more general) "Ontology 1". The expectation is, that a specialized concept from one of the specialized ontologies can be more easily translated into concepts of the other specialized ontology, because they both use the common "vocabulary" of the general ontology "Ontology 1".

Guarino (1998) also suggests another kind of distinction related to the dynamic aspects of computer-implemented ontologies. This distinction is based on their usage in the process of software development and use: the time that the ontology is used (the temporal dimension), and the function it is used for (the functional dimension).[18]

In the temporal dimension, one can distinguish between usage of ontologies at development time of the software vs. usage at run time of the software. In the functional dimension, the distinction can be made between usage as the basis for the software's user interface, the application component, or the database component.

An ontology may be categorized using these two dimensions into any of the six combinations of temporal and functional usage, or a combination of the six. Figure 3 (on the next page) shows a matrix of these two dimensions. The matrix is divided into six regions, one for each of the possible combinations.

An example: Protégé builds a tree of classes from the class-hierarchy of an ontology and creates a user interface to enter slot values, etc. Therefore, Protégé is an instance of a tool that uses an ontology at run time to create a user interface. This is indicated in figure 3 by the ellipse labeled "A Protégé-ontology".

The three distinctions presented here—content, time of use, and function—will be used in the next chapter to categorize the KMOD ontology.

This distinction serves well for the classification of existing ontologies. However, for the *design* of computer-implemented ontologies, design criteria are needed. Gruber

18 The latter is referred to as the *structural dimension* in [Guarino 1998].

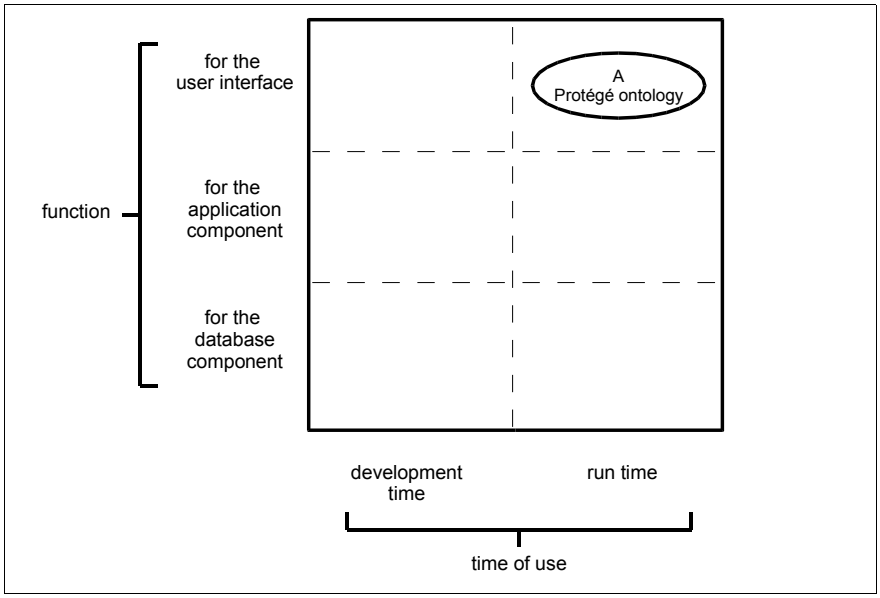

Figure 3
Distinguishing ontologies by dynamic aspects: time of use and function.

(1993) suggests the following criteria for the design of formal ontologies for knowledge sharing and reuse between computer agents:

- **Clarity**: Terms should be rigorously defined using logical axioms whenever possible, and should include an informal description.
- **Coherence**: Inference should not lead to contradictions. Formal definitions and informal description should not contradict each other.
- **Extendibility**: Future use should be anticipated and the introduction of new terms should not necessitate changes of existing definitions.
- **Minimal encoding bias**: The implementation should not depend on a particular symbol-level encoding. Results if design decisions are made because of convenience of notation or implementation.
- **Minimal ontological commitment**: For a particular knowledge sharing purpose, only the weakest assumptions possible about the domain should be modeled.

For the design of situated ontologies, these criteria may not be suitable. Uschold and Jasper (2003) observed that workers at Boeing resisted the imposition of a controlled vocabulary. Neither of the five design criteria above addresses this kind of problem. Instead, the four enabling characteristics of boundary objects, introduced in section 1.4 "Situated ontologies in communities of practice", could serve as general guidelines for the development of situated ontologies. For the evaluation of the KMOD ontology, these characteristics will be used.

1.6 The notion of ontologies in this thesis

The KMOD ontology is intended for knowledge sharing between humans, enabling people from different communities of practice to communicate about knowledge at Airbus. Currently, the ontology represents knowledge about different aspects of the company: The types of aircrafts which Airbus produces are represented, as well as the processes involved in designing such aircrafts, and the people who actually design them. This information can be used by managers from different departments to assess critical knowledge areas.

To identify the relevant concepts a number of practitioners were interviewed. These interviews served as the foundation for the design of the ontology. Though the KMOD ontology is implemented in such a way as to support limited automated reasoning, it is not an ontology for software agents. Thus, the KMOD ontology will be presented in this thesis primarily as a *situated ontology*.

Part 2

The KMOD project

Part 2 takes a look at the KMOD project. A brief introduction of related work is followed by a description of the tools which were used for the KMOD project, including the Protégé meta-model. The chapter concludes with an explanation of the process that was used for creating the KMOD ontology.

2 Part 2: The KMOD project

2.1 Project overview

The project's goal was to create an account of Airbus' knowledge including an assessment of that knowledge. Here, assessment means to assess knowledge using different criteria, like knowledge which is critical and knowledge which is common —always with respect to Airbus' business. This "map of knowledge" was to be made available to users via a web portal. The web portal was to be built using an ontology, the so called KMOD ontology. The portal should not only be used to browse the ontology but also to answer queries with respect to the assessment of the knowledge. This kind of company-wide knowledge assessment had never before been done at Airbus. In the beginning it was expected to be useful for two kinds of users:

- New employees who want to get an overview of what their own department knows and how it is related to the knowledge of others.
- Managers who are interested in a complete account and an assessment of Airbus' knowledge.

The KMOD team had hoped to be able to consider both views, but during the course of the project the focus turned more and more to the latter. The reason was that it was too difficult to keep track of both views.

While the final goal was an account and the assessment of all of Airbus' knowledge, the project was conducted in one department of Airbus, EGA. The intention was to successfully develop a web portal for EGA and then modify it for the needs of other departments. If all portals use the same underlying KMOD ontology it should be easy to join them into a single portal for Airbus as a whole.

When I joined the project, work had already commenced for over a year. A detailed document (including a mind map) about EGA's knowledge with results from interviews had been compiled. Mainly from this document with the interview results the KMOD ontology had to be designed as a basis for the web portal.

2.2 Related work

The KMOD project was not the first to aim at building an ontology in the context of knowledge management for an enterprise. Related efforts are a recent knowledge-management project at Boeing [Uschold and Jasper 2003], and, larger and more general in scale, the TOVE-project [Fox1992], and the Enterprise Ontology [Uschold1998].

On the subject of knowledge assessment, the KMOD team was not able to locate any publicly available information.

The case of Boeing

The Boeing project aimed to support Boeing's service representatives using a combination of document retrieval yellow pages and system. A service representative is a person who is contacted by Boeing customers to assist in solving specific problems which the customer is not able to solve alone. The goal was to create a system that the service representative can easily query to locate relevant Boeing experts and documents which may have or contain necessary information for the task at hand.

The service representative does this by selecting a *request type* (like "Part Substitution"), the wanted *resource* (this includes both names of experts and documents), and the relevant *concepts* (like "Flight Control") from a predefined list. The system would interpret the submitted form as a query meaning e. g. "Show me the names of experts which have expertise in *Part Substitution* with regard to *Flight Control*." This query is run against a metadata repository—the database—that contains the relevant information about the documents and the experts. The result of the query is then displayed to the user.

The system is build around a number of technologies, the most important being RDF[19], F-Logic, and the *Boeing thesaurus*. RDF and F-Logic rules are used in a metadata repository that contains information about the experts and documents. The *Boeing thesaurus* serves as a "lightweight ontology" [Uschold and Jasper 2003, 235] for this metadata repository, with approximately 37,000 concepts and 100,000 relations among them. The thesaurus has originally been developed and maintained by Boeing for the purpose of company-wide document classification and retrieval. Uschold and Jasper emphasize the successful exploitation of the thesaurus in a context it was not originally designed for: the search for experts.

The project at Boeing and the KMOD project share the general context of knowledge management with semantically enriched technologies for the aircraft industry. But while Boeing wants to support a very specific task at hand, KMOD aims at creating a new system for a new task. Boeing wants to support a task that was previously conducted without a specialized system on a regular basis, while KMOD aims at enabling the task of company-wide assessment of knowledge.

A controlled vocabulary like the Boeing thesaurus would have been helpful for the development of the KMOD ontology. But because Airbus does not currently have a controlled vocabulary the KMOD ontology had to be developed by other means, namely, by conducting interviews with domain experts.

The Boeing portal assists users in querying the system as described above, by letting them select words from predefined lists. This allows for very easy creation of queries, but it offers limited flexibility, because all queries have essentially the same structure. For the Boeing system this is not a drawback, because it wants to support exactly this one kind of query.

---

19 *Resource Description Framework*, an XML-dialect that can be used for semantic annotation of resources. Cf. http://www.w3.org/RDF/

The KMOD portal will provide assistance as well, but a more general approach seemed necessary. For this purpose a simple template language has been developed that supports the creation of query templates to support different kinds of queries. The user is presented with an informal description of the query, and is prompted to fill in the template values, e. g. "Who knows something about the knowledge area X?" Here, the user will be prompted with a list of all knowledge areas. She can choose one or more knowledge area from the list and submit the query. Other, more complex queries are possible as well.

TOVE and the Enterprise Ontology

Both TOVE[20] [Fox and Fadel 1993] and the Enterprise Ontology (hereafter EO) [Uschold et al. 1998] aim at modeling a complete enterprise. They do this with a number of ontologies which together form a complex framework.

The KMOD team expected to be able to partially reuse concepts from either of these enterprise ontologies. This proved to very difficult owing to the size and complexity of both frameworks. A brief analysis showed

- that TOVE and EO contain many concepts that would not be needed for the KMOD ontology, and
- that it would be very difficult to only reuse the needed concepts because they depend on other concepts.

Thus, the idea of reusing either TOVE or EO was abandoned.

2.3 Tools for the KMOD project

This section describes the tools that were used to create the KMOD ontology. The description of how the tools were actually put into use can be found in the section "Designing the ontology" at the end of this chapter.

2.3.1 Overview of the Tools

The central tools are Protégé-2000, Flora-2, which were used for the purpose of knowledge representation and inference, and a combination of Tomcat/JSP, which was used to create an integrated user interface for the final application:

- **Protégé-2000**[21] as an ontology editor,
- **Flora-2** as an F-Logic-based query engine,

20 TOVE is the acronym for "Toronto Enterprise Project".

21 We actually used a derived version called OntoWorks that was developed by DaimlerChrysler based on Protégé-2000 version 1.8. Because we did not use any of the special abilities of OntoWorks I will continue to use the name Protégé-2000 instead.

- **Java Server Pages** and **Tomcat** (hereafter Tomcat/JSP) to implement a web application.

The main reason for choosing these tools was the experience that DaimlerChrysler Research & Development Berlin had gained in previous projects. The combination of tools had already been successfully used by them to implement other ontology-based applications, including web applications.[22]

Protégé-2000

Protégé-2000 is an open-source, frame-based ontology editor developed at Stanford University's medical informatics department.[23] It is written in Java and supports plug-ins. Protégé can be used to create a hierarchy of classes and instances of those classes.

The user interface is divided into different tabs which offer different views on the current model[24]: the *class-browser tab* to create and view properties of classes, the *instance tab* to create and view instances, etc. New tabs can be added via the plug-in mechanism. We used a plug-in called *OntoQuery*[25] to connect Protégé-2000 to Flora-2.

Protégé also offers a Java-API that can be used from any Java-program to access a Protégé model without the Protégé user interface.

The KMOD team uses Protégé together with the OntoQuery plug-in to create and maintain the ontology and the knowledge base, including the Flora-2 rules and queries. The Protégé-API is used in a web portal to access the KMOD ontology. Note that the KMOD ontology can currently only be manipulated using Protégé or the Protégé-API, but not via the web portal.

Protégé supports the creation of a class-hierarchy and instances, including the propagation of properties via multiple inheritance. (See the next section "The Protégé meta-model" for more information.)

Flora-2 and the OntoQuery plug-in

Flora-2 is an implementation of the F-Logic language which supports complex objects, inheritance and deduction.[26] Being also a frame-based system, Flora-2 lends itself easily to extend the expressive power of a Protégé model. Its meta-model is

---

22 The previous projects were concerned with solving time constraints. These ontologies could not be reused for KMOD.

23 The Protégé homepage is at http://protege.stanford.edu

24 In Protégé parlance the class hierarchy by itself is called the ontology, while the classes together with instances is called a knowledge base. To avoid confusion with the concept of ontology introduced in chapter 2, I will use the generic term *model* instead, referring to both a Protégé ontology and knowledge base.

25 OntoQuery was developed by DaimlerChrysler Research & Development Berlin based on the Flora-Tab plug-in by Micheal Kifer.

26 In fact, Flora-2 also integrate features of Transactional Logic which we did not use. Se the Flora-2 homepage at http://flora.sourceforge.net for more information.

more general than Protégé's, e. g. in Flora-2 a frame does not have to be an instance of another frame. It also provides an inference mechanism that goes beyond simple inheritance: using rules (axioms) one can intensionally define properties of frames. These rules, together with the ability to create complex queries, were the most important reasons for extending Protégé with Flora-2.

This is achieved with OntoQuery, a plug-in that integrates Flora-2 into Protégé, enabling the user to query a Flora-2 database from within Protégé. A query is executed by OntoQuery via Flora-2 in a two step process:

1. First, the Protégé model is converted to Flora-2 syntax which is then passed to Flora-2 for execution. The query itself and any Flora-2 rules that are stored in special parts of the Protégé model are also passed directly to Flora-2.
2. The result is returned by Flora-2 and is parsed by OntoQuery for further use. (Usually for display in the OntoQuery tab.)

Note that this is a one-way process: the Protégé model is translated to Flora-2, but there is no translation back into a Protégé model. The Protégé model does not change.

Also note that the first step does not only translate the Protégé model to Flora-2, but that it also allows to include rules, which will become part of the final Flora-2 model. This combination of rules and queries provides a powerful extension to any Protégé model. In KMOD this mechanism was used to allow the intensional definition of properties.

When not using the Protégé plug-in, one must use a text-editor to write the Flora-2 source code and then compile the source using the Flora-2 compiler. Alternatively, Flora-2 may be used in an interpreted mode which allows for the interactive editing of the program.

Tomcat / JSP

JSP is a well known server-side programming language based on the Java technology. Tomcat is a *JSP-container*, i. e. a web-server that can execute programs written in JSP. Using JSP and Tomcat made it easy to access the Protégé model trough Protégé's Java-API. Tomcat/JSP was used to create the user interface in the form of a web portal.

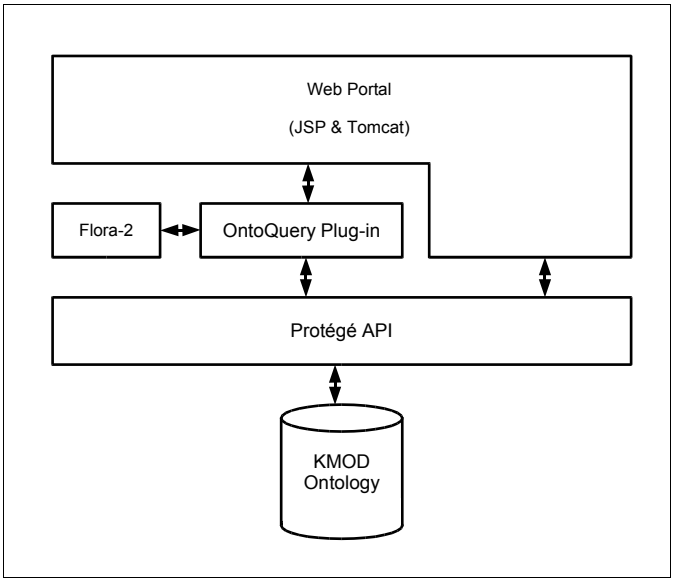

Figure 4
Architecture of the KMOD web portal. The arrows represent communication between different parts of the system.

The KMOD web portal

Together, Protégé-2000, Flora-2, and Tomcat / JSP, make up the KMOD web portal called OntoPortal. The architecture of the portal is in shown in figure 4.

The basis of the system is the KMOD ontology, which is currently stored as files in the native Protégé format. It is accessed by the web portal and the OntoQuery plug-in using the Protégé-API. The OntoQuery plug-in is used to add Flora-2's functionality and make it available to the web portal. This includes handling the storage and retrieval of Flora-2 rules within the KMOD ontology as well as converting query results back to Protégé objects. The web portal uses both OntoQuery and the Protégé-API to create the user interface for navigation and to query the KMOD ontology, which can then be accessed with a web browser by the user.

Other tools

Besides the aforementioned tools, a word processor was used to write informal documentation and a drawing program to create graph representations of the concepts in the KMOD ontology. This was necessary to communicate effectively with other team members about the KMOD ontology. Using Protégé-2000 for this purpose was not an option because the tree-like view of Protégé is not readily understood by people which are unfamiliar with the program.

Object-Frame (properties being determined)	**Schema-Frame** (determining properties)
instance	*class*
class	*metaclass*
slot	*metaslot*
metaclass	*metaclass*
metaslot	*metaclass*

Table 1
Schema-object relation of Protégé frames.

2.3.2 The Protégé-2000 meta-model

This section introduces some concepts of Protégé-2000. The description is not a complete account of the Protégé meta-model but concentrates instead on aspects which are relevant for the KMOD ontology. The information is based on the description of Protégé by Noy, Fergersen and Musen (2000).

All first-class objects of Protégé are referred to as *frames*[27]. The most common *frames* are *class*, *instance,* and *slot*. Two other kinds of *frames* are *metaclasses* and *metaslots,* which are special kinds of *classes* and *slots,* respectively.

An important relation in any Protégé model is the *instance-of* relation which is used to determine the properties of a *frame*. Two *frames* may be related to each other by the *instance-of* relation, which results in the one *frame*'s properties being determined by the other *frame*. To distinguish between these two frames it is helpful to name them according to the role they play in that relation: The *frame* that determines the properties is called the schema-frame, while the *frame* of which the properties are being determined is called the object-frame. Thus, the *instance-of* relation may be paraphrased as follows: In an *instance-of* relation, the schema-frame determines the properties of an object-frame.

Most types of *frames* may assume either of the two roles: A *class* determines the properties of an *instance*, while the properties of the *class* are determined by a *metaclass*. In the former relation, the *class* assumes the role of the schema-frame with respect to the *instance*, in the latter case, the *class* assumes the role of the object-frame.

Instance- and *slots-frames* are restricted to the object-frame role, i. e. they cannot determine properties of other *frames*. The *metaclass-frame* on the other hand serves as a schema-frame for *class*, *metaslot* and itself. The *metaclass* is therefore the only *frame* that can be used to define its own properties.

Table shows all possible combinations of the *instance-of* relation by listing which type of *frames* can be used to determine the properties of which other type of *frame.*

27 A note on notation: For the description of the Protégé meta-model I use *italics* to mark words that refer to elements of the Protégé model.

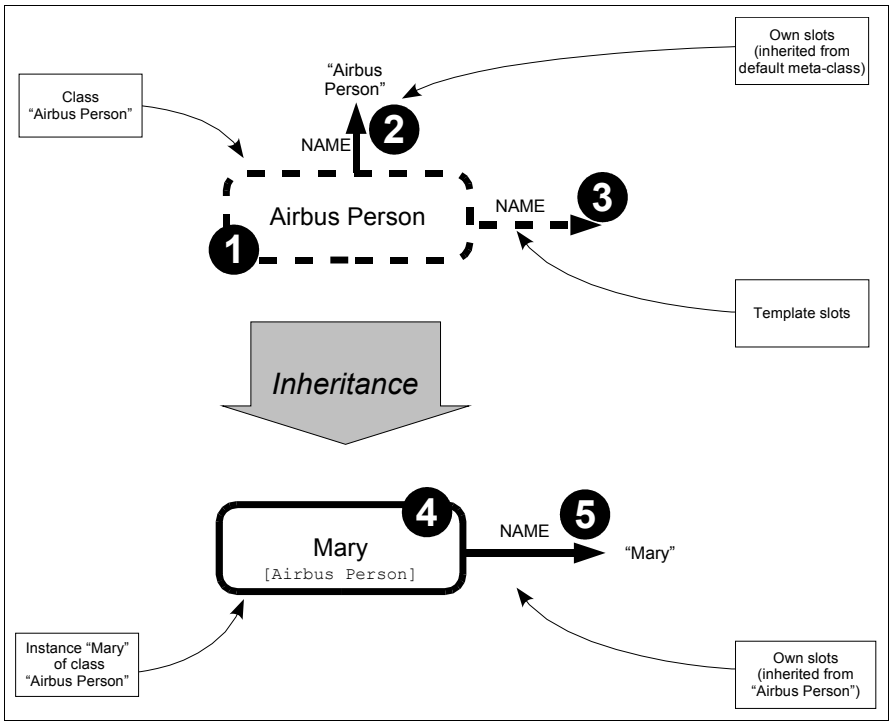

Figure 5
Template slots of a class become own slots of instances of that class. The dashed rounded squares represent classes, the solid rounded squares represent instances. Dashed arrows represent template slots, thick solid arrows represent own slots.

Classes are arranged in a familiar hierarchy of super- and sub-classes. Note that *classes*, *metaclasses* and *metaslots* are all part of this one class hierarchy. Except for the standard root class `THING`[28], a *class* must have one or more direct super-classes and may have one or more direct sub-classes. Every *instance* is assigned to exactly one *class*, making it an instance of that class and all its super-classes.

Slots are first-class objects in Protégé which means they can exist independently of other elements. A *slot* can be attached to a *frame* in two different ways: either as a so-called *own slot* or as a *template slot*. An *own slot* represents a property of the *frame* it is attached to, while a *template slot* determines the *slots* of instances of that *frame*. Figure 5 shows an example. E. g. the *slot* `NAME` is attached twice to the *class* `Airbus Person` (1). Once as an *own slot* (2), where it carries the name of the *class* ("Airbus Person"), and once as a *template slot* (3) where it determines that all *instances* of `Airbus Person` will also have a `NAME` *slot*. An *instance* inherits the *template slot* as an *own slot*. In figure 5, the *instance* `Mary` (4) inherits its slots from its *class* (and all the classes' superclasses), but for the *instances* the *slot* will be an *own slot*[29] (5), in this case containing the value *"Mary"*. Note that the *class* `Airbus Person` has also

28 `THING` is the standard root class of all Protégé models.

Name of System Class	Description
THING	The root class of any Protégé model. The only class that does not have a superclass.
SYSTEM-CLASS	The superclass of all special system classes (except THING).
CLASS	The superclass of all *metaclasses*.
STANDARD-CLASS	The default *metaclass*.
SLOT	The superclass of all *metaslots.*
STANDARD-SLOT	The default *metaslot.*

Table 2
Hierarchy of important default system classes.

inherited its *own slot* (2), in this case from the default meta-class which is not displayed.

Generally, a *template slot* of a schema-*frame* determines the *own slot* of all object-frames of that schema. Thus the *own slots* of an *instance* are determined by attaching *template slots* to its *class*. The *own slots* of a *class* are defined by attaching *template slots* to its *metaclass.* Finally, the *own slots* of a *metaclass* are defined by its *metaclass*. (Cf. table)

The *classes* and *slots* together are called the *ontology* in Protégé, while the *ontology* and *instances* together are called the *knowledge base*. I use instead the more general term *Protégé model* or simply *model*, referring to both the *Protégé ontology* as well as the *knowledge base*.

Every Protégé model includes a certain number of so called *system classes*. These are special *classes*, *metaclasses*, and *metaslots*, like the standard root class THING. The system classes may be used to customize a Protégé model. E. g. instead of using the default *metaclass* STANDARD-CLASS, one may create a custom *metaclass* by sub-classing either CLASS or STANDARD-CLASS. The default system classes cannot be changed or deleted. A list of important system classes can be found in table 2.

29 This is similar to class variables and instance variables in object-oriented programming: class variables determine values of the class, while instance variables determine values of the instance.

2.4 Designing the KMOD ontology

2.4.1 The interviews

The interviews were designed and conducted before I joined the project. The interviews were *qualitative* interviews in the form of guided open-questions. In this form of interview the interviewer uses an interview guideline, i. e. a list of questions. Instead of simply asking the questions one after the other, she encourages the interviewee to speak freely, only guiding him from time to time or asking for clarification on a specific point. [Flick 1995]

The people interviewed were selected from four sub-departments of EGA, ranging across all of the four national companies. The people were selected based on two criteria: their "professional" view, and their "national" view. The professional view is made up of three groups of people: **managers**, **experts**, and **youngsters**. The national view is made of five groups: the views of **France**, the **United Kingdom**, **Spain**, **Germany**, and a **transnational view**. The distribution of the interviewees across the national view can bee seen in table 3:

Department	*Airbus France*	*Airbus UK*	*Airbus Spain*	*Airbus Germany*	*trans-national*
Shape Design	3	3	1	3	2
Data	1	1		2	
Support	3	2		1	
Complete Aircraft	3	2		2	
Total	**10**	**8**	**1**	**8**	**2**

Table 3
Number of interviewees from each department and national company.

For the distribution of members across the categories of managers, experts, and youngsters the exact numbers are not available.

The overall process, from the individual interviews to the final documents, may be described as a four step process (cf. illustration in figure 6 below):

1. The interviews were conducted in all four national companies—France, UK, Spain, and Germany—with each country having its own interviewer.
2. During the interviews notes were taken by the interviewers which were later used to write a summary of each interview (labeled "F1", "F2", "U1" and so on in figure 6).

3. These summaries were then organized by each interviewer into a single document, one for each national company (labeled "France", "UK", "Spain", and "Germany" in figure 6).
4. As the last step these three documents were compiled by the project lead into the final summary document (labeled "Airbus" in figure 6). This final document includes a mind map of the interview results.

This final document is the basis for the creation of the KMOD ontology.

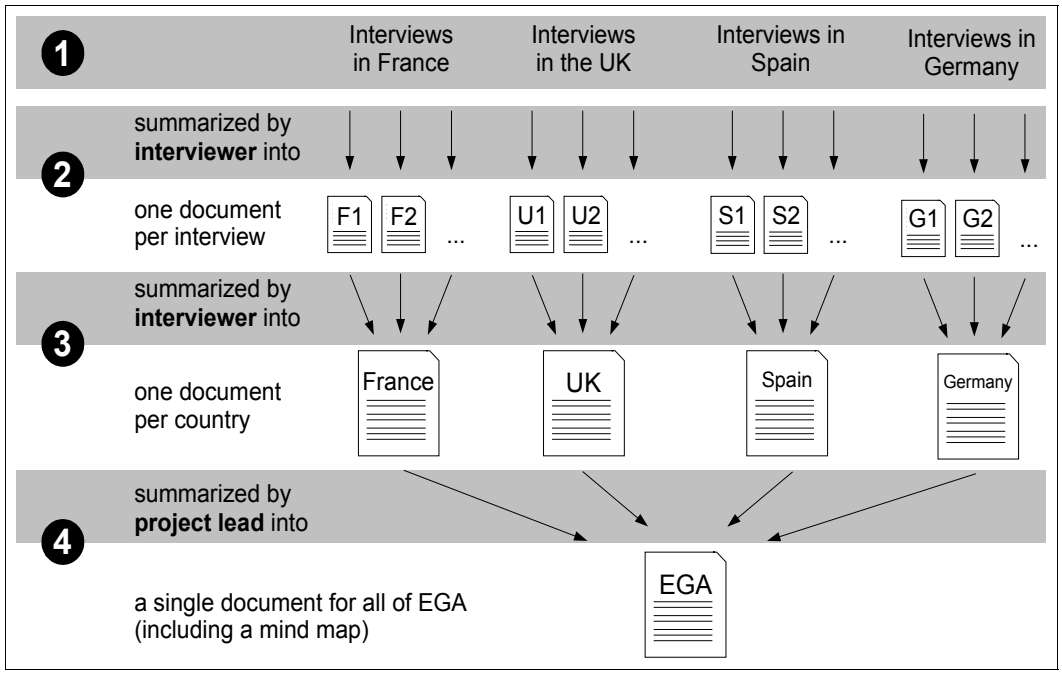

Figure 6
Creating the summary document from the interviews.

2.4.2 Creating the KMOD ontology

The process of designing and creating the KMOD ontology was mainly my responsibility in close cooperation with my company supervisor. It can be seen as creating a series of versions, ranging from simple informal description to the final KMOD ontology as Protégé models with Flora-2 rules.

The first ideas about the KMOD ontology were drafted in the form of notes, sketches, and informal descriptions. The descriptions were created from the notes and sketches with a word processor for easy distribution via email. These documents were used to regularly discuss the ideas with my supervisor at DaimlerChrysler. These discussions had two main purposes: inform my supervisor about the current status of the ontology (including suggestions and new ideas from his side), and discussing the future proceedings of the development process.

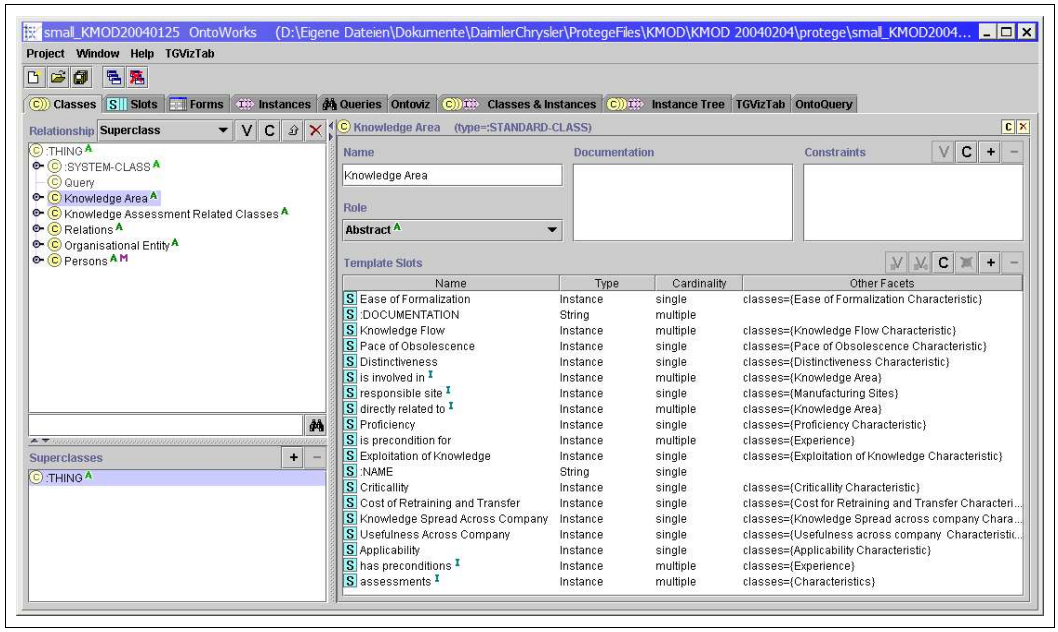

Figure 7
The KMOD ontology in Protégé-2000. On the left is the class hierarchy showing the top-level classes. The right hand side shows details of the class "Knowledge Area".

With the increasing complexity of the drafts, the descriptive documents were becoming more and more difficult to create and maintain. Additionally, the ambiguity of the informal descriptions was increasingly a source of misunderstanding. While the informal descriptions were useful for the description of coarse ideas in the beginning, they were now inhibiting further development and the precise definition of concepts.

We therefore decided to replace the informal descriptions by Protégé models instead.[30] (Cf. figure 7) The initial Protégé model was created from the last informal description. This initial model was the first in a series of continuously revised models. These models became the basis for the communication of the current status of the ontology. As a result of the formal nature of Protégé models, ambiguity was no longer a problem.[31] Instead, three drawbacks of Protégé became apparent:

- Lack of support for effective communication
- Lack of support for multiple versions of a model

30 The use of Protégé for the ontology creation was intended from the beginning, but it had not been clear, when to actually start using Protégé.

31 A negative side effect that we were not aware of a the time was, that we started restricting our way of thinking about the domain to concepts that could be easily implemented with Protégé. This will be discussed in chapter 4 "Evaluation".

- Lack of support for collaboration

Effective communication is increasingly difficult when the models become larger and more complex. A Protégé model can be browsed with Protégé using different views, usually the class-view, the instance-view, or a combination of both. Figure 7 shows a screen-shot of Protégé's class view. On the left the class hierarchy can be seen, with the class "Knowledge Area" highlighted. The large part on the right displays details of the selected class: the value of its own slots (Name, Documentation etc.) and a list of the template slots below.

Here, the problem is that the only relation which can be intuitively viewed and browsed is the super-/sub-class relation of the class hierarchy. All other relations lie "hidden"in the template slots. For a routine user of Protégé this is not a problem when working with a familiar model. But even a routine user will have problems familiarizing herself with models created by others. This was sometimes the case for my supervisor when I had made significant changes to the model: if the changes were radical they could be very hard to comprehend using only Protégé.

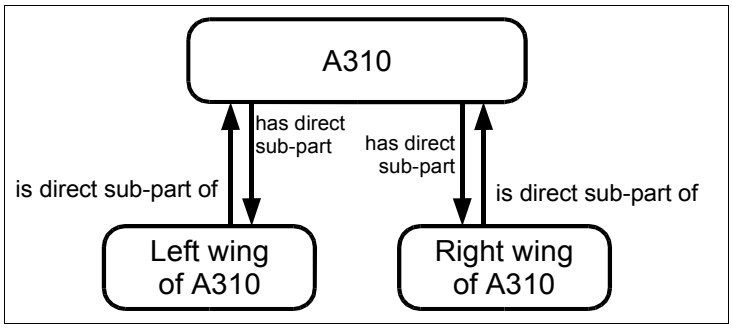

Figure 8
Example of a partial informal graph representation of the KMOD ontology.

To overcome this problem, I drew informal graph representations of selected details of the Protégé model. (Cf. figure 8) Such graphs proofed useful for communicating changes to the ontology with my company supervisor, and for communicating the main concepts to team members who were not familiar with Protégé.

The latter was especially important because neither my supervisor nor I were experts on the domain of aircrafts and the related issues. We sometimes had the opportunity to discuss the ontology (using these graphs representation) with a team member who was more familiar with the domain but not with Protégé.

A sample graph as used for communication is shown in figure 8. The rounded squares represent concepts, in this case instances of the class "Knowledge Area". The arrows represent relations between the concepts, in this case a part-of relation.

Protégé's lack of support for versions and collaboration became apparent when both my company supervisor and myself made changes to our copy of the model independently. Protégé does not provide support for joining two models, which therefore had to be done by hand. Being aware of these complications, we agreed that changes to the model should always be made by the same person.

Flora-2 was not used until the KMOD ontology had a considerable size. It was used to formulate rules (or axioms) and queries. The rules were first formulated as natural language assumptions and afterwards implemented in Flora-2. They were then tested by issuing queries and manually checking the results against the KMOD ontology.

Figure 9 shows a screen shot of the Flora-2-plug-in by DaimlerChrysler called OntoQuery. The center and right show details of the query "Test-Query". The very right hand side shows part of the axioms, which are in Flora-2 syntax. The bottom shows the result of the query after pressing the button "query".

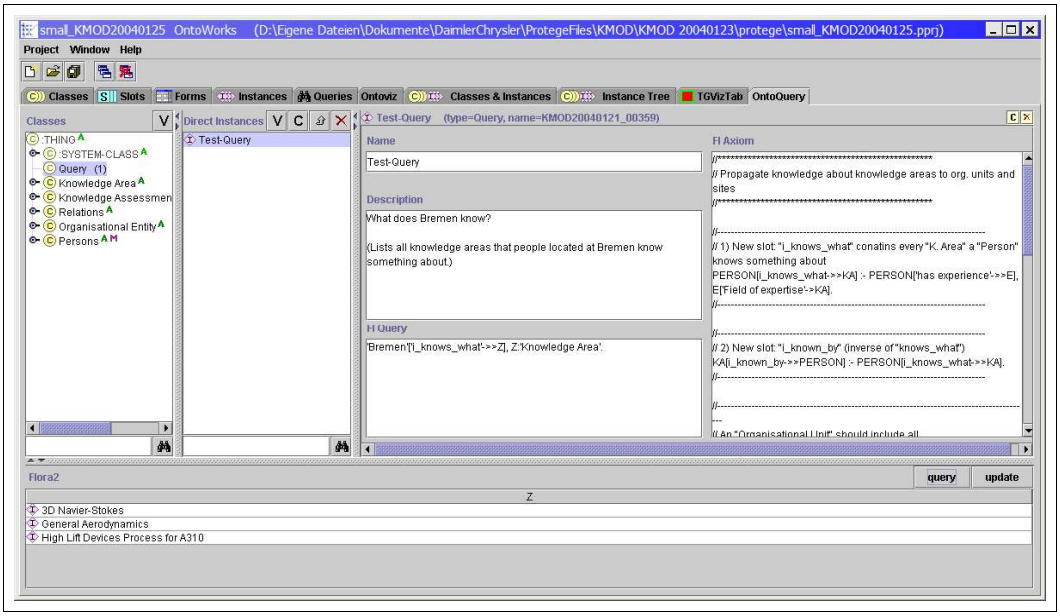

Figure 9
A Flora-2-query in Protégé-2000 (OntoWorks).

This work flow was very time consuming when using the complete KMOD ontology which contained 7067 frames at the time. Most of these frames, roughly 6600, were in fact instances. Introducing a new axiom or changing an existing axiom resulted in a complete recompilation of the Flora-2-knowledge-base. This process took several minutes even on a comparatively fast computer. Though once the compilation was complete, queries were very fast.

To decrease the time needed for the complete compilation of the Flora-2-knowledge-base I removed a number of instances, creating a smaller version of the KMOD ontology which contained only 644 frames. This reduced the compilation time to a few seconds.

Part 3

The KMOD ontology

Part 3 contains a detailed description of the KMOD ontology, including a look at the conceptual view that underlies the ontology and a list of the main classes.

3 Part 3: The KMOD Ontology

3.1 Rationale of the KMOD ontology

The two basic criteria for the development of the KMOD ontology are 1) the separation of domain and assessment knowledge and 2) the (technically imposed) demand to create a class-hierarchy that can be used as a navigation-menu.

Many of the concepts of the KMOD conceptual view and ontology are related to knowledge management. For brevity, the term "knowledge management" will often be abbreviated to "KM" in the names of these concepts.

The central criterion for the design of the KMOD ontology is the separation of the domain knowledge and the assessment of that knowledge. The separation is expected to allow for a reuse of the assessment related concepts of the ontology with other domain ontologies.

The KMOD web portal (OntoPortal) is currently build in such a way that it constructs the the navigation-menu directly from the class hierarchy of the underlying KMOD ontology. This imposes a constraint on the class-hierarchy: The class-hierarchy needs to be structured in a way that will enable users to easily navigate the site.

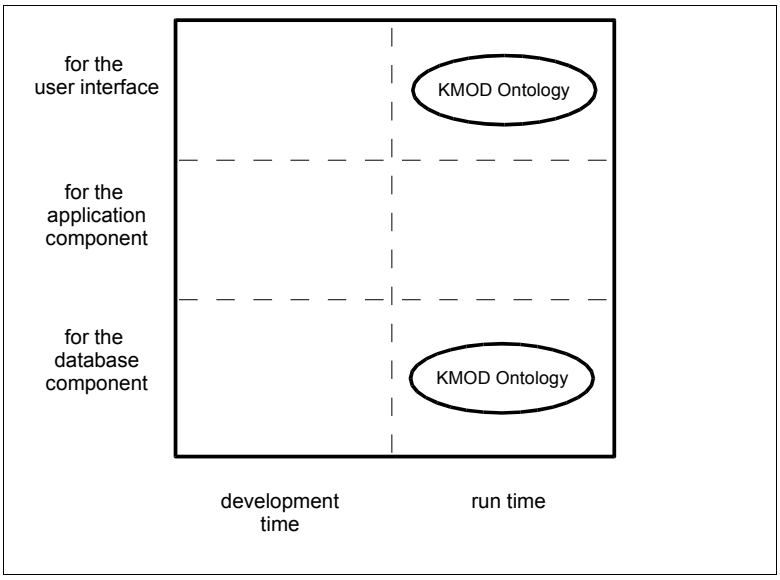

Figure 10
Usage of the KMOD ontology

The meta-model of Protégé is able to accommodate this demand by allowing a class to have multiple super-classes. (See the description of the class `Person`, below.)

Using the static and dynamic criteria introduced in chapter 2, the KMOD ontology can be classified statically as an application-ontology, because it combines both a task- and a domain-ontology. The task is that of assessing knowledge, while the domain is the domain knowledge that is being assessed—currently EGA's knowledge.

Dynamically, the KMOD ontology can be classified as being used for two run time aspects: for the database component and the user interface (cf. figure on the previous page).

3.2 A look at the conceptual view of KMOD

This description is an attempt to spell out *from my point of view*[32] the conceptual view that the KOMD team developed over the course of the project—as opposed to the KMOD ontology which was implemented in Protégé and Flora-2. This conceptual view underlies the implemented KMOD ontology but is not identical with it (cf. chapter 2). The team's conceptual view changed over time while we were working on the KMOD ontology. The state described here is the one that is most closely related to the description of the KMOD ontology in the next section.

32 A specific conceptual view was never documented or explicitly agreed upon. That is why I describe my idea of the KMOD conceptual view.

3.2.1 Knowledge assessment

The goal of KMOD was to create a detailed account of the department's (and eventually all of Airbus') knowledge, including an assessment of that knowledge. Here, *knowledge assessment* means how the knowledge relates to EGA's business (how critical is the knowledge for the core business processes, how unique is it etc.).[33] The criteria for this assessment are called the *knowledge assessment criteria*, or simply *assessment criteria* (no. 2, fig. 11 below). The criteria were taken from the interview results document.[34] Some of these criteria are listed in Table 4 (on the next page)

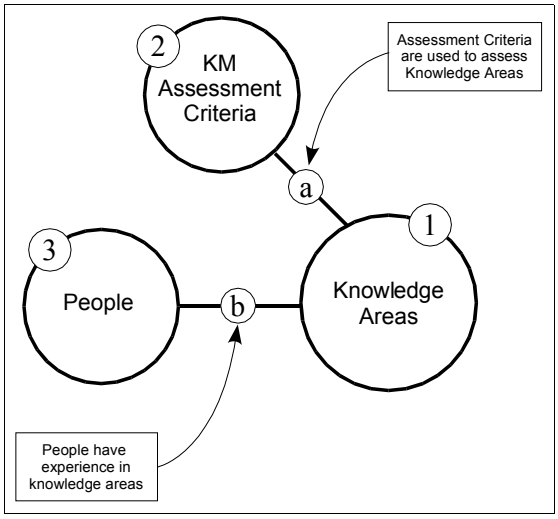

Figure 11
Relation between knowledge areas and people on the one hand, and knowledge area and KM assessment criteria on the other.

33 The notion of *knowledge assessment* used in this thesis is not connected with a different notion which refers to the assessment *of people's knowledge*. The latter is used in the context of learning and training to assess how well a person has learned something.

34 These criteria were actually compiled before the interviews were conducted. They were not inferred from the interviews.

Descriptive question / Example	*Scale*	*KMOD Name*
How fast will the knowledge (k.) grow old and obsolete? *Knowledge about the method 3D-Navier-Stokes is still changing and would be rated either (2)* evolving *or even (3)* dynamic.	1) static k. 2) evolving k. 3) dynamic k.	Pace of Obsolescence
How distinct is the knowledge (k.)? *Knowledge about common basic laws in physics would be rated (1)* fundamental.	1) fundamental k. 2) advanced k. 3) innovative k.	Distinctiveness
How much has the knowledge penetrated the company, or how far is it spread? *Knowledge that is only shared by one or two individuals would be rated (1)* individual.	1) individual 2) limited spread 3) collectively shared	Knowledge Spread Across Company
How effectively is the knowledge used?	1) not at all 2) partially 3) effectively	Exploitation of Knowledge
How critical is the knowledge for the successful functioning of the company? *Knowledge that will not effect the functioning of Airbus would be rated (1)* low.	1) low 2) medium 3) high	Criticality

Table 4
Selected KMOD knowledge assessment criteria.

3.2.2 People and experience

A meaningful account of EGA's knowledge is not complete without the concept of people that work there. *People* (3, fig. 11, on the previous page) are the ones who actually *know* something, they have experience in one or more knowledge areas (b, fig. 11).

Figure 11 hides some details about the relation between *people* and *knowledge areas* (b): A person might have a certain *level of expertise* for a given knowledge area (beginner, expert etc.). We call this the person's *experience*.

A similar concept of experience also relates different knowledge areas. The following informal description is an example of a case where one knowledge area depends on another:

> *In order to successfully design the high lift device of an airplane, one needs be an expert in general aerodynamics.*

Consequently, a person can only master the knowledge area *high lift design* if she is an expert on the knowledge area *general aerodynamics.* We call this the *precondition* of a knowledge area. These two concepts (experience and precondition) are represented by a single class "*experience"* in the KMOD ontology.[35]

3.2.3 Knowledge management concerns

Besides representing knowledge areas in order to assess them, the KMOD ontology was expected to model knowledge management (henceforth KM) related concerns and possible solutions. Another example in the form of an informal description:

> *Sometimes there is only one expert for a certain knowledge area. If this expert retires without passing on her knowledge to other members of Airbus, the knowledge is lost. To avoid such loss of knowledge the use of knowledge management techniques would be helpful.*

This example raises some general questions about the relation of *concerns*, knowledge *development*, *sharing*, *reusing*, and *knowledge management techniques.*

- What are the *concerns* of department's members with respect to critical knowledge?
- How do these concerns influence *developing*, *sharing*, and *reusing* of knowledge?
- Which *knowledge management techniques* could be used to positively influence these concerns?

35 In the beginning we had doubts about this way of representation, because experience of a person and experience as a precondition are obviously not identical concepts. In retrospect, it was a choice that not only worked but that also follows Hobbs idea of basing semantics on how we talk about the world [Hobbs 1985].

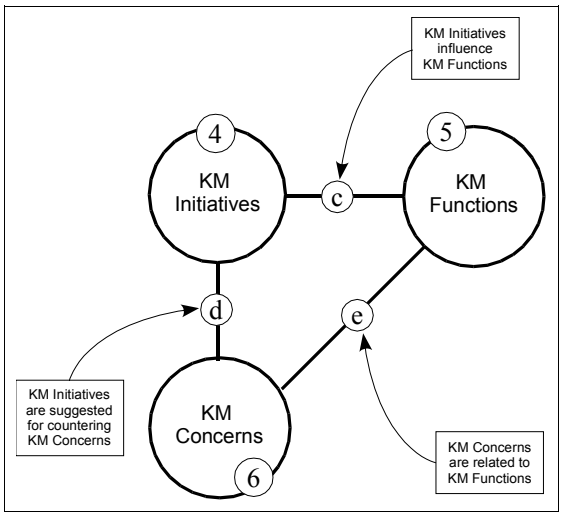

Figure 12
Relation between KM initiatives, KM functions, and KM concerns.

The answers to these questions come from two sources: the interviews, and KMOD's expert on knowledge management. For the KMOD ontology we identified three elements which we used to model the answers (cf. figure 12): *KM concerns* (6), *KM functions* (5) (development, sharing, reusing)[36], and *KM initiatives* (4).

KM functions are influenced by both *KM concerns* and *KM initiatives* (e and c in fig. 12)*:* In the example above, the loss of knowledge through retirement (*KM concern*) has a negative impact on the ability to share the knowledge (KM function "share")—knowledge which the company has lost can obviously no longer be shared. One known possibility to positively influence the sharing of knowledge (again, KM Function "share") is the establishment of experience boxes[37] (a KM initiative).

The interview result document also included a list of suggested KM initiatives for each KM concern (d, fig. 12). This list was compiled by Airbus' knowledge management experts. We included these suggestions as a relation between KM initiatives and KM concerns.

36 During the course of the project more KM functions were added to this list, like the transformation between tacit and explicit knowledge.

37 Experience boxes refers to the practice that people that do similar things should be located close to each other, in the expectation that they will share their knowledge.

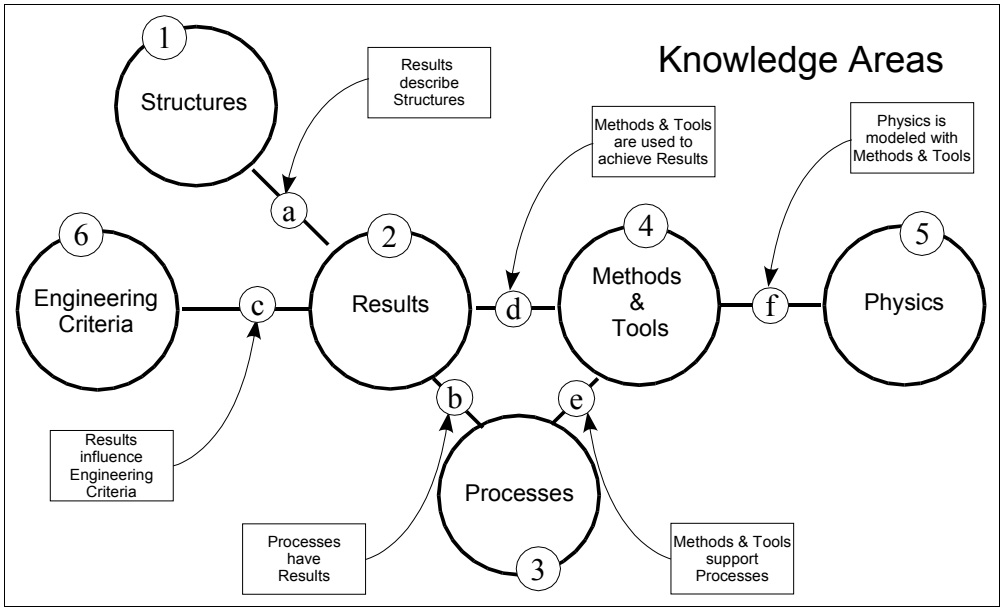

Figure 13
The six main kinds of knowledge areas and their relationship.

3.2.4 Six kinds of knowledge areas

So far, knowledge areas are something that can be assessed using the assessment criteria. To enable the modeling of relationships *between* certain knowledge areas we need a more detailed concept. The project team identified six main kinds of knowledge areas based on the interview results as well as the team members' knowledge about industrial engineering (numbers 1-6 in figure 13):

1. *Structures*: Any kind of physical object (currently only airplanes' parts).
2. *Results*: A document like the design of an airplane or a test result.
3. *Processes*: A process like designing an aircraft or a part of an aircraft.
4. *Methods & Tools*: A method (like eXtreme Programming) or a tool (like a word processor).
5. *Physics*: The subject of physics, especially aerodynamics.
6. *Engineering Criteria*: Aircrafts are designed to have certain properties, like having a low weight and low drag while being able to carry many people. These three (and other) conflicting goals are called *engineering criteria* in KMOD.

Using relations between these specializations of the knowledge area concept, the inner structure of the domain can be expressed. Between the six, the following relations are assumed to exist (letters a-f in figure 13):

a) **A *structure* may be defined by some *result*.**
(An airplane (*structure*) is defined in the design blueprints for that plane (*result*).)

b) **A *process* may yield a *result*.**
(The design phase of a new aircraft (*process)* results in a number of design documents (*result*).)

c) **A *result* may have an influence on an *engineering criterion*.**
(The design documents *(result)* for an aircraft have an influence on the drag of the aircraft *(engineering criterion)*.)

d) ***Methods & tools* may be used to create a certain *result*.**
(3D-Navier-Stokes (*method*) is used in creating the design document of an airplane (*result*).)

e) ***Methods & tools* may be used to support a certain *process*.**
(3D-Navier-Stokes (*methods & tools*) is used during the design phase of a new airplane (*process*).

f) ***Methods & tools* may be used to model (certain aspects of) *physics*.**
(3D-Navier-Stokes (*methods & tools*) can be used to model the flow of air (*physics*).)

The concepts of *process* and *result* are more specific than the chosen names suggest. The term *process* in the context of KMOD does only include such processes which yield a result in the form of a document. This excludes all kinds of manufacturing processes. Similarly, the term *result* is meant to refer exclusively to documents. Thus, the current version of KMOD cannot be used to represent the actual manufacturing process of an aircraft, but it can be used to represent the process of planning and designing such an aircraft.

The main reason for this restriction to "mental" processes and results lies in the department's field of work: The KMOD project took place in a department that designs aircrafts but that does not manufacture them. Physical processes, like manufacturing, played no role in the interview result document that was the basis for the ontology.

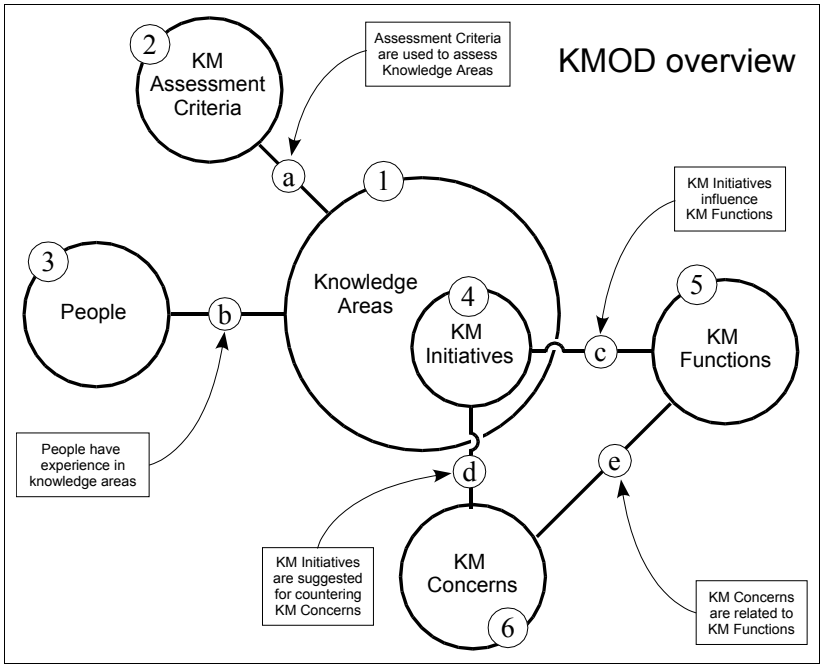

Figure 14
Overview of important KMOD concepts (1-6) and their relations (a-e).

3.2.5 Overview of the conceptual view of KMOD

Figure 14 is an overview of the main concepts of the KMOD conceptual view. The central concept is *knowledge area* (1). A knowledge area can be assessed (a) with the help of *knowledge assessment criteria* (2). *People* (3) can have experience (b) in a certain knowledge area.

A *knowledge management initiative* (4) is a special kind of knowledge area (indicated by the placement of (4) inside the boundaries of (1)) that is related (c, d) to both *knowledge management functions* (5) and *knowledge management concerns* (6), which are themselves related to each other (e).

- THING
  - Knowledge Area
    - Structure
    - Process
    - Result
    - Methods and Tools
    - Engineering Criterion
    - Physics
  - Person
    - Airbus Person
    - Non Airbus Person
  - Organizational Entity
    - Organizational Unit
    - Manufacturing Site
    - Person
  - Relation
    - Experience
    - Function Influence on Criteria
  - Knowledge Assessment Related Class
    - KM Function
    - KM Concern
    - KM Practice
    - Characteristic

Table 5
Hierarchy of the main classes of the KMOD ontology. Note that the same class "Person" appears twice: as a subclass of "THING", and as a subclass of "Organizational Entity". See the description of "Person" in this section for more information.

3.3 The main classes of the KMOD ontology

The complete first two levels[38] of the KMOD ontology's class-hierarchy are shown in table 5. THING is the root-class of every Protégé model and is listed only for reference.

38 Not counting Protégé's standard root-class THING.

Sample Class

This `Sample Class` is introduced by an informal description followed by a list of its properties. The informal description refers to slots of a class by using one or more numbers. In this example (1) refers to the slot called `first slot`.

Sample Class		
Template slots:	(1)	**`first slot`** `(value type of first slot)`
	(2)	`...`
Superclass:	`List of superclasses`	
Subclasses:	`List of subclasses`	

Table 6
Example of a description for a class called "Sample Class".

The following introduction of the main classes lists for each class a short informal description of the class followed by a table of its properties (*template slots*, *super-*, and *subclasses*). Table 6 shows an example for a class called "Sample Class".

The informal description of a class will contain references to the template slots of that class in parenthesis, e. g. (1). The numbers refer to the number of the template slot from the property list of the class. In the example of table 6 the number (1) refers to the slot `first slot`.

The names of classes, slots and types are set using a `typewriter font`. Class names start with an `Uppercase` letter, while slot names start with a `lowercase` letter. The default slots of Protégé (like `name` and `documentation`) are omitted for brevity.

The list of template slots gives the type of the slot value in parenthesis. E. g.

```
has precondition (Experience)
```

means that the value of the slot named `has precondition` is of type `Experience` (which is a class). If the type of the template slot equals the class the slot belongs to, then the class name is omitted, e. g. the following slot is a direct slot of the class `Process`:

`is direct sub-process of`

This means that the slot is of the type it belongs to, in this case `Process`, because no other type is given in brackets.

Classes with names in the plural form will also be referred to in their single form and vice versa to avoid grammatical errors.

3.3.1 Complete list of top-level classes[39]

Some classes, like the `Relation` class, are referred to as *convenience classes*. Such classes have no formal meaning in the KMOD ontology, which means that they could be removed without harming or changing the rest of the ontology. The purpose of these classes is to make the ontology easier to understand and maintain, because they provide a description for their subclasses and make them more conveniently accessible in Protégé.

Knowledge Area

The `Knowledge Area` class is the central class of the KMOD ontology, having more template slots than any other class. Instances of this class represent what an `Organizational Entity` can have knowledge about. The two classes are connected via the class `Experience`. Two `Knowledge Areas` can be related to each other in three different ways (from the most specific to the most general):

- one is a precondition for the other (10,13),
- one is in some way involved in the other (11,12),
- one is in some other way related to the other (6).

For some `Knowledge Areas` there exists a `Manufacturing Site` that is officially responsible for that `Knowledge Area` (18). The rest of the slots have the purpose of assessing the `Knowledge Area` using instances of a subclass of `KM Characteristics` (1, 2, 3, 5, 6, 7, 12, 13, 14, 15, 17).

39 Again, Protégé's default root-class `THING` is omitted.

Knowledge Area	
Template slots:	(1) **applicability** (Applicability Characteristic) (2) **cost of retraining and transfer** (Cost of Retraining and Transfer Characteristic) (3) **criticality** (Criticality Characteristic) (4) **directly related to** (5) **distinctiveness** (Distinctiveness Characteristic) (6) **ease of formalization** (Ease of Formalization Characteristic) (7) **exploitation of knowledge** (Exploitation of Knowledge Characteristic) (8) **has precondition** (Experience) (9) **involved knowledge areas** (10) **is involved in** (11) **is precondition for** (Experience) (12) **knowledge flow** (Knowledge Flow Characteristic), (13) **knowledge spread across company** (Knowledge Spread Across Company Characteristic) (14) **pace of obsolescences** (Pace of Obsolescence Characteristic) (15) **proficiency** (Proficiency Characteristic) (16) **responsible site** (Manufacturing Site) (17) **usefulness across company** (Usefulness Across Company Characteristic)
Superclass:	THING
Subclasses:	Engineering Criteria Method and Tool Physics Process Result Structure

Organizational Entity

An `Organizational Entity` is the superclass of classes which represent anyone or anything that is regarded part of an organization, like an employee (e. g. John Smith), an organizational unit (e. g. EGA), or a manufacturing site (e. g. Airbus Bremen). An `O.` can have experience in some `Knowledge Area` (1).

Organizational Entity	
Template slots:	(1) **`has experience`** `(Experience)`
Superclass:	`THING`

Subclasses:	`Manufacturing Site`
	`Person`
	`Organizational Unit`

Person

`Person` is the superclass for the two classes `Airbus Person` and `Non Airbus Person`. Note that `Person` appears twice in the table above (table 5) because of the design criteria of the KMOD ontology: From the perspective of the user the concept of `Person` is central enough to appear on the first level of the web-portal's navigation-menu. From the modeling perspective of KMOD a `Person` is a kind of `Organizational Entity`.

Person	
Template slots:	*none*
Superclass:	`THING` `Organizational Entity`
Subclasses:	`Airbus Person` `Non Airbus Person`

Relation

This is the superclass of all relations that cannot be directly modeled as slots. (I. e. *n*-ary relations with $n > 2$). This class is currently a *convenience class* (see above) that has no formal meaning, nor does it appear in the navigation menu of the web portal.

Future versions of the KMOD ontology may introduce a formal meaning and a taxonomy of relations.

Relation	
Template slots:	*none*
Superclass:	THING
Subclasses:	Experience Function Influence on Criterion

Knowledge Assessment Related Classes

Like Relation, this class only serves as a collecting point for its subclasses.

Knowledge Assessment Related Class	
Template slots:	*none*
Superclass:	THING
Subclasses:	Characteristic KM Concern KM Function KM Practice

3.3.2 Direct subclasses of Knowledge Area

Structure

Structure is the superclass of all classes which represent an aircraft (Complete Aircraft) or a part of an aircraft (Aircraft Parts). The instances of Structure can be organized into a part-of hierarchy (1,3). Also, a Structure may be specified by a Result (2).

Structure	
Template slots:	(1) **has direct sub-parts,** (2) **inverse of details structural component** (Result) (3) **is direct sub-part of**
Superclass:	Knowledge Area
Subclasses:	Aircraft Parts Complete Aircraft

Process

A Process in KMOD represents any process which results in some kind of document. (Thus, excluding processes like manufacturing an airplane.) A Process can be part of a sequence of Processes (1,2), and can also be part of a hierarchy of processes (3,4). Usually a Process is supported by Methods and Tools (5) and renders some Result (6). The subclasses correspond to kinds of processes that where identified by the KMOD team. Although the subclasses currently have no formal meaning, they are useful for the web portal to group similar processes together.

Process	
Template slots:	(1) **directly follows,** (2) **directly precedes,** (3) **has direct sub-process,** (4) **is direct sub-process of** (5) **supporting methods & tools** (Methods and Tools) (6) **yields result** (Result)
Superclass:	Knowledge Area
Subclasses:	Total Process Development Process Concept Design Process Definition Process Physical Test Process Feasibility Process Simulation & Calculation Process

Results

A `Result` represents a result which can be documented. It may be the result of a `Process` (5). It usually has impact on some `Engineering Criterion` (4), is usually achieved by employing some `Method and Tool` (6), and can be part of a hierarchy of `Results` (2,3). In addition, it may specify some part or parts of an airplane or even a complete airplane (1).

Results	
Template slots:	(1) **details structural component** (Structure) (2) **has direct sub-result** (3) **has direct super-result** (4) **has impact on** (Engineering Criteria) (5) **is result of** (Process) (6) **method/tool used** (Methods and Tools)
Superclass:	Knowledge Area
Subclasses:	Complex Result Concept Result Detailed Design Result Physical Test Result Simulation and Calculation Result

Methods and Tools[40]

The class `Methods and Tools` represents a hierarchy (4,5) of methods and tools that are usually used for modeling some aspect of physics (3) or for supporting some process (6). A `Method and Tool` may have a `Result` (1) and a certain level of maturity (2).

40 The name of this is especially problematic. From modeling perspective a name like "Method or Tool" would be more appropriate. But having the web portal in mind it makes sense, because the user will select this for a list of all methods and tools.

Methods and Tools	
Template slots:	(1) **method maturity level** (Method Maturity Level Characteristic) (2) **more general method or tool** (3) **more specific method or tool** (4) **results in** (Result) (5) **used for modeling** (Physics) (6) **used to support process** (Process)
Superclass:	Knowledge Area
Subclasses:	KM Initiatives KM Tools

Engineering Criterion

An Engineering Criterion represents an aspect of an engineering solution, like the *optimum weight solution for the wings*. It is usually influenced by a certain Result (1), and may be part of a hierarchy of Engineering Criteria (2,3). E. g. the *overall optimum solution* would have the above mentioned optimum weight solution as one sub criterion.

Engineering Criterion	
Template slots:	(1) **is influenced by** (Result) (2) **sub criteria,** (3) **super criteria**
Superclass:	Knowledge Area
Subclasses:	*none*

3.3.3 KM Function, KM Concern, and KM Initiative

KM Function

A KM Function represents a knowledge related action, like sharing or reusing knowledge. A KM Functions may be influenced by KM Concerns and/or KM Initiatives (1).

KM Function	
Template slots:	(1) **influenced by these km concerns and initiatives** (KM Concerns, KM Initiatives)
Superclass:	Knowledge Assessment Related Classes
Subclasses:	*none*

KM Concern

KM Concern represents concerns that may have an impact on Airbus's KM efforts. A concern in this sense could be an employment stop due to a budget cut, or the problem of high personnel turn-over. A KM Concern has a certain importance (2) may influence KM Functions (1) and may be influenced by KM Initiatives (3).

Knowledge Assessment Related Class	
Template slots:	(1) **impacted km functions** (KM Functions) (2) **km concern importance** (Importance Of Concern Characteristic) (3) **suggested km initiatives** (KM Initiatives)
Superclass:	Knowledge Assessment Related Classes
Subclasses:	*none*

KM Initiative

A KM Initiative represents a type of knowledge management effort, like *workshops* (i. e. not one specific workshop but the notion of workshops for knowledge sharing, reusing etc.). A KM Initiative will usually have some impact on one or more KM Functions (1), and may be suggested for solving (or at least positively influencing) certain KM Concerns (2).

KM Initiative	
Template slots:	(1) **impacted km functions** (KM Functions) (2) **suggested for solving these km concerns** (KM Concerns)
Superclass:	Methods and Tools
Subclasses:	*none*

3.4 Representing knowledge with the KMOD ontology —selected examples

The main classes of the KMOD ontology which have been introduced in the previous section were used to represent statements that were extracted from the interview results document and other sources. A formal representation of a statement was created by identifying central concepts of that statement which could be "mapped" to corresponding classes or relations of the KMOD ontology. This process of mapping usually meant creating a new instance of a class. E. g. to represent a statement involving an A310 airplane, we would create an instance of `Complete Aircraft` (a subclass of `Structure`) called `A310`.

It must be pointed out that the A310 is not a single airplane but stands for a whole class of airplanes. Would it not therefore make sense to represent the concept of an A310 as a class instead of an instance? In KMOD the instance `A310` represents the class of A310 airplanes, because the department for which the KMOD ontology was developed is not concerned with individual airplanes but with classes of airplanes.

This section gives some examples of how the KMOD ontology can be used to create formal representation of informal descriptions.

3.4.1 Syntax of the representations used

To illustrate the examples in this section, parts of the KMOD ontology are represented by using graphs. The graphs consist of rounded rectangles and arrows connecting the rectangles. The rectangles represent instances and the arrows represent own slots of the instances.

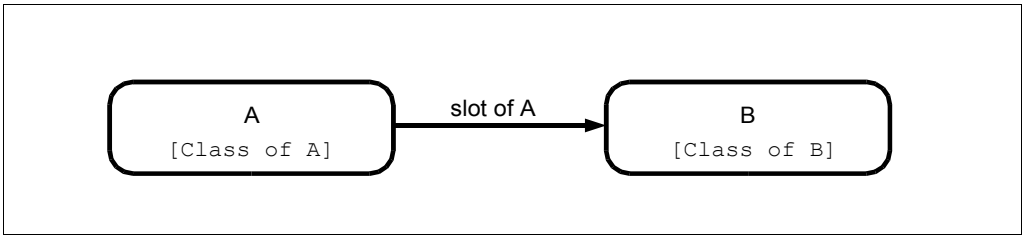

Figure 15
Graph representation of two instances connected via a slot.

Figure 15 shows two instances, "A" and "B", and an own slot of A labeled "slot of A". The slot value of "slot of A" is "B". Another way to state this is to say "*A* and *B* are connected via *slot of A*". The classes are given in square brackets below the instance names. Note that only slots are displayed that are relevant in the current context and that both instances may have further own slots which are not shown.

The Flora-2 rules, however, cannot be adequately represented using the simple graphs above. Thus the rules will be given in the original Flora-2 syntax. The relevant concepts of the Flora-2 syntax are summarized below:

- Words starting with a lowercase letter and words in 'single quotes' are literals.
- Words starting with an Uppercase letter are variables.
- `f1::f2` defines
  - that the frame `f1` is a subtype (subclass) of frame `f2`.
- `i:f[s1->>{v1, v2, ..., vn}]` defines
  - an instance `i` of type `f`,
  - a slot `s1` having the values `v1` through `vn`.
- Rules in FLora-2 are stated as inverse implications:
  `statementB :- statementA` defines
  - a rule that makes `statementB` true if `statementA` is true.
  - A sample rule:

    ```
    X[lastname->>SNAME] :- X[surname->>SNAME]
    ```

    defines a rule that for all frames `X` that have a slot called `surname`, will "add" a slot called `lastname` to that frame with `lastname` having the same value as `surname`.
- Lines starting with "`//`" are comments.

3.4.2 Representing relations

A *relation* (or *predicate*) can be defined as a function that maps its arguments to the truth values *true* or *false*. In Protégé relations have to be defined *extensionally*, i. e. one must list all possible combinations of arguments for which the relation holds (is *true*). Using Flora-2 offered the additional possibility of defining relations *intensionally*, by defining rules that state the conditions under which the relation holds.

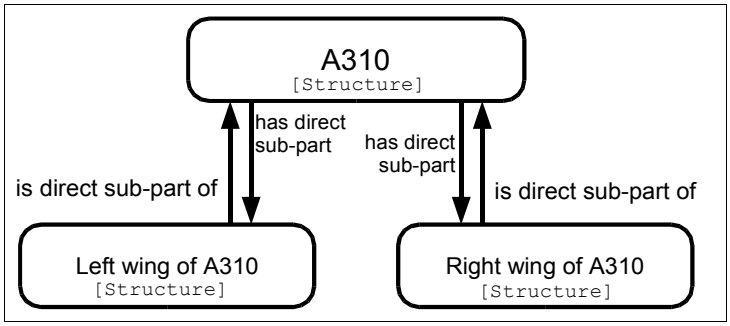

Figure 16
The binary relation "direct-part-of" is represented using two inverse slots "has direct sub-part" and "is direct sub-part of".

Binary relations between instances

For binary relations there were two possibilities: to model them using either first-class objects or using slots. It was decided to model them as slots, because it is straight-forward to do so using Protégé's *inverse slots*[41].

A binary relation is thus represented by one or two slots, each connected to a class. Two slots are only used when the relation must be navigable in both directions in the web portal. E. g. the direct-part-of relation for `Structure` is implemented using two slots, both attached to `Structure` itself: `is direct sub-part of (Structure)` and `has direct sub-parts (Structure)`. Figure 16

41 A list of inverse slots can be found in Appendix A.

If two *slots* are defined as *inverse slots* (or one *slot* as an *inverse* of itself) and a value is assigned to one of them, then Protégé will automatically add the correct value to the other. E. g. the `has direct sub-part` and `is direct sub-part of` *slots* are defined as *inverse slots*. Adding an *instance* A to the `has direct sub-part` *slot* of B will automatically add B to A's `is direct sub-part of` *slot* as well.
Note that this will only work, if the inverse slots are defined before assigning the values. Changing existing *slots* with already assigned values to *inverse slots* will not have this effect.

illustrates how the direct-part-of relation between an *A310 (*a specific type of aircraft) and its wings is modeled.[42] For brevity, subsequent illustrations will only show one of the inverse slots of the binary relation.

Another relation, the part-of relation, is defined in terms of the direct-part-of relation: Using Flora-2 rules it was possible to define the part-of relation as the transitive closure of the direct-part-of relation:

```
// All direct sub-parts are also sub-parts:
A['has sub-parts'->>B] :-
      A['has direct sub-parts'->>B].
```
[43]

This defines a new slot `has sub-parts` as containing all instances which the existing `has direct sub-parts` contains. A second rule makes sure, whenever a A has a sub-part B, and B has a sub-part C, then A also has C as a sub-part:

```
// Transitive closure:
A['has sub-parts'->>C] :-
      A['has direct sub-parts'->>B],
      B['has sub-parts'->>C].
```

Without Flora-2 the part-of relation would have had to be modeled the same way as the direct-part-of relation. The slot values would need to be filled in by hand which would be very time-consuming and error-prone.

A similar approach was used to define a relation called *knows about.* The Protégé model only contained assertions about which knowledge areas a person knows something. Using Flora-2 the relation was extended to `Organizational Units`: A rule was introduced which asserts that an `Organizational Unit` knows about the same knowledge areas which all its members know about, and that an `Organizational Unit` also knows about everything that all its sub-units know about.

---

42 Note that the instances are actually instances of subclasses of `Structure`. The purpose of the illustration is to show the modeling of a binary relation.

43 Constraints are omitted here for better readability. For a better run time performance of the rules it is useful to add type-constraints for A, in this case restricting A to subclasses of `Structure`.

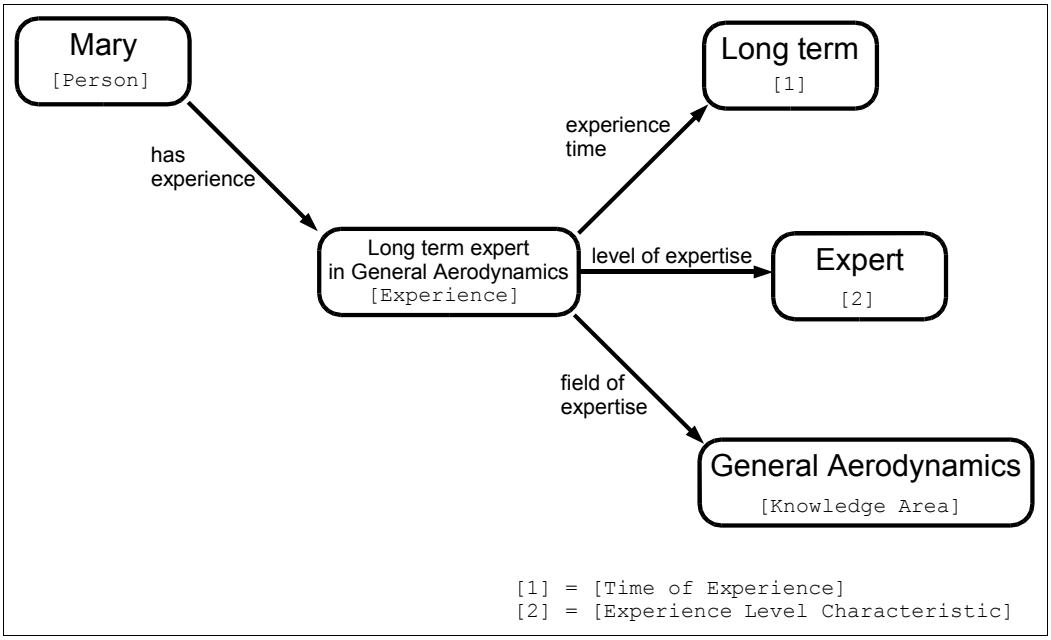

Figure 17
"Mary has long term experience as an expert in general aerodynamics."

n-ary relations between instances

The *n*-ary relations with $n > 2$ are modeled as first-class objects. Using first-class objects means creating different classes for different kind of relations, one class for each kind. These classes have a slot for each partner in the relation.

This approach is used to model people's *experience,* which is necessary in order to represent a certain level of experience of a person—as opposed to representing that the person knows something (without qualifying how well she knows it). For example:

(1) John has experience in general aerodynamics.

(2) Mary has long term experience as an expert in general aerodynamics.

Statement (1) could be modeled by a slot `knows` attached to the class `Persons`. Then `John` could be related to `General Aerodynamics` by using this slot. If the same slot would be used for (2), then the extra information about Mary's expertise will be lost. Therefore, a new class `Experience` was created with three slots `level of expertise`, `field of expertise`, and `experience time`. The class together with these three slots can be used to qualify a person's experience in more details than a simple slot. To represent statement (2) one would create an instance of `Experience` (labeled `Long term expert in General Aerodynamics` in figure 17) and fill its slot with the instances `Long Term`, `Expert`, and `3D-Navier-Stokes`. This instance of `Experience` would then be added to the `has experience` slot of `Mary`.

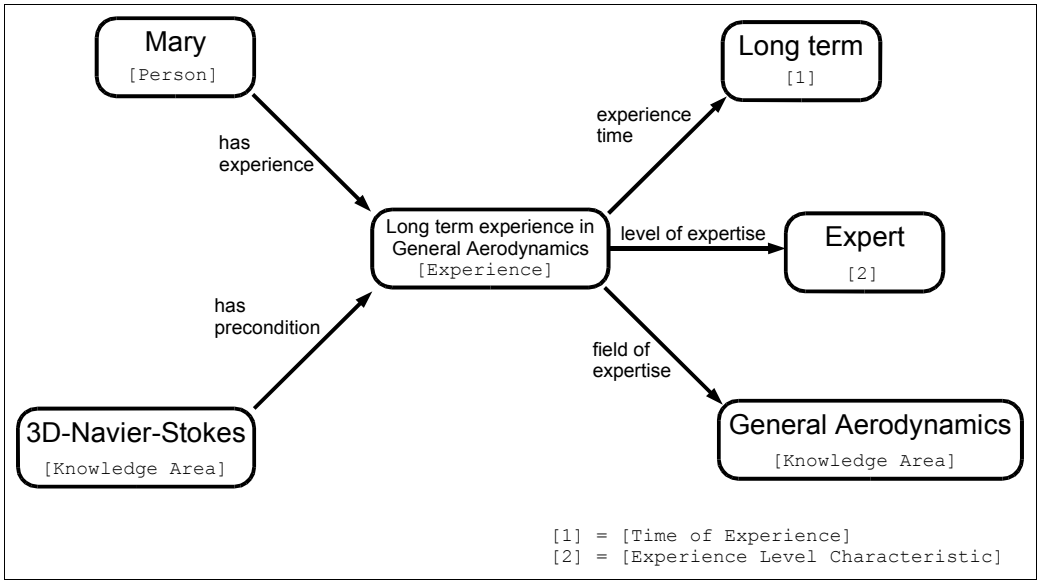

Figure 18
`Experience` *representing Mary's experience and a precondition:*
"Being a long term expert in general aerodynamics is a precondition for 3D-Navier-Stokes."

To represent statement (1), one would create another instance of `Experience`. While the `field of expertise` slot would be filled with the same instance of `General Aerodynamics`, the other two slots would be left blank. This instance of `Experience` would then be added to the `has experience` slot of `John`.

So far, instances of `Experience` are only used to model the experience that a person has of a certain knowledge area. Because the goal of the KMOD ontology was the assessment of knowledge, but not primarily the knowledge a certain person has, a related notion of experience was also modeled using the same class `Experience`. This notion of experience represents a kind of *precondition*[44], in the sense that knowledge in one area is a necessary precondition to gain knowledge in some other area. Thus, `Experience` can represent a person's *experience* as well as a *precondition* for a knowledge area. The two cases are shown in figures 17 and 18: An instance of `Experience` that represents a person's experience (figure 17 on the previous page). The same instance additionally represents the necessary experience (the precondition) for a knowledge area in figure 18.

Note that the same `Experience` instance may be assigned to any number of `Knowledge Areas` and `Persons`. Thus, an instance of `Experience` does not represent an individual's knowledge (i. e. not *"John's long term expertise in aerodynamics"*) but instead a notion of a more general kind (*"Long term expertise in*

44 See the description of the conceptual view underlying KMOD, page 43, for the concept of *precondition of a knowledge area.*

aerodynamics"). The notion of individual knowledge is represented by assigning an `Experience` instance to a `Person`. The same is true when using an instance of `Experience` to represent the precondition of a knowledge area.

As a result, when a `Person's` experience changes (or when it is determined that the necessary experience for some knowledge area changes) and this different experience must be represented, then, instead of changing the properties of the assigned experience instance (e. g. replacing the slot-value `Long Term` with some other instance of `Time of Experience`), the whole `Experience` instance should be replaced with another instance. Otherwise, all `Persons` and `Knowledge Areas` which were assigned the original `Experience` would now be connected to the changed instance of `Experience`, which is probably not intended.

3.4.3 Representing knowledge assessment criteria

The idea of the KMOD ontology was to enable the assessment of the Airbus department's knowledge. Assessment of a knowledge area is achieved by creating an instance of a certain class (a subclass of `KM Characteristic`) and assigning it to a certain slot of the knowledge area.

A total number of 13 characteristics were identified for the purpose of assessment, which are all represented by subclasses of `KM Characteristic`. Instances of eleven of the 13 characteristics are used directly as slot-values for slots of the `Knowledge Area` class (cf. the property list of `Knowledge Area`, slots 1, 2, 3, 5, 6, 7, 12, 13, 14, 15, 17). The instances of the two remaining classes are used as a slot-value for a slot of the `Method and Tool` class (the slot `method maturity level`), and for the slot-value of a slot of the `Experience` class (the slot `level of experience`).

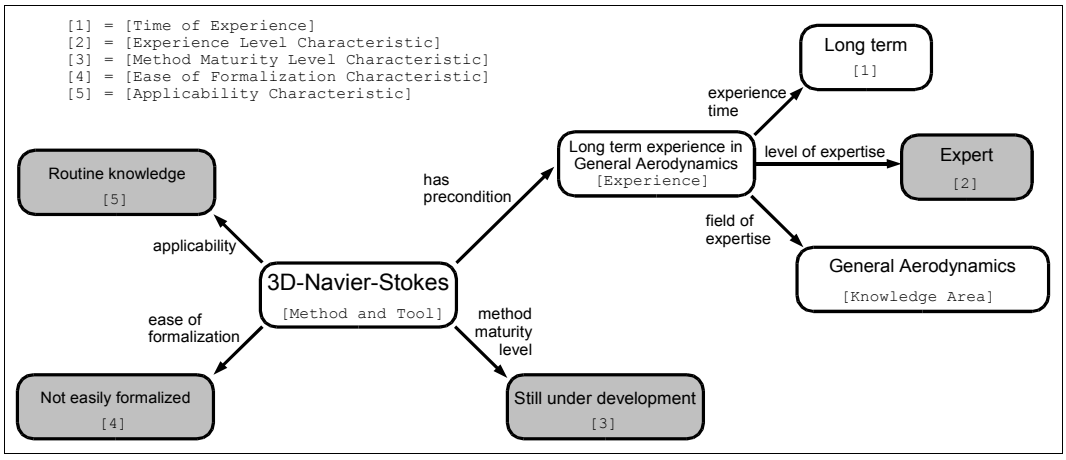

Figure 19
Assessment example of `3D-Navier-Stokes`. *Instances of* `KM Characteristics` *are marked gray.*

A (fictitious) example assessment of the instance 3D-Navier-Stokes is shown in figure 19 (on the previous page). The instances of `KM Characteristics` are marked with a gray background.

The example represents the following statement:

> *3D-Navier-Stokes is a method or tool, which is still under development. Knowledge about 3D-Navier-Stokes is routine knowledge that is not easily formalized. A precondition to having knowledge about 3D-Navier-Stokes is long term expertise in the area of general aerodynamics.*

The same instance of a `KM Characteristic` may be assigned to more than one `Knowledge Area`, indicating that these knowledge areas share the same value of a certain assessment characteristic. In the example above, the characteristic `Not easily formalized` may be assigned to any number of `Knowledge Areas`, indicating that all these knowledge areas cannot easily be formalized.

Part 4

Evaluation

Part 4 is an evaluation of the results of the KMOD project. This includes an evaluation of the KMOD ontology itself, the approach, and the tools which were used for the creation.

The chapter includes a number of suggestions for possible improvements in these three areas.

4 Part 4: Evaluation

The KMOD ontology has not yet been presented to "real" users in a production or demonstration system. Therefore, an evaluation of the ontology can only commence from a theoretical point of view, although the practical evaluation of users remains the final test of its adequacy.

4.1 Evaluation of the KMOD ontology

The KMOD ontology is an ontology for knowledge sharing between humans and as such it is primarily a situated ontology. It aims to provide a common means for the assessment of Airbus' knowledge, enabling people from different departments within Airbus to access these assessments and communicate about them.

Thus, an evaluation of the KMOD ontology needs to take into account the enabling-characteristics for boundary objects which were introduced in section 1.4 "Situated ontologies in communities of practice": accommodation, modularity, abstraction, and standardization.

Accommodation describes the ability to adapt to the different needs of different communities. In a sense, the KMOD ontology accommodates to the needs of the ontology developers through Protégé, and to the needs of users through the web portal. How it accommodates to the needs of different users (managers, new employees etc.) remains to be seen when it is put into use.

Modularity means that the ontology should contain different parts which address different groups of people. KMOD's distinction of a domain ontology part and a knowledge assessment ontology part provides a very simple kind of modularity. Exchanging the domain ontology part may offer a basic means to address a different group of users. The knowledge assessment ontology part can be seen as uniting the different domain ontologies under a common roof.

Abstraction means that the ontology should contain only relevant details. The KMOD ontology's level of detail was decided on the basis of the interview result documents. Again, it remains to be seen when the ontology is put into use whether the level of detail serves the purpose of the ontology.

Standardization means to provide an explicit and uniform way of use. The actual way of use of the KMOD ontology will be determined by the web portal which will provide the user interface. The KMOD ontology itself currently does not provide an explicit description of how it should be used. An informal description of the concepts in the ontology could be a first step in this direction.

The KMOD ontology exhibits some of the characteristics of a potential boundary object. Whether this is enough to actually become part of the daily practice of different communities will only become apparent when it is put into use. In any case, there is room for improvement with respect to all of the aforementioned characteristics. In particular, the introduction of informal descriptions of the

concepts offers a chance to enhance standardization without the need to change the formal parts of the ontology.

Recalling the problems of imposing a standard vocabulary encountered by Uschold and Jasper (2003) in the Boeing project, such informal descriptions might also be used to allow different communities to map their local terms to the global terms of the ontology, which aids accommodation. Alternatively, synonyms could be added by introducing a new slot `synonyms` of type `String` for every concept. This slot would contain a list of synonyms for a certain concept, which would require only minimal changes to the ontology. Other, more complex solutions are possible but require more complex changes to the ontology.[45]

Another improvement pertains to the aspect of modularity. Currently, the KMOD ontology's class-hierarchy serves two purposes: model the domain and serve as the basis for navigation in the web portal. This led to certain peculiarities, like the class `Person` appearing in two places of the hierarchy. A possible separation of these two different demands would consist of the actual modeling part (the ontology) and an explicit mapping part, that would map concepts from the ontology to navigation elements in the web portal. Making these possibly conflicting demands explicit results in a heightened awareness of them and may result in a more flexible and overall more adequate design.

Though being primarily a situated ontology, the formal aspects of the KMOD ontology should not be neglected. Gruber suggests a number of design criteria which may be used to evaluate a formal ontology [Gruber 1993], which have been introduced in section 1.5 “Computer-implemented ontologies”.

Clarity requires that the concepts of an ontology should be rigorously defined whenever possible and be documented by an informal description. The concepts of the KMOD ontology are neither rigorously defined nor do they provide an informal definition. They were developed from the interview results document. The results document together with this thesis may be viewed as a first step towards an informal description.

Coherence requires, that both the formal definitions and informal descriptions of the concepts will not lead to contradictions. To date, contradictions have not occurred in the KMOD ontology. Still, the point is that the KMOD ontology does in no way constrain the creation of such instances.[46] Therefore, the possibility of contradictions cannot be excluded for certain.

---

45 A more complex solution would be, for example, to use a slot of type `Instance` or even create a new class that serves as a synonym-relation (similar to `Experience`).

46 One reason for this is that such constraints would be difficult, in same cases maybe impossible to implement. Because only ontology experts are allowed to manipulate the ontology and create instances, constraints were avoided whenever possible.

Extendibility requires that new kinds of use should be anticipated and new concepts may be added as specializations without the need to change existing concepts. The KMOD ontology facilitates new kinds of using the ontology by distinguishing between concepts of the domain and concepts for knowledge assessment. It is expected that the domain concepts may be specialized or even replaced without a need to change the concepts for knowledge assessment and vice versa.

Minimal encoding bias occurs when design decisions are made for reasons of convenience or notation. As has been mentioned before, in the KMOD ontology this is the case with binary relations, which are represented as slots. These slots are attached to classes, thus, making the relations (slot) part of the definition of the class. This was done because treating relations as slots is easier to handle both in Protégé and in the web portal. Detaching the relations and representing them as independent entities might have been more faithful to the underlying conceptual view. For n-ary relations (with $n > 2$) this has been done (e. g. `Experience`), because using simple slots was not possible in these cases.

Minimal ontological commitment requires the ontology to make as few claims about the world as possible to support the given application. The KMOD ontology adheres to this principle by constraining the slot-values as little as possible. Thus, slots will usually allow more then one instance, even in cases where this does not seem necessary. For example, one may assume that something can only be a direct sub-part of a single super-structure. The KMOD ontology does not impose such a constraint. It allows an instance to be part of more than one super-structure.

Another example: One could also create an instance of `Experience` with the slot-values `Long Term` and `Beginner`, which may be understood as someone being a "long term beginner" in some field of expertise. The term "Beginner" implies that the person has learned the subject only recently, which in a sense contradicts the meaning of the phrase "long term".[47] Yet, this lack of constraint is a an example of minimal ontological commitment of the KMOD ontology.

If changes to the ontology are required, like in the case separating the representational and navigational concerns, the difficulty of the change should be weighed against the possible benefits. The evaluation of both situated and formal aspects suggests that the creation of informal description will be beneficial for users and ontology developers alike, while not requiring changes to the actual ontology[48]. Such descriptions should state the conceptual view of each concept and its use in the ontology. They would provide a significant improvement to the current ontology.

47 This is of course not a logical contradiction, but the seems to disagree with one's intuition, almost like an oxymoron. Giving this instance the benefit of the doubt, one might say that "beginner" is probably a misnomer and should rather be named "basic knowledge or experience".

48 The standard slot `Documentation` may be used for this purpose.

4.2 Evaluation of our approach

The development of the KMOD ontology was a central activity in the KMOD project. The KMOD project itself was managed using a standard method for projects at Airbus. While the overall project was guided by a method, the development of the KMOD ontology was not.

Our approach to the development of the ontology was mainly based on the other team member's previous experience from similar projects. The idea was to first conduct the interviews and then create from them a series of increasingly complex versions of the ontology, which should as a final step be deployed in a system for demonstration and testing purposes. The anticipated time frame for this whole process was approximately one year.

My work for the project ended before the KMOD ontology was actually presented to any users. At the end of this period the possible need for more feedback from future users and domain experts was discussed. This was after more then six month of ontology development.

Here, a problem of the chosen approach becomes apparent. It was assumed that the KMOD team will be able to build an adequate ontology solely based on the interview results. Thus, no measures were taken to ensure frequent evaluation and feedback from the domain experts (which are not part of the KMOD team). The interviews were the only means of letting the future users (and domain experts) take part in the development process.

In application-oriented software development, prototyping is a way to ensure the incorporation of user feedback into the development process [Züllighoven 2005]. Three kinds of prototyping can be distinguished based on their purpose: exploratory, evolutionary, and experimental prototyping [Floyd 19984]. The right kind of approach must be chosen based on the specific project and the stage of development of the ontology.

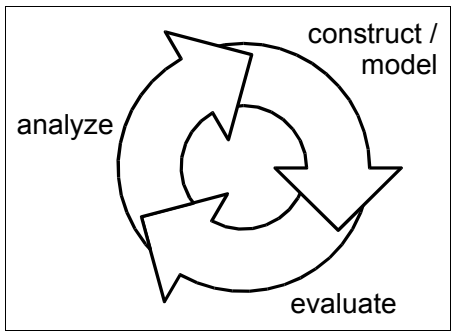

Figure 20
Recurring steps in the development of a prototype (based on [Floyd and Oberquelle 2004])

Evolutionary prototyping would have been well suited for the development of the successive versions of the KMOD ontology. Figure 20 shows an abstract illustration of the prototyping process. Based on the interviews (analyze) an initial version may be build (construct/model). The versions would serve as the prototypes and would be criticized

by the domain experts[49] (evaluate). This feedback is analyzed to build the next version of the ontology.

The time interval for a complete cycle of analysis, construction and evaluation depends on the specific project. I developed the KMOD ontology within six month, not including the interviews. If we had used a prototyping approach, three complete cycles seem to be a realistic estimate in retrospect.

Two factors must be taken into account when considering prototyping for ontology development:

- participation of domain experts
- availability of evaluation criteria

These are critical for the success of the approach. The participation of domain experts is of course crucial for the process. If they do not participate there is no improvement over the original approach used for the KMOD ontology.

The evaluation criteria are important for both domain experts and ontology developers to judge if the development is going in the right direction. These criteria should be directed towards the future use to ensure that the ontology is adequate for the intended purpose. The criteria mentioned in the previous section do not meet these requirements.

Competency questions ([Gruninger and Fox 1994]) are well suited to evaluate the adequacy of an ontology for a particular purpose:

> "[Competency questions] serve as benchmarks for the development of ontologies [...]. These questions not only characterize existing ontologies but also drive the development of new ontologies that are required to solve the competency questions."
>
> *[Gruninger and Fox 1994, 1]*

Competency questions are a set of questions that the future ontology should be able to answer. Also, they can be used to identify relevant concepts. Thus, competency questions can guide the ontology developers, while at the same time providing criteria to evaluate the fitness for the intended purpose. As an additional benefit competency questions may also serve as a starting point for the creation of informal descriptions.

Competency questions could also counter the negative effect which the early use of a tool like Protégé may have. In KMOD the effect was that we restricted our way of thinking about the ontology to constructs which Protégé would easily support. This happened after we switched from sketching the ontology by the informal documents to using Protégé. Gradually, we imposed Protégé's technological restrictions on the design of the ontology.

---

49 In the KMOD project the users regarded as the domain experts.

Another approach from software development, the creation of a *glossary*, might have also been fruitfully used for the development of the KMOD ontology. A glossary is "a dictionary of terms relevant in an application domain." ([Züllighoven 2005, 475]). Its purpose is the establishment and documentation of terms, that the project team agreed upon. It is important that the terms in the glossary should come from the application domain language and not from the developers [Züllighoven 2005, 477]. A list of terms can be compiled from the interviews, the competency questions, and from discussions with the domain experts.

Besides serving as the central repository for the common project language, the glossary may also be used to supply informal descriptions of ontology concepts.

Glossary and competency questions together form a solid basis for the creation of informal description that are firmly grounded in the practice of domain experts. Both glossary and competency questions can guide the development process. Using a prototyping approach ensures frequent feedback of domain experts about the adequacy of the ontology. Competency questions can be used to evaluate the progress.

4.3 Evaluation of the tools[50]

Protégé was the main tool for the creation of the KMOD ontology. It proofed flexible enough to accommodate the different demands of the KMOD ontology, specifically the combination of modeling a domain and building a user interface based on that domain.

Some problems we encountered when using Protégé can be easily solved. E. g. the standard slot `Name` must contain a unique value for every frame. This prevents the creation of classes, instances etc. with the same name. It can be solved by adding a new slot which will not have this restriction, e. g. `KMOD Name`, to the default meta-class `STANDARD-CLASS`.

There are some features that Protégé does not offer but which would have been helpful for the creation of the KMOD ontology:

1. support for refactoring
2. consistency check of inverse slot values[51]
3. a list of "favorite classes" or "bookmarks" that can be quickly accessed
4. effective visualization of the ontology

The features (1) and (2) would have been helpful to ensure that changes to existing concepts will not result in inconsistencies. E. g. changing the type of a template slot

50 I will concentrate on Protégé and the integration of Flora-2. I will not evaluate Tomcat/JSP because it was not used to create the ontology, but is only used to create the web portal.

51 When an existing slot is made inverse, Protégé will not ensure that instances that have this slot as an own slot contain the correct values. This has to be done by hand.

of some class may result in inconsistencies, if instances of that class exist. An instance of the class may have a slot-value in the changed slot that is not of the (newly changed) type. In Protégé a slot with an incorrect value is marked with a red border. That is the only way to identify the mistake. Therefore, after changing the slot type one has to manually check every instance for erroneous slot value. A refactoring feature could warn beforehand that a certain change will result in such inconsistencies and list the effected frames.

A list of "favorite classes" or "bookmarks" would have made the convenience classes (e. g. the class `Knowledge Related Classes`) dispensable. Recall that convenience classes served as a collection point for quick access to related classes. Making a class a "favorite class" should add that class to a special list which can be accessed quickly from anywhere in Protégé. This should in no way effect the actual class-hierarchy. This feature would be helpful to quickly navigate to important classes without the need to artificially change the class-hierarchy.

In the KMOD project, graph representations of the Protégé model were drawn by hand. The possibility to automatically generate simple graphs[52] from a Protégé model would enable effective communication with domain experts.

One feature we did not miss was the possibility to handle competency questions in an effective manner. Such feature could be used to support the evaluation phases of a prototyping approach mentioned above.[53]

The integration of Flora-2 into Protégé via the OntoQuery-plug-in is still very basic. A syntax error in an axiom or query will be passed to Flora-2 and will result in an empty search result. Thus, one always has to check the plug-in's debug-output to verify that an empty search is correct or if it is the result of a syntax error. It would be an improvement if the plug-in reports these syntax errors instead.

Other improvements of Flora-2 integration into Protégé, e. g. making the results of the axioms accessible from the Protégé user interface, would be much more difficult to implement.

52 Some graph generating plug-ins were available but the resulting graphs were even more complex then the Protégé user interface and therefore not suitable for effective communication.

53 A commercial ontology editor that integrates competency questions is, for example, OntoEdit (http://www.ontoprise.com).

5 Summary and outlook

In this thesis I have described the KMOD project, the KMOD ontology which I helped to develop during my participation in the project, and the tools we used.

The first part introduced three meanings of the term ontology: Ontology as a philosophical discipline, an ontology as the goal of Ontology, and ontology in informatics as a conceptual artifact. Based on the distinction of knowledge sharing between computer agents on the one hand, and knowledge sharing between humans on the other, the notions of *formal* and *situated ontologies* were distinguished.

In the second part the KMOD project and the tools we used were described. This included the description of related work, which had been considered for reuse in the project, and an introduction to the Protégé meta-model. The chapter ends with a characterization of the ontology's development process as a series of increasingly complex ontology versions. This led over to the description of the KMOD ontology, which is the subject of part three.

Part three presents the KMOD ontology as a situated ontology for knowledge sharing between humans. It begins with an introduction to the underlying conceptual view, followed by an account of the central concepts of KMOD ontology.

Finally, in part four the KMOD ontology, our approach, and the tools were evaluated.

The KMOD ontology was evaluated with respect to the enabling-characteristics of boundary-objects and the criteria suggested by Gruber (1993). It was concluded that the adoption of informal descriptions will significantly improve the ontology's level of standardization and clarity.

A prototyping approach was suggested as a systematic and controlled alternative to the way we created the ontology as a series of ontology versions. This would also ensure early and frequent feedback from domain experts. To support the evaluation of the ontology during its development, the use of competency questions and the creation of a glossary was suggested. The use of a project glossary was also identified as a means to creating informal descriptions for ontology concepts.

The evaluation of Protégé pointed to a number of possible improvements. Especially enhancements in the area of effective visualization of ontologies and facilities for the integration of competency questions would be needed to support a prototyping approach for ontology development.

Outlook

The KMOD project aimed at the creation of an ontology-based system for knowledge sharing between humans. For this purpose the KMOD ontology was developed. Future users will need to judge whether this original goal was achieved.

Ontologies are expected to solve many problems in the field of automated knowledge sharing between both humans and software agents. Yet this subject seems to be treated as pertaining mainly to knowledge sharing between software

agents while neglecting what I have called the situatedness of ontologies. Future work on ontologies in informatics should further clarify and investigate the notion of situated ontologies as a means for knowledge sharing and reuse between humans.

Both application-oriented software development and the study of boundary objects seem a good starting point to learn how situated ontologies can be developed and how they may support knowledge sharing and reuse between humans. A promising approach may be the adoption of existing classifications for the creation of ontologies. Empirical evidence is needed to show if and how application-oriented software development methods can be successfully used for the creation of situated ontologies.

Future work could also investigate how tools may adequately support the creation of situated ontologies. Effective visualization is not well supported by existing tools. I believe that this is a key factor to support the communication between ontology developers, users, and domain experts.

6 Bibliography

[Berners-Lee et al. 2001]
Berners-Lee, T.; Hendler, J.; Lassila, O.: *The Semantic Web*, in: Scientific American, May 2001.
Available at (accessed: 1/10/2004):
http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

[Bowker and Star 1999]
Bowker, G. C.; Star, S. L.: *Sorting things out: classification and its consequences,* The MIT press, Cambridge, Massachusetts, 1999.

[Clancey 1993]
Clancey, W. J.: *The knowledge level reinterpreted: Modeling socio-technical systems,* in: Ford, K. M.; Bradshaw, J. M. (editors): Knowledge Acquisition as Modeling, John Wiley & Sons: New York, 1993, pp. 33-50.
Available at (accessed: 02/06/2005):
http://cogprints.org/312/00/125.htm

[Curd & Cover 1998]
Curd, M.; Cover, J. A.: *Philosophy of Science: The Central Issues*, Norton & Company, New York, USA, 1998.

[Domingue et al. 2001]
Domingue, J; Motta.E.; Buckingham Shum, S.; Vargas-Vera, M.; Kalfoglou, Y.; Farnes, N.: *Supporting Ontology Driven Document Enrichment within Communities of Practice,* in: Proceedings of the First International Conference on Knowledge Capture, Victoria, British Columbia, Canada, October 21-23; ACM Press, New York, 2001, pp. 30-37.
Available at (accessed 12/05/2004):
http://kmi.open.ac.uk/projects/akt/kcap01_john_final.pdf

[Flick 1995]
Flick, U.: *Qualitative Forschung: Theorie, Methoden, Anwendung in Phsychologie und Sozialwissenschaften*, Rowohlt Verlag, Hamburg,Germany, 1995.

[Floyd 1984]
Floyd, C.: *A systematic look at prototyping*, in: Budde, R.; Kuhlenkamp, K.; Mathiassen, L.; Züllighoven, H.: Approaches to Prototyping, Springer-Verlag, Berlin, 1984, 1-18.

[Floyd, Reisin and Schmidt 1989]

Floyd, C.; Reisin, F.-M.; Schmidt, G.; *STEPS to Software Development with Users*, in: Ghezzi, C.; McDermid, J. A. (editors): ESEC '89, Lecture Notes in Computer Science no. 387, Springer, Berlin / Heidelberg, Germany, 1989, pp. 48-64.

[Floyd and Oberquelle 2004]

Floyd, C.; Oberquelle, H.: Lecture notes for the 2004/05 lecture "Softwaretechnik und Softwareergonomie" at the University of Hamburg. Available at (accessed: 01/15/2005):
`http://swt-www.informatik.uni-hamburg.de/attachments/LVTermine/WS04-05_VL-06_STE_Prototyping.pdf`

[Fox and Fadel 1993]

Fox, M.; Chionglo, J. F.; Fadel, F. G.: *A Common Sense Model of the Enterprise*, in: Proceedings of the 2nd Industrial Engineering Research Conference, Norcross, Georgia; Institute for Industrial Engineers, 1993, pp. 425-429.
Available at (accessed: 12/4/2003):
`http://www.eil.utoronto.ca/enterprise-modelling/papers/fox-ierc93.pdf`

[Genesereth and Nilsson 1987]

Genesereth, M. R.; Nilsson, N. J. : *Logical Foundation of Artificial Intelligence,* Morgan Kaufmann, Los Altos, California, 1987.

[Guarino 1995]

Guarino, N.: *Formal Ontology, Conceptual Analysis and Knowledge Representation,* in: Guarino, N.; Poli, R. (editors): International Journal of Human and Computer Studies, Special issue on Formal Ontology, Conceptual Analysis and Knowledge Representation, vol. 43, no. 5/6, 1995, pp. 625-640.

[Guarino 1997]

Guarino, N.: *Understanding, building, and using ontologies,* International Journal of Human and Computer Studies 46, 1997, pp. 293-310.

[Guarino 1998]

Guarino, N.: *Formal Ontology in Information Systems,* in: Guarino, N. (editor): Formal Ontology in Information Systems 1998 (Proceedings), IOS Press, Amsterdam, The Netherlands, 1998.

[Gruber 1993]

Gruber, T. R.: *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*, in: Guarino, Nicola; Poli, Roberto (editors): Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishing, 1993, pp. 907-928.

[Gruninger and Fox 1994]

Gruninger, M.; Fox, M. S.: *The Role of Competency Questions in Enterprise Engineering*, in: Proceedings of the IFIP WG5.7 Workshop on Benchmarking-Theory and Practice, Trondheim, Norway, 1994.
Available at (accessed: 11/21/2004):
`http://www.eil.utoronto.ca/enterprise-modelling/papers/benchIFIP94.pdf`

[Hobbs 1985]

Hobbs, J. R.: *Ontological Promiscuity*, Proceedings of the 23rd conference on Association for Computational Linguistics, Association of Computer Linguistics, Morristown, New Jersey, 1985, pp. 60-69.

[Maedche and Staab 2001]

Maedche, A.; Staab, S.: *Ontology Learning for the Semantic Web*, in: *IEEE Intelligent Systems*, volume 16, no. 2, 2001, pp. 72-79.

[Mahesh and Nirenburg 1995]

Mahesh, K.; Nirenburg, S.: *A Situated Ontology for Practical NLP*, in Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence (IJCAI-95), Aug. 19-20, Montreal, Canada, 1995.

[Noy, Fergersen and Musen 2000]

Noy, N. F.; Fergerson, R. W.; Musen, M. A.: *The knowledge model of Protege-2000: Combining interoperability and flexibility*, in: Dieng, R.; Corby, O. (editors): Knowledge Engineering and Knowledge Management. Methods, Models, and Tools: Proceedings of 12th International Conference, EKAW 2000, Juan-les-Pins, France, October 2-6, Springer-Verlag, Berlin, Germany, 2000.
Available at (accessed 12/12/2004):
`http://www-smi.stanford.edu/pubs/SMI_Reports/SMI-2000-0830.pdf`

[Noy and McGuinness 2001]

Noy, N. F., McGuinness. D. L.: *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, University of Stanford, California, March 2001.

[Nygaard 1986]
Nygaard, K.: *Program Development as a Social Activity*, in: Kugler, H.-J. (editor): Information Processing 86. Proceedings of the 10th IFIP World Computer Congress '86 Dublin, Elsevier Science Publishers, Amsterdam, The Netherlands, 1986, pp. 189-198.

[Oppermann, Schnurr and Studer 2001]
Oppermann, H.; Schnurr, H.-P.; Studer, R.: *Die Bedeutung von Ontologien für das Wissensmanagement*, in: wissensmanagement 6, 2001, pp. 33-36.

[Prechtl and Burkard 1996]
Prechtl, P.; Burkard, F.-P. (editors): *Metzler Philosophie Lexikon*, Verlag J. B. Metzler, Stuttgart, Germany, 1996.

[Smith and Welty 2001]
Smith, B.; Welty, C. (editors): *2nd International Conference on Formal Ontology in Information Systems: FOIS 2001,* Ogunquit, Maine; October 17-19, 2001; ACM Press, New York, 2001.

[Sowa 2000]
Sowa, J. F.: *Knowledge Representation: logical, philosophical, and computational foundations*, Brooks / Cole; Pacific Grove, California, 2000.

[Staab 2002]
Staab, Steffen: *Wissensmanagement mit Ontologien und Metadaten*, in: Informatik Spektrum, Springer-Verlag, Vol. 25, No. 3, 2002, pp. 124-209.

[Star 1989]
Star, S. L.: *The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving*, in: Gasser, L.; Huhns, M. (editors): *Distributed Artificial Intelligence*, Volume 2, Pitman / Morgan Kaufmann Publishers, London, UK, 1989.

[Ukena 2003]
Ukena, S.: *Ontologieansätze in Internet-basierten IT-Systemen am Beispiel der elektronischen Verwaltung*, Studienarbeit, University of Hamburg, Germany, 2003.
Available at (accessed: 12/12/2004):
`http://swt-www.informatik.uni-hamburg.de/publications/files/Stud/Stefan%20Ukena%20-%20Ontologieansaetze.pdf`

[Uschold et al. 1998]

Uschold, M.; Kind, M.; Moralee, S.; Zorgios, Y.: *The Enterprise Ontology*, in: Uschold, M.; Austin, T. (editors): The Knowledge Engineering Review, Vol. 13, Special Issue on Putting Ontologies to Use, 1998.
Available at (accessed: 10/20/2003):
`http://www.aiai.ed.ac.uk/project/pub/documents/1998/98-ker-ent-ontology.ps`

[Uschold and Jasper 2003]

Uschold, M.; Jasper, R.: *Enabling Task-Centered Knowledge Support through Semantic Markup,* in: Fensel, D.; Wahlster, H.; Lieberman, J. H. (editors): Spinning the Semantic Web, MIT-Press, 2003, pp. 223-251.

[Welty et al. 1999]

Welty, C.; Lehmann, F.; Gruninger G.; Uschold M.: *Ontology: Expert Systems All Over Again?*, Opening panel at the American Association for Artificial Intelligence National Conference 1999 (AAAI-99), Powerpoint-presentation, Austin, Texas, 1999.
Available at (accessed: 12/12/2004):
`http://www.cs.vassar.edu/faculty/welty/presentations/aaai-99/`

[Wenger 1998]

Wenger, E.: *Communities of Practice*, Cambridge University Press, New York, USA, 1998.

[Züllighoven 2005]

Züllighoven, H.: *Object-Oriented Construction Handbook: Developing Application-Oriented Software with the Tools & Material Approach,* dpunkt.verlag / Morgan Kaufmann Publishers, Heidelberg, Germany, 2005.

[Zúñiga 2001]

Zúñiga, G. L.: *Ontology: Its Transformation from Philosophy to Information Systems,* in: Smith, Barry; Welty, Christopher (editors): 2nd International Conference on Formal Ontology in Information Systems: FOIS 2001, Ogunquit, Maine, USA. October 17-19, 2001, ACM Press, New York, 2001, pp. 187-197.

7 Appendix A – List of inverse slots

If two *slots* are defined as *inverse slots* (or one *slot* as an *inverse* of itself) and a value is assigned to one of them, then Protégé will automatically add the correct value to the other. E. g. the `has direct sub-part` and `is direct sub-part of` *slots* of `Structure` are defined as *inverse slots*. Adding an *instance* A to the `has direct sub-part` *slot* of B will automatically add B to A's `is direct sub-part of` *slot* as well.

Note that this will only work, if the inverse slots are defined before assigning the values. Changing existing *slots* with already assigned values to *inverse slots* will not have this effect.

Represented relation or property	***Inverse slot(s)***
a method or tool is used to create certain results	• `results in (Result)` • `method/tool used (Method and Tools)`
a process yields a result	• `yields result (Result)` • `is result of (Process)`
a results describes a structure	• `details structural component (Structure)` • `inverse of details structural component (Result)`
a results may impact engineering criteria	• `has impact on engineering criteria (Engineering Criteria)` • `is influenced by (Result)`
direct part-of relation	• `has direct sub-parts (Structure)` • `is direct sub-part of (Structure)`
direct subprocess hierarchy	• `had direct sub-process (Process)` • `is direct super-process (Process)`
directly related to	• `directly related to (Knowledge Area)`
has experience	• `has experience (Experience)` • `organizational entity having this experience (Organizational Entity)`
has precondition	• `has precondition (Experience)` • `experience is necessary for knowledge area (Knowledge Area)`
hierarchy of physics knowledge areas	• `physics sub knowledge area (Physics)` • `physics super knowledge area (Physics)`

Represented relation or property	***Inverse slot(s)***
influence of KM function on KM criteria (1)	• km function slot (KM Function) • influence relation on function side (Function Influence on Criteria)
influence of KM function on KM criteria (2)	• km criterion slot (KM Criteria) • influence relation on criteria side (Function Influence on Criteria)
is involved in	• is involved in (Knowledge Area)
level of expertise	• level of expertise (Experience Level Characteristic) • expertise value applies to these preconditions (Experience)
method and tool hierarchy	• more general method or tool (Method and Tools) • more specific method or tool (Method and Tools)
method or tool can support a process	• supporting methods & tools (Method and Tools) • used to support process (Process)
organizational units at manufacturing site	• organizational units at this site (Organizational Unit) • located at site (Manufacturing Site)
physics can be modeled using a method or tool	• used for modeling (Physics) • can be modeled with (Method and Tools)
process hierarchy	• has direct sub-process (Process) • is direct sub-process (Process)
process sequence	• directly follows (Process) • directly precedes (Process)
responsible site	• responsible site (Manufacturing Site) • has system leadership for (Knowledge Area)
result hierarchy	• has direct sub-result (Result) • has direct super-result (Result)
time of experience	• experience time (Time of Experience) • inverse of experience time (Experience)

Erklärung*

Ich bestätige, dass ich die vorliegende Arbeit ausschließlich unter der Verwendung angegebenen Quellen und Hilfsmittel angefertigt habe.

Hamburg, 4. April 2005

* This statement declares, that this thesis has been prepared by myself exclusively with the literature and tools that are mentioned in the text.